USERS

¡CONSEJOS APLICABLES A LA PRÁCTICA PROFESIONAL ACTUAL!

XHTML + CSS + JAVASCRIPT + AJAX

DESARROLLADOR WEB

CLAVES Y TÉCNICAS PARA OPTIMIZAR SITIOS

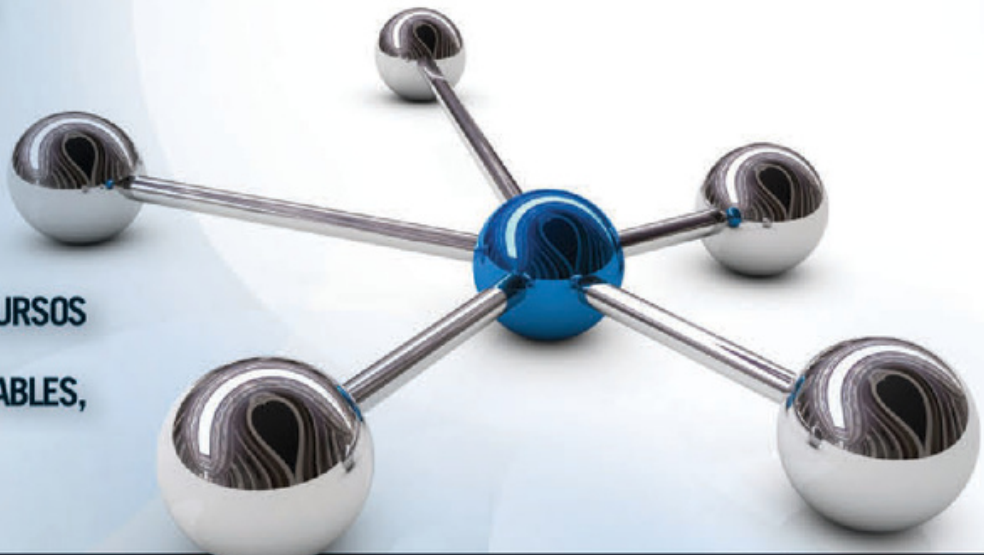
OBTENER MAYOR COMPATIBILIDAD EN NAVEGADORES

METODOLOGÍAS PARA ESCALAR POSICIONES EN LOS BUSCADORES

DISMINUIR LOS TIEMPOS DE RESPUESTA Y EL CONSUMO DE RECURSOS

DESARROLLO DE APLICACIONES ESTABLES, RÁPIDAS Y SEGURAS

por Francisco Minera



MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS MAN

SOLUCIONES PROFESIONALES Y EFICIENTES

CONÉCTESE
CON LOS MEJORES LIBROS DE COMPUTACIÓN

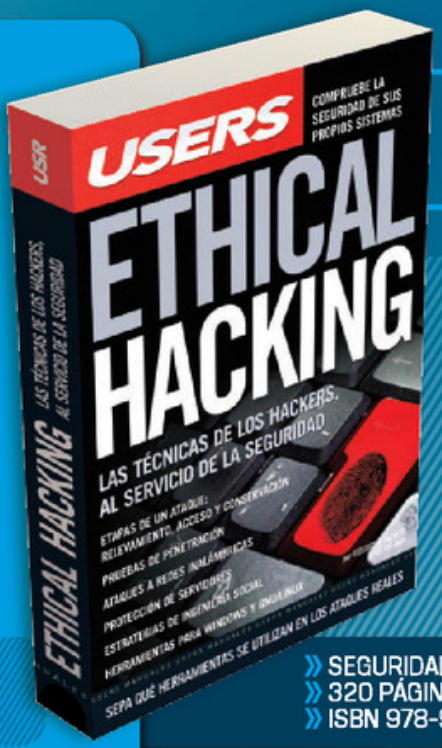
LLEGAMOS A TODO EL MUNDO
VÍA OCA * Y DHL **

usershop.redusers.com
usershop@redusers.com

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

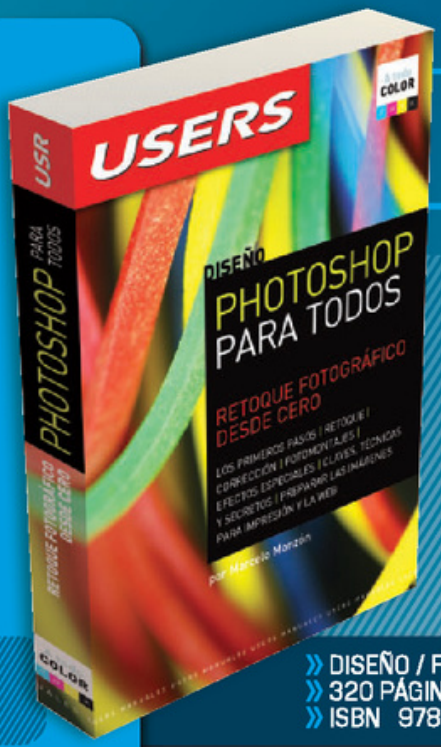
DESCUBRA UN NUEVO NIVEL EN EL DESARROLLO DE APLICACIONES WEB

» DESARROLLO / INTERNET
» 352 PÁGINAS
» ISBN 978-987-663-010-8



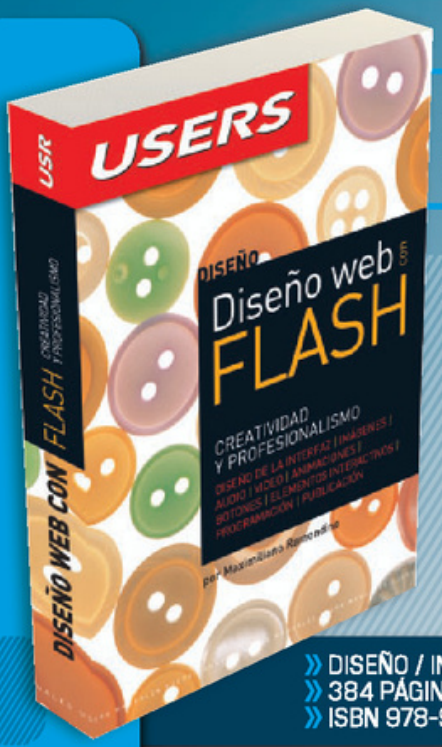
CONOZCA LAS TÉCNICAS DE LOS HACKERS Y CONSIGA UN SISTEMA SEGURO

» SEGURIDAD / INTERNET
» 320 PÁGINAS
» ISBN 978-987-1347-93-3



RETOQUE Y EDITE FOTOGRAFÍAS CON LA HERRAMIENTA MÁS PODEROSA

» DISEÑO / FOTOGRAFIA
» 320 PÁGINAS
» ISBN 978-987-1347-57-5



ELABORE IMPACTANTES SITIOS WEB CON FLASH

» DISEÑO / INTERNET
» 384 PÁGINAS
» ISBN 978-987-1347-34-6

DESARROLLADOR WEB

CLAVES Y TÉCNICAS PARA OPTIMIZAR SITIOS

por Francisco Minera

RedUSERS

TÍTULO: DESARROLLADOR WEB

AUTOR: Francisco Minera

COLECCIÓN: Manuales USERS

FORMATO: 17 x 24 cm

PÁGINAS: 400

Copyright © MMXI. Es una publicación de Fox Andina en coedición con Gradi S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. Esta publicación no puede ser reproducida ni en todo ni en parte, por ningún medio actual o futuro sin el permiso previo y por escrito de Fox Andina S.A. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en I, MMXI.

ISBN 978-987-1773-09-1

Minera, Francisco
Desarrollador web. - 1a ed. - Buenos Aires : Fox Andina; Banfield - Lomas de Zamora: Gradi, 2011.
400 p. ; 24x17 cm. - (Manual users; 204)

ISBN 978-987-1773-09-1

1. Informática. I. Título

CDD 004.1

LÉALO ANTES GRATIS

EN NUESTRO SITIO PUEDE OBTENER, DE FORMA GRATUITA, UN CAPÍTULO DE CADA UNO DE LOS LIBROS

RedUSERS
COMUNIDAD DE TECNOLOGIA

redusers.com

Nuestros libros incluyen guías visuales, explicaciones paso a paso, recuadros complementarios, ejercicios, glosarios, atajos de teclado y todos los elementos necesarios para asegurar un aprendizaje exitoso y estar conectado con el mundo de la tecnología.

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

Francisco Minera

Analista de Sistemas y Programador Superior, sus tareas están relacionadas con el desarrollo de sistemas y plataformas para pequeñas y medianas empresas en diversos lenguajes y arquitecturas. Es autor de los libros *PHP y MySQL*, *Proyectos con PHP*, *XML*, *PHP 5*, *Ajax*, *PHP Master*, *PHP 6* y *PHP Avanzado*.

Si se quiere contactar con el autor, siempre es posible enviarle sugerencias o comentarios a la dirección de correo electrónico **francisco.minera@gmail.com**.

Dedicatoria

Este libro está dedicado a todos los que hicieron posible su publicación. Saludos y gracias.

PRÓLOGO

El proceso de desarrollo de un sistema está constituido de múltiples capas o procesos, cada uno con un destino específico, y los factores que intervienen a la hora de generar de manera exitosa una aplicación son muchos. Saber cuáles son las opciones que tenemos y elegir la mejor alternativa disponible para llevar a cabo una determinada tarea es de una importancia prioritaria.

Al momento de escribir este libro surgieron dos premisas: que la estructura fuera lo más simple posible (en general cada capítulo está orientado a una regla de optimización en particular, http://developer.yahoo.com/performance/rules.html) y traducir cada concepto a un nivel terrenal, a casos de uso diario, es decir que la parte práctica tuviera un papel importante dentro de la obra.

Este libro no está orientado a ningún lenguaje o herramienta en particular, por lo que se espera que sea de utilidad para cualquier tipo de desarrollo web, independientemente de las tecnologías utilizadas. De hecho tratamos de abarcar la mayor cantidad de herramientas para que el lector pueda apreciar el panorama general y no se sienta atado a una opción en particular. Además está organizado de una manera tal que posibilita consultas rápidas de acuerdo a las necesidades del lector, por lo cual cada capítulo puede leerse de manera independiente, sin un orden prefijado. Además tratamos de incluir ejemplos prácticos y enlaces a sitios web para permitir una profundización de los temas tratados.

Si bien es cierto que las velocidades de conexión a Internet aumentan sin cesar, es importante mantener nuestros sistemas lo mas optimizados que se pueda. A lo largo de los diferentes capítulos intentaremos enumerar los principios y las herramientas necesarias para llevar a cabo esta tarea. Cada regla es simple y no conlleva demasiados inconvenientes aplicarlas, por lo cual lo más importante será la incorporación de conceptos: saber cómo y porque aplicar tal o cual regla.

En definitiva, este libro apunta a introducir y afianzar a sus lectores en los resultados que puedan lograrse a partir del uso de las herramientas de desarrollo mencionadas a los largo de los diferentes capítulos.

EL LIBRO DE UN VISTAZO

La optimización de sitios web es un conjunto de tareas que abarcan diferentes aspectos de un desarrollo, y en este libro veremos las opciones disponibles para aplicar a nuestras páginas. En cada capítulo analizaremos tecnologías, herramientas y aplicaciones para llevar a cabo las tareas necesarias y lograr resultados visibles por nuestros visitantes.

Capítulo 1

CLAVES PARA LA OPTIMIZACIÓN DE SITIOS

En este primer capítulo haremos un recorrido por las principales motivaciones existentes para mantener nuestros sitios optimizados. También veremos cuáles son las características y las opciones con las que contamos para efectuar los cambios necesarios en nuestras páginas.

Capítulo 2

XHTML

Surgido para extender y direccionar la funcionalidad de HTML, XHTML nos permite desarrollar páginas válidas y bien formadas, y en esta sección veremos cómo aplicar las reglas existentes para lograr explotar todas sus ventajas y respetar los estándares actuales.

Capítulo 3

CSS

Las hojas de estilo en cascada son muy populares desde hace ya mucho tiempo y en este capítulo veremos cuáles son las posibilidades para incluir CSS y ligar las expresiones a los elementos de nuestros documentos, tratando de ahorrar la mayor cantidad de código posible.

Capítulo 4

JAVASCRIPT

Este lenguaje del lado cliente es ya un clásico conocido por la gran mayoría de los autores de páginas y lo utilizaremos en los distintos tipos de desarrollos que emprendamos.

Capítulo 5

IMÁGENES

En este apartado haremos un recorrido por los formatos de imagen más habituales, revisaremos sus características y los casos de utilización de cada uno. Además, veremos algunas de las herramientas disponibles para tratar de incluir imágenes en nuestras páginas web. También conoceremos los formatos de imágen más utilizados y analizaremos los conceptos relacionados con la precarga de imágenes.

Capítulo 6

AJAX

A través de este capítulo nos dedicaremos a realizar un repaso por los principios de Ajax y además analizaremos algunas de las herramientas que necesitamos para proceder con su implementación, entre ellas conoceremos las desventajas que presenta con respecto al modelo tradicional y su relacion con otros lenguajes de programación como Java, .Net y PHP.

Capítulo 7

HTTP

Disminuir el número de peticiones HTTP es una de las reglas más importantes para optimizar nuestros sitios, y en este capítulo intentaremos analizar las características de este protocolo y las herramientas disponibles para controlar su funcionamiento. Además, veremos cómo comprimir contenido web y los usos más importantes del archivo .HTACCESS.

Capítulo 8

HERRAMIENTAS COMPLEMENTARIAS

En la actualidad contamos con un sinnúmero de herramientas y aplicaciones para controlar el desempeño de nuestras páginas, y en este capítulo haremos un repaso por algunas de las más importantes, e incluiremos los detalles acerca de su utilización.

Capítulo 9

POSICIONAMIENTO DE SITIOS EN GOOGLE Y HERRAMIENTAS DE CONTROL PROVISTAS

El buscador de Internet más popular del momento mantiene algunas directivas que no sólo nos permiten ubicar nuestros sitios en los listados de búsqueda sino también escalar posiciones dentro de ellos. Veremos cuáles son esas reglas y cómo Google nos da la posibilidad, mediante sus herramientas para webmasters, de obtener toda la información acerca de nuestras páginas.

Anexo A

REFERENCIA JAVASCRIPT

Este lenguaje de propósito general es sin dudas uno de los más empleados en todo tipo de desarrollos, y en este apéndice observaremos sus cualidades y la sintaxis que utiliza para llevar a cabo tareas e interactuar con los elementos de una página web.

Anexo B

REFERENCIA CSS 2

El estándar CSS evoluciona en cada nueva versión e incluye posibilidades que ayudan al desarrollador a incrementar las capacidades referidas al diseño de sus páginas. En este anexo haremos un repaso de CSS 2, la versión aceptada por la mayoría de los navegadores.

Anexo C

REFERENCIA XML

Este metalenguaje se ha convertido en una tecnología aceptada y utilizada por aplicaciones de todo tipo. Comprender sus reglas, claras y a la vez estrictas, nos permitirá entender la lógica en el manejo de XHTML.

Anexo D

REFERENCIA JSON

Gracias a la popularidad que ganó en el último tiempo y a que se posicionó como una alternativa válida a la hora de intercambiar datos entre aplicaciones, JSON se expandió en base a su simplicidad y a la gran cantidad de versiones para distintos lenguajes.

Servicios al lector

En este último apartado encontraremos un índice de palabras utilizadas en el libro para encontrar rápidamente lo que necesitamos.

! INFORMACIÓN COMPLEMENTARIA

A lo largo de este manual encontrará una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados.
Cada recuadro está identificado con uno de los siguientes iconos:

CURIOSIDADES E IDEAS

ATENCIÓN

DATOS ÚTILES Y NOVEDADES

SITIOS WEB

CONTENIDO

Sobre el autor 4
Prólogo 5
El libro de un vistazo 6
Información complementaria 7
Introducción 12

Capítulo 1

CLAVES PARA LA OPTIMIZACIÓN DE SITIOS

Situación actual 14
Conceptos generales 15
Importancia de los estándares 17
Reglas de optimización 18

Herramientas tratadas 19
Niveles de optimización 20
Conocimientos requeridos 21
Resumen 21
Actividades 22

Capítulo 2

XHTML

Origen e historia 24
Tipos de documento 26
- Strict 27
- Transitional 28
- Frameset 29
- XHTML 1.1 29

Reglas de marcado 30
Juegos de caracteres 34
Cabeceras 35
Validación 36
Limpieza con HTML Tidy 39
- Técnicas de reducción 41
- Compatibilidad entre navegadores 42

Estructuración de documentos 44
Uso de frames 45
DHTML (HTML Dinámico) 46
Inclusión de JavaScript 48
Inclusión de CSS 49
Elementos disponibles 50
Módulos en XHTML 57
XHTML 2.0 58
Resumen 61
Actividades 62

Capítulo 3

CSS

Consejos prácticos 64
Inclusión de documentos 64
Posicionamiento 70
Escritura reducida 71
- Fuentes 74
- Fondos 75
- Listas 75

Agrupación de estilos 75
Colores RGB 77

Imágenes en conjunto 79
Opciones adicionales 81
- Aplicaciones para el tratamiento de hojas de estilo 82

Juego de caracteres 88

Validación de documentos 88
Compatibilidad entre navegadores 89
Diseño sin tablas 92
Resumen 93
Actividades 94

Capítulo 4

JAVASCRIPT
Ubicación de los scripts 96
Unificación de archivos 97
Minify 97
Reducción de documentos 104
JSMin 104
Packer 108

Bananascript.com 109
ShrinkSafe 111
Pack:tag 111
YUI Compressor 112
Escribir menos código con Prototype 113
Características y modo de uso 114
Resumen 129
Actividades 130

Capítulo 5

IMÁGENES
Situación y necesidades 132
Tipos de compresión 132
Formatos disponibles 132
JPEG 133
GIF 134
PNG 136
Tratamiento de imágenes 138
PhpThumb 140
IrfanView 145
Precarga de imágenes 147
Resumen 151
Actividades 152

Capítulo 6

AJAX
Características y tecnologías 154
componentes 154
Cambios introducidos 157
Principios y origen del término 158
El objeto XMLHttpRequest 159
Compatibilidad 162
Desventajas en relación con el modelo tradicional 165
JavaScript 165
Internet Explorer 165
Navegadores actuales 166
CSS 166
Seguridad 166
Características del modelo 166
Implementación con Prototype 167
Disminución del tiempo de respuesta y uso del caché 173
Ajax y los lenguajes de programación 173

Java 174
.Net 176
PHP 176
JavaScript 178
Resumen 179
Actividades 180

Capítulo 7

HTTP
Introducción 182
Contenido comprimido 184
Reducción de peticiones HTTP 189
Contenido distribuido 190
Redes de distribución 191
Cabeceras 191
Posicionamiento en buscadores 194

El archivo .HTACCESS 197
Control de errores 199
Índices de directorio 201
Control de acceso por IP 202
Redirecciones 203
Fecha de expiración 203
Control de acceso por directorio 204
URLs cortas y amigables 205
Directivas PHP con php_value y php_flag 212
Herramientas de control 213
Fiddler 213
LiveHTTPHeaders 216
HttpWatch 216
Resumen 217
Actividades 218

Capítulo 8

HERRAMIENTAS COMPLEMENTARIAS
Introducción 220
Firebug 220
Web Developer 247
Web Inspector 257
Opera Dragonfly 258
XRAY 260
DebugBar 260
Companion.JS 263
Internet Explorer Developer Toolbar 264
Resumen 265
Actividades 266

Capítulo 9

POSICIONAR SITIOS EN GOOGLE
Introducción 268
Herramientas disponibles 269
Sitios de calidad 271
Rastreo 273
Googlebot 275
Índices 276
Sitemaps 278
Resumen 279
Actividades 280

Anexo A

REFERENCIA JAVASCRIPT
Introducción 282
Sintaxis básica 284
Comentarios 285
Variables 285
Operadores disponibles 288
Tipos de datos 289
Estructuras de control 291
Funciones 296
Implementación DOM 301

Eventos 305
Manejo y control de errores 308

Anexo B

REFERENCIA CSS 2

Introducción 310
Opciones disponibles 311
Comentarios 311
Selectores 311
Pseudo clases y elementos 320
Contadores 322
Unidades de medida 325

Colores 326
Cursores 328
Textos 329
Fuentes 331
Tablas 333
Fondos 333
Bordes 334
Dimensiones 336
Listas 336
Márgenes 337
Posicionamiento 338
Visibilidad 339

Anexo C

REFERENCIA XML

Orígenes y principios generales 342
Sintaxis 343
Reglas de marcado 344
Comentarios 349
Prólogo de un documento 350
DTD 352
Tipos de documento 356
Creación y visualización de documentos 359
Casos de uso 363
Relación con las bases de datos 365
Espacios de nombres 365
Inclusión de hojas de estilo 367
Transformaciones 367

Anexo D

REFERENCIA JSON

Introducción 370
Implementaciones 371
XML y JSON 372
Sintaxis de JSON 377
JavaScript y JSON 377
Ejemplo: Listado de productos con Prototype y JSON 383
Implementación PHP de JSON 389

Servicios al lector

Índice temático 394

INTRODUCCIÓN

El termino optimización es usado de manera frecuente en este libro, y con él nos referimos a la necesidad no solo de que los sistemas que desarrollamos funcionen, sino que lo hagan utilizando la menor cantidad de recursos posible. Para lograr una eficacia razonable en nuestros scripts, en los capítulos que siguen intentaremos adentrarnos en las técnicas planteadas por Yahoo!, que nos servirán de punto de partida para avanzar sobre un gran número de herramientas disponibles en el mercado, la gran mayoría de ellas de uso libre y gratuito.

Una de las grandes virtudes de Yahoo! fue haber concentrado en un mismo documento varios de los conceptos más importantes en cuanto a la optimización de sitios se refiere. Las explicaciones son simples y claras. En este libro lo que pretendemos es presentar cada una de ellas de manera individual y aplicar un grado mayor de profundidad.

Uno de los objetivos que tiene este libro es brindar soluciones que estén al alcance de todos, es decir que se apliquen a toda clase de proyectos, ya sean simples o complejos. Y además brindar la posibilidad de tomar un proyecto ya terminado y modificarlo aplicando las reglas de optimización. En cuanto a la estructura de cada capítulo, hemos tratado de aplicar el mismo método en todos ellos: primero avanzar a paso firme sobre la teoría de cada regla y luego rápidamente ir hacia la parte práctica, para fijar conceptos.

En general cuando escuchamos hablar de optimización pensamos directamente en el posicionamiento de nuestros sitios en los principales motores de búsqueda. Si bien una parte de esta obra está dedicada específicamente a ese tema, el concepto de optimización es llevado un poco mas allá, intentando abarcar temas referidos a la velocidad de respuesta de nuestras páginas y a la forma en que los usuarios acceden a los contenidos ofrecidos.

En síntesis, la idea es brindar una serie de conceptos claros y precisos para que podamos incorporarlos rápidamente en el día a día de nuestros desarrollos, y además contar con las herramientas adecuadas para implementar las reglas de optimización en nuestros sitios.

En cuanto a los requerimientos, sólo se le demanda al lector experiencia en el desarrollo de páginas web desde cualquiera de los papeles posibles: programación, diseño o alguna actividad relacionada. Eso bastará para conocer los conceptos básicos y ubicarse en el contexto oportuno según el caso.

Capítulo 1

Claves para la optimización de sitios

En este capítulo haremos una breve introducción a los conceptos y temas tratados a lo largo del libro y pondremos énfasis en las ventajas y los principios en los que se basa la optimización de las aplicaciones web.

Situación actual	14
Conceptos generales	15
Importancia de los estándares	17
Reglas de optimización	18
Herramientas tratadas	19
Niveles de optimización	20
Conocimientos requeridos	21
Resumen	21
Actividades	22

SITUACIÓN ACTUAL

Con el paso del tiempo las necesidades de los sitios web variaron. Más allá de la aparición de nuevas tecnologías, el perfeccionamiento de las ya existentes, e incluso de los nuevos usos dados a las herramientas que formaron parte y estuvieron presentes en los comienzos de Internet, los servicios demandados por los usuarios hicieron que las aplicaciones necesitaran dar una respuesta consistente y estandarizada. Algo que no dependiera totalmente de las buenas intenciones o del conocimiento adquirido del desarrollador.

En la mayoría de los casos, los usuarios de sitios web exigen rapidez para obtener la información en el menor tiempo posible sin importarles la estructura interna, la estabilidad o incluso la seguridad de las aplicaciones a las que acceden. Entonces, es responsabilidad del desarrollador mantener sistemas bien organizados que puedan ser modificados y ampliados cada cierto tiempo sin tener que recomenzar el trabajo de manera total. La situación ideal es que haya desarrollos técnicamente competentes que logren abastecer las demandas de los usuarios.

Hasta hace no mucho tiempo había principalmente dos puntos que atentaban contra el logro de estas premisas. El primero estaba relacionado con las herramientas disponibles para desarrollar sitios. Las alternativas eran pocas y las funcionalidades ofrecidas requerían por parte del autor de sitios una dedicación y una preparación mucho mayores a las de la actualidad. La segunda tiene que ver con las velocidades de conexión, que es algo que sin dudas restringía las posibilidades y obligaba a desechar opciones solo porque la mayoría de los usuarios no iba a poder acceder a ellas correctamente.

En la actualidad, el número de alternativas es enorme, las herramientas disponibles, en muchos casos libres y gratuitas, son de nivel profesional y nos permiten resolver de manera simple la mayoría de los desafíos propuestos. Esto sin contar las aplicaciones ya desarrolladas puestas a disposición de cualquiera para adecuarlas, modificarlas y vincularlas casi automáticamente para aumentar los servicios de nuestro sitios. Además, existe una gran comunidad de desarrolladores que destina su tiempo y esfuerzo a publicar artículos y responder las inquietudes de los demás. Sin dudas, esto ayuda a ampliar el conocimiento y completar los proyectos particulares en tiempo y forma.

III CARGA MÁS RÁPIDA

Es importante tener en cuenta que lograr el menor tamaño posible en los archivos que conforman un sitio web nos ayudará a reducir el tiempo de carga que será necesario para nuestras páginas. En los próximos capítulos observaremos cuáles son las alternativas para lograrlo y también las herramientas necesarias para llevarlo a cabo.

Figura 1. *La oferta de aplicaciones referidas al tratamiento y el control de sitios web cada vez es mayor. Muchas de ellas incluso son de uso gratuito.*

CONCEPTOS GENERALES

Optimizar sitios requiere de mucha paciencia para lograr resultados óptimos. Cada capítulo de este libro enseña cómo abordar una aplicación web y reconocer qué partes de nuestros sitios pueden lograr una mayor velocidad de respuesta. Las tecnologías que se tratan aquí son reconocidas por entes representativos que especifican reglas básicas necesarias con las cuales comparar y ajustar nuestros desarrollos. También abordaremos qué herramientas son apropiadas para validar cada uno de nuestros proyectos dependiendo de las caracteristicas de los mismos.

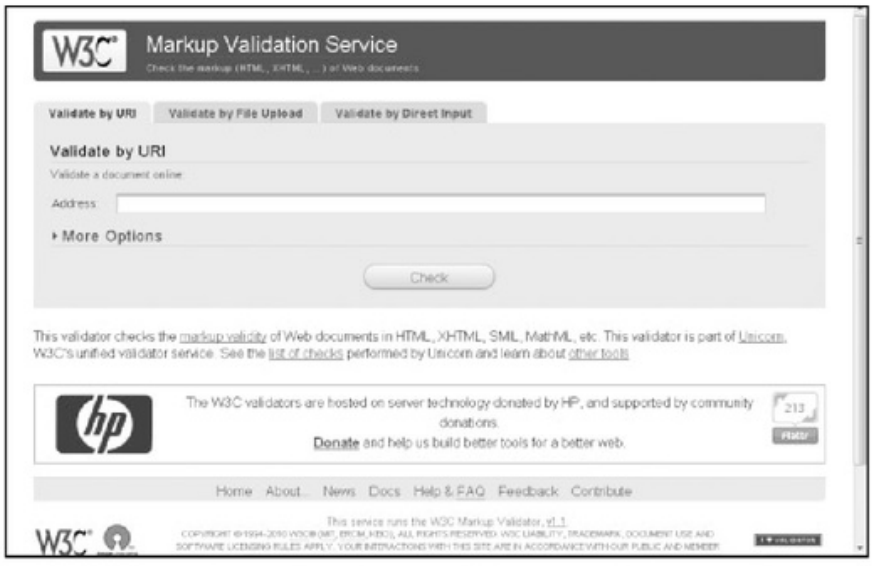


Figura 2. *Las aplicaciones para validar la correcta formación de código en relación con los estándares web son de uso libre.*

Los organismos oficiales que tienen vigencia en la actualidad están formados por importantes empresas. Ellas fueron las principales propulsoras de la creación, el mantenimiento y la ampliación de los estándares. Esto les traería como beneficio la posibilidad de competir desarrollando sus productos sobre la base de reglas claras y consensuadas. Algunos casos notorios son la **W3C** (*World Wide Web Consortium*) o la **ECMA** (*European Computer Manufacturers Association*). Éstos son ejemplos de entidades que marcan direcciones y hacen eco de las necesidades y las opiniones no solo de las principales compañías sino de los propios desarrolladores, que al fin y al cabo son los que en última instancia eligen utilizar una herramienta u otra.

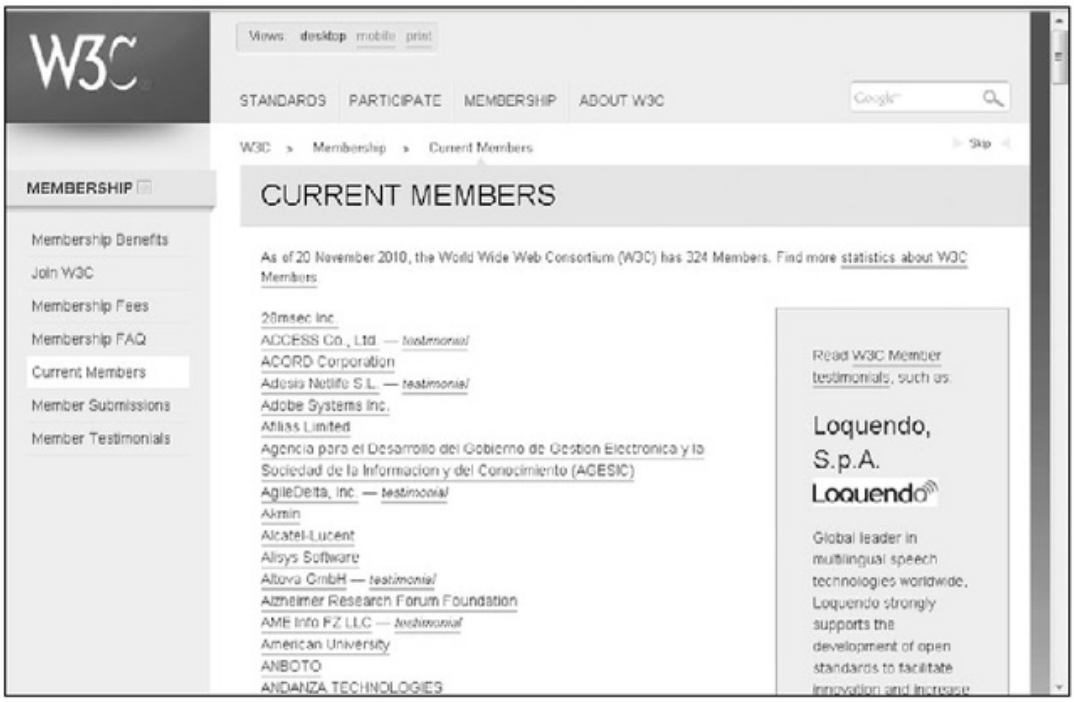


Figura 3. Apple, AT&T, Cisco, ERICSSON, HP, IBM, Microsoft, Mozilla y Yahoo! son solo algunas de las empresas que forman parte de la W3C.

Un punto interesante es que la gran mayoría de las tecnologías que componen los estándares web es de uso libre y gratuito, es decir, que están al alcance de cualquiera que posea los conocimientos necesarios para implementar soluciones. Sin dudas, esto no es casual. Las herramientas seleccionadas han sido probadas durante largos períodos (**CSS** y **JavaScript** son solo algunos ejemplos) y las nuevas responden a las necesidades surgidas en los últimos tiempos (podríamos nombrar a **Ajax**, **XML** y **JSON**), pero ninguna de ellas es propietaria. No será necesario descargar software adicional o específico, ni actualizaciones de ningún tipo. Esto deriva en la igualdad de posibilidades para todos los que quieran participar en el desarrollo de aplicaciones web. Si volvemos al inicio, ante este panorama se vuelve indispensable contar con una manera concreta de validar la calidad técnica de las aplicaciones. Las reglas impuestas por los organismos oficiales cumplen ese propósito.

IMPORTANCIA DE LOS ESTÁNDARES

¿Para qué apegarnos a los estándares? Hay más de una razón para hacerlo, pero una de las más importantes es garantizar de manera real y tangible la calidad de una aplicación. Un desarrollo puede funcionar perfectamente y cumplir con los requisitos solicitados por nuestros clientes. Sin embargo, cada día es más necesario demostrar además que internamente todas las partes componentes están basadas en tecnologías reconocidas a nivel mundial, que mantienen sus principios en documentos públicos y que tienen un alto concepto por parte de los participantes en el desarrollo de aplicaciones web. Los organismos internacionales que mantienen estas tecnologías nos brindan la posibilidad de comparar nuestro código contra aplicaciones que se encargan de procesarlo y hacernos saber si éste es válido o no. El proceso es simple y gratuito, y a lo largo de los capítulos que siguen mostraremos en detalle cuáles son las opciones que se encuentran disponibles y cómo utilizarlas en nuestros propios proyectos.

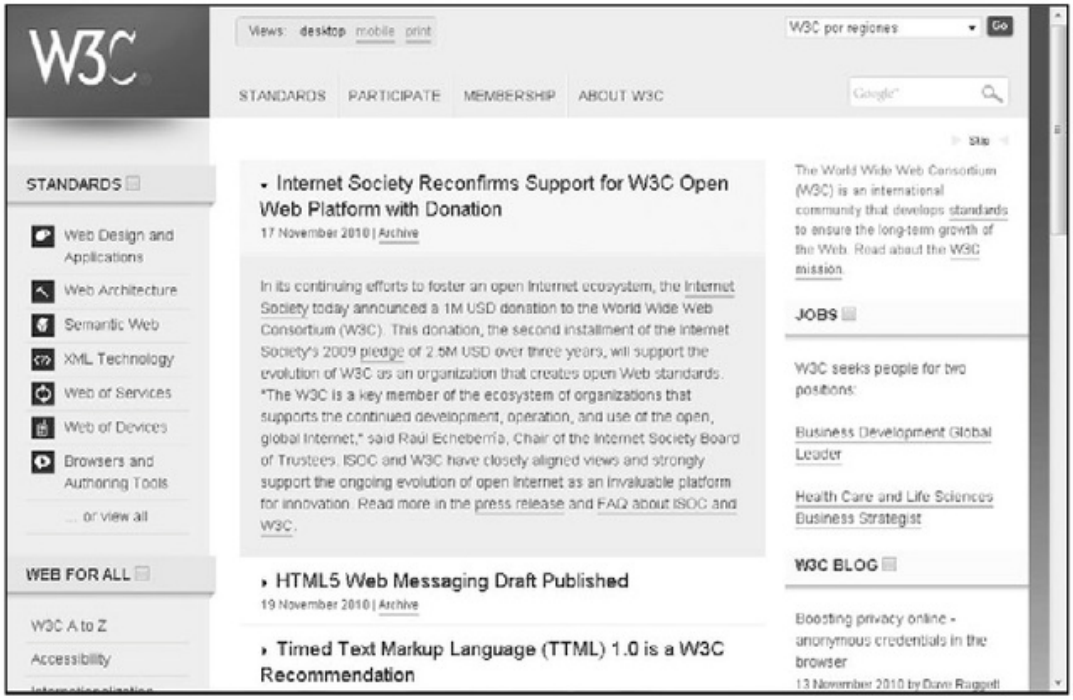


***Figura 4.** La W3C es una de las organizaciones más importantes dedicada a mantener estándares web.*

Hasta hace algún tiempo existía la creencia de que los estándares iban por un camino y las aplicaciones reales por otro, es decir que era casi imposible aplicar los conceptos definidos en la documentación oficial. Esto cambió gracias a aquellos que necesitaban contar con algunas reglas claras para desarrollar sus sitios, y actualmente es común observar cómo aplicaciones web profesionales, populares y en algunos casos complejas, le dan gran importancia al hecho de respetar los modelos pautados por los organismos oficiales.

REGLAS DE OPTIMIZACIÓN

El equipo de desarrollo de la empresa **Yahoo!** impulsó la creación de un documento que contiene un listado con las técnicas necesarias para optimizar sitios web. Estas reglas fueron tomadas como base por muchas otras compañías y se convirtieron en un punto de partida que les sirve mucho a quienes intentan desarrollar sistemas de manera equilibrada, mediante la utilización de la menor cantidad de recursos posible.

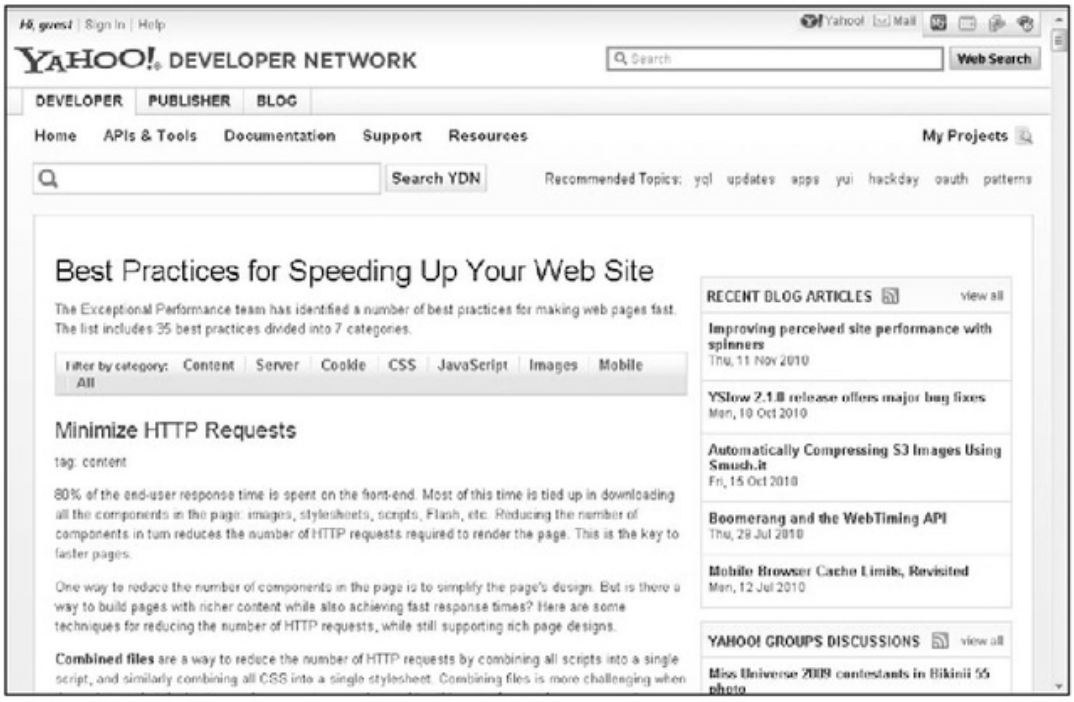


***Figura 5.** Yahoo! realiza uno de los aportes más importantes a la optimización de sitios web con sus reglas.*

Estas catorce normas se han forjado en base a la experiencia obtenida por los empleados en cuanto a la generación de sitios. Yahoo! es una de las empresas que más importancia le presta a sistematizar desarrollos web. Además, mantiene a disposición de los programadores un gran número de herramientas de propósito general. Todas son libres y gratuitas. El listado completo de pautas se puede obtener en **http://developer.yahoo.com/performance/rules.html**, y nos servirá de gran ayuda para mantener una

III VISIÓN A FUTURO

Por ser consensuadas y no pertenecer de manera directa a una empresa en particular, las tecnologías componentes de los estándares web nos dan la seguridad de que nuestros desarrollos no deberán ser modificados con cada versión nueva. Sin dudas, esto nos brinda la seguridad necesaria para implementar soluciones en nuestros proyectos.

idea general acerca de qué es lo que podemos hacer y cuáles son las alternativas para lograrlo. Además, cada regla es ampliada y complementada por los mismos usuarios, que dejan mensajes con alternativas y mejoras.

Apegarse a los estándares oficiales es un buen comienzo para empezar el recorrido en busca de la optimización de nuestras aplicaciones. En los capítulos que siguen observaremos cómo todo se mueve en conjunto y las acciones se complementan unas con otras. No se trata de una casualidad: más allá de unificar principios y plasmar consensos en documentos oficiales, los organismos responsables buscan en todo momento tratar de acelerar el intercambio de información entre clientes y servidores, sin dejar de lado la consistencia de los desarrollos. Como ejemplo inicial podemos describir la inclusión de código CSS y JavaScript en documentos XHTML. Como veremos, hay distintas maneras de hacerlo, pero las ventajas de utilizar archivos externos son claras. Por un lado, esto nos permite seguir las recomendaciones de las especificaciones oficiales y al mismo tiempo optimizar nuestros desarrollos y facilitar su mantenimiento.

Figura 6. *Tanto Yahoo! como Google y otros buscadores ponen en su página de resultados el tiempo que llevó arrojarlos. Un claro ejemplo de potencia y optimización.*

III PRESENTE Y FUTURO

En la actualidad, distintos **agentes de usuario** y no solo los tradicionales navegadores pueden acceder a las aplicaciones web. En los próximos capítulos de este libro analizaremos la forma en que las tecnologías actuales pueden ayudarnos a mantener distintas versiones de un mismo sitio, exigiendo un mínimo esfuerzo por parte del desarrollador.

HERRAMIENTAS TRATADAS

Como mencionamos anteriormente, las tecnologías analizadas y comentadas son de uso libre, sin restricciones. Por lo tanto, las herramientas que incluimos para controlar y verificar el correcto funcionamiento del código también son, en la mayoría de los casos, de uso libre. Se trata de aplicaciones intuitivas y casi todas accesibles mediante un navegador web. Esto nos permitirá mantenernos independientes del sistema operativo y la arquitectura desde la que trabajemos.

***Figura 7.** Los navegadores web actuales integran herramientas para controlar los aspectos relativos al comportamiento de las aplicaciones.*

Además, tratamos de incluir más de una alternativa en cada caso, para mantener opciones y que sea el propio lector el que seleccione la herramienta que más y mejor se adapte a sus posibilidades y necesidades.

NIVELES DE OPTIMIZACIÓN

Dentro de las muchas opciones posibles, podemos definir las alternativas relacionadas con la optimización de sitios en dos categorías: una a nivel de sistema y la otra a nivel de código. La primera está relacionada con aquellas cuestiones relativas a la configuración de los servidores que alojan las aplicaciones, que si bien en la mayoría de los casos no están al alcance de los usuarios comunes, pueden sernos de ayuda a la hora de comprender qué aspectos pueden interferir en el desempeño de nuestros

proyectos. También podrán servirnos a la hora de contratar un servicio de alojamiento y optar por alguno de los planes ofrecidos.
En cuanto a la optimización de código, sin dudas se trata de la parte desde la cual podemos sacar más rédito y ampliar las capacidades de nuestros sitios web.
La intención es que el recorrido por ambas opciones logre ayudar al lector a conseguir una visión completa que abarque todos los aspectos relacionados al diseño y a la programación de páginas web, y que la aplicación de las soluciones planteadas no requiera anular de manera completa los desarrollos sino complementarlos para enriquecer y aumentar su rendimiento.

CONOCIMIENTOS REQUERIDOS

La optimización de sitios es un tema abierto y en continua evolución, por lo que en cierta manera nunca termina. Se trata de una tarea constante que requiere una atención especial. Más allá de este aspecto, lo cierto es que mantener una idea general acerca de los puntos habituales que necesitan cuidado nos ayudará a automatizar algunos conceptos para aplicarlos de manera sistemática en todos nuestros desarrollos. Sin dudas, éste uno de los objetivos que plantea nuestro libro.

El desarrollo web en la actualidad se vale de la utilización de algunas herramientas o tecnologías muy conocidas y abiertas, con historia y con mucho futuro por delante: **XHTML**, **CSS**, **JavaScript** y **Ajax**, entre otras. Éstas son parte de una realidad ineludible para cualquiera que implemente aplicaciones web, y a lo largo de los distintos capítulos asumiremos que por lo menos los conceptos básicos y las ideas generales de funcionamiento acerca de cada una están al alcance del lector.

RESUMEN

En este capítulo observamos los principios generales acerca de la utilidad que nos brindan las reglas de optimización de sitios y el por qué del respeto a los estándares web que recomiendan los organismos internacionales. Además, planteamos los lineamientos fundamentales sobre los que se apoyarán los siguientes apartados.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué entiende usted cuando se habla de la optimización de sitios web?

2 ¿Qué entiende por estándar web?

3 ¿Cuál cree que es la importancia real de respetar los estándares?

4 ¿Qué necesidad motivó el surgimiento de los estándares web?

5 ¿Qué beneficio les traen los estándares a las empresas?

6 ¿Qué beneficio les traen los estándares a los desarrolladores?

7 ¿Qué beneficio les traen los estándares a los usuarios de las aplicaciones?

EJERCICIOS PRÁCTICOS

1 Plantee cinco fallas que nota en sus desarrollos.

2 Busque más información acerca de las reglas propuestas por Yahoo!.

3 Busque información acerca de las tecnologías mantenidas por W3C.

4 Busque información acerca de las empresas que componen la W3C.

Capítulo 2

XHTML

XHTML es una especificación XML y el reemplazo natural de HTML. En este capítulo haremos un recorrido por sus características más destacadas y veremos todo lo necesario para generar y validar nuestros propios documentos.

Origen e historia	**24**
Tipos de documento	**26**
Strict	27
Transitional	28
Frameset	29
XHTML 1.1	29
Reglas de marcado	**30**
Juegos de caracteres	**34**
Cabeceras	**35**
Validación	**36**
Limpieza con HTML Tidy	**39**
Técnicas de reducción	41
Compatibilidad entre navegadores	42
Estructuración de documentos	**44**
Uso de frames	**45**
DHTML (HTML Dinámico)	**46**
Inclusión de JavaScript	**48**
Inclusión de CSS	**49**
Elementos disponibles	**50**
Módulos en XHTML	**57**
XHTML 2.0	**58**
Resumen	**61**
Actividades	**62**

ORIGEN E HISTORIA

El nacimiento del **lenguaje de marcado HTML** (*HyperText Markup Language - Lenguaje de Marcado de Hipertexto*) fue para dar formato y estructurar la información contenida en una página. Las demandas sobre la capacidad inicial de HTML hicieron necesaria la creación y el planeamiento de una nueva versión que permitiera concretar varias necesidades, por ejemplo que la información y la forma de presentarla estén claramente separadas (evitar la utilización de elementos como **font**, o la etiqueta **style**, por ejemplo). Para esto hay que ubicar documentos HTML y hojas de estilo en distintos lados. Esto genera independencia del dispositivo desde el que se accede a la información (la presentación puede quedar a cargo de tecnologías externas como **CSS**).

XHTML forma parte de un estándar desde el 26 de enero de 2000 (la versión 1.1 fue publicada el 31 de Mayo de 2001), y está mantenido por la **W3C** (*World Wide Web Consortium*, **www.w3.org**). Centralizar los lineamientos fundamentales de una tecnología en un texto público permite que para visualizar esta clase de documentos los programas específicos lo hagan de manera similar, independientemente del sistema operativo o la arquitectura utilizados.

XHTML se considera el sucesor de HTML, y más allá de las diferencias puntuales, la idea es que sea más estricto y más ordenado (está basado en XML, por lo que cumple sus reglas), y que disminuya aún más el grado de interpretación posible por parte de los navegadores. Esto generaba inconvenientes para los diseñadores, ya que tenían que escribir código compatible, incluso para las distintas versiones de un mismo navegador. Este problema no ha sido resuelto del todo, pero con XHTML logró controlarse un poco más.

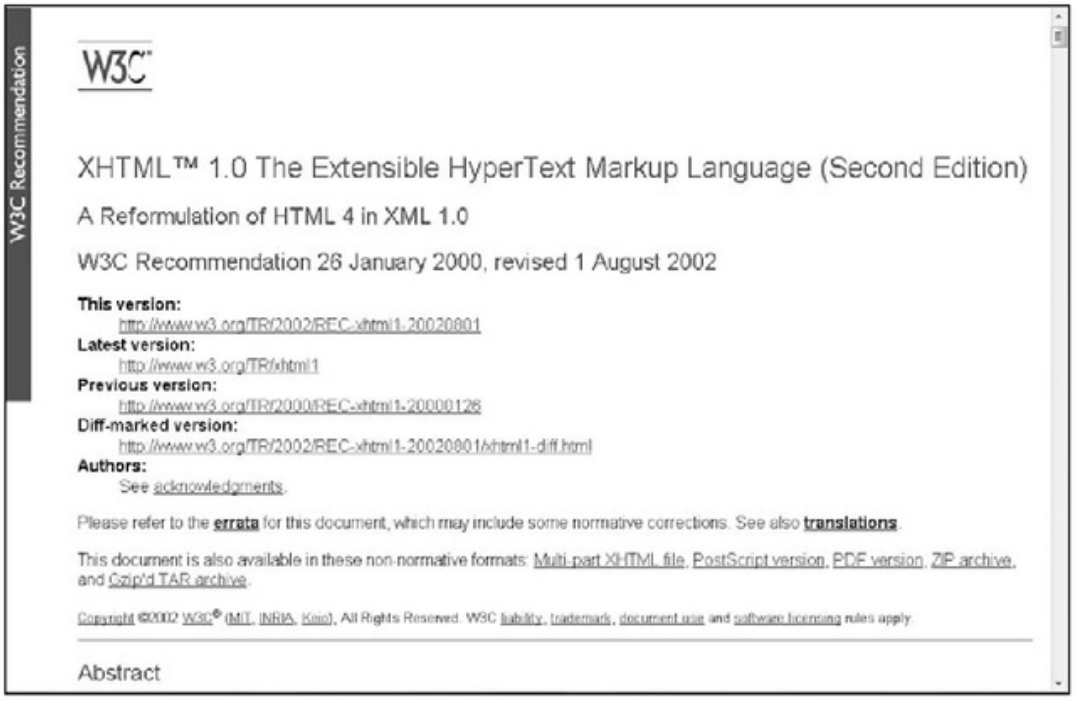


***Figura 1.** XHTML es uno de los múltiples estándares mantenidos por la organización **W3C (World Wide Web Consortium)**.*

Debemos saber que, comparado con XHTML, HTML es un lenguaje más flexible. Por ésta razón, si queremos visualizar un documento mal formado, por ejemplo sin cerrar etiquetas o mediante directivas que no forman parte del estándar o que no están soportadas por todos los programas cliente, el navegador intentará solucionar el inconveniente interpretando el código a su propia manera, según sus propias reglas, lo que se traduce en que el usuario pueda acceder a la ágina pero generando problemas para los diseñadores. Como un documento XHTML válido debe estar bien formado, no habría lugar para distintas interpretaciones, o por lo menos se reduciría el margen.

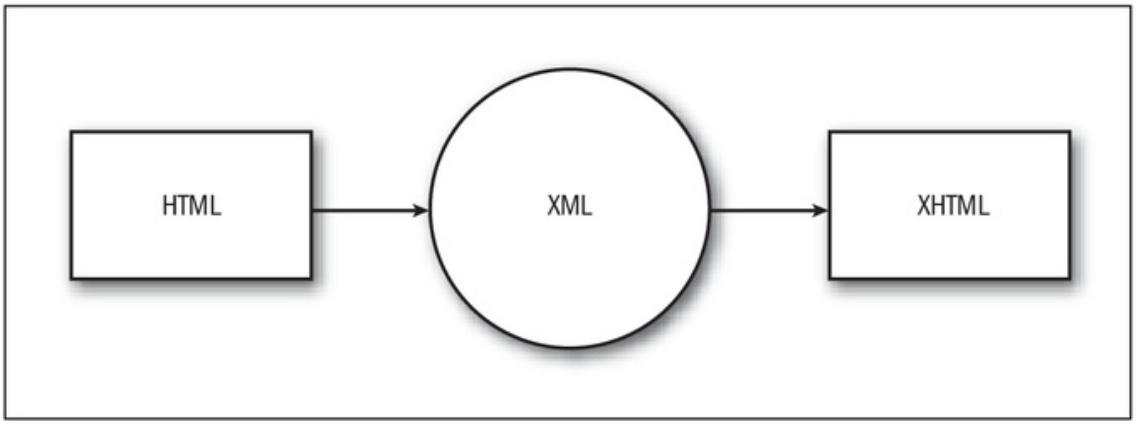


Figura 2. *XHTML es una especificación XML y por lo tanto respeta las reglas impuestas en el popular metalenguaje.*

Orientar HTML hacia XML es algo muy positivo, que permite mantener un lenguaje popular y muy aceptado como HTML, y a la vez es un primer paso hacia el soporte masivo del formato de documentos XML por parte de los navegadores web. En resumen, así surge XHTML (*eXtensible HyperText Markup Language - Lenguaje Extensible de Marcado de Hipertexto*), que no es más ni menos que la versión XML de HTML. Éste mantiene las características de su predecesor (sus elementos predeterminados y su facilidad de implementación), además de la necesidad de cumplir las especificaciones establecidas para XML. XHTML puede tomar ventaja de las especificaciones basadas en XML, como por ejemplo **MathML** y **XSL**, por solo citar dos de los más populares derivados de este metalenguaje.

III MATHML

Es importante tener en cuenta que la especificación conocida como **MathML (Mathematical Markup Language)** tiene como objetivo describir fórmulas matemáticas. Podemos encontrar más información visitando el sitio que se encuentra en la dirección **www.w3.org/Math**. Todavía no es soportado de manera natural por la mayoría de los navegadores.

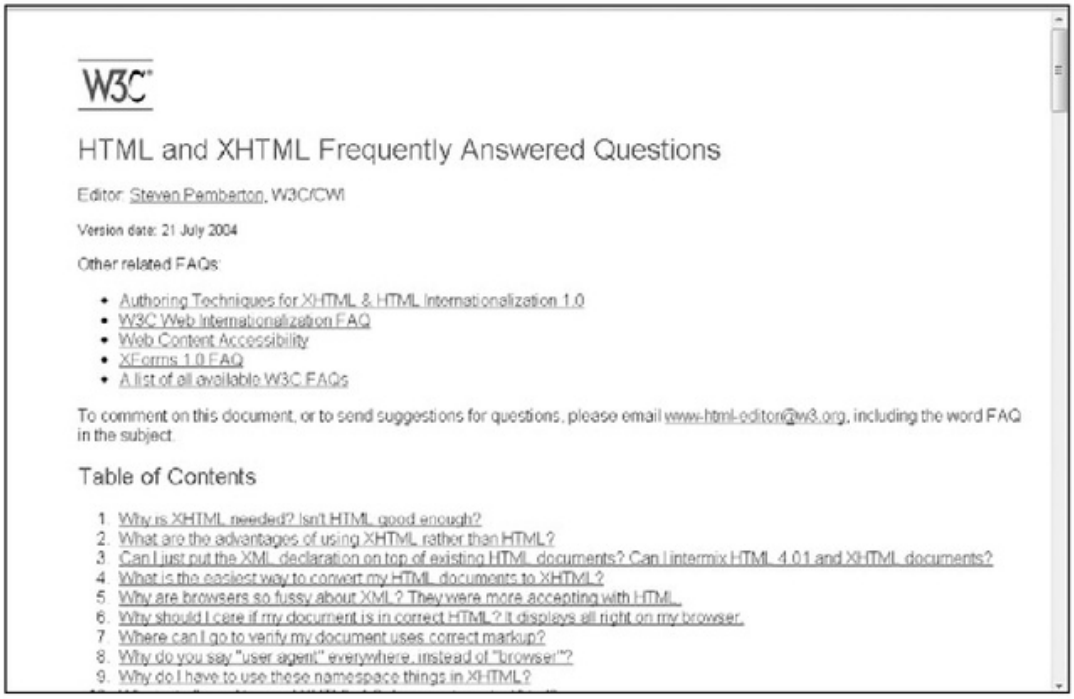


***Figura 3.** Del sitio oficial podemos obtener un listado con las preguntas más frecuentes acerca de XHTML (**www.w3c.org/MarkUp/2004/xhtml-faq**).*

El lenguaje XHTML está compuesto por los mismos elementos definidos en **HTML versión 4.01**. Podemos disponer de la especificación original en el sitio web oficial de W3C, en la dirección **www.w3.org/TR/xhtml1**.
A continuación repasaremos algunas de las opciones referidas a la estructura y la validación de documentos creados utilizando XHTML.

TIPOS DE DOCUMENTO

Si tomamos como base lo que dice el estándar oficial de la W3C, los documentos HTML deben poseer cabeceras específicas para definir la versión XHTML que se utiliza. Esto, aunque no es obligatorio, es útil para validar nuestros documentos, ya que las aplicaciones disponibles para tal fin toman la información del doctype (o sea el elemento que define el tipo de documento) para aplicar las reglas correspondientes.
Hay distintas técnicas para validar documentos XML, una de ellas es mediante un **DTD** (*Document Type Definition* - Definición del Tipo de Documento). Poseen una sintaxis estricta y un documento XML se considera como válido si respeta las reglas impuestas en un DTD asociado. Cada DTD incluye, entre otras informaciones, los elementos y los atributos permitidos y su forma de inclusión en un documento. La inclusión del tipo de documento ya existía en HTML pero era opcional y pocas veces se utilizaba. Hay tres tipos de documentos en XHTML, cada uno de los cuales corresponde a un documento DTD distinto. Cada uno sirve para indicarle al

navegador qué versión XHTML le corresponde a la página. De no especificar el tipo de documento, normalmente un navegador interpretará el documento como HTML, aun cuando en la práctica sea XHTML. A continuación veamos los tipos de documentos disponibles para XHTML 1.0.

Strict

Esta definición se utiliza cuando en el documento XHTML no se incluye información relativa a la presentación de los datos, es decir, que está almacenada en hojas de estilo externas, por ejemplo CSS. Debemos saber que la cabecera para definir cualquier tipo de documento strict es la siguiente:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Como se puede intuir a partir de su nombre, este nivel de validación es preciso y no permite la inclusión de elementos en desuso, es decir, aquellos que no formarán parte de las próximas versiones del lenguaje XHTML, como **applet**, **center**, **dir**, **menu**, **basefont**, **font**, **isindex**, **s**, **strike**, y **u**, entre otros, ni de ciertos atributos de determinados elementos. Entre las restricciones más importantes podemos citar:

- Para el elemento **table**, el atributo **height** ya no es soportado.
- Los elementos denominados **table**, **tr** y **td** no soportan el atributo **background**, el cual se usa para ubicar una imagen como fondo.
- Para el elemento **img** el atributo **alt** es obligatorio.
- El elemento **a** no puede contener otro elemento **a**.
- El elemento llamado **pre** no puede contener ninguno de los elementos que se mencionan a continuación: **img**, **object**, **big**, **small**, **sub** o **sup**.
- El elemento **button** no puede contener ninguno de los elementos **input**, **select**, **textarea**, **label**, **button**, **form**, **fieldset**, **iframe** o **isindex**.

III XSL

El estándar conocido como **XSL** (**eXtensible Stylesheet Language**) está compuesto por tecnologías que permiten recuperar determinados contenidos de un documento XML y modificar su estructura, para así conseguir distintos tipos de salidas para un mismo origen de datos. Su sitio oficial en Internet es **www.w3.org/Style/XSL**.

- El elemento **label** no puede contener otro elemento **label**.
- El elemento **form** no puede contener otro elemento **form**.
- No está permitida la utilización del elemento **frame**.

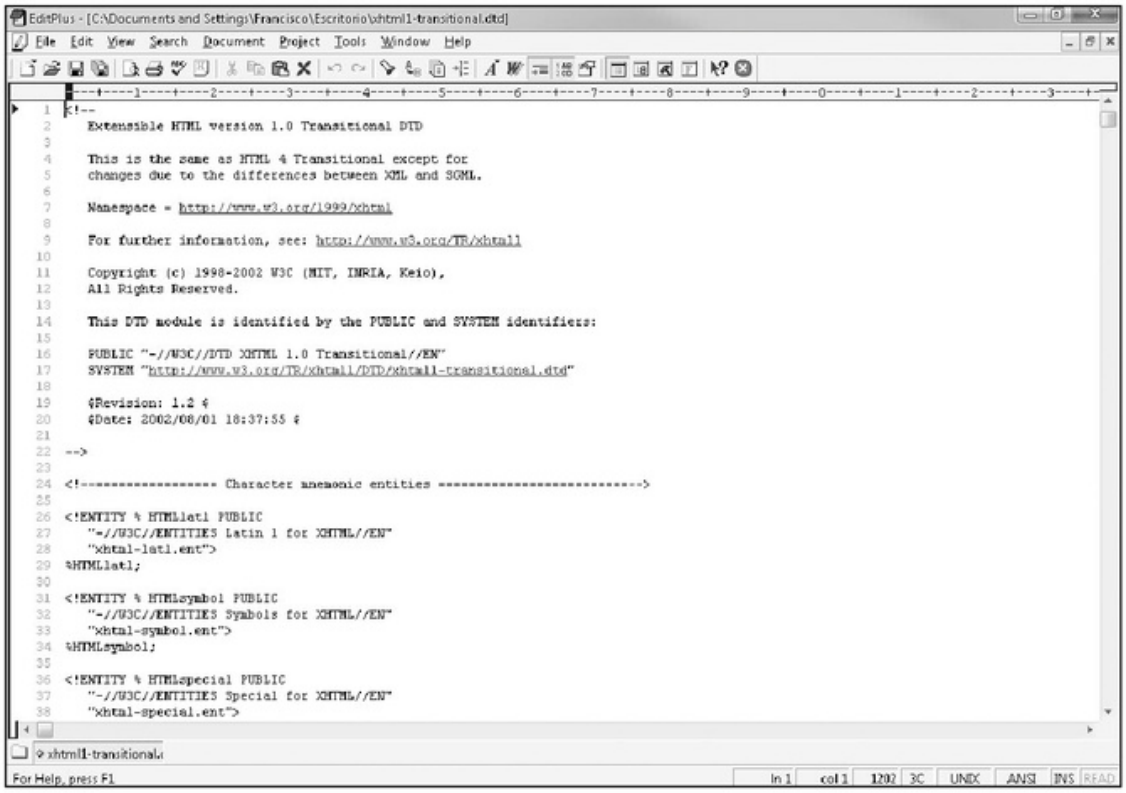


***Figura 4.** Los **DTD** para validar documentos XHTML son de libre acceso y se pueden descargar del sitio oficial de la W3C.*

Transitional

Se conoce a **Transitional** como el tipo de documento que se utiliza por defecto, y se considera una transición hacia el nivel **Strict**. La diferencia principal en éste radica en la posibilidad de incluir elementos y atributos que no se incluirán en próximas versiones de tecnologías a utilizar para el diseño de sitios web.

```
<!DOCTYPE html
PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
```

Además, es necesario saber que podemos incluir aspectos relacionados con la presentación mediante el uso de etiquetas o estilos en línea, sin la utilización de CSS o XSLT. Permite manejar elementos de XHTML y HTML 4.01, y se recomienda su implementación y uso cuando nos enfrentamos con un navegador cliente que no soporta correctamente los estilos CSS.

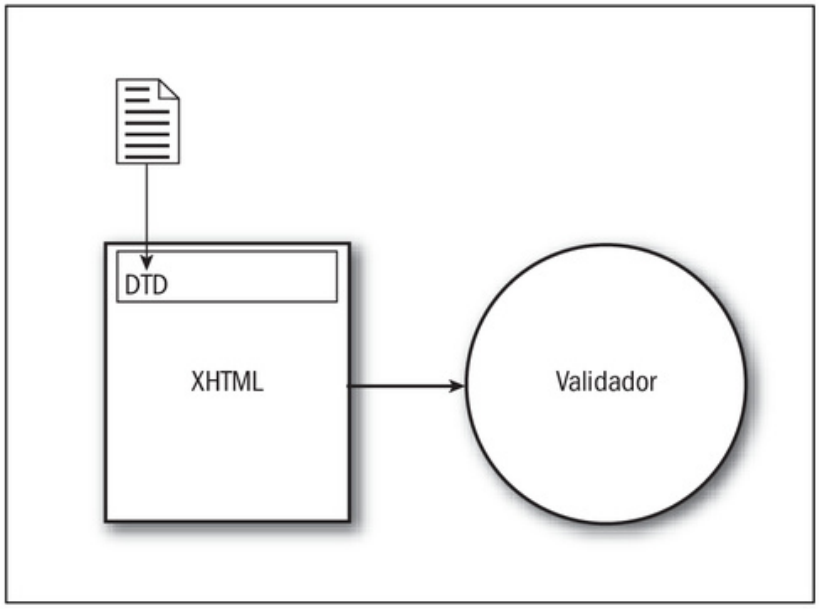


Figura 5. *Para validar un documento XHTML es necesario definir previamente un documento DTD asociado.*

Frameset

Es un nivel similar a **Transitional** con la diferencia de que además permite el uso de marcos (elemento **frame**) dentro de la estructura del documento HTML. La forma de definición la vemos en el siguiente código:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

También debemos tener en cuenta que todos los componentes de la declaración de los tipos de documento (**Strict**, **Transitional** y **Frameset**) son del tipo **Case Sensitive**, o sensibles a mayúsculas y minúsculas.

XHTML 1.1

Para la versión 1.1 de XHTML, el Doctype utilizado es:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Podemos darnos cuenta de que no se define el tipo de documento ya que se considera como una opción equivalente a **Strict**, vista anteriormente.

REGLAS DE MARCADO

A diferencia de HTML, XHTML es un lenguaje estricto como todas las especificaciones derivadas de XML. Algunas de sus características más importantes son:

- El atributo **xmlns** del elemento **html** debe estar presente. Hace referencia al espacio de nombres utilizado por XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

- El elemento **html** también dispone del atributo opcional **lang**, que en caso de ser utilizado debe estar acompañado por el atributo denominado **lang** el cual pertenece al espacio de nombres **xml**:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
```

- Los nombres de los elementos y atributos deben escribirse en minúsculas (solo para los nombres, sin incluir los valores).

```
<table><tr><td>TEXTO</td></tr></table>
```

- Es necesario tener en cuenta que los valores de los atributos deben encerrarse entre comillas, que pueden ser simples o dobles.

```
<input type="text" id="texto" value="valor" />
```

- Todos los elementos, incluyendo aquellos que se encuentran vacíos, deben contar con su correspondiente etiqueta de cierre.

```
<img src="imagen.png" width="500" height="500" border="0" alt="texto" />
```

Deben incluir un espacio antes de **/>**, por motivos de compatibilidad que no tienen que ver con la especificación, o también:

```
<img src="imagen.png" width="500" height="500" border="0" alt="texto"></img>
```

Cabe destacar que no todos los elementos pueden estar vacíos, aun cuando no registren ningún contenido, por ejemplo:

```
<!– incorrecto –>
<p />
<!– correcto –>
<p></p>
<!– correcto –>
<br />
```

- Los elementos se deben anidar en forma correcta. Para esto es necesario cerrarlos en el orden inverso al que fueron abiertos.

```
<center><p><strong> texto </strong></p></center>
```

- En XHTML hay ciertos elementos que deben estar presentes obligatoriamente: **html**, **head**, **title** (siempre dentro de **head**) y **body**.

- Al igual que sucede en XML, cada documento XHTML debe contener un elemento raíz, que en este caso será el elemento **html**. Un elemento raíz engloba a todos los demás, y es el primero que se abre y el último que se cierra.

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<!DOCTYPE html
PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
 <head>
  <title> título del documento </title>
```

III TIPOS DE ACCESO

Debemos saber que una de las motivaciones más importantes, que se encargó de generar la creación e implementación de XHTML fue la necesidad de poder visualizar el contenido de Internet desde dispositivos diferentes a los navegadores web tradicionales, entre ellos encontramos **PDAs** y teléfonos celulares, por citar solo algunos ejemplos.

```
 </head>
 <body>
 ...
 </body>
</html>
```

- Reemplazar el atributo **name** por el atributo **id** (o utilizar ambos).

```
<div id="div1" name="div1"></div>
```

- Debemos evitar la minimización de atributos y proceder a reemplazarlos por la inclusión de pares compuestos por atributo / valor:

```
<option selected>Opcion 1</option>
```

```
<option selected="selected">Opcion 1</option>
```

- Reemplazar caracteres especiales por entidades. Si el valor de un atributo contiene el carácter **&**, deberá reemplazarse con la entidad equivalente **&** y si el valor de un atributo contiene el carácter **<** debe reemplazarse con la entidad equivalente **<**.

```
<a href="pagina.php?id=10&amp;mode=a">Texto enlace</a>
```

- Un punto relativo a la compatibilidad entre navegadores es el de no incluir saltos de línea para definir un mismo elemento, por ejemplo:

```
<!- incorrecto ->
<img src="imagen.png"
     width="500"
     height="500"
     border="0"
     alt="texto" />

<!- correcto ->
<img src="imagen.png" width="500" height="500" border="0" alt="texto" />
```

- Debemos tener en cuenta que los elementos de línea deben insertarse dentro de elementos de bloque (para DTD estrictos):

```
<!– incorrecto –>
<em><h1>texto ...</h1></em>

<!– correcto –>
<h1><em>texto ...</em></h1>
```

- El texto no se debe insertar directamente en el cuerpo del documento, es decir, dentro de la etiqueta **body** (para DTD estrictos):

```
<!– incorrecto –>
<body>texto ...</body>

<!– correcto –>
<body><span>texto ...</span></body>
```

- En cuanto a los enlaces internos, se podrán referenciar partes de una página mediante el uso del atributo denominado **id**:

```
<a href="#pie">Texto enlace</a>
<p id="pie">Pie de página</p>
```

Podemos obtener un listado completo con todas las reglas necesarias para elaborar documentos XHTML válidos en el sitio web de la W3C, **www.w3.org/TR/xhtml1/#guidelines**. Siempre es conveniente tener esta documentación para repasarla antes de iniciar el desarrollo de un sitio o una aplicación web. Nos permitirá ahorrar mucho tiempo en la búsqueda de ésta por Internet.

III DERIVACIONES

XHTML es una especificación XML. Esto tiene muchas consecuencias positivas necesarias para los tiempos que corren y los que vendrán. Una de ellas es sin dudas la intención de estructurar la información y realizar la separación del contenido y diseño, dejándolo en manos de algunas tecnologías externas como CSS, por ejemplo.

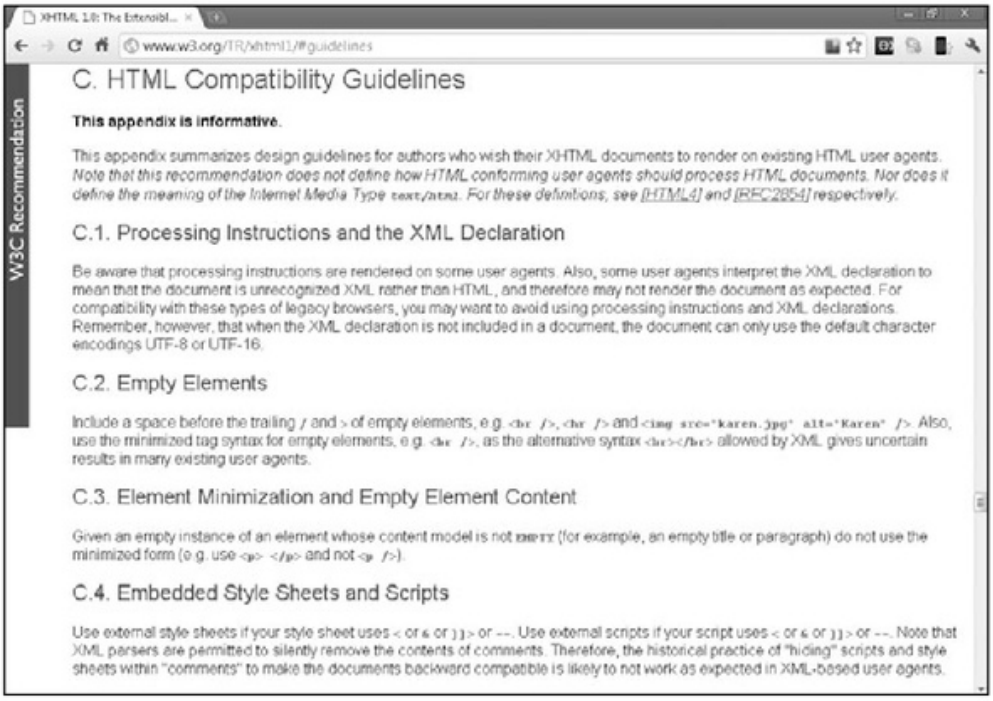


Figura 6. *La migración de documentos HTML a XHTML es algo natural que puede llevarse a cabo si se cumplen reglas claras y estrictas.*

JUEGOS DE CARACTERES

Existen varias alternativas para definir el juego de caracteres utilizado en un documento XHTML. La primera consiste en incluir como primera línea la declaración de la versión XML utilizada más el atributo **encoding**:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

XHTML admite la inclusión de esta línea. La W3C recomienda su uso, pero no es obligatorio cuando no se incluye el atributo **encoding**. En caso de no incluirla se asume que se utiliza la versión XML 1.0 con el juego de caracteres **UTF-8**.

III COMPATIBILIDAD POR PARTE DE LOS NAVEGADORES

Hay que destacar el hecho de que ciertas versiones de algunos navegadores web, generalmente las más antiguas, no ofrecen soporte XHTML y por lo tanto lo interpretan como HTML tradicional. Por el contrario, lo más común es que las versiones más actuales de los navegadores web no presenten dificutades para admitir XHTML.

Figura 7. *El juego de caracteres definido para una página se puede establecer de distintas maneras y es un aspecto muy importante que influye en cómo se visualizará el contenido del sitio.*

Una alternativa interesate es incluir la codificación mediante la cabecera meta **http-equiv**, que se ubica dentro del elemento **head**:

```
<meta http-equiv="Content-type" content='text/html; charset="iso-8859-1"' />
```

En el código que vemos en el apartado anterior podmeos darnos cuenta de que el valor de **charset** determina el juego utilizado en el documento XHTML.

CABECERAS

Según la especificación oficial, un documento XHTML básico está compuesto por las secciones o partes que listamos a continuación:

- Texto básico (incluidos los párrafos, las cabeceras y las listas).
- Hipervínculos y vínculos a documentos relacionados.
- Formularios básicos.
- Tablas básicas.
- Imágenes.
- Metainformación.

Dentro de la sección **head** de un documento XHTML se pueden incluir múltiples informaciones, entre las cuales podemos mencionar aquellas referidas a la propia página (**metadatos**). Además de las referencias a las hojas de estilo externas, a los documentos **JavaScript**, etcétera, existe el elemento **title**, dentro del cual se incluye el título de la página. Este texto puede ser diferente para cada página y la debe describir, ya que luego podrá ser tomado e incluido por los buscadores en los resultados. Solo se admite la utilización de solo un elemento **title** por página. En el atributo **profile** la etiqueta **head** permite definir la URL de un documento externo que contiene el perfil de los metadatos de la cabecera del documento. Por su parte, los elementos **meta** permiten incluir información descriptiva.

VALIDACIÓN

Existen diversas alternativas para validar archivos XML en general y XHTML en particular, según las reglas impuestas en los documentos DTD vistos anteriormente (los DTD en general varían según la versión XHTML a utilizar).

- xhtml1-strict.dtd
- xhtml1-frameset.dtd
- xhtml11.dtd
- xhtml1-transitional.dtd

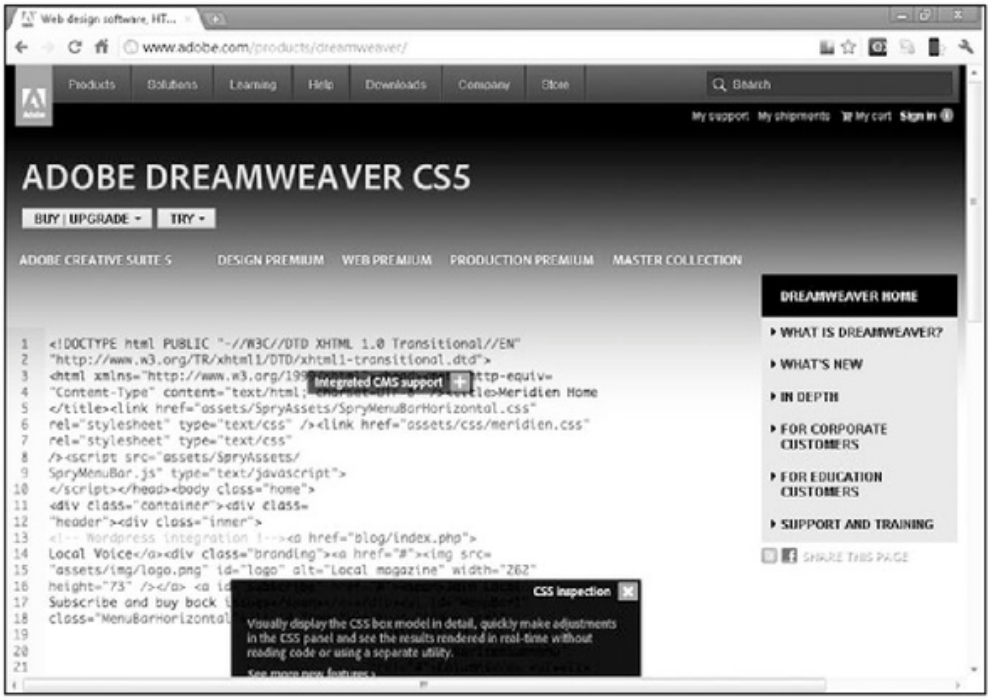


***Figura 8. Adobe Dreamweaver** incluye la posibilidad de validar documentos XHTML contra DTD. Debemos acceder al menú **File/Check Page/Validate Markup** o presionar **SHIFT + F6**.*

Para hacerlo necesitamos contar con programas específicos denominados **parsers**, que son analizadores sintácticos, o simplemente validadores XHTML. En general, podemos utilizarlos desde editores, que nos brindan la posibilidad de validar documentos offline, o desde servicios web, que nos permiten realizar la validación de los documentos de forma online, o a partir de librerías.
La W3C incluye un servicio que permite validar documentos XHTML online, y podemos visitarlo en **http://validator.w3.org/**. Otra opción disponible es **http://htmlhelp.com/tools/validator/**.

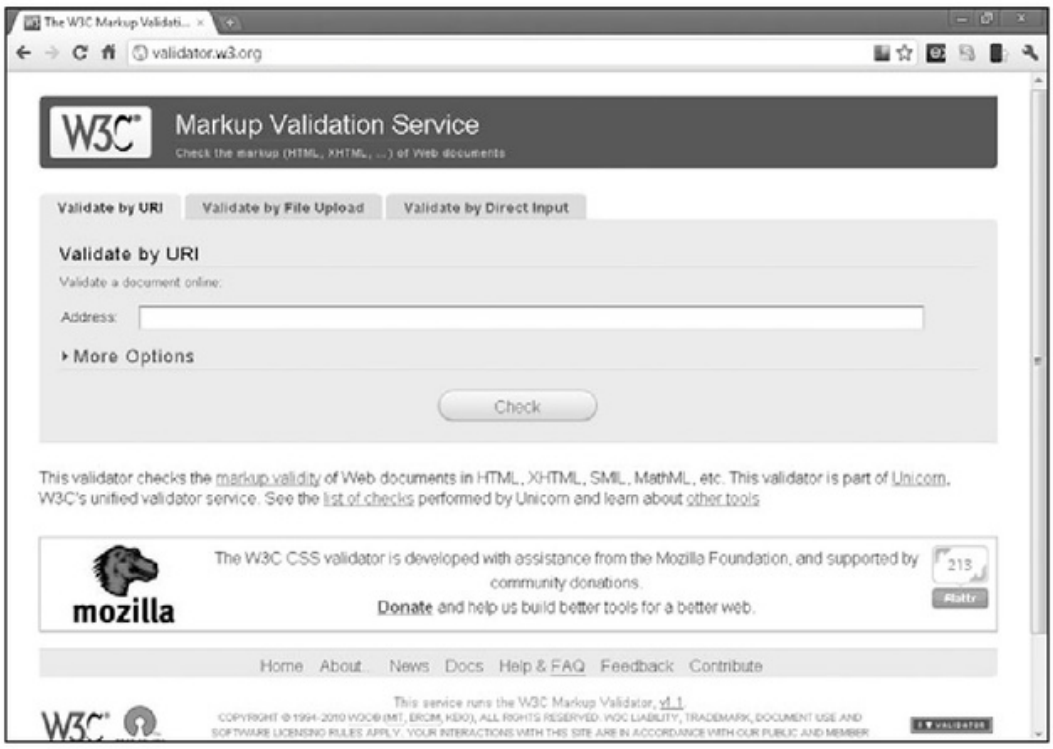


***Figura 9.** La validación de un documento XHTML puede realizarse de varias maneras. Una de ellas es mediante el servicio provisto por la W3C.*

Fuera del lado oficial, existe **XHTML-CSS**, que es una herramienta online que permite validar documentos XHTML y CSS al mismo tiempo, sin la necesidad de utilizar aplicaciones por separado para evaluar estas tecnologías utilizadas. Podemos acceder a su funcionalidad en **http://xhtml-css.com/**.

III ACTUALIZACIÓN HACIA XHTML

Es importante destacar que hay cierto consenso en aseverar que los sitios web no migran de manera masiva de HTML a XHTML porque los navegadores actuales que son compatibles con XHTML también leen HTML correctamente. También hay que decir que el cambio provoca múltiples beneficios y se produce de manera lenta pero constante.

Figura 10. *La herramienta* ***XHTML-CSS*** *nos permite validar tanto documentos XHTML como hojas de estilo CSS asociadas.*

El navegador web **Firefox** incorpora extensiones que pueden descargarse de manera gratuita, las cuales nos permiten validar aspectos relacionados a XHTML y CSS. A continuación listamos algunos ejemplos:

- **HTML Validator** (extensión basada en **Tidy** y **OpenSP**, **SGML Parser**) **https://addons.mozilla.org/en-US/firefox/addon/249**.
- **Offline Page Validation** (permite enviar páginas a los servicios de la W3C). **https://addons.mozilla.org/en-US/firefox/addon/2520**.
- **CSS Validator** (validación CSS mediante la W3C). **https://addons.mozilla.org/en-US/firefox/addon/2289**.
- **Webdeveloper Toolbar** (múltiples tipos de validación, entre las que podemos encontrar CSS, **RSS**, HTML, enlaces, etcétera). **https://addons.mozilla.org/en-US/firefox/addon/60**.

III INTERACCIÓN ENTRE ESPECIFICACIONES XML

Una de las características que nos ofrece XML es que se trata de un lenguaje extensible: las distintas especificaciones pueden utilizarse en conjunto y se logran resultados sorprendentes. XHTML no es la excepción y mediante el uso de espacios de nombres es posible incorporar alternativas como **MathML** y **Scalable Vector Graphics**.

- **Total Validator** (validaciones de manera sencilla) **http://totalvalidator.com/tool/extension.html**.
- **Checky** (valida HTML, XHTML, CSS, **RDF**, **RSS**, **XML**, **hipervínculos**, y **metadatos**, entre otras alternativas) **https://addons.mozilla.org/en-US/firefox/addon/165**.

Figura 11. *El navegador* ***Mozilla Firefox*** *incluye un gran número de extensiones para distintos propósitos. Uno de ellos es la validación de documentos XHTML.*

Podemos obtener más información acerca de Firefox en el sitio web **http://es-es.www.mozilla.com/es-ES/firefox/about**.

LIMPIEZA CON HTML TIDY

Actualmente, hay múltiples herramientas para revisar código XHTML y hacerlo compatible con las directivas detalladas en los estándares. Una de ellas es **HTML TIDY (http://tidy.sourceforge.net)**, desarrollada por Dave Raggett (**www.w3.org/People/Raggett/tidy**). Es una aplicación de distribución libre y código fuente abierto, disponible para ser utilizada en diferentes modos: online, mediante la línea de comandos del sistema operativo o como librería para incluirla en aplicaciones.

```
C:\> tidy opciones nombreArchivo(s)
```

HTML TIDY recibe como argumento un código XHTML y devuelve un listado con las incompatibilidades o los errores encontrados, más una versión actualizada del documento original, siempre y cuando las correcciones sean posibles. Entre las reformas manejadas se encuentran la inclusión de elementos de cierre, el anidamiento de elementos, el tratamiento de listas, el agregado de comillas a los atributos, el reporte de elementos o atributos no disponibles en la especificación, el reemplazo de ciertos caracteres por entidades HTML según corresponda, el reemplazo de etiquetas (**font** y **center**, por ejemplo) por estilos CSS y la eliminación de elementos vacíos (** **, por ejemplo), entre muchas otras características más. Todas son útiles para nuestro trabajo de optimización.

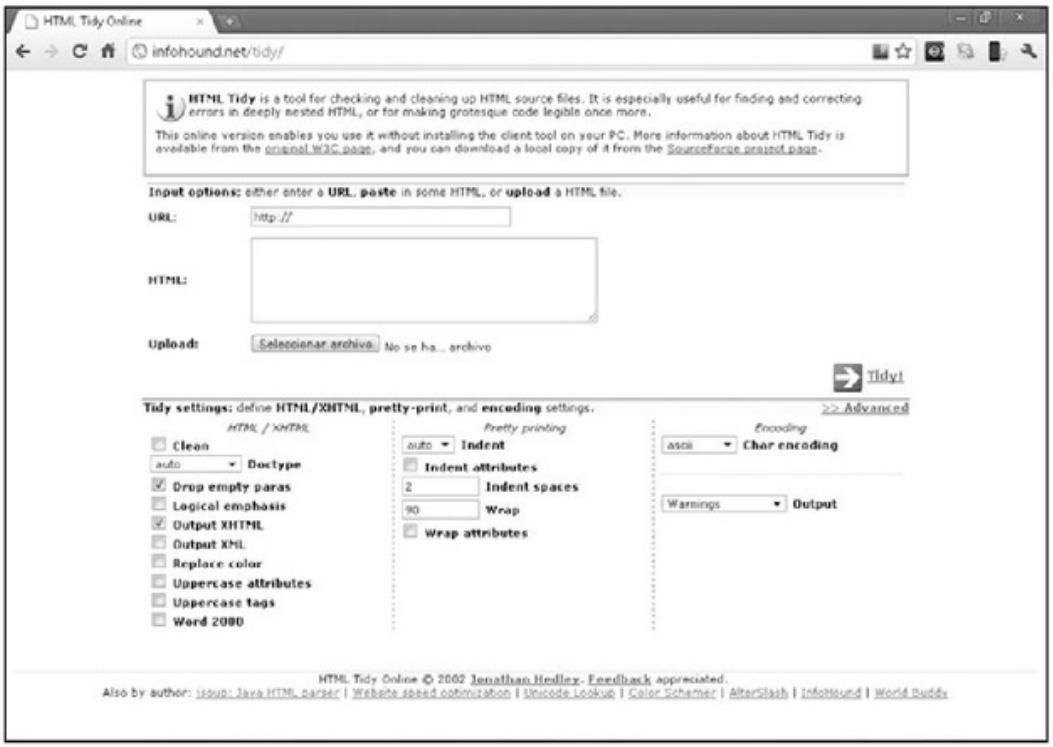


***Figura 12. HTML TIDY** es una herramienta popular para pulir documentos XHTML en particular y de marcado en general.*

Es importante saber que existen múltiples versiones online de HTML TIDY, a continuación destacamos algunas de ellas:

III DIFERENCIAS

Podemos obtener un listado con las diferencias más notables entre HTML y la especificación XHTML si visitamos el sitio oficial de la W3C, donde se incorpora un recorrido detallado por los cambios introducidos, desde los más notorios hasta los detalles mínimos, lo encontramos en la siguiente dirección: **www.w3.org/TR/xhtml1/#diffs**.

- **www.w3.org/services/tidy** (en el sitio de la W3C).
- **http://valet.htmlhelp.com/tidy**.
- **http://infohound.net/tidy**.

Se trata de una herramienta con un gran número de opciones de configuración para personalizar el tratamiento de documentos, como por ejemplo la indentación de líneas, el juego de caracteres utilizado, la versión XHTML, etcétera.

***Figura 13. HTML TIDY** ofrece múltiples opciones para procesar documentos y lograr los resultados esperados.*

Podemos obtener más información acerca de HTML TIDY en su sitio web oficial: **http://tidy.sourceforge.net/docs/quickref.html**.
Otra opción para migrar documentos HTML a XHTML es mediante el navegador web **Amaya**, que permite guardar documentos en formato XHTML. Al respecto podemos obtener más información en el sitio web **www.w3.org/Amaya**.

Técnicas de reducción

Entre las acciones que reducen el tamaño de los archivos XHTML se encuentran:

- incorporar los estilos desde archivos externos.
- incorporar las rutinas realizadas en JavaScript desde archivos externos.
- eliminar código innecesario dentro del documento.
- eliminar comentarios.

Compatibilidad entre navegadores

Mas allá de las reglas detalladas en la sección anterior, un documento XHTML normalmente se puede visualizar sin problemas en navegadores de Internet que no sean actuales (XHTML está compuesto por los mismos elementos definidos en HTML versión 4.01). Los navegadores compatibles con XML perciben los documentos XHTML como cualquier otro archivo XML sin necesidad de plugins o intérpretes de por medio. Podemos obtener un listado completo de estos en el sitio **www.w3schools.com/xml/xml_browsers.asp**.

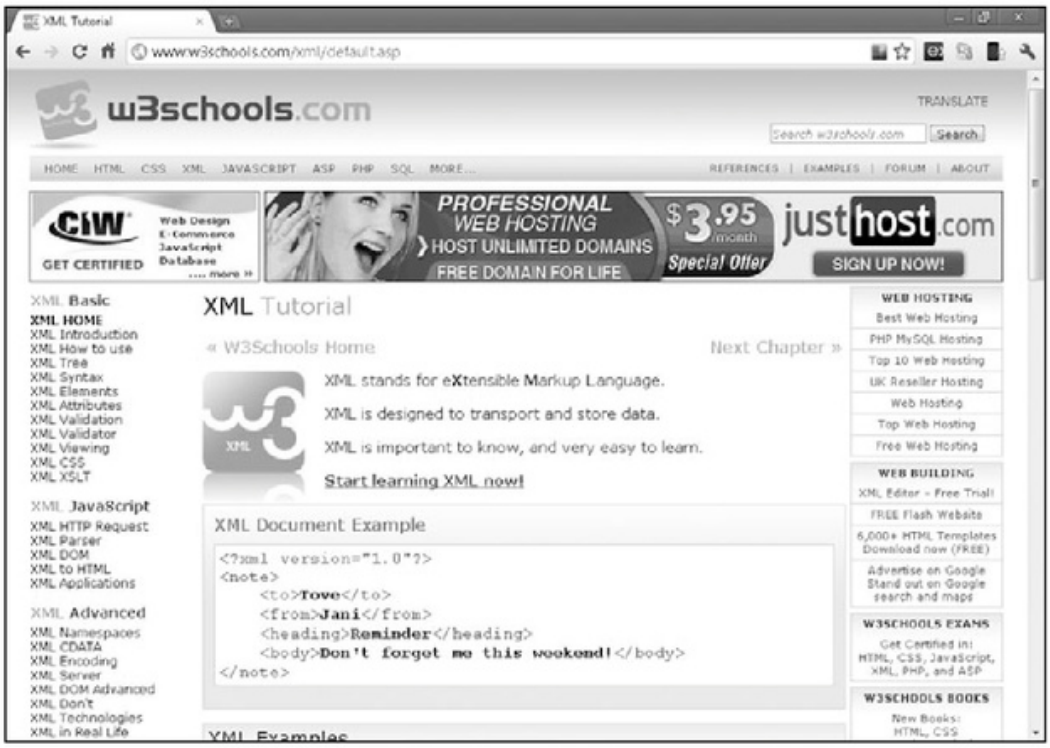


Figura 14. *Cada vez son más los navegadores que les dan soporte directo a documentos en formato XML.*

Cabe destacar que antes de la llegada de XHTML, los distintos navegadores web incorporaron elementos propietarios al lenguaje HTML, es decir, aquellos que no forman parte del estándar oficial. En este sentido podemos mencionar varios ejemplos, entre ellos se encuentran los siguientes **layer**, **marquee** y **blink**, aunque hay más. Lo

III CAMBIOS ENTRE HTML Y XHTML

Debemos saber que las modificaciones que han sido introducidas en **XHTML 1.x** no son mayores, ya que una de las intenciones iniciales era que HTML fuera compatible con XML. Podemos daros cuenta que las reglas que validan un documento XHTML así lo demuestran, ya que todas son similares a las impuestas por el lenguaje de marcado XML a sus documentos.

mismo sucede con los atributos: algunos de ellos se encuentran en desuso o no forman parte de la especificación (entre ellos se encuentran **bordercolor**, **marginwidth**, **marginheight**, **topmargin** o **leftmargin**, por ejemplo).

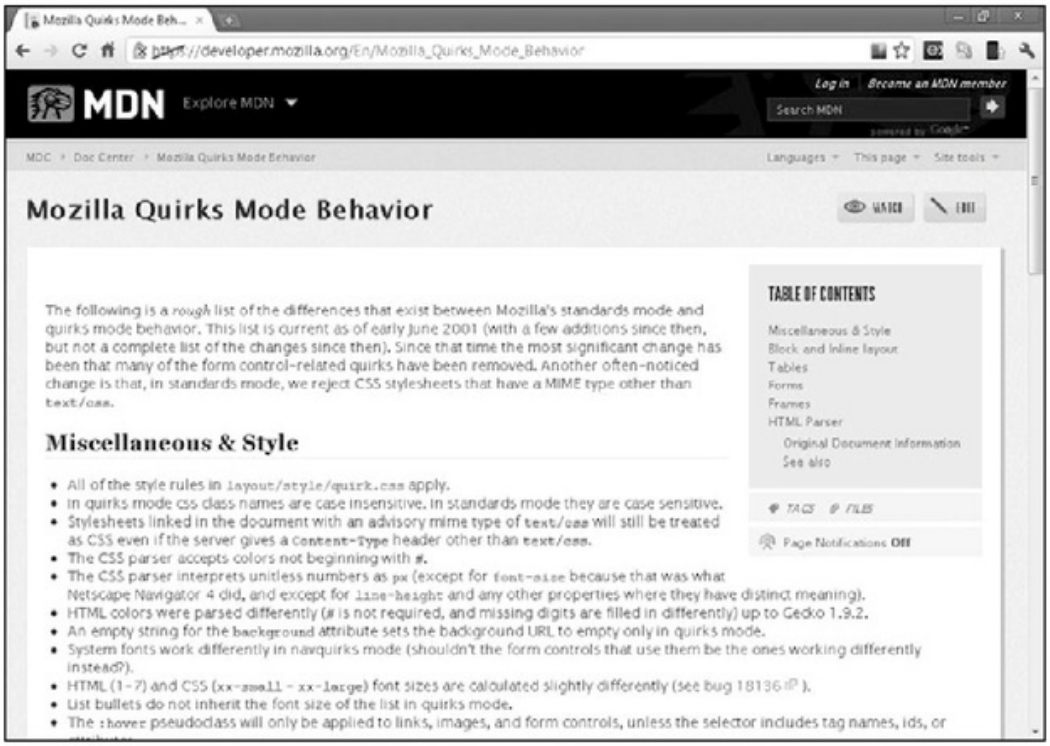


Figura 15. *El navegador* ***Mozilla Firefox*** *es uno de los tantos que mantiene distintos modos según el grado de validez de una página.*

Ahora bien, la tarea de un navegador es mostrar de la forma más fiel posible un sitio web, sin importar si éste cumple los estándares o no. Las versiones recientes de los navegadores les dan cada vez más importancia a los estándares mantenidos por la W3C, para conservar un alto grado de compatibilidad para toda clase de documentos, es decir, los válidos y los no válidos. Por eso, algunas de las empresas más importantes del sector (las primeras fueron **Microsoft** y **Mozilla**) decidieron solucionar el tema mediante la incorporación de dos modos de visualización: el modo compatible (**quirk mode**, para aquellos documentos que no respetan de forma absoluta los estándares) y el modo estándar (**standard mode**, para aquellos documentos válidos, apegados a la especificación). La decisión se toma de acuerdo con el doctype especificado: para un tipo de documento en particular se pasa al modo estándar y para los demás al modo compatible (que se utiliza en caso de no haber declaración alguna). Podemos obtener más detalles acerca de cómo cada navegador opta por un modo u otro en los siguientes sitios:

- **https://developer.mozilla.org/en/Mozilla's_Quirks_Mode**.
- **http://msdn.microsoft.com/en-us/magazine/cc163763.aspx**.
- **www.opera.com/docs/specs/doctype**.

ESTRUCTURACIÓN DE DOCUMENTOS

Es normal que en las distintas páginas de un sitio se repitan secciones como menús, encabezados, pies, etcétera, y lo ideal en estos casos es que el código que se incluye en más de una página se almacene en un archivo externo, es decir, el mismo criterio que suele utilizarse para las hojas de estilo CSS o los documentos JavaScript.

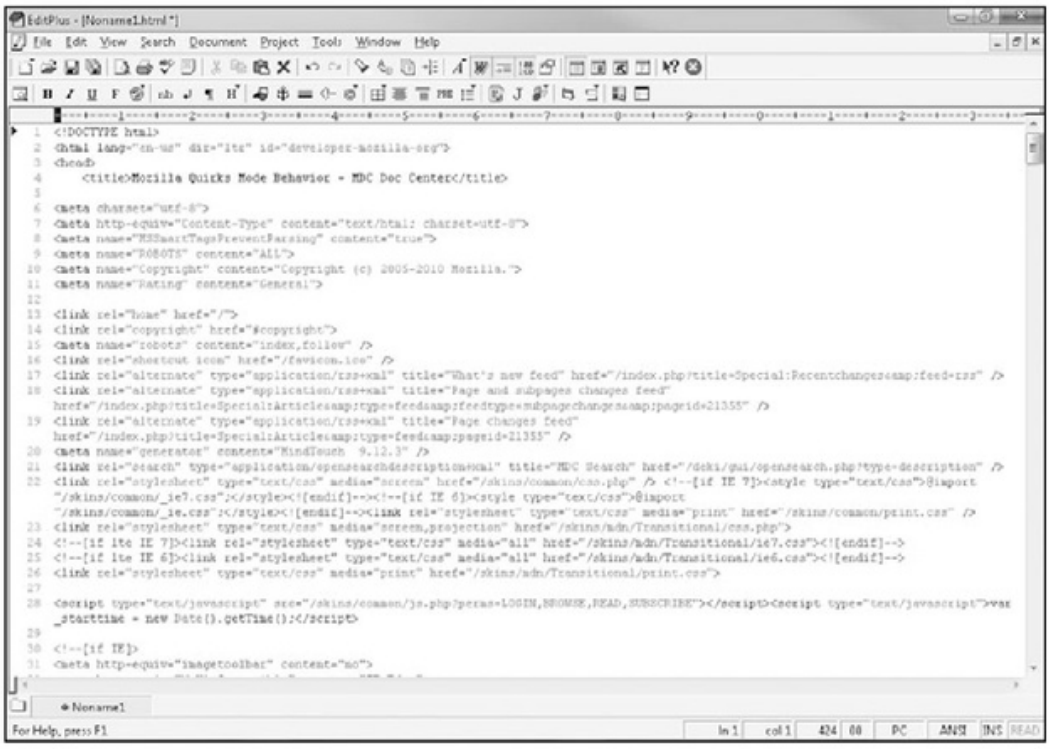


***Figura 16.** La inclusión de documentos externos es una técnica muy utilizada que permite modular sitios y volver a utilizar un mismo código en varias páginas.*

Existen varias alternativas que nos permitirán llevar a cabo la inclusión: por ejemplo utilizar **templates** y/o inclusiones mediante algún lenguaje del lado servidor como **PHP**, **ASP.net**, **JSP**, **PERL**, etcétera, o también mediante el uso de **SSI** (**Server Side Includes**), el cual no precisa ningún lenguaje en particular y trabaja mediante un sistema similar. Debemos tener en cuenta que no todos los servicios de **web hosting**, o alojamiento web, admiten esta opción.

RESPETAR LOS ESTÁNDARES

Hay quienes prefieren escribir documentos no válidos con la condición de que se visualicen bien en la mayor parte de los navegadores, cuando en realidad son las empresas desarrolladoras de esta clase de software las que deberían preocuparse por la compatibilidad 100% de los sitios web, cualquiera sea la tecnología con la que hayan sido desarrollados.

USO DE FRAMES

La **utilización de frames** para estructurar sitios está en desuso y hay un consenso general para en lo posible no utilizarlos. Originalmente surgieron como un mecanismo para dividir una página en secciones o subpáginas, lo que permitía ubicar cada parte en un archivo diferente y recargar una o varias partes según correspondiera. Esto permite consumir menos recursos. Este comportamiento puede emularse hoy en día si se utilizan técnicas como por ejemplo **Ajax**, que es una alternativa más transparente, segura y profesional.

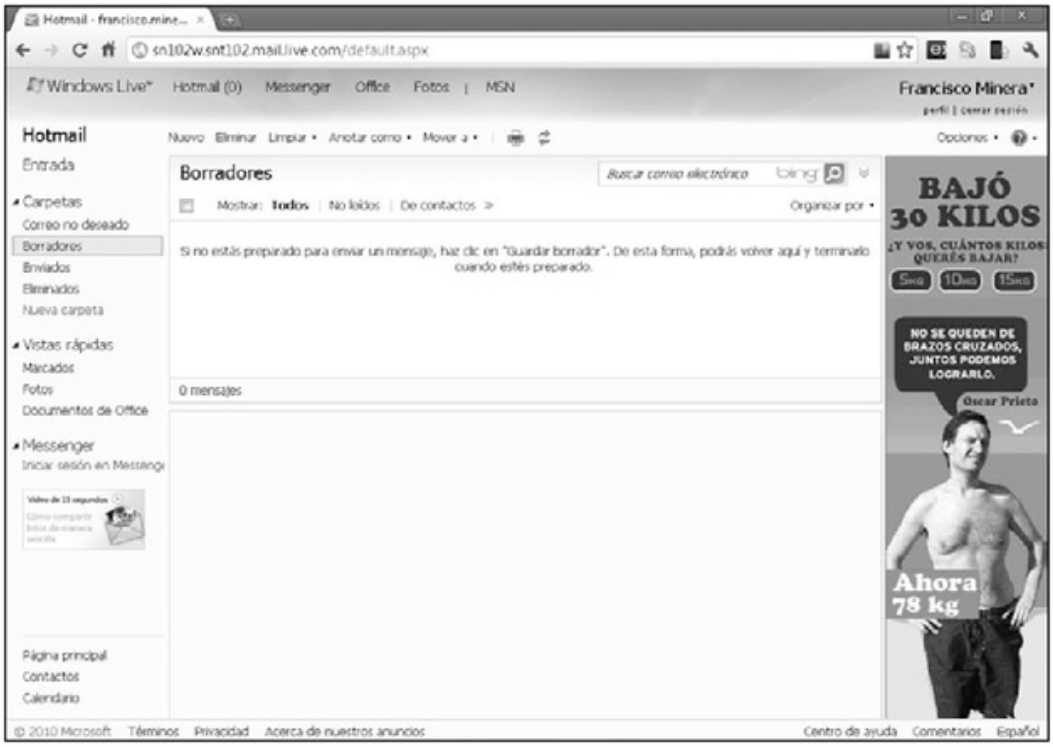


***Figura 17.** Aun cuando algunos sitios continúen utilizando **frames** para estructurar su contenido, es una técnica discutida por un gran sector de los diseñadores.*

Con respecto a la división en partes, la utilización de **templates** mediante algún lenguaje del lado servidor (como **PHP**, **ASP.net**, **JSP**, **PERL**, entre otros), es una técnica muy popular y probada. Otra alternativa es **SSI** (**Server Side Includes**) que, de estar habilitada en el servidor, permite incluir templates en el lado cliente.
Otra desventaja es el diseño: un sitio estructurado a partir de frames pierde frescura y da la sensación de ser antiguo o de no aprovechar las posibilidades existentes hoy en día. Además, en general solo es posible promocionar la página inicial, ya que las páginas de contenido se cargan en frames interiores, sin enlaces públicos, lo que también atenta contra el agregado a favoritos, la impresión de páginas o el uso de la barra de navegación del navegador, por ejemplo. Los frames desaparecieron en la versión 1.1 de XHTML, aunque en la actualidad se trabaja en la reestructuración del modelo. Podemos obtener más información acerca de este proceso en **www.w3.org/TR/xframes**.

Figura 18. *La especificación* ***XFrames*** *intenta reformular el trabajo mediante marcos y se espera que esté disponible para XHTML 2.0.*

En la actualidad, el uso de frames está siendo desplazado y poco a poco reemplazado por alternativas visualmente más atractivas, como la ya mencionada AJAX.

DHTML (HTML DINÁMICO)

El lenguaje de marcado de hipertexto dinámico (**DHTML**, *Dynamic HyperText Markup Language*) surge a partir de la conjunción de tecnologías que se unen para ubicar la funcionalidad de una página en módulos que interactúan entre sí.

El objetivo de este lenguaje es entregar la posibilidad de manipular elementos de XHTML (en general **div** o **span**) y modificar su contenido o su forma de visualización (por ejemplo poder ocultarlos, cambiarles el tamaño, intercambiar estilos, variar su posición, entre otra sposibilidades) a partir de JavaScript junto con la utilización de hojas de estilo CSS, por ejemplo.

El lenguaje de marcado de hipertexto dinámico o DHTML es una parte importante del modelo conocido como Ajax ya que permite actualizar partes de una página sin recargarla completamente en el lado cliente. Con Ajax se agrega la posibilidad de recuperar contenidos desde el lado servidor.

```
<span id="texto"></span>
```

Es importante saber que al contenido de los elementos denominados **div** y **span** se puede acceder mediante el uso de la propiedad **innerHTML**, podemos ver un ejemplo de esto en el código que mostramos a continuación:

```
<script type="text/javascript">
//<![CDATA[

document.getElementById("texto").innerHTML = "v";

//]]>
</script>
```

El nombre **div** es un apócope de división, y surge en HTML a partir de la necesidad de estructurar una página en varias secciones (en general un contenedor, una cabecera, un menú de opciones, un contenido principal y un pie de página) con el objetivo de poder identificar cada parte mediante el atributo denominado **id** o asociarla a una clase CSS por medio del atributo **class**:

***Figura 19.** Cada vez son más los sitios web que utilizan elementos como div o span para estructurar su contenido internamente.*

Luego en la hoja de estilo podemos definir las características correspondientes a cada una de las partes del documento, como por ejemplo el ancho, el alto, el color de fondo y la fuente, entre otras opciones adicionales.

INCLUSIÓN DE JAVASCRIPT

Hay dos maneras de incluir código JavaScript en una página XHTML: definir el contenido dentro del documento o referenciar los **archivos JS externos**. También es posible usar ambas en forma simultánea. En cuanto a la primera alternativa se recomienda definir el código correspondiente dentro una sección **CDATA**, lo que previene que los caracteres que tengan un significado especial para XML generen conflictos. Estas secciones se delimitan mediante la apertura **<![CDATA[** y el cierre **]]>**.

```
<script type="text/javascript">
//<![CDATA[

//código JavaScript

//]]>
</script>
```

La inclusión de referencias a archivos externos es la opción más recomendable ya que un mismo documento **JS** puede llegar a ser utilizado por más de una página XHTML. Esto deriva en un mantenimiento más sencillo ya que la modificación de un archivo tendrá repercusión en todas las páginas que lo referencien. Si optamos por esta opción, la sintaxis requerida es la siguiente:

```
<script type="text/javascript" language="javascript"
   src="archivo.js"></script>
```

El atributo **type** es obligatorio para la etiqueta **script**. Hay que tener en cuenta que ante cada apertura de este elemento se invoca al intérprete JavaScript. Esto puede llegar a demandar un consumo de recursos innecesario: hay que tratar de incluir la mínima cantidad necesaria de elementos **script**.

III JAVASCRIPT

Actualmente y desde hace ya mucho tiempo JavaScript está presente en la mayoría de los sitios web, independientemente de las características de los mismos. Este histórico lenguaje esta más vivo que nunca y cada vez son más los frameworks que amplían sus virtudes, por esta razón se muestra como una eficiente herramienta para los desarrolladores web.

Otro aspecto importante es el lugar de la página en el cual se incluyen estas referencias. Cuando se descarga una página, al encontrar una etiqueta **script** los navegadores detienen el proceso hasta que el interprete JavaScript haya terminado su trabajo. Por tal motivo, las recomendaciones apuntan a incluir estos elementos lo más cerca posible del final del documento XHTML, antes del cierre del elemento **body**.

INCLUSIÓN DE CSS

En la inserción de **hojas de estilo CSS** en páginas XHTML ocurre algo similar a lo visto con los scripts JS. Para incluir código o estilos CSS podemos contener el código en un archivo externo o ubicarlo dentro de una sección **CDATA**.

```
<link rel="stylesheet" href="archivo.css" type="text/css" />
```

Al incluir estilos CSS dentro de una sección **CDATA**, los caracteres con significado especial para XML son reemplazados por entidades equivalentes para evitar conflictos. Esta sección se delimita con **<![CDATA[** y **]]>**. Tanto en JavaScript como en CSS, **CDATA** está comentado según corresponda (con **//** en JavaScript y con **/******/** en CSS).

```
<style type="text/css">
/*<![CDATA[         */

body {
     background-color: #990000;
     font-size: 14px;
}
/*]]>               */
</style>
```

III DISPOSITIVOS Y CSS

Debemos tener en cuenta que uno de los beneficios más importantes, entre los derivados de la utilización de las hojas de estilo es la posibilidad de seleccionar un dispositivo en particular: de esta forma un documento puede adoptar diseños distintos en diferentes dispositivos, por ejemplo uno para impresión y otro distinto para la web.

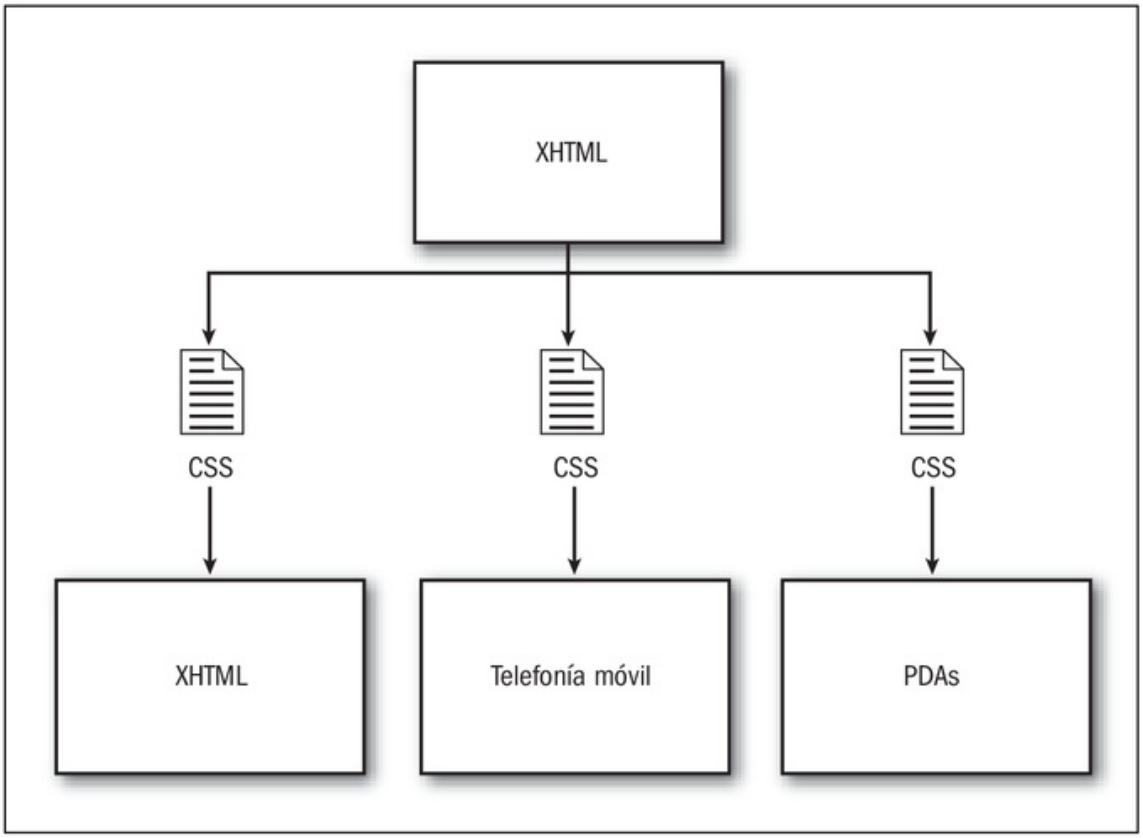


Figura 20. *Un mismo documento XHTML puede utilizar diferentes hojas de estilo según el dispositivo mediante el cual se tendrá acceso.*

A diferencia de lo que sucede con JavaScript, es recomendable incluir las referencias CSS lo más cerca posible del inicio de la página, preferentemente dentro de la etiqueta **head**. Esto se debe a que los navegadores, por lo menos la mayoría de ellos, no muestran ninguna salida hasta que los estilos se hayan cargado.

ELEMENTOS DISPONIBLES

XHTML nos provee un conjunto de elementos sumamente útiles para trabajar en el desarrollo de páginas. Las opciones disponibles (divididas en categorías) para implementar y utilizar son las siguientes:

NOMBRE	DESCRIPCION	ATRIBUTOS
address	Información acerca del autor.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
blockquote	Cita (larga).	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, cite

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
div	Contenedor.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align
dl	Definición de una lista.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, compact
fieldset	Agrupa los campos de un formulario.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
form	Formulario.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, action, method, enctype, accept, name, onsubmit, onreset, accept-charset, target
h1, h2, h3, h4, h5, h6	Niveles para encabezados.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align
hr	Línea horizontal.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align, size, width, noshade
noframes	Contenido alternativo para las páginas que utilizan marcos en navegadores que no los soportan.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
noscript	Contenido alternativo para las páginas que utilizan scripts en navegadores que no los soportan.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
ol	Lista ordenada.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, type, start, compact
p	Párrafo.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align
pre	Texto preformateado.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, width
ul	Lista desordenada.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, type, compact

Tabla 1. *Elementos de bloque. Son útiles para reemplazar el formato estándar propio del lenguaje HTML, obteniendo los mismos resultados.*

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
a	Enlace interno o externo.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, charset, type, name, rel, rev, href, hreflang, target, shape, coords, accesskey, tabindex, onfocus, onblur
abbr	Abreviatura.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
acronym	Acrónimo.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
b	Negrita.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
bdo	Texto bidireccional.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
big	Fuente grande.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
br	Salto de línea.	id, class, style, title, clear
button	Botón.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, name, value, type, disabled, accesskey, tabindex, onfocus, onblur
cite	Cita.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
code	Código.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

EDITORES ACTUALES

Muchas herramientas de diseño web aún no producen código XHTML correcto. Esto crea la necesidad de seleccionar cuidadosamente un editor que proporcione las características que mejor se adecuen a lo que queremos alcanzar en cuanto a la creación de páginas web. Es aquí donde entran en juego los elementos que provee XHTML.

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
dfn	Definición.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
em	Énfasis.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
i	Itálicas.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
iframe	Marco interno.	id, class, style, title, londesc, name, src, frameborder, marginwidth, marginheight, scrolling, align, height, width
img	Imagen.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, src, alt, longdesc, name, height, width, usemap, ismap, align, border, hspace, vspace
input	Control de formulario para la entrada de datos. Los disponibles son: text, password, checkbox, radio, submit, image, reset, button, hidden y file.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, type, name, value, checked, disabled, readonly, size, maxlength, src, alt, ismap,usemap, onselect, onchange, accept, align, accesskey, tabindex, onfocus, onblur
kbd	Texto a ingresar.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
label	Título para los campos de un formulario.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, for, accesskey, onfocus, onblur
map	Mapa.	xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, id, class, style, title, name
object	Objeto embebido genérico.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, declare, classid, codebase, data, type, codetype, archive, standby, height, width, usemap, name, tabindex, align, border, hspace, vspace
q	Cita (corta).	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, cite
samp	Salida simple de los scripts.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
select	Combo de selección.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, name, size, multiple, disabled, tabindex, onfocus, onblur, onchange
small	Fuente pequeña.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
span	Contenedor genérico.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
strong	Énfasis en negrita.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
sub	Texto inferior.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
sup	Texto superior.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
textarea	Cuadro de texto de múltiples líneas.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, name, rows, cols, disabled, readonly, onselect, onchange, accesskey, tabindex, onfocus, onblur
tt	Texto con espaciado simple.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
var	Instancia de una variable.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Tabla 2. *Elementos en línea.*

XML Y HTML

Debemos aaber que los documentos XML no fueron creados con el fin de reemplazar a los tradicionales documentos HTML. Aunque es posible hacerlo, es importante recordar que estas tecnologías tienen objetivos diferentes. Mientras que XML apunta a estructurar la información, HTML se enfoca en darle formato para presentarla.

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
caption	Título de la tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown,onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align
col	Columna.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, span, width, valign, align, char, charoff
colgroup	Grupo de columnas.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown,onkeyup, span, width, valign, align, char, charoff
table	Tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, summary, width, border, frame, rules, cellspacing, cellpadding, align, bgcolor
tbody	Cuerpo de la tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align, char, charoff, valign.
td	Celda.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, abbr, axis, headers, scope, rowspan, colspan, align, char, charoff, valign, bgcolor, nowrap, width, height
tfoot	Pie de la tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align, char, charoff, valign
th	Celda de título.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, nmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown,onkeyup, abbr, axis, headers, scope, rowspan, colspan, align, char, charoff, valign, bgcolor, nowrap, width, height
thead	Encabezado de la tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align, char, charoff, valign
tr	Fila de la tabla.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, align, char, charoff, valign, bgcolor

Tabla 3. *Elementos de tabla.*

LA VERSIÓN DE XML

En caso de ser incluida, la línea de código que se encarga de contener la versión de XML (**<?xml version="1.0" ... ?>**) deberá figurar en primer lugar en nuestros documentos. No podemos dejar líneas en blanco ni espacios antes de esta declaración porque esto produciría errores al intentar reconocer y validar los documentos.

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
area	Áreas de un mapa.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, shape, coords, href, nohref, alt, target, accesskey, tabindex, onfocus, onblur
base	Dirección base del documento.	href, id, target
body	Cuerpo del documento.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, onload, onunload, background, bgcolor, text, link, vlink, alink
dd	Definición.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick,onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
del	Texto eliminado.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, cite, datetime
dt	Definición de un término.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
frame	Marco.	id, class, style, title, longdesc, name, src, frameborder, marginwidth, marginheight, noresize, scrolling
frameset	Contenedor de marcos.	id, class, style, title, rows, cols, onload, onunload
head	Encabezado del documento.	xml:lang, lang, dir, id, profile
html	Elemento raíz.	xmlns, id, version, xml:lang, lang, dir
ins	Texto insertado.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, cite, datetime
legend	Leyenda.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, accesskey, align
li	Lista.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, type, value

III FUNCIONALIDAD

El lenguaje XML se ha convertido en una tecnología que se ve en pequeños, medianos, y también grandes proyectos. Esto nos entrega una prueba de lo importante que es para los desarrolladores conocer sus principios fundamentales y sus posibilidades. El nivel de complejidad de la aplicación en la que se implemente no influye.

NOMBRE	DESCRIPCIÓN	ATRIBUTOS
link	Enlace independiente del dispositivo de acceso. Entre los tipos disponibles se encuentran: alternate, stylesheet, start, next, prev, contents, index, glossary, copyright, chapter, section, subsection, appendix, help y bookmark.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, charset, href, hreflang, type, rel, rev, media, target
meta	Metainformación.	xml:lang, lang, dir, id, http-equiv, name, content, scheme
optgroup	Grupo de opciones.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, disabled, label
option	Opción seleccionable.	id, class, style, title, xml:lang, lang, dir, onclick, ondblclick,onmousedown onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup, selected, disabled, label, value
param	Parámetro.	id, name, value, valuetype, type
script	Script.	id, charset, type, src, defer, language
style	Definición de estilos.	xml:lang, lang, dir, id, type, media, title
title	Título del documento.	xml:lang, lang, dir, id

***Tabla 4.** Otros elementos.*

MÓDULOS EN XHTML

Es importante saber que XHTML está compuesto por módulos y cada uno de ellos contiene elementos que están a disposición del diseñador. Algunos de los módulos más utilizados de XHTML 1.1 son los siguientes:

NOMBRE	ELEMENTOS
Módulo Estructural	body, head, html, title
Módulo Texto	abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Módulo Hipertexto	a
Módulo Lista	dl, dt, dd, ol, ul, li
Módulo Objeto	object, param
Módulo Presentación	b, big, hr, i, small, sub, sup, tt
Módulo Editar	del, ins
Módulo Texto Bidireccional	bdo
Módulo Formularios	button, fieldset, form, input, label, legend, select, optgroup, option, textarea
Módulo Tablas	caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr
Módulo Imagen	img

NOMBRE	ELEMENTOS
Módulo Mapa lado Cliente	area, map
Módulo Metainformación	meta
Módulo de Scripts	noscript, script
Módulo Hoja de estilos	style
Módulo Link	link
Módulo Base	Base

***Tabla 5.** Módulos en XHTML 1.1.*

Podemos obtener más información acerca de la totalidad de los módulos en la especificación oficial publicada por la W3C.

XHTML 2.0

Uno de los aspectos más importantes de XHTML 2.0 consiste en la decisión por parte de W3C de no mantener una compatibilidad con respecto a las versiones previas (ya sea de HTML como de XHTML). Entre las innovaciones se incluyen:

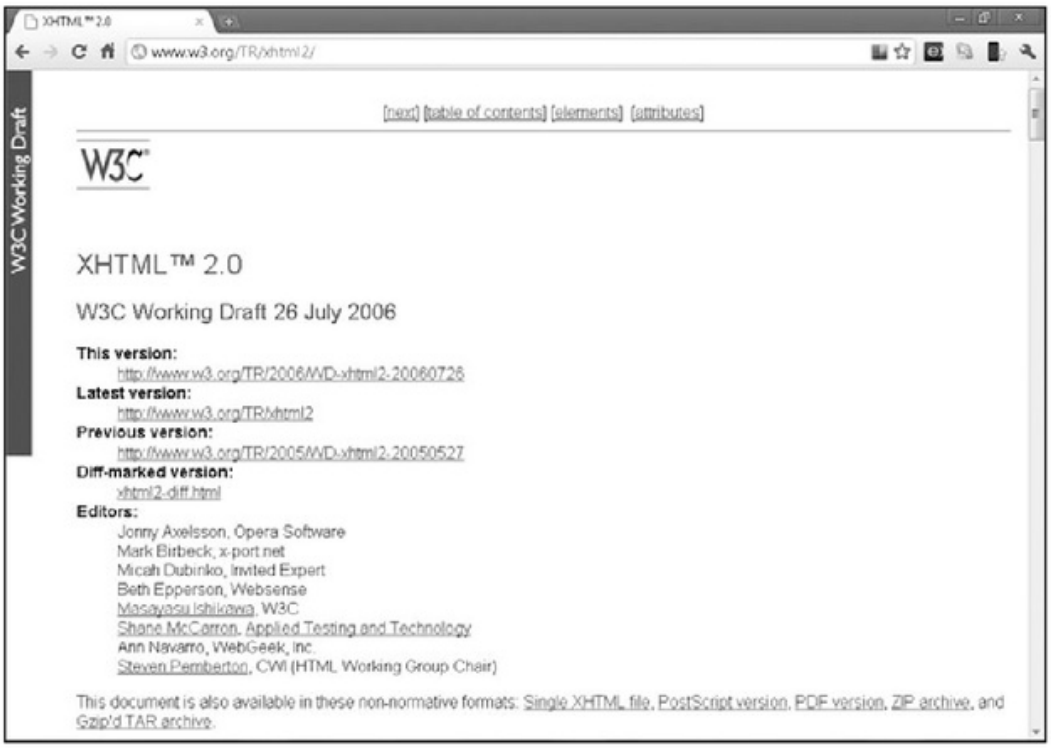


***Figura 21.** La especificación XHTML 2.0 está siendo revisada y se esperan cambios importantes que tendrán repercusión en el diseño y el desarrollo de aplicaciones.*

- **section** (con la posibilidad de anidarse para obtener niveles ilimitados) en reemplazo de los **Hn** (**H1** a **H6**). Dentro de cada sección se podrá definir un elemento

llamado **H** para estipular títulos:

```
<section>
  <h>Título #1</h>
  <section>
    <h>SubTítulo #1</h>
  </section>
  <h>Título #2</h>
  ...
</section>
```

- Elemento separador **Separator** en lugar de **HR**.
- El elemento **BR** es reemplazado por los elementos **l** y **line**.
- Se introduce **BLOCKCODE** en lugar de **PRE**.
- Modificaciones en el funcionamiento del elemento **P**.
- Reemplazo de los elementos **INS** y **DEL**.
- Especificación de metadatos mediante **META** y **LINK**.
- El elemento **SCRIPT** pasa a denominarse **HANDLER**.
- Inclusión de eventos XML (**XML Events**).
- Implementación del atributo **HREF** para cualquier elemento.
- Implementación del atributo **SRC** para cualquier elemento (recomendado por la especificación para cargar imágenes, por ejemplo).
- Integración con estándares en vías de desarrollo como **XForms** (formularios), **XFrames** (marcos) y **DIAL** (aplicaciones independientes al dispositivo).
- Debemos saber que se podrá incluir contenido desde documentos externos XML (mediante el elemento XML denominado **include**) sin necesidad de usar un lenguaje de programación del lado servidor:

```
<span><ns:include href="info.xml" /></span>
```

- el elemento **object** se utiliza en reemplazo de **style**, **img** y **applet**, y para definir el

RECOMENDACIÓN

Para obtener la recomendación oficial por parte de la W3C sobre la última versión disponible de XML, podemos visitar el sitio web que se encuentraq en la dirección **www.w3.org/XML**. En este sitio encontraremos una gran cantidad de información relevante y enlaces a sitios que poseen datos sobre otras tecnologías basadas en XML.

tipo **MIME** se utiliza el atributo **srctype**:

```
<object src="imagen.png" srctype="image/png"></object>
```

Otro de los cambios importantes es que se espera que la especificación sea más breve y concreta que las anteriores, sobre todo porque se centrará más en dispositivos en general (denominados **agentes de usuario**) y no en navegadores web de computadoras de escritorio. Además, la validación XML será obligatoria. De lo contrario los navegadores no aceptarán las páginas como XHTML 2.0. Recordemos que hoy no solo se utilizan especificaciones XHTML para los navegadores web de escritorio, sino que también se implementa esta tecnología en los navegadores de celulares, entre otros.

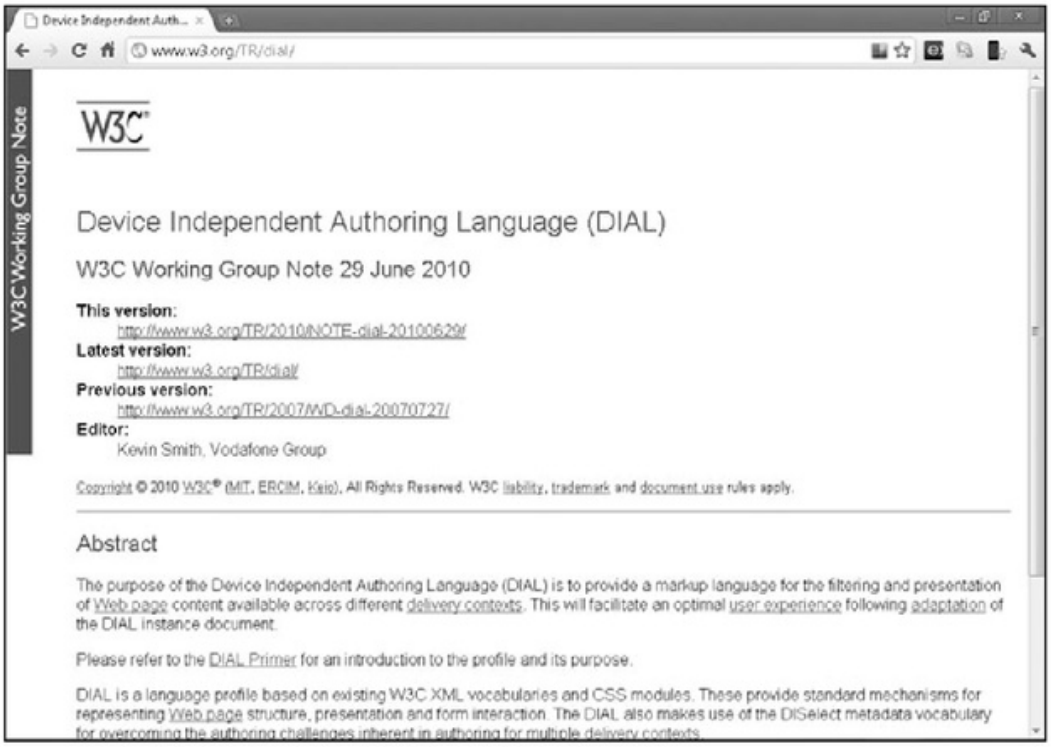


***Figura 22. DIAL** propone el desarrollo de aplicaciones independientes al dispositivo, lo que pone énfasis en la accesibilidad.*

Si bien la puesta en práctica de la nueva versión de XHTML no será inmediata, siempre es interesante observar en qué sentido se mueve el estándar y cuál es el nivel de apoyo que dan las empresas desarrolladoras de dispositivos para acceder a esta clase contenidos. También cabe destacar que, según la especificación oficial, la gran parte de **XHTML 2** ya funciona en los navegadores actuales. Podemos obtener más información al respecto en las siguientes direcciones:

- **www.w3.org/TR/xhtml2** (XHTML).
- **www.w3.org/MarkUp/Forms** (XForms).
- **www.w3.org/TR/xframes** (XFrames).

- **www.w3.org/TR/dial** (DIAL).

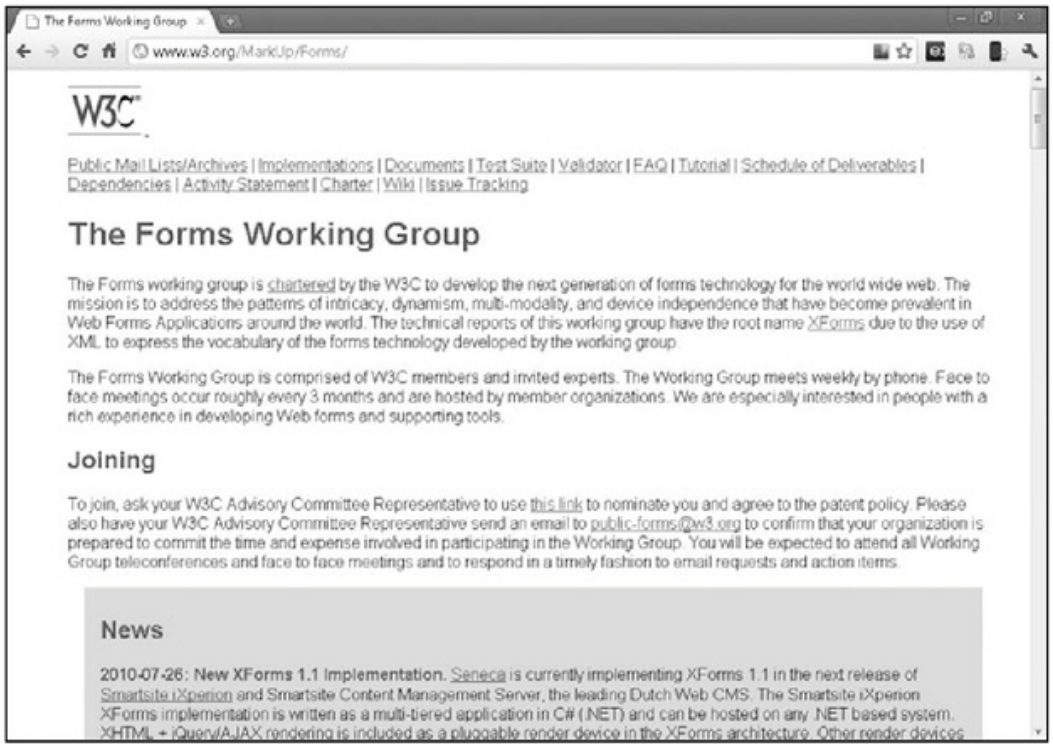


Figura 23. *Desde hace un tiempo que se trabaja en los* ***XForms*** *y se espera que en la versión 2.0 de XHTML ya sean una realidad.*

... RESUMEN

Las ventajas ofrecidas por XHTML en comparación con HTML son múltiples y en este capítulo observamos las más destacadas. Además, hicimos un recorrido por los puntos necesarios para construir documentos XHTML válidos y bien formados, y terminamos con las características que se esperan para XHTML versión 2.0.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son las diferencias principales entre los lenguajes HTML y XHTML?

2 ¿Cuál es la relación entre XHTML y XML?

3 ¿Cuáles son las características principales de XHTML?

4 ¿Qué hay de nuevo en XHTML 2.0?

5 ¿Qué es un Doctype? ¿Para qué sirve?

6 ¿Cuáles son los Doctype disponibles para XHTML 1.0?

7 ¿Cómo se valida un documento XHTML?

8 ¿Para qué sirven las cabeceras HTTP?

9 ¿Para qué sirve la herramienta HTML Tidy?

10 ¿Qué es un agente de usuario?

EJERCICIOS PRÁCTICOS

1 Genere un documento XHTML y revíselo con la herramienta HTML Tidy.

2 Genere un documento XHTML y aplique una validación con el servicio disponible en el sitio de la W3C.

3 Descargue la extensión Webdeveloper Toolbar para Firefox y visualice las funcionalidades disponibles.

4 Busque más información acerca de la cabecera Robots y su vinculación con los motores de búsqueda.

5 Nombre 7 reglas que debe cumplir un documento XHTML para considerarse como válido.

CSS

En este capítulo se exponen algunos consejos prácticos para optimizar nuestras hojas de estilo y lograr que la carga de las páginas requiera la menor cantidad de recursos posible.

Consejos prácticos	**64**
Inclusión de documentos	**64**
Posicionamiento	**70**
Escritura reducida	**71**
Fuentes	74
Fondos	75
Listas	75
Agrupación de estilos	**75**
Colores RGB	**77**
Imágenes en conjunto	**79**
Opciones adicionales	**81**
Aplicaciones para el tratamiento de hojas de estilo	82
Juego de caracteres	**88**
Validación de documentos	**88**
Compatibilidad entre navegadores	**89**
Diseño sin tablas	**92**
Resumen	**93**
Actividades	**94**

CONSEJOS PRÁCTICOS

Es indudable que el perfeccionamiento de un sitio es una tarea compleja que requiere muchas acciones por parte del autor o responsable. Una de las más importantes es la referida al tratamiento de las hojas de estilo, y a continuación veremos en distintos puntos cómo disminuir el tamaño y agilizar la carga de un archivo CSS (*Cascading Style Sheets*).

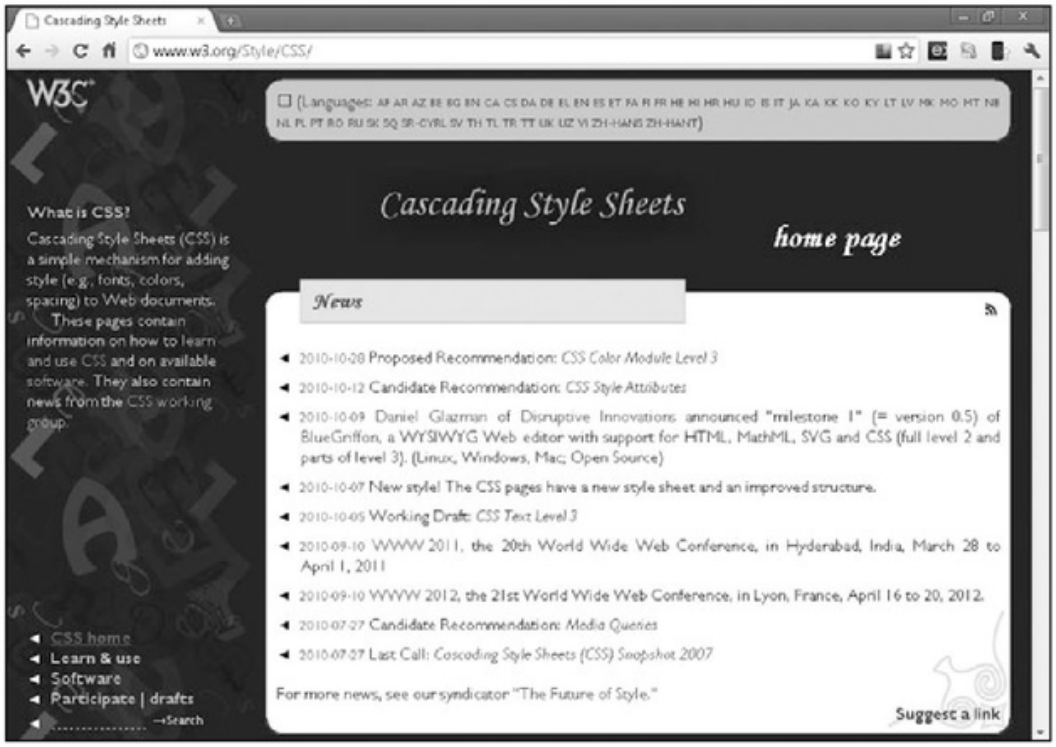


Figura 1. *CSS es uno de los múltiples estándares mantenidos por la organización* ***W3C (World Wide Web Consortium).***

CSS posee su propio sitio web en el que podemos encontrar enlaces a tutoriales y ejemplos acerca de la utilización hojas de estilo, además de las últimas novedades. El sitio web al que debemos acceder es **www.w3.org/Style/CSS**.

INCLUSIÓN DE DOCUMENTOS

Existen distintas alternativas a la hora de ligar estilos CSS a un documento XHTML.

1. Enlazar a un documento externo.
2. Embeber estilos dentro del documento.
3. Importar hojas de estilo.
4. Mediante estilos en línea.

1. Enlazar a un documento externo mediante el elemento **link**, que se debe ubicar dentro del elemento **head**. El atributo opcional **type** sirve para especificar el tipo de salida (el **Content-type**) y si el servidor reconoce los documentos de extensión **CSS** como **text/css** no es necesario incluirlo de manera explícita.

```
<link rel="stylesheet" href="archivo.css" type="text/css" />
```

Otro atributo opcional es **media**, que especifica el tipo de dispositivo sobre el que se aplicará la hoja de estilos. Entre los valores posibles se encuentran:

- **screen** (el valor por defecto, sitios web habituales).
- **print** (para las versiones imprimibles).
- **projection** (presentaciones).
- **aural** (sintetizadores).
- **braille** (presentaciones mediante el código táctil **Braille**).
- **tty** (presentaciones en modo carácter).
- **tv** (televisión).
- **all** (para todos los dispositivos disponibles).

Podemos incluir una hoja de estilo diferente para uno o más dispositivos o utilizar la misma, en cuyo caso los valores de media se separan por comas.
El atributo **rel** se utiliza para definir relaciones entre un documento XHTML y la hoja de estilos. Puede tomar dos valores: **StyleSheet** (para las hojas de estilo preferidas o que tendrán ascendencia sobre las demás) y **Alternate StyleSheet** (alternativas, que se utilizarán en los casos en los que las hojas predeterminadas no definan ninguna acción). No todos los navegadores trabajan con hojas de estilo alternativas. Es posible incluir más de una hoja preferida siempre y cuando tengan el mismo atributo **title**.

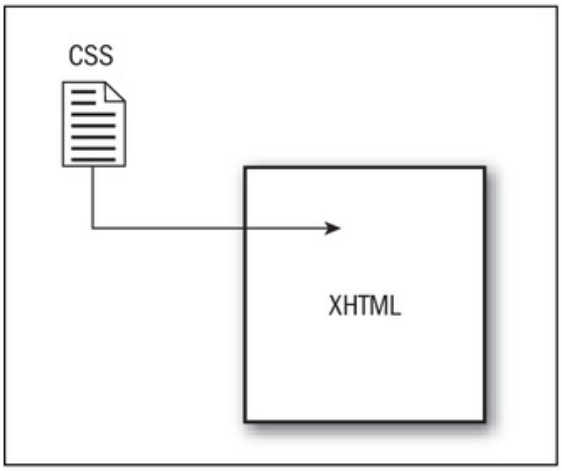


***Figura 2.** Es posible relacionar una hoja de estilos CSS externa a un documento XHTML.*

En resumen, es de mucha importancia incluir estilos desde hojas externas para centralizar los aspectos de diseño y poder modificar los distintos comportamientos del sitio web que desarrollamos en conjunto a partir de la reforma de un solo archivo. Esto hará nuestra tarea mucho más práctica y de fácil actualización sin riesgo a alterar la configuración y retrasar nuestras otras labores.

2. Embeber estilos dentro del documento. Esta opción nos permite incluir reglas de estilo que solo tendrán validez para la página actual. Se definen dentro del elemento **head** entre las etiquetas **style** de apertura y cierre.

```
<STYLE TYPE="text/css" MEDIA="screen">

BODY  {
  color: #909090;
  margin: 10px;
}

P {
  background: blue;
  color: #000;
}

.estilo {
  padding: 5px;
  text-align: center;
}
</STYLE>
```

Debemos tener en cuenta que si se utilizan ambos métodos conjuntamente, y tanto en la hoja externa como en las definiciones internas hay un mismo estilo, este tendrá preferencia por sobre el estilo externo.

CAMBIOS DISPONIBLES

CSS versión 2 es actualmente la más utilizada, y podemos obtener un listado con el detalle de los cambios ocurridos si tomamos como base la versión 1 en el sitio oficial de la W3C, que encontramos en la dirección **www.w3.org/TR/REC-CSS2/changes.html**. Allí se enumeran las nuevas posibilidades y las opciones que ya no estarán disponibles.

Figura 3. *Existe la posibilidad de incluir o embeber estilos CSS directamente en un documento XHTML.*

La idea detrás de esta clase de inclusión es que los estilos definidos solo serán utilizados en la página que los incorpore. De lo contrario, o sea si un mismo estilo existe en más de una página, es conveniente incluirlo en un archivo externo y vincularlo a las páginas correspondientes. Trabajar con hojas de estilo externas beneficia el mantenimiento de un sitio. Podemos asociar un archivo de estilo a varios documentos, por lo que en vez de modificar cada uno de ellos por separado, solo necesitaremos realizar cambios sobre la hoja de estilo.

En algunos navegadores no tan actuales, la definición de estilos se incluía en la salida de la página, y para evitarlo se encerraba el contenido dentro de comentarios **SGML** (los mismos de XHTML). Esto es válido para HTML pero no para XHTML, que no los admite dentro de la etiqueta **style**.

Figura 4. *Firefox es uno de los navegadores más utilizados y a la vez uno de los que más respeto tiene hacia los estándares web.*

3. Importar hojas de estilo. Esta alternativa es similar a las anteriores pero en conjunto. Dentro del estándar CSS existe una cláusula **@import** que recibe como argumento la ruta a una hoja de estilos externa:

```
@import url(/css/estilos.css);
```

Puede incluirse en conjunto con estilos internos, siempre al comienzo:

```
<STYLE TYPE="text/css" MEDIA="screen">
@import url(/css/estilos.css);

BODY  {
  color: #909090;
  margin: 10px;
}

P {
  background: blue;
  color: #000;
}

.estilo {
  padding: 5px;
  text-align: center;
}

</STYLE>
```

Si se utilizan los dos métodos en conjunto y tanto en la hoja externa como en las definiciones internas hay un mismo estilo, el interno tendrá preferencia sobre el externo.

MEDIOS DISPONIBLES

CSS no solo nos permite aplicar estilos para obtener una salida que pueda ser comprendida por un navegador web, sino que puede especificarse el tipo de salida que se requiere (por ejemplo, código braille, textos impresos, computadoras de mano, etcétera). Podemos obtener más información sobre los medios disponibles en el sitio **www.w3.org/TR/CSS2/media.html.**

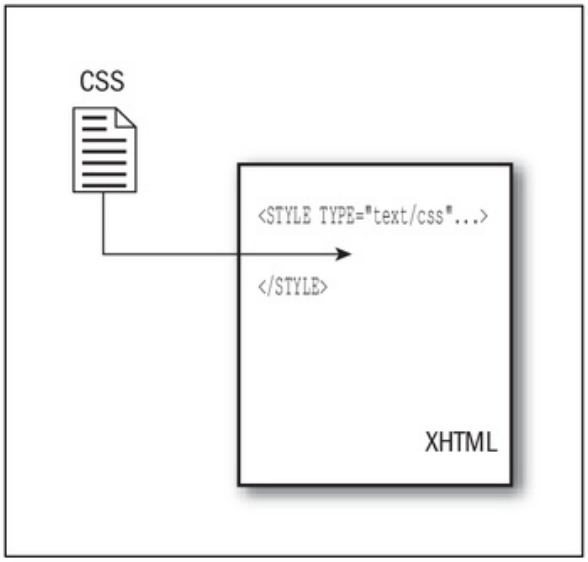


Figura 5. *Con la cláusula import podemos utilizar una hoja externa y estilos internos al mismo tiempo.*

4. Mediante estilos en línea. También existe la posibilidad de formatear elementos dentro del documento mediante el atributo **style**, que es un método conocido como **inline**:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body style="color: #909090; margin: 10px;">
 ...
 </body>
</html>
```

III VISIÓN Y PREVISIÓN

Es necesario recordar que la utilización y el respeto hacia los estándares que han sido establecidos nos darán beneficios a corto y largo plazo. El mantenimiento y la ampliación de las características de las partes de un sitio se vuelven más sencillos si la estructura es clara y bien definida. Esto les permite trabajar no solo al autor original sino a terceros.

El atributo **style** sobrescribe los estilos definidos en un archivo CSS en caso de que exista un conflicto en el sentido de que se intente darle formato a un mismo elemento desde ambas opciones. No es soportado por algunos navegadores y como no es posible definir el medio (atributo **media**), por defecto se utiliza **all**. Puede utilizarse en cualquier elemento contenido en **body** (los únicos no habilitados son **basefont**, **param** y **script**). Para habilitar esta opción en los distintos navegadores debemos incluir la cabecera correspondiente dentro de **head**, aunque en la mayoría de los casos no es necesario:

```
<META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">
```

Este método es la manera menos recomendable de darle formato a un documento porque con el tiempo se volverá dificultoso modificar e incluso interpretar documentos XHTML. Además, ubicar en un mismo documento la información contenida y el formato de estilo no es recomendable y va en contra de lo que propone el estándar.

POSICIONAMIENTO

Es recomendable incluir las referencias CSS lo más cerca posible del inicio de la página, preferentemente dentro de la etiqueta **head**. Esto se debe a que la mayoría de los navegadores, no muestran ninguna salida hasta que los estilos se hayan cargado.

Figura 6. *La ubicación de las hojas de estilo CSS dentro de documentos XHTML es un punto importante que puede agilizar la carga de las páginas.*

Podemos obtener más información al respecto en el sitio oficial de la W3C: **www.w3.org/TR/html4/struct/links.html#h-12.3**.

ESCRITURA REDUCIDA

Una de las características de CSS es disminuir el peso de los documentos XHTML mediante archivos externos. El estándar provee algunas directivas para reducir el tamaño de las hojas si resumimos múltiples líneas en solo una. Por ejemplo:

```
.nombreClase {
  margin-top: 1px;
  margin-right: 2px;
  margin-bottom: 3px;  margin-left: 4px;}
```

podría resumirse en una sola línea y obtendríamos los mismos resultados:

```
.nombreClase {
  margin: 1px 2px 3px 4px;}
```

La secuencia comienza por la parte superior, y sigue el sentido de las agujas del reloj:

- arriba (**top**).
- abajo (**bottom**).
- derecha (**right**).
- izquierda (**left**).

***Figura 7.** Un mismo sitio puede modificar su estilo de manera completa a partir de CSS, como puede observarse en el sitio **www.csszengarden.com/tr/espanol**.*

El mismo caso puede aplicarse a directivas como **border** o **padding**:

```
/* formato original */
.nombreClase {
  padding-top: 1px;
  padding-right: 2px;
  padding-bottom: 3px;
  padding-left: 4px;
}

/* formato acotado */
.nombreClase {
  padding: 1px 2px 3px 4px;
}
```

Si necesitáramos incluir el mismo valor para todas las opciones, podríamos detallar solo un valor. Por ejemplo, las siguientes definiciones son equivalentes:

```
/* # 1 */
.nombreClase {
  padding-top: 10px;
  padding-right: 10px;
  padding-bottom: 10px;
  padding-left: 10px;
}

/* # 2 */
.nombreClase {
  padding: 10px 10px 10px 10px;
}

/* # 3 */
.nombreClase {
  padding: 10px;,
}
```

En caso de definir solo dos valores, el primero sería para arriba (**top**) y abajo (**bottom**), mientras que el segundo para izquierda (**left**) y derecha (**right**). Por ejemplo, las siguientes definiciones son equivalentes:

```
/* formato original */
.nombreClase {
  padding-top: 20px;
  padding-right: 10px;
  padding-bottom: 20px;
  padding-left: 10px;
}

/* formato acotado */
.nombreClase {
  padding: 20px 10px;
}
```

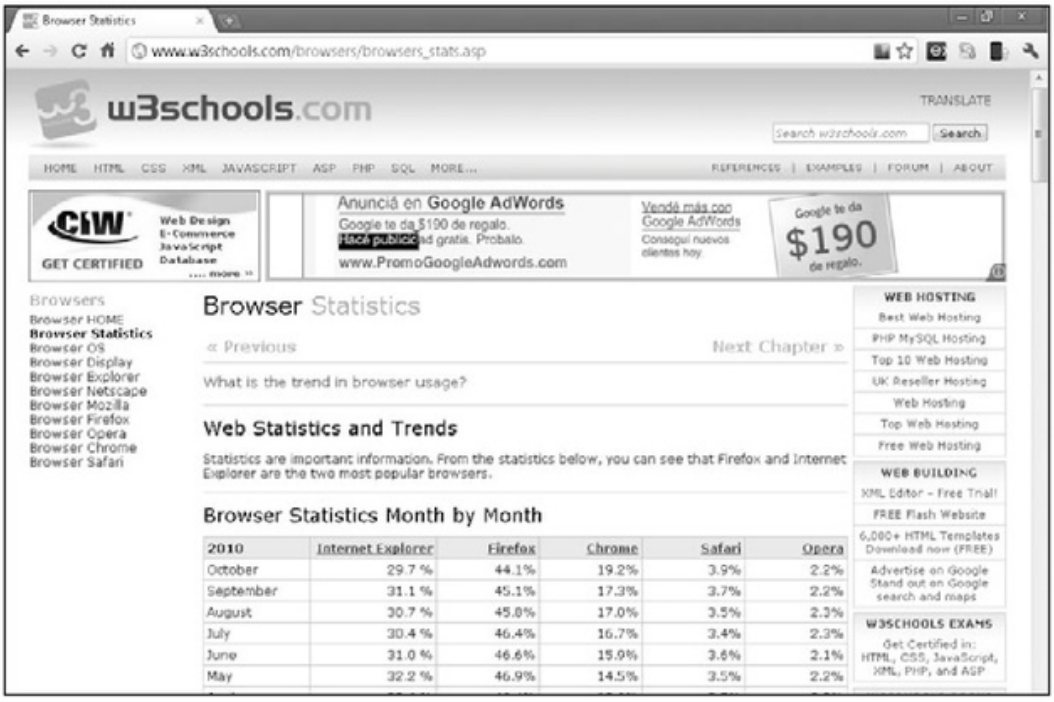


Figura 8. *La uniformidad de la hoja de estilos en relación con un navegador es un objetivo constante de los diseñadores (**www.w3schools.com/browsers/browsers_stats.asp**).*

TRABAJO EN CONJUNTO

Es necesario que tengamos en cuenta que CSS se encuentra vinculado a documentos estructurados: se utiliza desde hace tiempo en conjunto con HTML (se proveen ciertas funcionalidades aplicables sólo a los elementos propios de este lenguaje) y XML, y se encarga de añadirles estilos a los elementos componentes de los documentos.

Con tres valores, el primero sería para arriba (**top**), el segundo para izquierda (**left**) y derecha (**right**), y el último para abajo (**bottom**). Por ejemplo, las siguientes definiciones son equivalentes:

```
/* formato original */
.nombreClase {
  padding-top: 20px;
  padding-right: 10px;
  padding-bottom: 30px;
  padding-left: 10px;
}

/* formato acotado */
.nombreClase {
  padding: 20px 10px 30px;
}
```

Cada alternativa tiene su utilidad en distintos casos: la inclusión de un valor tiene sentido solo cuando en el formato extendido todos los valores son idénticos, la inclusión de dos valores solo cuando coinciden los valores horizontales y verticales, mientras que los tres valores cuando coinciden los lados izquierdo y derecho.

Fuentes

También es posible incorporar varias propiedades en una misma línea (**font-style**, **font-variant**, **font-weight**, **font-size**, **line-height** y **font-family**) relativas a la tipografía utilizada,

```
.nombreClase {
  font: italic bold 14pt Verdana, serif;
}
```

que sería equivalente a utilizar el formato completo:

```
.nombreClase {
  font-style: italic;
  font-weight: bold;
  font-size: 14pt;
```

```
  font-family: Verdana, serif;
}
```

En la forma abreviada, no es necesario especificar valores para todas las propiedades (solo son obligatorios **font-size** y **font-family**) aunque hay que respetar cierto orden preestablecido: las propiedades **font-weight**, **font-style** y **font-variant** se pueden especificar en cualquier orden, seguidas por el **font-size**, seguidas por una barra (/) y **line-height**, para terminar con **font-family**.

```
font: bold 1em/1.4em Verdana, Arial, sans-serif;
```

Fondos

Con la propiedad **background** y si separamos los valores con espacios es posible incorporar en una misma línea más de una propiedad (**background-image**, **background-color**, **background-repeat**, **background-attachment** y **background-position**, en este orden), tal como vimos en el apartado sobre las fuentes.

Listas

Con la propiedad **list-style** y si separamos los valores mediante espacios es posible incorporar más de una propiedad en una misma línea (**list-style-image**, **list-style-position** y **list-style-type**), tal como vimos en el apartado sobre las fuentes.

AGRUPACIÓN DE ESTILOS

La **herencia** es una de las características más importantes dentro de CSS. Un elemento **B** contenido dentro de un elemento **A** heredará los estilos de **A**, a menos que la definición de **B** incluya las propiedades que aparecen en **A**, solo que con valores diferentes (las reglas de estilo se aplican con precedencia a las más genéricas).

```
.nombreClase1 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}
```

```
.nombreClase2 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}

.nombreClase3 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}
```

Con esta cualidad podemos reducir el tamaño de nuestras hojas de estilo si agrupamos los estilos coincidentes, por lo que el código anterior podría convertirse en:

```
.nombreClase1, .nombreClase2, .nombreClase3 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}
```

Cada elemento está separado por una coma y un espacio. Un estilo heredado luego podrá ampliarse o modificarse en relación con su estado original:

```
.nombreClase1, .nombreClase2, .nombreClase3 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}

.nombreClase2 {
  background-color: #3399FF;
  padding: 10px;
}

.nombreClase3 {
  text-align: center;
}
```

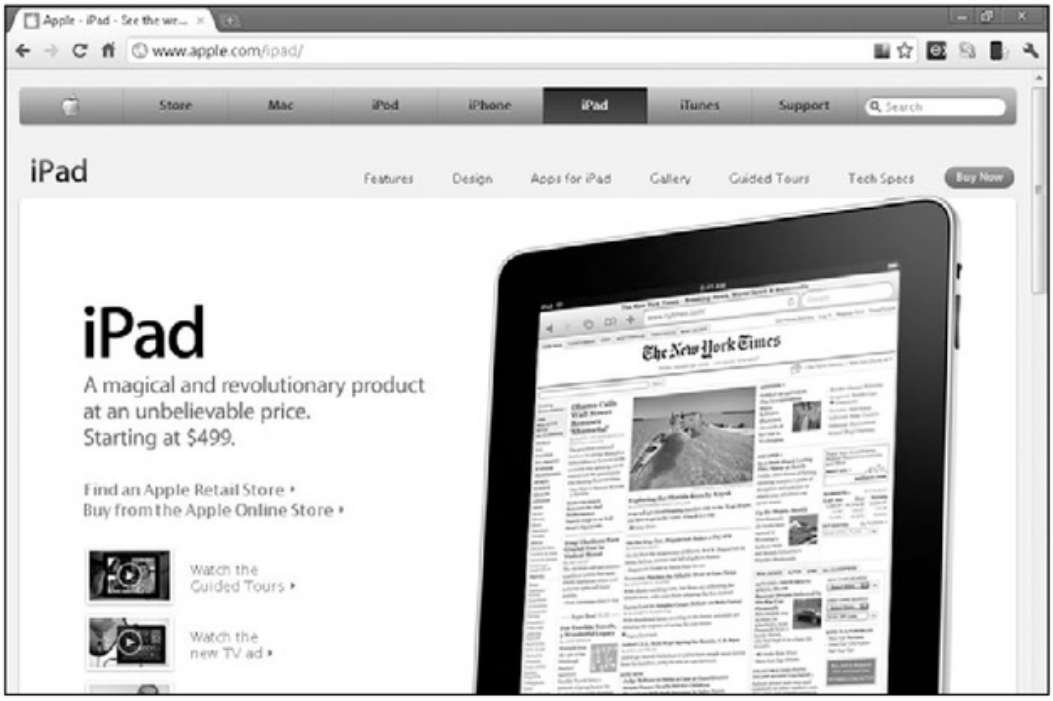


Figura 9. *CSS nos permite generar salidas para múltiples dispositivos, no solo computadoras de escritorio.*

Podemos obtener más opciones acerca de estos temas en la especificación oficial de CSS, en el sitio del organismo W3C.

COLORES RGB

Existen múltiples maneras para definir colores en CSS. Si utilizamos el modo **hexadecimal** podemos realizar una abreviatura en el código de los colores, siempre y cuando los pares que indican el rojo, el verde y el azul sean iguales. Así, el siguiente código

```
.nombreClase1 {
  background-color: #666666;
  padding: 10px;
}

.nombreClase2 {
  background-color: #FFFF00;

  color: #000000;
  padding: 5px;
}
```

Figura 10. *En la actualidad existe una gran cantidad de editores que nos permiten crear hojas de estilo de manera sencilla e intuitiva.*

podría convertirse en su equivalente:

```
.nombreClase1 {
  background-color: #666;
  padding: 10px;
}

.nombreClase2 {
  background-color: #FF0;
  color: #000;
  padding: 5px;
}
```

Las palabras reservadas de CSS para representar colores también ocupan menos espacio que la definición hexadecimal completa.

III DESARROLLO Y PRODUCCIÓN

Debemos tener en cuenta que en el mercado existen muchas herramientas que nos permiten eliminar de nuestro código aquellos caracteres que no interfieren en la funcionalidad del mismo. El precio de esto es la legibilidad del mismo, por lo que deberemos aplicar estas soluciones solo antes de pasar a producción.

IMÁGENES EN CONJUNTO

Cada imagen implica una nueva petición al servidor, y mientras menos peticiones hagamos en nuestras páginas, mejor. Una de las alternativas que puede servirnos en determinados casos es concatenar dos imágenes y luego, con CSS, mostrar solo la parte correspondiente. En el siguiente ejemplo contamos las siguientes imágenes:

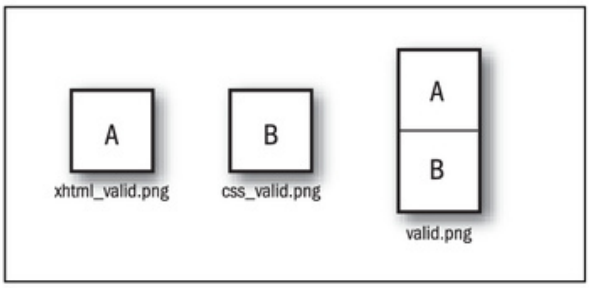


***Figura 11.** El acceso a partes de una misma imagen puede ayudarnos a reducir el número de peticiones.*

Si utilizamos las dos imágenes iniciales a mostrar en el documento por separado, tendremos el código a continuación:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <style type="text/css" media="screen">
  #xhtml {
    background-image: url(xhtml_valid.png);
     background-repeat: no-repeat;
    width: 16px;
```

CSS VERSIÓN 1

Debemos saber que la primera versión de CSS (CSS1) es aceptada por la mayoría de los navegadores web de la actualidad y prácticamente ha sido reemplazada por la segunda versión. Podemos encontrar la referencia oficial más ejemplos prácticos de uso si visitamos el sitio web que se encuentra en la dirección **www.w3.org/TR/REC-CSS1**.

```
    height: 16px;
  }

  #css {
    background-image: url(css_valid.png);
     background-repeat: no-repeat;
    width: 16px;
    height: 16px;
  }
  </style>
 </head>
 <body>

 <div id="xhtml"></div>
 <div id="css"></div>

 </body>
</html>
```

En cambio, y si realizamos una petición en lugar de dos, tendremos éste:

```
<style type="text/css" media="screen">
#xhtml {
  background-image: url(valid.png);
  background-position: center top;
  width: 16px;
  height: 16px;
}

#css {
  background-image: url(valid.png);
  background-position: center bottom;
  width: 16px;
  height: 16px;
}
</style>
```

Estas soluciones pueden ser viables para que los desarrolladores trabajemos con imágenes de tamaño pequeño y de dimensiones similares.

OPCIONES ADICIONALES

Una técnica utilizada para reducir el tamaño de los documentos CSS consiste en eliminar los saltos de línea, los espacios y las sangrías que no aportan ni modifican el comportamiento de la hoja de estilos original. Por ejemplo, el código que se transcribe a continuación

```
.nombreClase1, .nombreClase2, .nombreClase3 {
  margin: 5px;
  padding: 3px;
  visibility: none;
}

.nombreClase2 {
  background-color: #3399FF;
  padding: 10px;
}

.nombreClase3 {
  text-align: center;
```

podría quedar reducido a lo siguiente:

```
.nombreClase1, .nombreClase2, .nombreClase3 { margin: 5px; padding: 3px; vi
sibility: none; }
.nombreClase2 { background-color: #3399FF; padding: 10px; }
.nombreClase3 { text-align: center; }
```

En las hojas de estilo extensas el tamaño disminuye considerablemente, y la única contrariedad es la visualización del código para editarlo: se supone que esta transformación se debe realizar cuando la etapa de desarrollo haya finalizado. También puede ser beneficioso recortar el nombre de los estilos, siempre y cuando éstos se mantengan comprensibles.

Otra manera válida y sencilla de ahorrar espacio es mediante la eliminación de comentarios, que si bien son útiles durante la etapa de desarrollo de un sitio, al momento de pasar a la etapa de producción, cuando las hojas de estilo ya no deberían sufrir modificaciones radicales, pueden omitirse y así disminuir la cantidad de bytes utilizados. A su vez, el intérprete tendrá menos bytes para recorrer y procesar.

Aplicaciones para el tratamiento de hojas de estilo

En la actualidad, hay una gran cantidad de programas que permiten automatizar estas tareas, lo que hace que el diseñador pueda enfocarse solo en la parte referida al desarrollo específico de las páginas del sitio. A continuación, algunas de las herramientas utilizadas para mejorar el procesamiento de las hojas de estilo: **CSSTidy**, **Dust-Me Selectors** y **CSS Optimizer**.

CSSTidy

Con esta herramienta, podemos comprimir las hojas de estilo para lograr que ocupen menos espacio y, por consiguiente, que carguen más rápidamente durante la navegación de los usuarios. Está disponible en dos formatos: **PHP** (para acceder por web) y **C++** (se distribuye tanto el código fuente como un ejecutable). Podemos conseguir cualquiera de las dos versiones en **http://csstidy.sourceforge.net/download.php**.

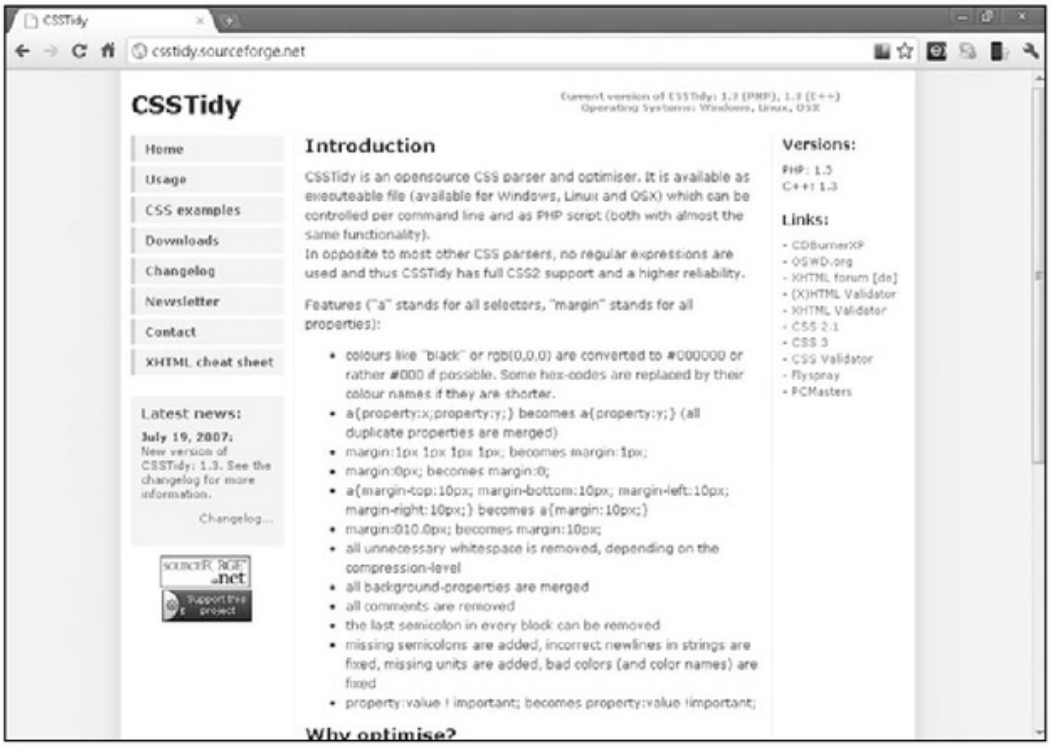


Figura 12. *CSSTidy es una herramienta* ***open source*** *disponible para distintas plataformas.*

CSS VERSIÓN 2

La segunda versión de CSS (**CSS2**) incorpora mejoras que tienen relación directa con XML y sus lenguajes derivados. Podemos visualizar la referencia oficial en el sitio web de la W3C, que se encuentra en la dirección **www.w3.org/TR/CSS21**. Debemos saber que allí podremos acceder a documentos técnicos específicos sobre la especificación.

En el primer caso, una vez descargada la distribución deberemos descomprimir los archivos y copiarlos a un directorio accesible mediante un servidor web, que puede ser local o externo. Luego accederemos al archivo **css_optimiser.php** que está dentro del directorio, y podremos observar una interfaz como la siguiente:

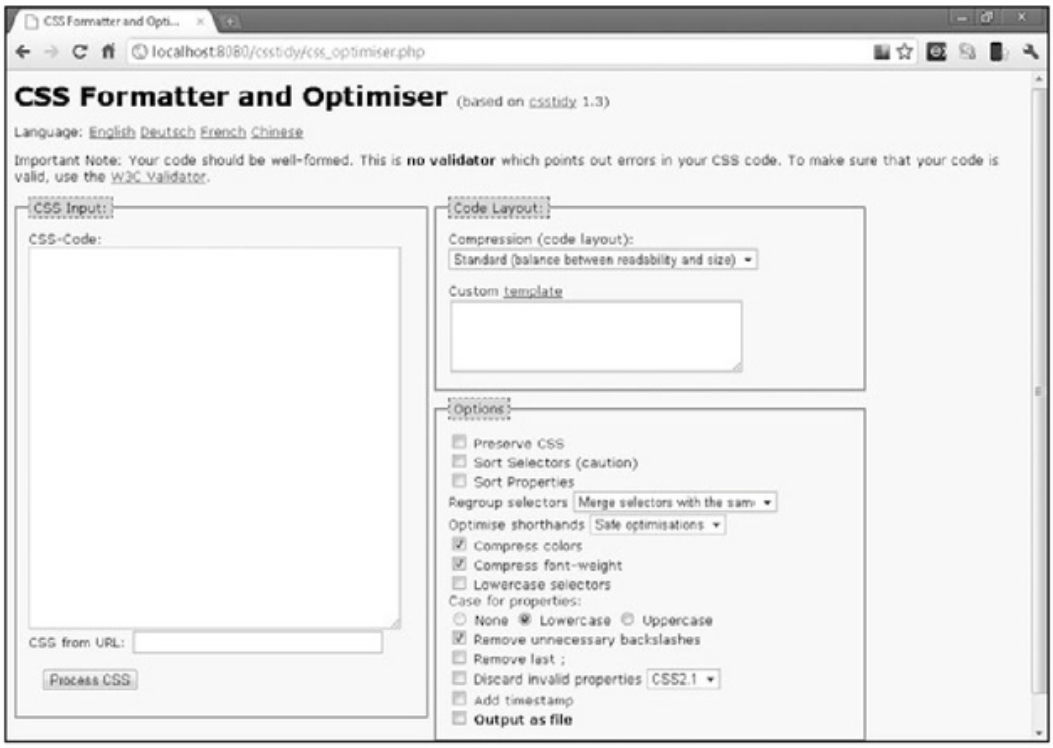


***Figura 13.** Antes de realizar el procesamiento deberemos configurar opciones relativas a la optimización de las hojas de estilo.*

Desde la sección **CSS Input** contamos con dos opciones: copiar el código CSS al cuadro de texto o definir la ruta absoluta al archivo que queramos procesar (por ejemplo, podría ser algo como **www.nombre-servidor.com/estilos/estilos.css**).
En la sección **Code Layout** definimos la compresión a realizar, con las opciones:

- **Highest** (una salida casi ilegible pero con muy poco peso).
- **High** (baja legibilidad, bajo peso).
- **Standard** (balance entre legibilidad y tamaño).
- **Low** (alta legibilidad).
- **Custom** (para personalizar la salida, más información en disponible en **http://csstidy.sourceforge.net/templates.php**).

Dentro de la sección **Options** contamos con la alternativa de preservar algunas opciones de la hoja de estilos (por ejemplo, comentarios), ordenar selectores, ordenar propiedades, agrupar selectores, reducir escritura, acortar nombres de colores, pasar a minúsculas selectores / propiedades, remover saltos de línea, remover el punto y coma de la última propiedad de cada estilo definido, descartar propiedades no

válidas, agregarle una marca de tiempo a la salida, y enviar el resultado a un archivo (para esto deberemos crear un directorio llamado **temp** en la distribución). Una vez concluido el procesamiento (que se inicia al presionar el botón **Process CSS**), podremos observar el listado de los cambios realizados, el radio de compresión logrado y el código fuente resultante (que según las opciones seleccionadas podrá encontrarse en un archivo dentro del directorio **temp**).

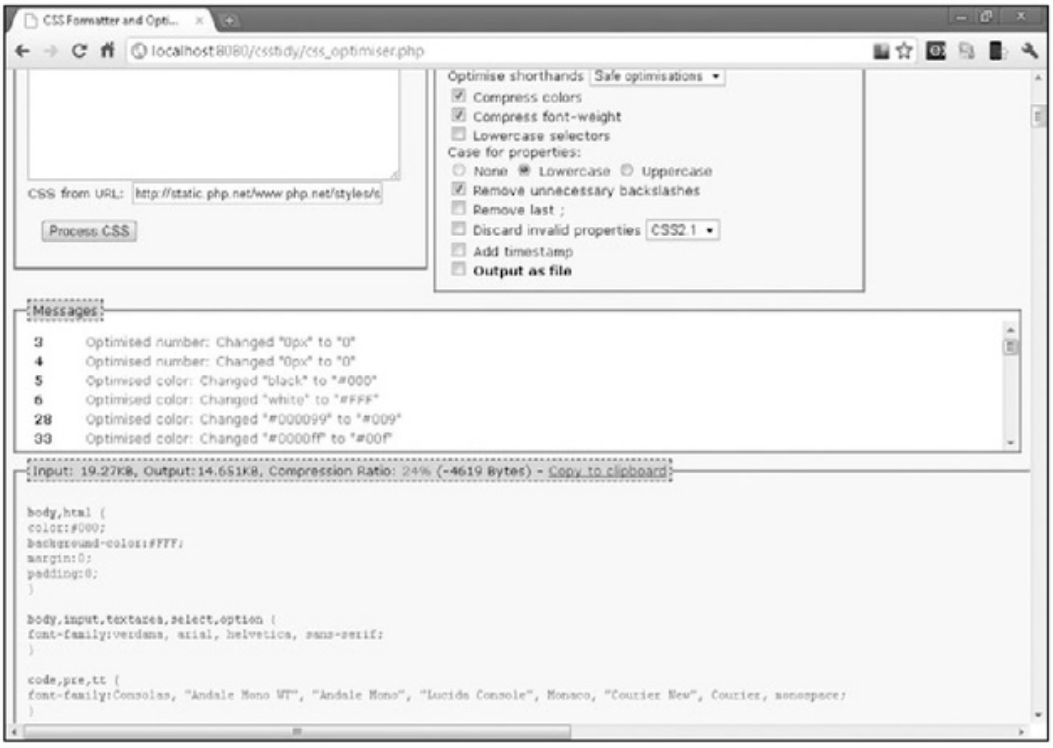


Figura 14. *La salida generada por* ***CSSTidy*** *podrá ser enviada a un archivo en disco para su posterior incorporación al sitio web.*

En cuanto a la versión ejecutable (**csstidy.exe**) deberemos acceder a una terminal e incluir las opciones necesarias, que son las mismas que aparecen en la versión web.

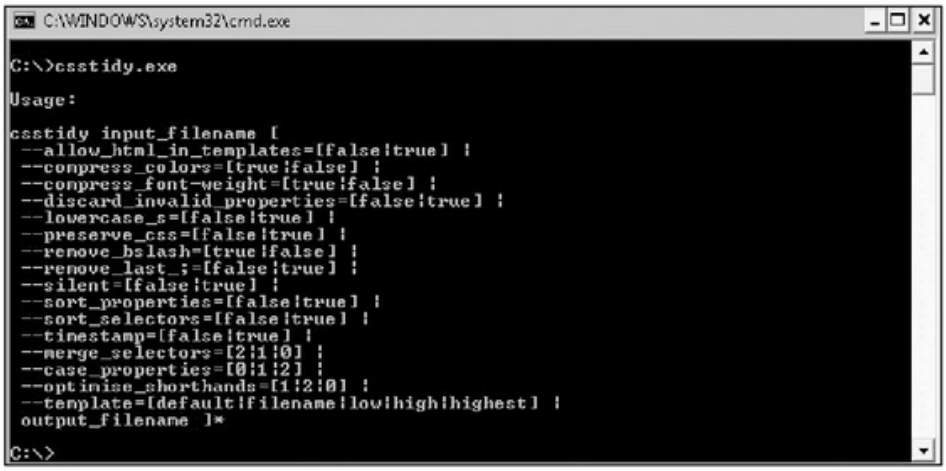


Figura 15. ***CSSTidy*** *puede ser utilizado en su versión de escritorio mediante el ejecutable incluido en la distribución oficial.*

Debemos tener en cuenta que es posible encontrar más información relevante acerca de esta herramienta si accedemos al sitio web oficial, el cual se encuentra en la dirección **http://csstidy.sourceforge.net**.

Dust-Me Selectors

Se trata de una extensión disponible para el navegador Mozilla Firefox y podemos descargarla de **www.sitepoint.com/dustmeselectors**.

Figura 16. *Firefox provee complementos de propósito general para mejorar las capacidades iniciales del navegador, uno de ellos es el acceso a las características de las páginas visitadas.*

Una vez hecho esto y después de reiniciada la aplicación, podremos observar un icono en la parte derecha de la barra de estado. Si hacemos clic, comenzará el proceso de revisión de las hojas de estilo asociadas a la página actual, y como resultado se brindará un listado con los estilos no utilizados.

CSS VERSIÓN 3

La organización W3C actualmente trabaja en la versión 3 de CSS, y podemos visitar la versión preliminar para saber cuáles serán las nuevas características y cuáles serán las opciones que dejarán de utilizarse, todo esto lo encontramos si visitamos el sitio web que se encuentra en la dirección **www.w3.org/Style/CSS/current-work**.

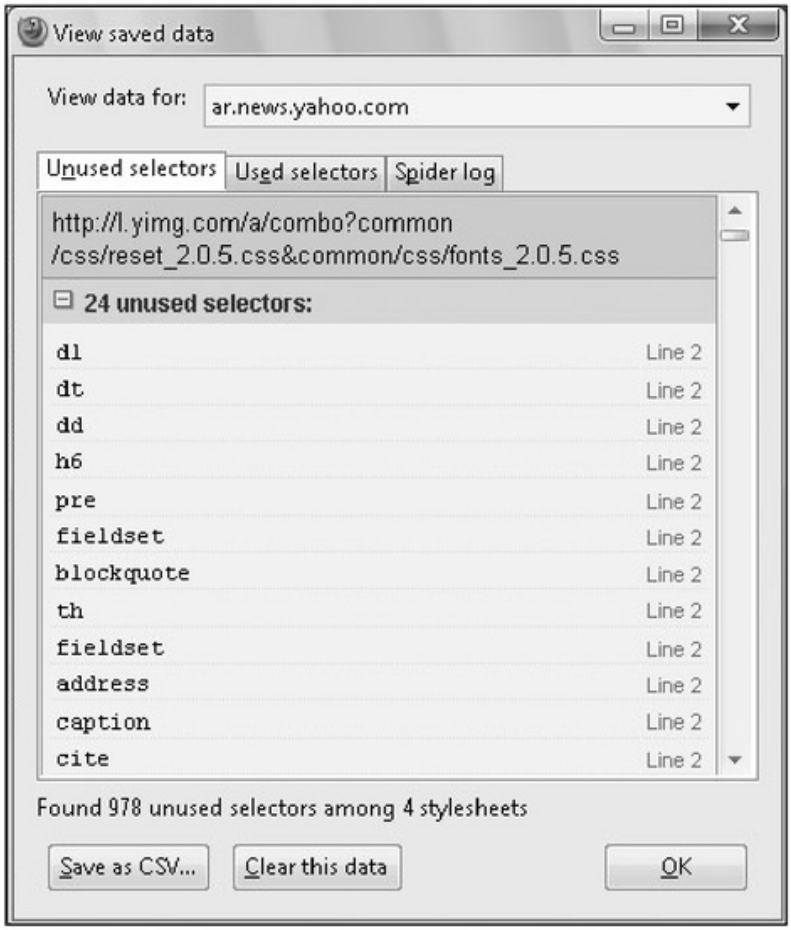


***Figura 17. Dust-Me Selectors** nos ofrece un resumen acerca de la utilización de los estilos en la página actualmente visitada.*

Debemos tener en cuenta que, además, se encarga de brindarnos la posibilidad de realizar un análisis del mapa del sitio (más conocido como **sitemap**), de esta forma podremos tener en cuenta las otras páginas.

CSS Optimizer

En el sitio web que podemos visitar en la dirección **http://flumpcakes.co.uk/css/optimiser** encontraremos una eficiente herramienta online que está desarrollada para permitirnos comprimir hojas de estilo. Su utilización es simple y podemos obtener beneficios de manera rápida y sin complicaciones.

III PROFESIONALISMO

Es necesario que tengamos en cuenta que una de las razones más importantes –en relación con la utilización de los estándares disponibles en busca de la optimización de las posibilidades de los sitios web en los cuales estamos trabajando–, es la de agregarle un mayor profesionalismo al trabajo diario del desarrollador o programador.

Figura 18. *Hoy en día existen numerosas aplicaciones que permiten la optimización de hojas de estilo de manera automatizada y personalizada.*

En cuanto a las opciones, podemos definir si eliminar saltos de línea, convertir colores de RGB a hexadecimal, pasar de rutas absolutas a relativas, agrupar estilos relacionados a los fondos utilizados, fuentes, listas, bordes y estilos en general.

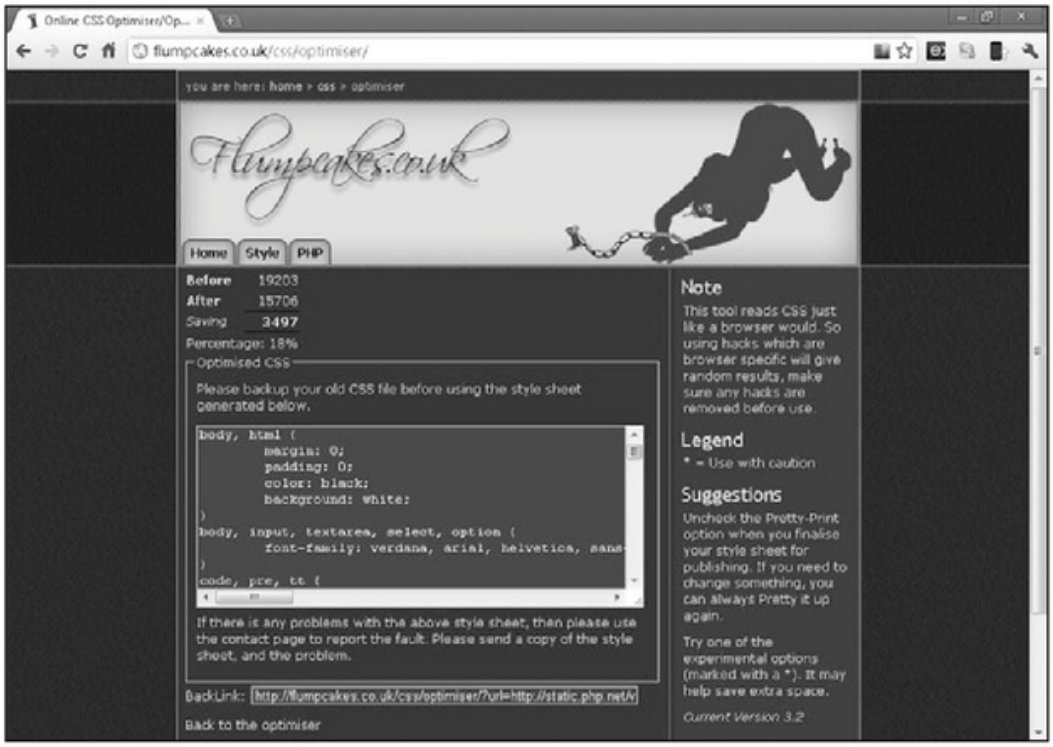


Figura 19. *Con esta herramienta podemos procesar hojas de estilos si especificamos la ruta al archivo o si copiamos el contenido al cuadro de texto.*

Una vez finalizado el proceso, podremos observar en la pantalla devuelta las estadísticas de la optimización realizada: tamaño inicial y posterior de la hoja de estilos, porcentaje ahorrado y la salida para copiar el código y utilizarlo en nuestro sitio.

JUEGO DE CARACTERES

Mediante **@charset** podemos definir el juego de caracteres a utilizar:

```
@charset "ISO-8859-1";
```

Utilizado desde una hoja de estilo, se debe ubicar antes de las declaraciones.

VALIDACIÓN DE DOCUMENTOS

En la actualidad existe un gran número de aplicaciones que nos permiten verificar si nuestras hojas de estilo están correctamente estructuradas o si hay inconsistencias como directivas en desuso, errores de sintaxis o repetición de sentencias. A continuación un recorrido por algunas de las alternativas disponibles:

- Existe un servicio online y gratuito de la W3C que nos permite comprobar la validez de las hojas de estilo CSS para saber si están correctamente formadas. Podemos acceder a este servicio en **http://jigsaw.w3.org/css-validator**.
- **Clean CSS**. Podemos obtener más información en el sitio oficial de la herramienta en **www.cleancss.com**.
- **XHTML-CSS**. Es una herramienta online que permite validar documentos XHTML y CSS al mismo tiempo. Podemos acceder a su funcionalidad en **http://xhtml-css.com**.

III DOCUMENTACIÓN

Si bien la documentación de código es un aspecto fundamental y necesario para cualquier sistema, es indudable que mientras más ordenados estén nuestros desarrollos, más fácil será la tarea de quienes en algún momento necesiten retomar la escritura del código para ampliar, adaptar o modificar el comportamiento del sistema.

- **CSS Validator**. Esta extensión para el navegador web Firefox admite características como la validación CSS mediante la W3C. Podemos instalarla si accedemos a **https://addons.mozilla.org/en-US/firefox/addon/2289**.

***Figura 20.** Así como es posible validar documentos XHTML, las hojas de estilo también pueden ser sometidas a revisión.*

COMPATIBILIDAD ENTRE NAVEGADORES

Es importante tener en cuenta que apegarse al estándar validado por las organizaciones como W3C permite que los desarrolladores no incurran en la posibilidad de utilizar extensiones generadas por las empresas creadoras de dispositivos para navegar páginas web. En las especificaciones oficiales se listan las propiedades habilitadas y mantenidas por el ente oficial, como vemos a continuación:

III VISIÓN A FUTURO

Por ser consensuadas y no pertenecer de manera directa a una empresa en particular, las tecnologías componentes de los estándares web nos dan la seguridad de que nuestros desarrollos no deberán ser modificados con cada versión nueva. Sin dudas, esto nos brinda la seguridad necesaria para implementar soluciones en nuestros proyectos.

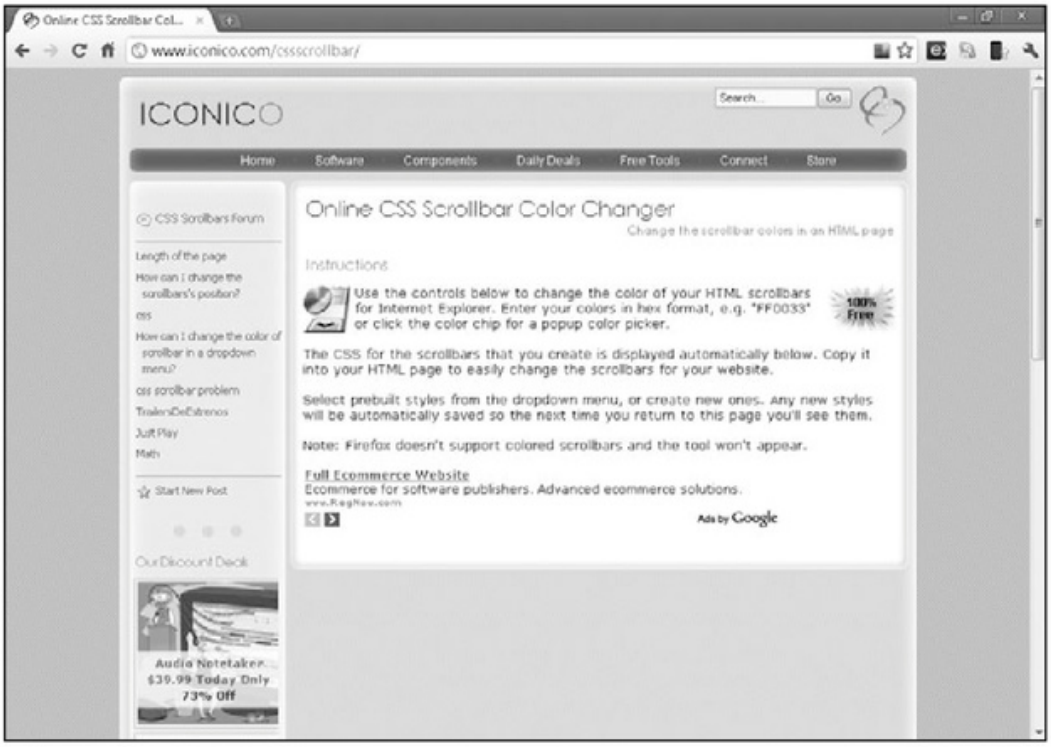


Figura 21. *Internet Explorer incorpora propiedades adicionales a CSS, que en la mayoría de los casos no son aceptadas por los demás navegadores, como es el caso de los estilos aplicados sobre las barras de desplazamiento.*

PROPIEDADES HABILITADAS		
azimuth	direction	padding-bottom
background	display	padding-left
background-attachment	elevation	page
background-color	empty-cells	page-break-after
background-image	float	page-break-before
background-position	font	page-break-inside
background-repeat	font-family	pause
border	font-size	pause-after
border-collapse	font-size-adjust	pause-before
border-color	font-stretch	pitch
border-spacing	font-style	pitch-range

ÚLTIMA VERSIÓN

Es interesante recordar que podemos encontrar el listado completo y actualizado con todas las propiedades (y sus posibles valores) incluidas en la versión 2 de CSS en el sitio oficial de la W3C, más precisamente en la dirección **www.w3.org/TR/REC-CSS2/propidx.html**. Además, allí se incluye la última información sobre el estado de la especificación.

PROPIEDADES HABILITADAS		
border-style	font-variant	play-during
border-top	font-weight	position
border-right	height	quotes
border-bottom	left	richness
border-left	letter-spacing	right
border-top-color	line-height	size
border-right-color	list-style	speak
border-bottom-color	list-style-image	speak-header
border-left-color	list-style-position	speak-numeral
border-top-style	list-style-type	speak-punctuation
border-right-style	margin	speech-rate
border-bottom-style	margin-top	stress
border-left-style	margin-right	table-layout
border-top-width	margin-bottom	text-align
border-right-width	margin-left	text-decoration
border-bottom-width	marker-offset	text-indent
border-left-width	marks	text-shadow
border-width	max-height	text-transform
bottom	max-width	top
caption-side	min-height	unicode-bidi
clear	min-width	vertical-align
clip	orphans	visibility
color	outline	voice-family
content	outline-color	volume
counter-increment	outline-style	white-space
counter-reset	outline-width	widows
cue	overflow	width
cue-after	padding	word-spacing
cue-before	padding-top	z-index
cursor	padding-right	

Tabla 1. *Propiedades habilitadas y mantenidas por la W3C.*

EDITORES CSS

Debemos saber que en el mercado actual hay una serie de editores especialmente dedicados a la creación de hojas de estilo CSS. Es posible encontrar un completo listado con los editores disponibles en la página oficial de la organización W3C, podemos localizarlo en el sitio web que se encuentra en la dirección **www.w3.org/Style/CSS/software**.

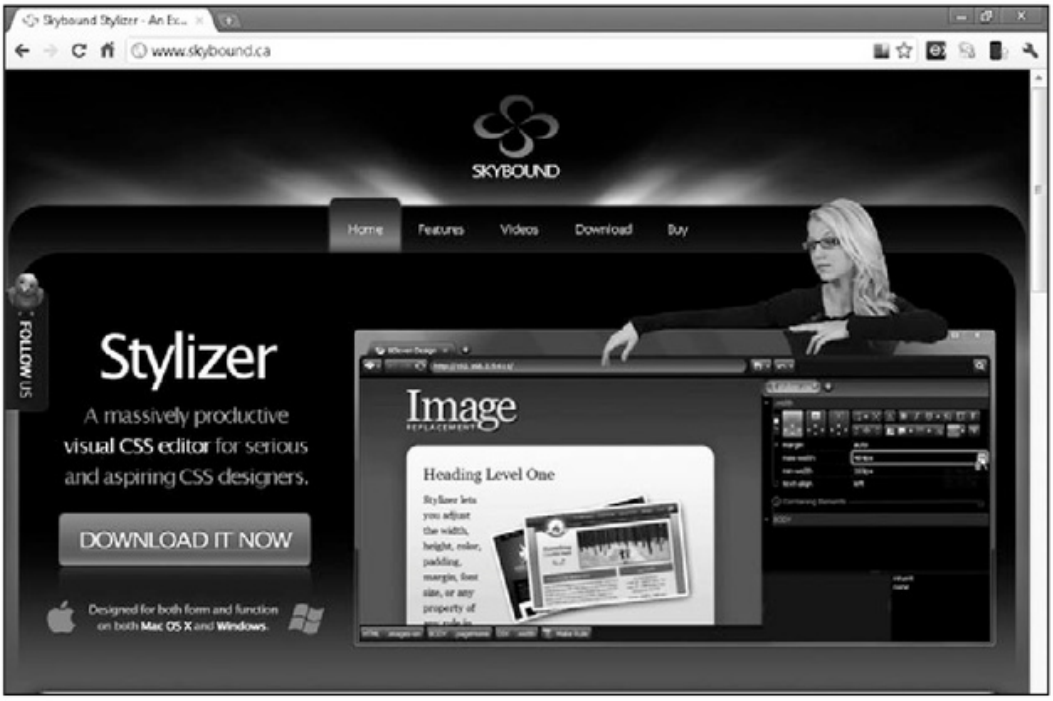


***Figura 22.** Al ser un estándar cerrado, las aplicaciones de edición de las hojas de estilo nos permiten realizar tareas rutinarias de forma rápida y eficaz.*

Otro punto por tener en cuenta es el de las expresiones (**expression**), que permiten incluir sentencias JavaScript en las hojas de estilo para asignarle un determinado valor a una propiedad CSS. Esto solo funciona en Internet Explorer, y es ignorado por el resto de los navegadores, por lo que su utilización no es recomendable.

En el sitio oficial de CSS, podemos encontrar la lista de los navegadores web que en la actualidad brindan soporte para las hojas de estilo en cascada. La última revisión del listado puede ser visitada en **www.w3.org/Style/CSS/software#browsers**.

DISEÑO SIN TABLAS

En un comienzo, las tablas fueron concebidas para mostrar datos de manera tabular, principalmente para listar registros en filas y columnas. HTML proporciona elementos tales como **table**, **tr**, **th** y **td**, entre otros, para incluir tablas en una página. Con el correr del tiempo y dadas las necesidades por parte de los diseñadores, que necesitaban organizar el contenido de las páginas de una manera simple y sin complicaciones, las tablas comenzaron a utilizarse principalmente para estructurar contenidos, y se alejaron así de su objetivo original.
En los últimos tiempos, han surgido cada vez con más fuerza las propuestas para evitar el uso de tablas para estructurar diseños, y en su lugar utilizar CSS para posicionar elementos (en general **div** o **span**, como vimos en el capítulo anterior).

Si bien esta técnica puede demandar más conocimientos por parte del diseñador web, el peso final de la página será mucho menor y el código podrá interpretarse más fácilmente, y se podrá evitar la anidación de tablas. No todos los navegadores interpretan de manera idéntica las propiedades relativas al posicionamiento de elementos, por lo que el responsable de la página deberá estar atento a esta situación y buscar la mejor solución para satisfacer la visualización en los distintos agentes de usuario. Al ser una técnica cada vez más popular, hay abundante bibliografía al respecto que brinda consejos y ejemplos de implementación.
Por otra parte, y al dejar el control en manos de CSS, la presentación en distintos dispositivos será más personalizable, lo que le brindará al autor un control profundo de cada detalle.

... RESUMEN

Una de las partes fundamentales en lo referido a la composición de un sitio web es la que trata sobre el trabajo con hojas de estilo. En este capítulo observamos, a partir de diversas reglas y distintos consejos, cómo disminuir el peso de los documentos lo más posible y cómo evitar que se demore la carga de nuestras páginas por este motivo.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué beneficio tiene incluir hojas de estilo desde documentos externos?

2 ¿Qué significa embeber estilos dentro del documento?

3 ¿Cómo se importa una hoja de estilos?

4 ¿Qué son los estilos en línea?

5 ¿Qué es un dispositivo CSS?

6 ¿En qué parte de la página se ubican los enlaces a las hojas de estilo? ¿Cuál es el beneficio?

7 ¿Qué es la agrupación de estilos?

8 ¿Qué es la herencia en CSS?

9 ¿Cuáles son las opciones para validar documentos?

10 ¿Para qué sirve la sentencia @charset?

EJERCICIOS PRÁCTICOS

1 ¿Cuáles son las ventajas y desventajas de utilizar estilos en línea?

2 Nombre cuatro dispositivos CSS.

3 Dé tres ejemplos de agrupación de estilos.

4 ¿Para qué podría servir la herencia de estilos?

5 ¿Para qué sirve la sentencia @import?

Capítulo 4

JavaScript

En la actualidad y desde hace mucho tiempo las aplicaciones web han tomado a JavaScript como el lenguaje principal para llevar a cabo tareas de propósito general. En este capítulo observaremos los principales aspectos en relación con la interacción entre este lenguaje y los documentos XHTML.

Ubicación de los scripts	**96**
Unificación de archivos	**97**
Minify	97
Reducción de documentos	**104**
JSMin	104
Packer	108
Bananascript.com	109
ShrinkSafe	111
Pack:tag	111
YUI Compressor	112
Escribir menos código con Prototype	**113**
Características y modo de uso	114
Resumen	**129**
Actividades	**130**

UBICACIÓN DE LOS SCRIPTS

Un tema de importancia a tener en cuenta es el de la posición que ocuparán las inclusiones de código JavaScript en nuestros documentos XHTML. Al encontrar un elemento **script**, los navegadores detienen la descarga de los demás elementos de la página y la retoman luego de que el contenido del archivo JS haya sido evaluado por el parser del navegador web. Esto vale tanto para las inclusiones de archivos externos como para el código JavaScript incluido directamente dentro de la página. Por tal motivo, es recomendable incluir esta clase de referencias justo antes del final del documento, lo más cerca posible de la etiqueta **body** de cierre.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 ...

 <script type="text/javascript" src="js/archivo1.js"></script>
 <script type="text/javascript" src="js/archivo2.js"></script>
 <script type="text/javascript" src="js/archivo3.js"></script>

 <script type="text/javascript">
 //<![CDATA[

 //codigo ...

 //]]>
 </script>

 </body>
</html>
```

Otra acción efectiva es utilizar pocos elementos **script** por página. Esto se logra incluyendo código JS en archivos externos, tratando de que sean la menor cantidad posible. A continuación veremos algunas herramientas de utilidad para lograrlo.

UNIFICACIÓN DE ARCHIVOS

Como mencionamos en los capítulos anteriores, cada referencia a un documento JavaScript externo implica una petición extra al servidor, algo que, como el número de requerimientos simultáneos desde un cliente es limitado, demandará más tiempo de procesamiento. Una de las técnicas que se utilizan para minimizar el número de peticiones a documentos JavaScript consiste en ubicar todos los scripts en un solo archivo, o por lo menos tratar de utilizar la menor cantidad posible. Más adelante veremos en este mismo capítulo cómo utilizar herramientas que permiten automatizar la tarea de reunir fragmentos de código en un mismo documento, mediante lenguajes de programación del lado servidor.

Minify

Esta herramienta nos permite minimizar y combinar múltiples documentos JavaScript (también funciona para archivos **CSS**) y necesitaremos contar con PHP versión 5.2.1 o superiores para utilizarla. Podemos descargar la última versión disponible de su sitio web oficial, **http://code.google.com/p/minify**. Una vez hecho esto, deberemos descomprimir la distribución a un directorio accesible en el servidor web desde un navegador.

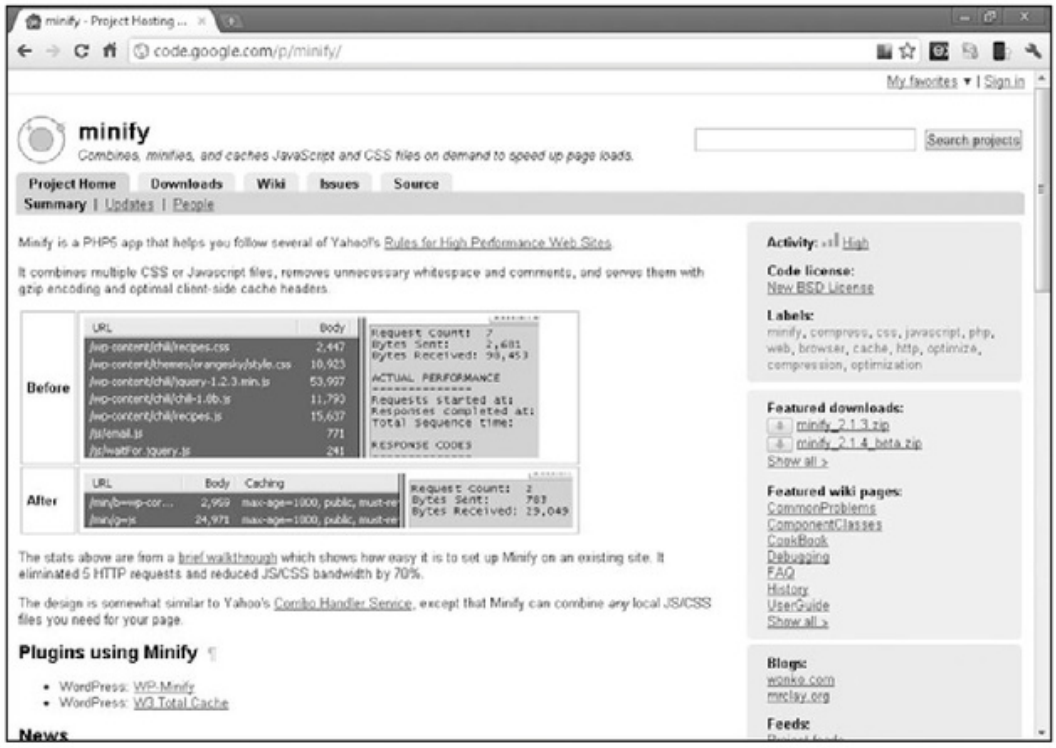


Figura 1. *Minify nos permite realizar dos tareas a la vez: minimizar y juntar archivos CSS y JS.*

En las últimas versiones han surgido cambios en cuanto a la forma de utilizar **Minify**. Estos son los pasos por seguir para utilizar esta herramienta:

- Descargar desde **https://code.google.com/p/minify/downloads/list**.
- Descomprimir y copiar el directorio **min** a nuestro sitio.
- Si navegamos hasta allí (esto puede variar según la configuración del sistema, pero la dirección podría ser algo como **http://localhost:8080/ejemplo-minify/min**); allí veremos un formulario que nos ayudará a generar el código para insertar en nuestras páginas. Luego, completamos la información de los archivos que necesitamos comprimir y, presionando el botón **Update**, generamos el código. Como el resultado será distinto según el tipo de archivo (CSS o JS), deberemos realizar este proceso por partes (en una subimos los CSS, generamos el código, copiamos y pegamos en el HTML, y luego lo mismo, pero esta vez con los JS).

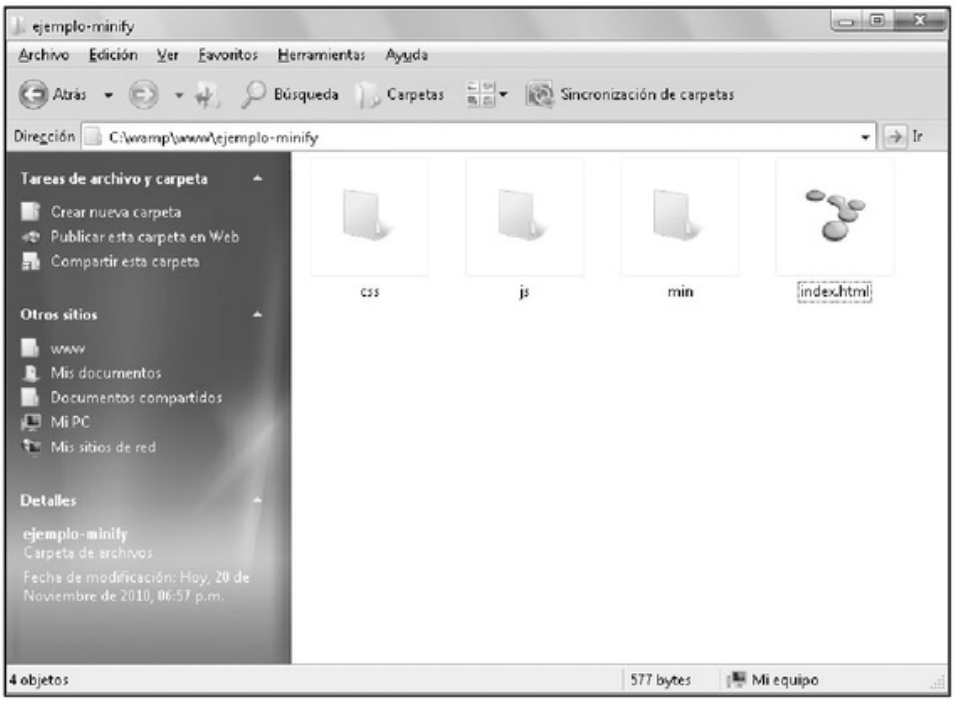


Figura 2. *Para utilizar Minify, contamos con una interfaz web, aunque podemos llevar a cabo el mismo proceso manualmente.*

El código generado es algo como lo siguiente:

```
<link type="text/css" rel="stylesheet" href="min/f=/ejemplo-
    minify/css/styles1.css,/ejemplo-minify/css/styles2.css,/ejemplo-
    minify/css/styles3.css" />
```

```
<script type="text/javascript" src="min/f=/ejemplo-
    minify/js/archivo1.js,/ejemplo-minify/js/archivo2.js,/ejemplo-
    minify/js/archivo3.js"></script>
```

Nuestro código inicial ahora se verá de la siguiente manera:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>

  <link type="text/css" rel="stylesheet" href="min/f=/ejemplo-
     minify/css/styles1.css,/ejemplo-minify/css/styles2.css,/ejemplo-
     minify/css/styles3.css" />

 </head>
 <body>

 ...

  <script type="text/javascript" src="min/f=/ejemplo-
     minify/js/archivo1.js,/ejemplo-minify/js/archivo2.js,/ejemplo-
     minify/js/archivo3.js"></script>

 </body>
</html>
```

Es importante saber que, dentro del archivo denominado **config.php**, es posible que definamos el valor deseado para algunas constantes que se encargará de modificará el comportamiento de la aplicación desarrollada.
Debemos tener en cuenta que una constante nos sirve para proceder a definir el directorio caché que será utilizado para la aplicación:

```
$min_cachePath = 'c:\\WINDOWS\\Temp';
```

Éste es un extracto del código CSS generado:

```
body{font-size:12px;color:#000;background-
     color:#fff;margin:0px;overflow:auto;background-
     image:url('/images/gradientfromtop.gif');background-repeat:repeat-
```

```
     x}body,p,h1,h2,h3,h4,table,td,th,ul,ol,textarea,input{font-
     family:verdana,helvetica,arial,sans-serif}h1{font-size:200%;margin-
     top:0px;font-weight:normal}h2{font-size:160%;margin-top:10px;margin-
     bottom:10px;font-weight:normal}h3{font-size:120%;font-
     weight:normal}h4{font-size:100%}h5{font-size:90%}h6{font-
     size:80%}h1,h2,h3,h4,h5,h6{background-

color:transparent;color:#000}iframe{margin:0px}div{width:100%}table,th,td,in
     put,textarea{font-size:100%}table.example{color:#000;background-
     color:#e5eecc;padding-top:8px;padding-bottom:8px;padding-
     left:10px;padding-right:10px;border:1px
solid #d4d4d4;background-image:url('/ejemplo-
     minify/css/images/bgfadegreen.gif');background-repeat:repeat-
     x}table.example_code{background-color:#fff;padding:4px;border:1px
solid #d4d4d4}table.example_code
td{font-size:110%;font-family:couriernew}table.code{outline:1px
solid #d4d4d4;border:5px
solid #e5eecc}table.code
td{font-size:110%;font-family:couriernew;background-color:#FFF;border:1px
solid #d4d4d4;padding:4px}table.example_code p,table.code p,td.code{font-
     family:couriernew}table.lamp{width:100%;padding:0px;border:1px
solid #d4d4d4;border-collapse:collapse}table.lamp
th{color:#000;background-color:white;padding:0px;border-right:1px solid
     #d4d4d4}table.lamp
td{color:#000;background-color:#e5eecc;padding:4px;padding-
     left:10px;background-image:url('/ejemplo-
     minify/css/images/bgfadegreen.gif');background-repeat:repeat-
     x}table.reference{border:1px
solid #c3c3c3;border-collapse:collapse}table.reference
th{background-color:#e5eecc;border:1px
solid #c3c3c3;padding:3px;vertical-align:top}table.reference
```

III VERSIONES MINIMIZADAS

Es importante tener en cuenta que cada vez son más las librerías y aplicaciones que incluyen en sus distribuciones versiones minimizadas o reducidas de sus archivos JavaScript. Como caso de ejemplo, podemos citar al cada vez más popular editor WYSIWYG **TinyMCE** (que encontramos en la dirección web **http://tinymce.moxiecode.com**).

Y a continuación, tenemos la posibilidad de ver un pequeño extracto correspondiente al código JS que se ha generado:

```
function AC_AddExtension(src,ext)
{if(src.indexOf('?')!=-1)
return src.replace(/\?/,ext+'?');else
return src+ext;}

function AC_Generateobj(objAttrs,params,embedAttrs)
{var str='<object ';for(var i in objAttrs)

str+=i+'="'+objAttrs[i]+'" ';str+='>';for(var i in params)
str+='<param name="'+i+'" value="'+params[i]+'" /> ';str+='<embed ';for(var
     i in embedAttrs)
str+=i+'="'+embedAttrs[i]+'" ';str+='
     ></embed></object>';document.write(str);}

function AC_FL_RunContent(){var ret=AC_GetArgs
(arguments,".swf","movie","clsid:d27cdb6e-ae6d-11cf-96b8-
     444553540000","application/x-shockwave-
     flash");AC_Generateobj(ret.objAttrs,ret.params,ret.embedAttrs);}
```

Figura 3. *Aunque no es el único lenguaje que soporta esta técnica, la minizacion se aplica generalmente sobre código JavaScript.*

REDUCCIÓN DE DOCUMENTOS

Además de Minify, en la actualidad existen múltiples herramientas para reducir el tamaño de los archivos JavaScript sin perder la funcionalidad de éstos. La reducción se logra al eliminar aquellas características que no modifican el correcto funcionamiento del script, pero que ocupan espacio, por ejemplo comentarios, tabulaciones o líneas nuevas. Este procesamiento nos entrega la posibilidad de que la carga de documentos sea más rápida que lo habitual.
A continuación, veremos algunas de las opciones que se encuentran disponibles para llevar a cabo estas tareas de manera rápida y automatizada.

JSMin

Esta herramienta de filtrado elimina los saltos de línea, las tabulaciones (reemplazadas por espacios simples) y los comentarios (simples y de múltiples líneas). Todo esto mantiene el correcto funcionamiento del script original. Podemos obtener más información acerca de esta herramienta en **http://crockford.com**.
Debemos saber que JSMin está disponible para múltiples lenguajes de programación, entre los que podemos encontrar los siguientes:

- C (**http://crockford.com/javascript/jsmin.c**).
- C# (**http://crockford.com/javascript/jsmin.cs**).
- Java (**http://inconspicuous.org/projects/jsmin/jsmin.java**).
- JavaScript (**http://fmarcia.info/jsmin**).
- Perl (**http://search.cpan.org**).
- PHP (**https://github.com/rgrove/jsmin-php**).
- Python (**http://crockford.com/javascript/jsmin.py.txt**).
- Ruby (**http://crockford.com/javascript/jsmin.rb**).

Incluso podemos descargar la ejecutable desde el sitio web oficial de la herramienta. En todos los casos las conversiones son en un solo sentido, lo que significa que, a partir de un documento, podemos obtener una versión reducida, pero no es posible lo contrario (a partir de un documento reducido obtener el original).

PHP

En algunos de los ejemplos de minimización y unificación de documentos JavaScript optamos por utilizar las versiones PHP coresponidentes a las librerías, y esto se debe a que en los últimos tiempos este lenguaje se ha convertido en uno de los más utilizados y estables. Además, la mayoría de los servicios de alojamiento web lo admite.

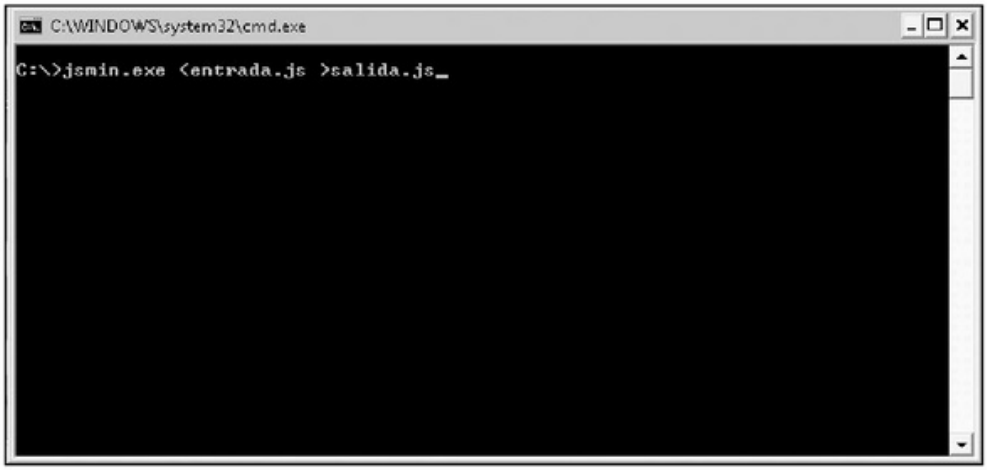


Figura 3. *Existen diferentes alternativas para acceder a la funcionalidad de JSMin, y una de ellas es mediante la línea de comandos del sistema operativo.*

Los mismos desarrolladores de esta aplicación proveen una herramienta que permite validar el código JavaScript (según parámetros establecidos por JSMin, ver **www.jslint.com/lint.html** para más información) antes de minimizarlo. Esto nos da la posibilidad de realizar la compresión de manera segura y sin errores. El programa se denomina **JSLint** y podemos descargarlo de su sitio web oficial.

Figura 4. *JSLint nos permite preparar el código JavaScript antes de ser comprimido por JSMin (**www.jslint.com**).*

Si tomamos como ejemplo la distribución de PHP, el primer paso que debemos realizar es proceder a incluir en nuestro script el archivo que corresponde a JSMin, esto lo realizamos de la siguiente forma:

```
require ‘jsmin-1.1.1.php’;
```

Luego, con el método **Minify** realizamos la conversión:

```
JSMin::minify($codigoArchivoOriginal);
```

Veamos un ejemplo práctico de utilización:

```
<?php

require 'jsmin-1.1.1.php';

$archivoOriginal = 'archivo.js';

$archivoComprimido = JSMin::minify(file_get_contents($archivoOriginal));

file_put_contents('jsmin_'.$archivoOriginal, $archivoComprimido);

?>
```

Debemos tener en cuenta que la necesidad de mantener el documento original se debe a que es muy difícil visualizar el resultado comprimido para realizar modificaciones en el código, cada vez que lo necesitemos.
En la siguiente imagen, tenemos la posibilidad de ver la forma en que luciría un archivo original, antes de realizar la compresión:

III MANTENER LOS ARCHIVOS ORIGINALES

Las herramientas vistas en esta sección reciben como entrada un documento JavaScript y se encargan de devolver la versión reducida, que generalmente es poco comprensible a simple vista por parte del desarrollador. Ante esto es recomendable mantener siempre la versión original y generar la compresión desde estos archivos.

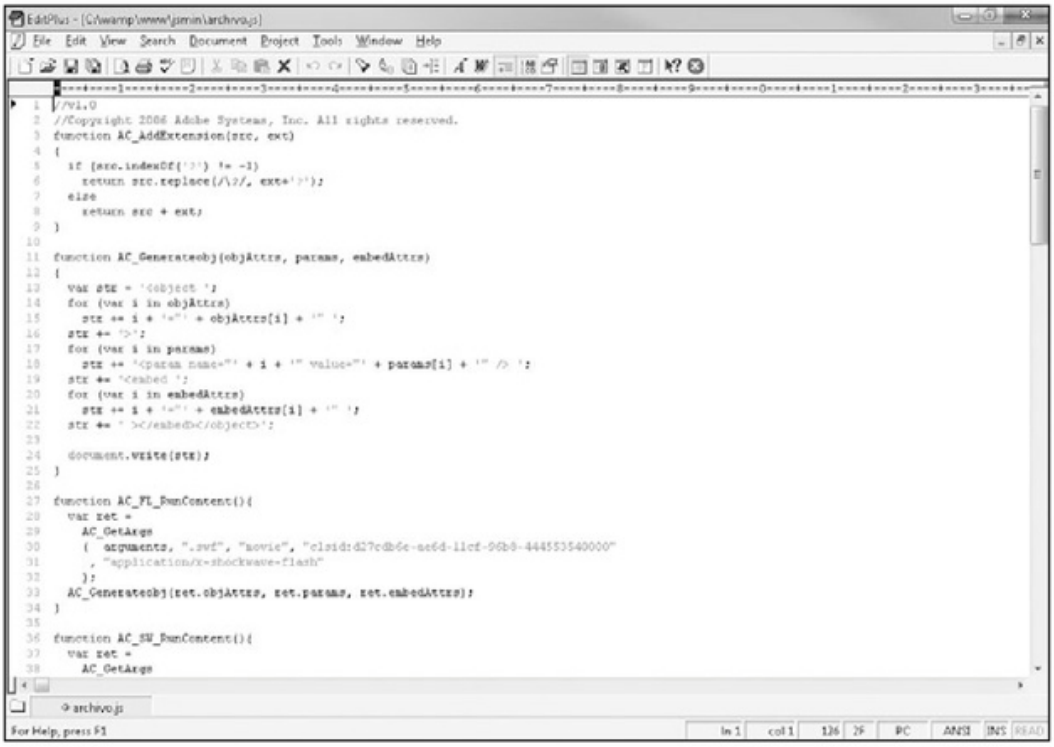


Figura 5. *Por un lado, la minimización provoca el ahorro de bytes y por el otro crea complejidad para comprender el documento a simple vista.*

Podemos darnos cuenta de que el resultado del archivo luego de realizar la compresión es muy distinto al contenido original. En la imagen que se muestra a continuación, podemos ver el archivo comprimido:

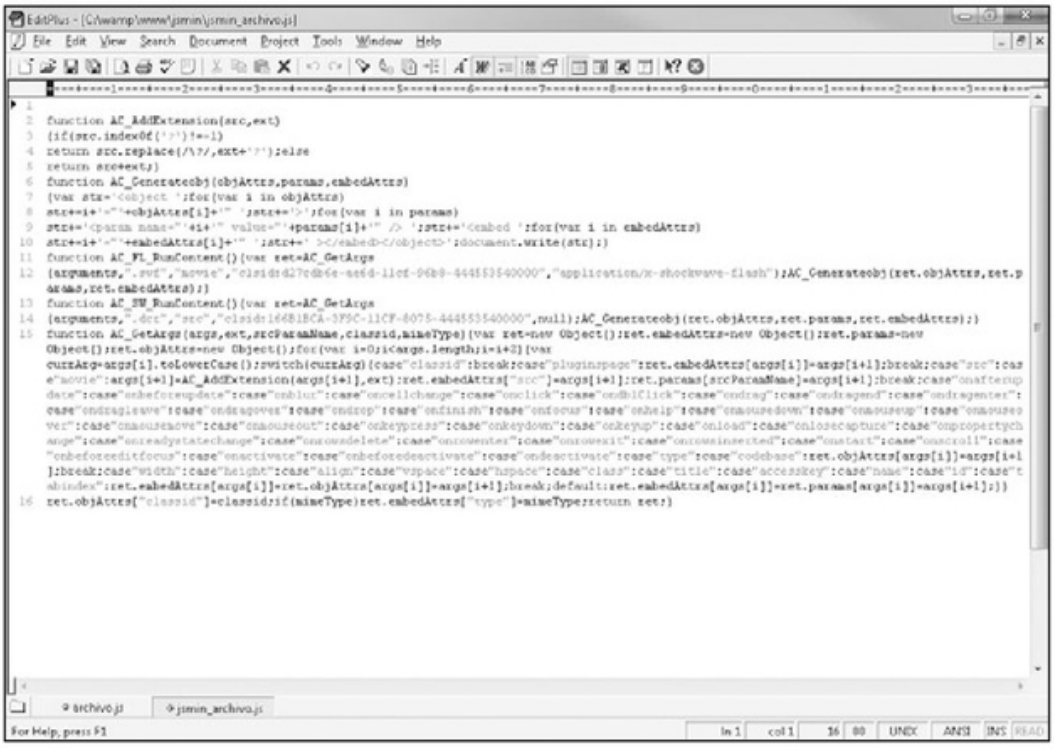


Figura 6. *Las técnicas de compresión suelen aplicarse en la etapa de producción de aplicaciones web.*

Lo que usualmente se hace al trabajar con este tipo de herramientas es aplicar los cambios necesarios en el original y luego volver a realizar el proceso de conversión. La distribución de JSMin requiere que contemos con PHP versión 5 o superiores.

Packer

Esta herramienta se encuentra disponible para ser utilizada desde distintos lenguajes y entornos, entre los que se encuentran:

- Plataforma **.NET** (requiere **Microsoft .NET Framework version 1.1**).
- **Perl**.
- **WSH** (*Windows Scripting Host*).
- **PHP** (requiere PHP versión 5 o superiores aunque en el sitio web hay disponible un versión para PHP 4, que también viene incluida en la distribución oficial).

Podemos descargar cualquiera de las opciones del sitio web oficial de la herramienta de **http://dean.edwards.name/download/#packer**.
Además de minimizar el tamaño de los scripts, Packer nos brinda la posibilidad de ofuscar el código de éstos. Ofuscación en este contexto significa modificar la visualización del código fuente haciéndolo más oscuro y difícil de interpretar para cualquiera, incluso para el propio desarrollador.
En el caso de utilizar PHP para comprimir nuestros scripts, debemos incluir el archivo **class.JavaScriptPacker.php**, crear un objeto **JavaScriptPacker** y por último invocar el método **pack** para obtener el resultado. Veamos un ejemplo:

```
<?php

require 'class.JavaScriptPacker.php';

$archivoOriginal = 'nombreArchivo.js';
$packer = new JavaScriptPacker($archivoOriginal);
```

III TIEMPOS

Es importante saber que la tarea de minimizar un documento implica que perderemos la posibilidad de recorrerlo visualmente en busca de los errores que puedan presentarse o solo para aplicarle modificaciones. Por este motivo, lo que se estila es minimizar los archivos luego de haber corregido y probado su funcionalidad por completo.

```
$archivoComprimido = $packer->pack();

file_put_contents('packer_'.$archivoOriginal, $archivoComprimido);

?>
```

El constructor denominado **JavaScriptPacker** admite otros argumentos adicionales como parámetros en su ejecución, además de la ruta al documento JavaScript: el tipo de codificación (**None**, **Numeric**, **Normal** o **High ASCII**), la posibilidad de recuperar el archivo original a partir del codificado (**true** o **false**, encontramos más información en la documentación oficial), y la utilización de caracteres especiales con algún significado para el lenguaje (**true** o **false**).

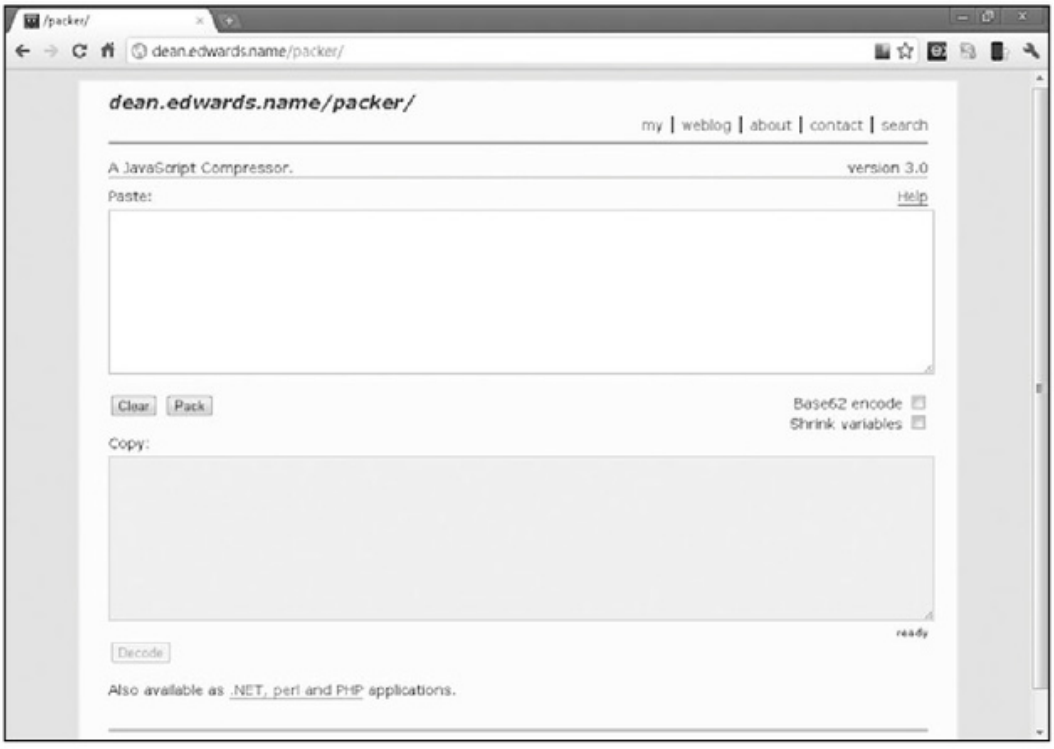


***Figura 7.** Packer mantiene una versión en línea desde la cual podemos comprimir nuestro código JavaScript.*

III ACCESO MEDIANTE SERVICIOS WEB

Debemos saber que dentro de las múltiples opciones disponibles para utilizar la aplicación de minimización de código JavaScript Packer podemos sumarle la alternativa de acceder a su funcionalidad mediante un servicio alojado en la siguiente dirección: **www.webserviceshare.com/computers/programming/tools/service/Trynt-Javascript-Compression.htm**.

Se trata de una herramienta en pleno desarrollo, por lo que habrá que tener cierta precaución al momento de emplearla en nuestros proyectos. Podemos obtener más información en su sitio web oficial: **http://dean.edwards.name/**.

Bananascript.com

Se trata de una interesante y útil herramienta a la que se puede acceder en el sitio web que se encuentra en la dirección **www.bananascript.com** y nos permite comprimir archivos JavaScript en línea de una forma muy sencilla.

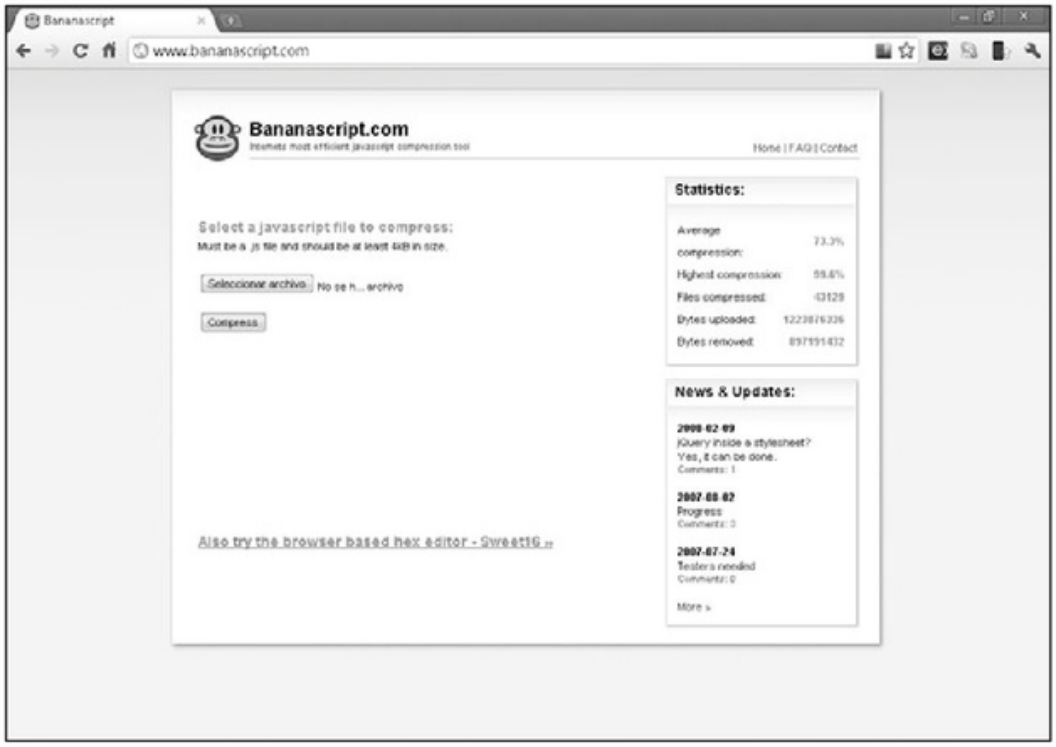


***Figura 8.** Los archivos ingresados deberán tener un peso superior a 4KB.*

La forma de funcionamiento es simple: en un formulario cargamos el documento a compactar y presionamos el botón **Compress**. Una vez hecho esto, podremos observar un resumen del proceso: el porcentaje de reducción, los bytes ahorrados en relación con el original y el link para bajar la nueva versión.

III OPCIONES

Entre las herramientas para minimizar código podemos distinguir dos tipos: las que funcionan online, es decir a través de un formulario web, y aquellas que podemos descargar e instalar en nuestros equipos. La decisión entre unas y otras tal vez tenga que ver con la cantidad de archivos que necesitemos procesar.

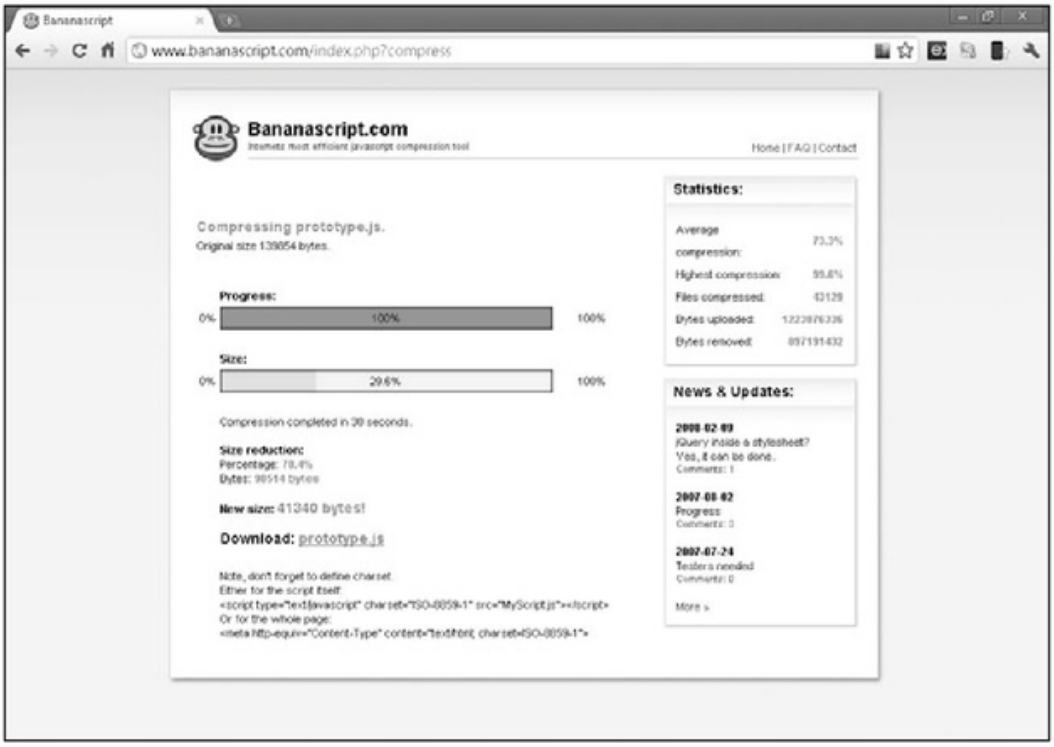


Figura 9. *La interfaz de acceso a* ***Bananascript*** *es simple e intuitiva.*

Como nota adicional, en el resumen de cada proceso se insta a definir el juego de caracteres para incluir el documento generado:

```
<script type="text/javascript" charset="ISO-8859-1"
   src="MyScript.js"></script>
```

O para la página completa:

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

El sitio maneja estadísticas acerca de los grados de compresión alcanzados, y, en promedio, un archivo comprimido llega al 72% si tomamos como parámetro el original.

ShrinkSafe

Esta herramienta forma parte de **Dojo Toolkit**, una librería popular de propósito general para desarrollos en JavaScript. Podemos obtener más información en su sitio web oficial: **http://dojotoolkit.org**.

Es posible acceder a la funcionalidad de **ShrinkSafe** mediante lenguaje Java, si descargamos la última versión de **http://shrinksafe.dojotoolkit.org**.

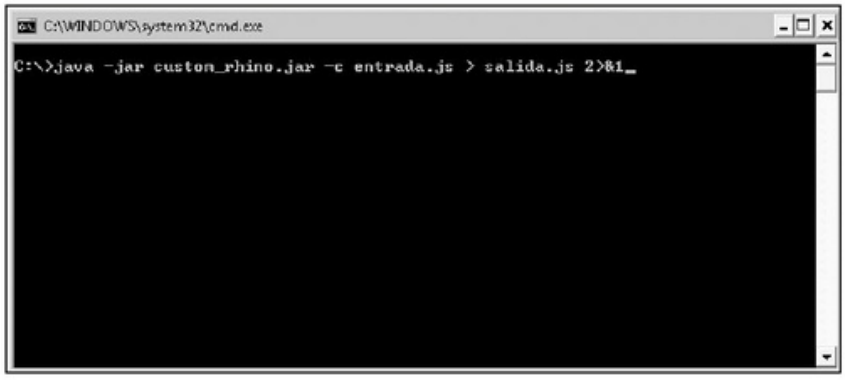


Figura 10. *Para utilizar* ***ShrinkSafe*** *debemos tener la plataforma Java previamente instalada.*

La instrucción para llevar a cabo la compresión es la siguiente:

```
java -jar custom_rhino.jar -c entrada.js > salida.js 2>&1
```

Podemos obtener más información acerca de esta herramienta en el sitio web que se encuentra en **http://dojotoolkit.org/docs/shrinksafe**.

Pack:tag

Esta opción funciona con el lenguaje **JSP** (*Java Server Pages*) y podemos descargar la última versión disponible del sitio web **http://sourceforge.net/projects/packtag**.

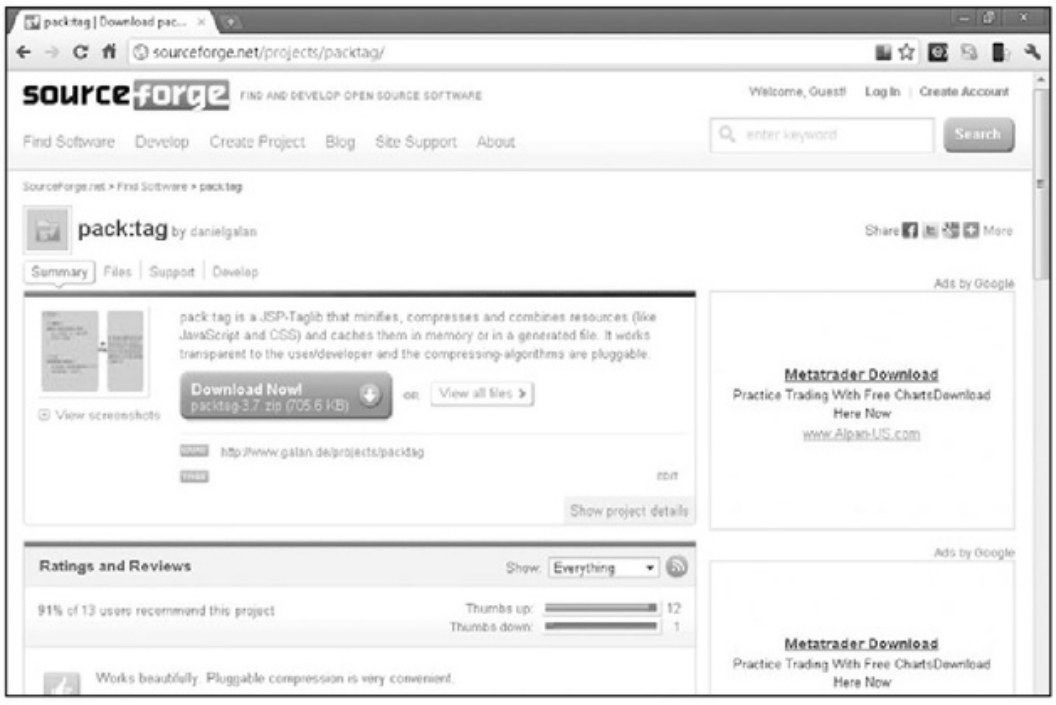


Figura 11. *Pack:tag permite un acceso preciso a la compresión de archivos mediante la utilización del lado servidor para desarrollar el proceso.*

También funciona con hojas de estilo CSS y además permite la combinación de llamadas para minimizar el número de peticiones HTTP. Admite la utilización de caché para volver a utilizar los procesamientos previos.

YUI Compressor

YUI (*Yahoo User Interface*) es un proyecto llevado adelante por la compañía **Yahoo!** (**www.yahoo.com**), que provee controles y servicios orientados al trabajo diario de los desarrolladores de sitios web. Una de las herramientas puestas a disposición es YUI Compressor, una aplicación escrita en Java (para versiones 1.4 o superiores), que permite comprimir tanto código JavaScript como hojas de estilo CSS.

```
$ java -jar yuicompressor.jar archivo.js

$ java -jar yuicompressor.jar archivo.css
```

Con la opción **–o nombreArchivo**, tenemos la posibilidad de enviar la salida generada a un documento externo, de la siguiente forma:

```
$ java -jar yuicompressor.jar archivo.js –o salida.js
```

```
$ java -jar yuicompressor.jar archivo.css –o salida.css
```

Debemos tener en cuenta que la aplicación denominada YUI Compressor es un proyecto de código abierto, por lo tanto, tenemos la posibilidad de acceder, revisar y modificar su código fuente en el caso de que sea necesario. Es importante saber que es posible obtener información adicional acerca de esta herramienta si visitamos el sitio oficial de Yahoo!, el cual se encuentra en la página accesible desde la dirección: **http://developer.yahoo.com/yui/compressor**.

III MAS OPCIONES

Yui provee controles sumamente útiles a la hora de desarrollar aplicaciones web, por ejemplo acordeones, editores WYSIWYG, fecha, hora, reproductores de video, slideshows, charts, interacción con Twitter, aumento y disminución de fuentes, JSON y formularios, entre muchos otros.

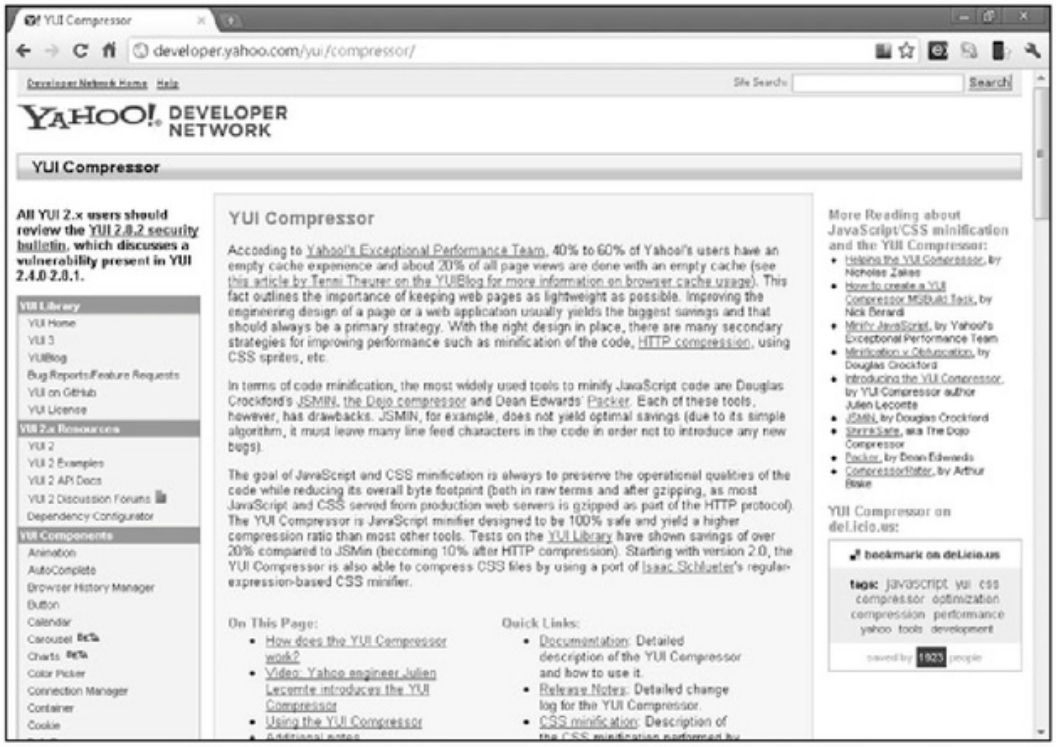


Figura 12. *YUI Compressor es el minimizador oficial de la empresa Yahoo! y está desarrollado en Java.*

ESCRIBIR MENOS CÓDIGO CON PROTOTYPE

Como vimos en apartados anteriores, muchas de las premisas para optimizar código están relacionadas con disminuir el tamaño de los archivos. Si la aplicación cuenta con una gran cantidad de código JavaScript podemos recurrir a **Prototype**, una librería que permite enmascarar algunas características ya definidas en el lenguaje haciéndolas más cortas, directas e intuitivas. Es decir que básicamente podemos definir sentencias JavaScript escribiendo una menor cantidad de código. Al mismo tiempo, agrega nuevas características. Por ejemplo, extiende la funcionalidad de ciertos objetos base como **Object**, **Array**, **Number** y **String**. Además de extender el lenguaje, Prototype permite anular las diferencias de implementación entre los distintos navegadores.

III RIVAL

En los últimos tiempos ha surgido una competencia para con Prototype: se trata de jQuery, una herramienta muy extendida en la actualidad y sobre la cual podemos encontrar una enorme cantidad de plugins que amplían su funcionalidad base.

Podemos descargarla si visitamos el sitio web oficial, que se encuentra en la dirección **www.prototypejs.org**. En los siguientes apartados veremos la forma en que podemos incluir Prototype en nuestros proyectos y también analizaremos cómo utilizar algunas de sus funcionalidades más atractivas.

Características y modo de uso

Es importante saber que Prototype es una librería JavaScript orientada a objetos. Es una herramienta de propósito general y entre las características más destacadas de esta biblioteca se encuentran las siguientes:

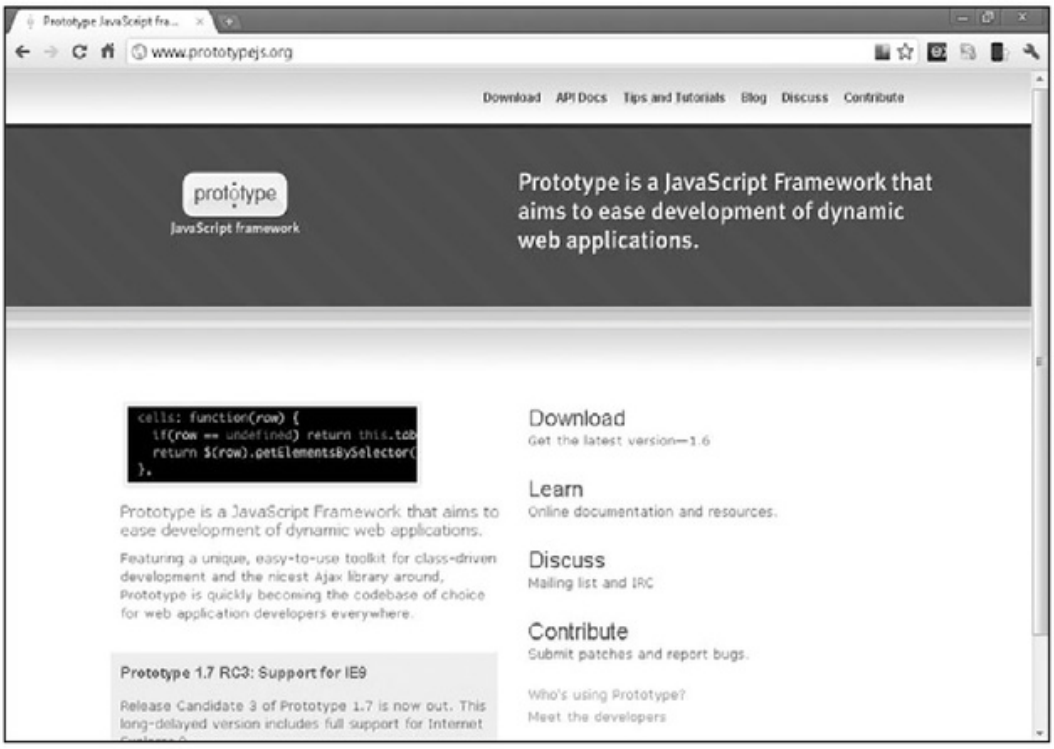


Figura 13. *Prototype es una de las opciones más profesionales y versátiles en lo relacionado con el desarrollo con JavaScript.*

- Soporte para trabajo en Ajax.
- Procesamiento de matrices.
- Referencia a elementos y enlace a eventos.
- Implementación DOM.
- Iteración de colecciones de elementos.
- Tratamiento de eventos.
- Trabajo con formularios y sus respectivos controles.
- Manipulación de arrays asociativos.
- Posicionamiento de elementos dentro de la página.
- Rangos de valores.
- Tratamiento de cadenas de caracteres.

Uno de los fines más importantes es lograr funciones de propósito general y de uso común, de esta forma podremos evitar la necesidad de lidiar con las particularidades de cada navegador y así facilitaremos la escritura de código JavaScript.
Debemos saber que la herramienta Prototype es totalmente compatible con el siguiente listado de navegadores web:

- **Microsoft Internet Explorer** para Windows desde la versión 6.0.
- **Mozilla Firefox** 1.5 y superiores.
- **Safari** 2.0 y superiores.
- **Opera** 9.25 y superiores.

Para que podamos incluir Prototype en nuestros proyectos simplemente debemos enlazar el archivo **prototype.js**, que podemos descargar del sitio oficial, a un documento XHTML de la siguiente manera:

```
<script type="text/javascript"

src="prototype.js"></script>
```

La implementación DOM de JavaScript permite obtener una referencia a un elemento del documento mediante el método denominado **getElementById**. Con la función **$**, Prototype se encarga de proveer un comportamiento similar, solo que puede recibir más de un argumento (más de un ID):

```
$("elemento1")
```

```
$("elemento1", "elemento2", "elementoN")
```

Veamos un ejemplo:

III PROTOTYPE

El desarrollador de la librería Prototype es **Sam Stephenson** (**www.conio.net**), es recomendable trabajar con su última versión que se encuentra disponible. Para que podamos saber cuál estamos empleando, se puede recurrir a la propiedad denominada **Version** del objeto **Prototype** o simplemente visualizar el archivo **prototype.js**.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  function mostrarContenido(id) {
    alert($(id).innerHTML);
  }

  //]]>
  </script>
 </head>
 <body>

  <div id="elemento1" onClick="mostrarContenido(this.id)">
  contenido del elemento 1
  </div>

  <div id="elemento2" onClick="mostrarContenido(this.id)">
  contenido del elemento 2
  </div>

  <div id="elemento3" onClick="mostrarContenido(this.id)">
  contenido del elemento 3
  </div>

 </body>
</html>
```

De manera similar, existe la función **$$**, con la que se obtienen referencias a los elementos que coincidan con la expresión CSS dada como parámetro:

```
$$("div #nombreClase");
```

Devuelve una lista de elementos. Veamos un ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  function cambiarEstado(nombreClase) {
    var lista = $$(nombreClase);
    lista.each(function(elemento){
       if (!elemento.style.color) {
         elemento.style.color = '#990000';
       } else {
         elemento.style.color = '';
       }
    });

  }

  //]]>
  </script>
 </head>
 <body>

  <div class="menu">
```

III INTERACCIÓN

Prototype viene incluida en la distribución oficial del framework llamado **Ruby on Rails** (que se encuentra en la dirección web **www.rubyonrails.org**) y, para hacer uso de ella, solo habrá que incluir la referencia en las vistas. Además sirve como base para otras librerías, como **Script.aculo.us** (**http://script.aculo.us**) y **Mochikit** (**www.mochikit.com**).

```
    contenido del elemento
    </div>
    <div class="menu">
    contenido del elemento
    </div>

    <div class="menu2">
    contenido del elemento
    </div>

    <div class="menu">
    contenido del elemento
    </div>

    <br />

    <input type="button" onclick="cambiarEstado('.menu')" value="Cambiar
       color" />

  </body>
</html>
```

Figura 14. *Script.aculo.us es una de las librerías más populares que basa su funcionamiento en Prototype.*

En el ejemplo anterior vimos cómo recorrer la lista devuelta por **getElementsByClassName** con el método **each**:

```
lista.each(function(elemento){
   ...
   ...
   ...
}
```

Prototype permite recorrer colecciones de elementos como si fuesen listas, utilizando características del objeto **Enumerable**, cuyos métodos son los siguientes:

- **each**: llama a la función de iteración y le pasa como parámetros el valor del elemento actual y la posición que ocupa.
- **all**: devuelve verdadero solo si todos los elementos son distintos de falso.
- **any**: devuelve verdadero si alguno de los elementos es distinto de falso.
- **collect**: ubica cada elemento de la colección en un array.
- **detect**: devuelve el primer elemento no falso de la colección.
- **entries**: para obtener más información podemos ver **toArray**.
- **find**: para más información podemos ver **detect**.
- **findAll**: devuelve un array con todos los elementos no falsos de la colección.
- **grep**: se encarga de recibir una expresión regular y nos devuelve todos los elementos que se encuentren en la colección y que coincidan.
- **include**: recibe como argumento un objeto y lo busca en la colección.
- **inject**: combina todos los elementos de la colección.
- **invoke**: por cada elemento de la colección aplica el método dado en el primer argumento. Los siguientes argumentos corresponden a los parámetros del método.
- **map**: para más información podemos ver **collect**.
- **max**: devuelve el elemento mayor de la colección.
- **member**: para más información podemos ver **include**.
- **min**: devuelve el elemento menor de la colección.
- **partition**: nos devuelve dos arrays: uno de ellos contiene los elementos no falsos y el otro corresponde al listado de los restantes.
- **pluck**: este método recibe como argumento una propiedad y nos devuelve el valor de cada elemento correspondiente a la colección.
- **reject**: devuelve un array con todos los elementos falsos de la colección.
- **select**: para más información podemos ver **findAll**.
- **sortBy**: ordena los elementos de la colección.
- **toArray**: devuelve un array con todos los elementos de la colección.
- **zip**: recibe un número variable de colecciones. En la primera posición de la primera

colección crea un array que contiene la primera colección. En la segunda posición de la primera colección, se encarga de crear un array que contiene la segunda colección, y así continúa con las demás.

Prototype define un objeto denominado **Hash**, que podría compararse con un array tradicional solo que con la posibilidad de incluir claves no numéricas. Un **Hash** posee los siguientes métodos:

- **keys**: devuelve un array con las claves del **Hash**.
- **values**: devuelve un array con los valores del **Hash**.
- **merge**: se encarga de recibir como argumento otro **Hash** y de esta forma nos devuelve la concatenación correspondiente.
- **toQueryString**: imprime los valores y forma un **querystring**.
- **inspect**: devuelve una representación en formato de cadena.

La función **$H** convierte el objeto recibido como argumento en un array asociativo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  var objeto = new Object();
  objeto.propiedad1 = 717;
  objeto.propiedad2 = 'Cadena';

  var hash = $H(objeto);

  alert(hash.inspect());

  //]]>
  </script>
 </head>
```

```
  <body>

  ...

  </body>
</html>
```

Prototype también amplía la implementación DOM original de JavaScript. Entre los métodos disponibles, se encuentran los siguientes:

- **addClassName**: se encarga de realizar la asignación a un elemento (primer argumento) una clase CSS (segundo argumento).
- **classNames**: recibe la referencia de un elemento y devuelve las clases CSS vinculadas.
- **cleanWhitespace**: quita los espacios del valor del argumento.
- **empty**: devuelve verdadero si el elemento recibido está vacío y falso, si no.
- **getDimensions**: devuelve las dimensiones del elemento recibido como argumento (objeto con dos propiedades: **height** y **width**).
- **getHeight**: devuelve el valor de la propiedad **offsetHeight** del elemento recibido.
- **getStyle**: devuelve el valor de la propiedad CSS (segundo argumento) del elemento dado en el primer argumento.
- **hasClassName**: recibe un elemento y una clase, y devuelve verdadero si corresponden.
- **hide**: recibe elementos separados por comas y los oculta.
- **makePositioned**: recibe un elemento y pone la propiedad **style.position** a **relative**.
- **remove**: quita el elemento pasado del documento.
- **removeClassName**: recibe un elemento y una clase CSS, y elimina la relación.
- **scrollTo**: posiciona la ventana a la altura del elemento que hemos entregado como argumento. Es como mover la barra de scroll.
- **setStyle**: se encarga de recibir como argumento un elemento y un **Hash** con propiedades CSS, y luego se las asigna.
- **show**: se trata de lo opuesto al método **hide**.

SUMA

Más allá de que desarrollar aplicaciones con Prototype puede ayudarnos a disminuir el tamaño de nuestros scripts, siempre hay que tener en cuenta que la biblioteca base ocupa un espacio (124 KB sin comprimir y aproximadamente 92 KB luego de ser minimizada). Las ventajas de su utilización bien pueden equilibrar la carga extra.

- **toggle**: si un elemento está en modo **hide**, lo pasa a **show**, y viceversa.
- **undoPositioned**: recibe un elemento y setea la propiedad **style.position** a vacío.
- **update**: le asigna un contenido (**innerHTML**) a un elemento (primer argumento).
- **visible**: si el elemento recibido es visible devuelve verdadero.

JavaScript posee un objeto **Array** y Prototype lo expande agregándole más funcionalidades, las cuales analizaremos a continuación:

- **clear**: se encarga de vaciar el array correspondiente.
- **compact**: devuelve un array sin los elementos nulos o indefinidos del original.
- **first**: devuelve el valor de la primera posición.
- **flatten**: devuelve un array de una dimensión y ubica los elementos de los subarrays en el primer nivel. Se aplica array de más de un nivel.
- **indexOf**: recibe como argumento un valor y devuelve la posición de la primera ocurrencia, o -1 si no se halló.
- **inspect**: devuelve una representación en formato de cadena de caracteres.
- **last**: devuelve el valor de la última posición.
- **reverse**: invierte las posiciones del array.
- **shift**: devuelve el primer elemento del array y lo quita del original.
- **without**: recibe valores como argumento y devuelve el array sin esos valores.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  var a = new Array('x', 'Y', 'z', null, '1', '2');

  alert(a.inspect());
  alert(a.first());
  alert(a.indexOf('Y'));
  alert(a.indexOf('Z'));
  alert(a.last());
```

```
  //]]>
  </script>
 </head>
 <body>

 ...

 </body>
</html>
```

Con la clase **ObjectRange** (que deriva del objeto **Enumerable**), Prototype permite representar rangos de valores. Recibe tres argumentos: límite inferior, límite superior y un valor que indica si se incluirán los límites dentro del rango (**false**) o no (**true**). La función **$R** permite crear una instancia de la clase **ObjectRange**. El método **include** recibe como argumento un valor y devuelve **true** si está dentro del rango o **false**, si no:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  var rango = $R(6, 24, true);
```

III REDUCCIÓN

En los próximos capítulos de este libro podremos conocer más alternativas para reducir y comprimir el tamaño, no solo de los documentos correspondientes a JavaScript, sino también de los documentos realizados con XHTML. Realizaremos todo esto de manera completamente automática con las funcionalidades ofrecidas por los servidores web.

```
  if (rango.include(6)) {
    alert("Esta incluido");
  } else {
    alert("No esta incluido");
  }

  //]]>
  </script>
 </head>
 <body>

 ...

 </body>
</html>

 </body>
</html>
```

Para obtener el valor de un elemento, contamos con la función **$F**, que recibe como argumento un ID:

```
$F('txtApellido')
```

Veamos un ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

AJAX

Debemos tener en cuenta que podemos obtener más información acerca de cómo implementar soluciones Ajax para nuestros proyectos utilizando Prototype en el capítulo dedicado a este modelo, en este libro. Allí veremos cómo realizarle peticiones al servidor sin tener que actualizar la página en forma completa, todo mediante el objeto **Ajax**.

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
  <script type="text/javascript">
  //<![CDATA[

  function mostrarValor(id) {
    alert($F(id));
  }

  //]]>
  </script>
 </head>
 <body>

  <input id="txtApellido" type="text" value="" />

  <input type="button" onclick="mostrarValor('txtApellido')" value="Mostrar
     Valor" />

 </body>
</html>
```

En cuanto a las cadenas de caracteres, Prototype amplía las capacidades del objeto **String** ya que le agrega las siguientes características:

- **stripTags**: elimina todos los elementos HTML o XML encontrados.
- **stripScripts**: elimina todos los bloques **script** encontrados.
- **escapeHTML**: escapa la cadena de caracteres.
- **unescapeHTML**: quita los caracteres de escape de **escapeHTML**.

III RECURSOS PARA PROTOTYPE

La biblioteca Prototype es cada vez más utilizada, y una prueba de ello es la creación de un repositorio de librerías y extensiones llamado **Scripteka**, desde donde podemos descargar complementos para nuestros proyectos. Podemos encontrar más información en su sitio web oficial que se encuentra en la dirección **www.scripteka.com**.

- **extractScripts**: devuelve un Array con los bloques **script** encontrados.
- **evalScripts**: evalúa cada bloque **script** encontrado.
- **toQueryParams**: transforma la cadena en un array asociativo; y ubica como índice el nombre de los parámetros y les asigna el valor de éstos, como en el ejemplo.
- **parseQuery**: ver **toQueryParams**.
- **toArray**: trasforma la cadena en un array y cada posición corresponderá a un carácter.
- **camelize**: se encarga de eliminar los guiones medios y pasa a mayúsculas el primer carácter de cada una de las palabras de la cadena.

Veamos algunos ejemplos:

```
<script type="text/javascript">
//<![CDATA[

array = 'ejemplo 1'.toArray();

for (var c=0; c<array.length; c++) {
  alert(array[c]);
}

//]]>
</script>

<script type="text/javascript">
//<![CDATA[

var cadena = 'x=1&y=11&z=111';

var array = cadena.toQueryParams();

alert(array['x']);
```

API

Desde el sitio que se encuentra en la dirección **http://api.prototypejs.org** podemos revisar cada una de las funcionalidades de Prototype con un nivel de detalle muy importante. Además existen muestras de código para visualizar de manera práctica la forma de implementación.

```
alert(array['y']);
alert(array['z']);

//]]>
</script>

<script type="text/javascript">
//<![CDATA[

var cadena = 'border-color';
alert(cadena.camelize());

//]]>

     </script>
```

En cuanto a los formularios, Prototype nos provee los objetos **Field** y **Form**. **Field** cuenta con los siguientes métodos:

- **clear**: este método le asigna una cadena vacía al valor del elemento correspondiente. Recibe como argumento un identificador.
- **present**: se encarga de recibir como argumento el identificador de un campo y valida que su contenido no esté vacío.
- **focus**: este método se encarga de darle el foco al elemento correspondiente al identificador que hemos pasado como argumento.
- **select**: se encarga de seleccionar el contenido del elemento correspondiente al identificador que pasamos como argumento.
- **activate**: nos permite activar un elemento.

```
if (Field.present('txtApellido')) {
```

III TUTORIAL

Desde el sitio oficial de Prototype podemos acceder a una sección en la cual nos explican paso a paso los principios y las características básicas para utilizar esta herramienta. Más información en el sitio que se encuentra en la dirección **www.prototypejs.org/learn**.

```
    ...
    ...
    ...
} else {

    ...
    ...
    ...

}
```

```
Field.select('txtApellido');
```

En cuanto a **Form**, los siguientes métodos están disponibles:

- **serialize**: devuelve un **querystring** con los campos y los valores de todos los elementos del formulario, como si se enviara mediante el método **GET**. Para serializar solo un elemento contamos con el objeto denominado **Form.Element**, que recibe como argumento el identificador adecuado.
- **findFirstElement**: devuelve el primer elemento no deshabilitado.
- **getElements**: devuelve un array con todos los campos.
- **getInputs**: devuelve un array con todos los campos **input** del formulario. Opcionalmente recibe dos argumentos: el tipo de elemento y su nombre.
- **disable**: deshabilita o anula el formulario.
- **enable**: habilita el formulario correspondiente.
- **focusFirstElement**: pasa el foco al primer elemento visible y habilitado.
- **reset**: limpia los campos del formulario.

Todos reciben opcionalmente como primer argumento el identificador del formulario. Debe especificarse si hay más de uno en la misma página.

BLOG

Para estar al tanto de las últimas mejoras introducidas en Prototype tenemos a disposición el blog oficial del proyecto, el cual podemos visitar accediendo a la dirección web www.prototypejs.org/blog ←http://www.prototypejs.org/blog→ . Allí además podremos dejar nuestros comentarios y proponer mejoras.

```
Form.disable("formBusqueda");
```

Además de **$F**, para obtener el valor de un elemento se puede utilizar el método **getValue** del objeto **Form.Element**:

```
alert(Form.Element.getValue('apellido'));
```

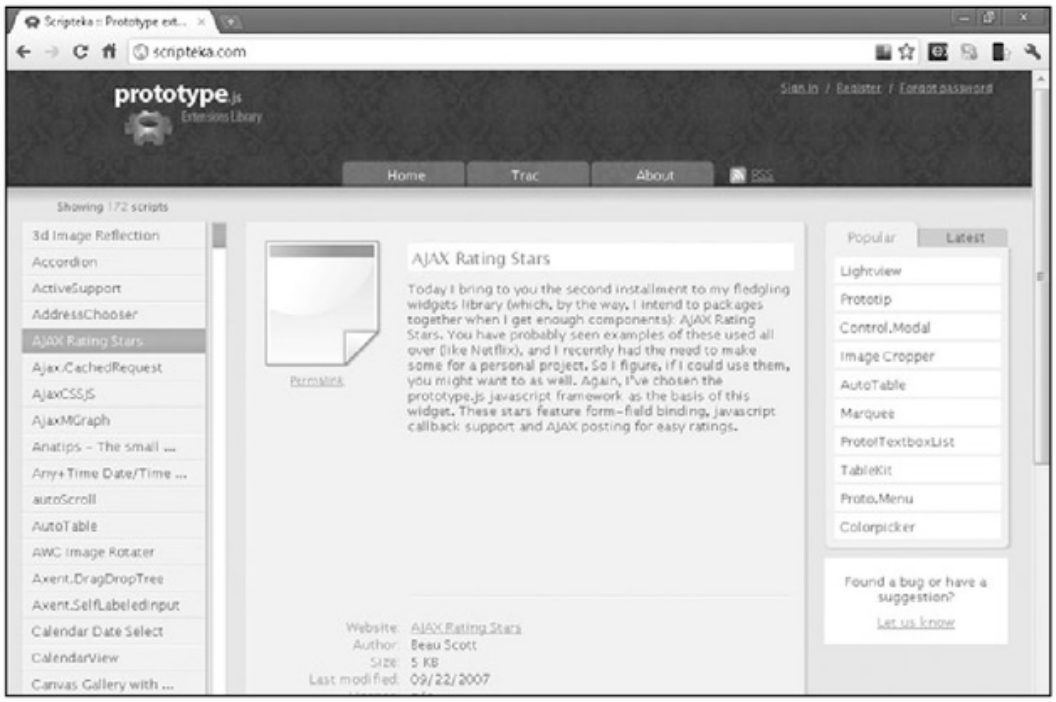


***Figura 15. Scripteka** es uno de los repositorios que se encarga de alojar aplicaciones que utilizan Prototype.*

Para controlar eventos se incluyen las siguientes alternativas:

- **Form.Element.Observer**: recibe como argumentos el identificador del elemento, los segundos entre actualización y actualización, y la función JavaScript que se invocará en cada una.
- **Form.Observer**: similar al anterior pero enfocado hacia formularios en general. Recibe como argumentos el identificador del formulario, la frecuencia en segundos entre actualizaciones y la función JavaScript que se invocará en cada actualización.
- **Form.Element.EventObserver**: recibe como argumentos el identificador del elemento y la función JavaScript que se invocará en cada actualización. Los eventos observados son **onclick** para los controles **radio** y **checkbox**, y **onchange** para los demás.
- **Form.EventObserver**: recibe como argumentos el identificador del formulario y la función JavaScript que se invocará en cada actualización. Los eventos observados son **onclick** para los **radio** y **checkbox**, y **onchange** para los demás.

De manera general, en Prototype es posible que podamos ligar o vincular elementos a eventos mediante sintaxis, de la siguiente forma:

```
$('idElemento').observe('click', nombreFuncion);
```

En el ejemplo anterior, al hacer clic en el elemento **idElemento** se invocará la función **nombreFuncion**.

La librería Prototype es muy completa y su uso está muy extendido en la actualidad. Para más información podemos acceder a los siguientes sitios, en los que se incluyen tutoriales y artículos para conocer su funcionamiento en detalle:

- **www.sergiopereira.com/articles/prototype.js.html**.
- **wiki.script.aculo.us/scriptaculous/show/Prototype**.
- **www.estadobeta.com/2006/08/04/prototype**.
- **http://particletree.com/features/quick-guide-to-prototype**.

... RESUMEN

En este capítulo hicimos un repaso acerca de cómo incluir código JavaScript en proyectos propios y acerca de cómo pulir nuestros documentos para disminuir su tamaño. En relación con este punto observamos las características de Prototype, una biblioteca que nos brinda la posibilidad de realizar distintas tareas con menor cantidad de código.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué sucede cuando un navegador encuentra la etiqueta script?

2 ¿En qué parte es conveniente ubicar el código JavaScript?

3 ¿Qué es la unificación de archivos?

4 ¿Qué es la reducción de archivos?

5 ¿Cuáles son las técnicas de reducción habituales?

6 ¿Qué es Prototype? ¿Para qué sirve?

7 ¿Cuáles son las ventajas de utilizar Prototype?

8 ¿Cuáles son las desventajas de utilizar Prototype?

9 ¿Para qué sirve la función $?

10 ¿Para qué sirve la función $$?

EJERCICIOS PRÁCTICOS

1 ¿En qué casos no utilizaría Prototype? ¿Por qué?

2 Minimice sus documentos JavaScript. ¿Cuánto espacio logró reducir?

3 ¿Cuántas peticiones logró reducir luego de unificar sus archivos?

4 Tome un documento JavaScript e intente modificarlo para utilizar las funciones disponibles en Prototype.

5 Busque información acerca de la librería Script.aculo.us.

Capítulo 5

Imágenes

Las imágenes constituyen una parte importante de las aplicaciones web, y por este motivo controlar aspectos como el tamaño (peso) y la calidad es una tarea a la que debemos prestarle atención. En este capítulo haremos un repaso por los formatos disponibles y las opciones existentes para comprimir nuestras imágenes al máximo.

Situación y necesidades	**132**
Tipos de compresión	**132**
Formatos disponibles	**132**
JPEG	133
GIF	134
PNG	136
Tratamiento de imágenes	**138**
PhpThumb	140
IrfanView	145
Precarga de imágenes	**147**
Resumen	**151**
Actividades	**152**

SITUACIÓN Y NECESIDADES

Si bien hubo un tiempo en el que las velocidades de conexión eran muy bajas y los sitios constituidos por texto con muy pocas imágenes eran regla general, hoy es habitual encontrar páginas con gráficos, fotografías y archivos de imagen que forman parte de la estructura de las páginas, por no hablar de videos y películas Flash. Más allá de estas diferencias, un punto buscado es la carga rápida de las páginas de un sitio, una demanda universal por parte de los usuarios en referencia a cualquier clase de aplicación, sin importar el servicio que preste. Responder qué porcentaje de la carga total deberían representar las imágenes dependerá del servicio y de las características del sitio. Por ejemplo, una aplicación que exhiba galerías de imágenes debe procesar más archivos que el resto de las páginas. El trabajo y las acciones para reducir el peso de las imágenes siempre son productivos, por ejemplo:

- evitar el uso de imágenes innecesarias, por ejemplo fondos uniformes.
- elegir correctamente el formato para cada una.
- procesar archivos para lograr el menor peso posible sin perder calidad.
- utilizar distintas versiones de una misma imagen, por ejemplo con miniaturas.

TIPOS DE COMPRESIÓN

Los distintos formatos de imagen admiten compresión de datos si se implementa un algoritmo específico. Hay dos clases de compresión: la que provoca pérdida de calidad en la imagen y la que no. A partir de la compresión sin pérdida de calidad es posible reconstituir la imagen original, y su principal característica es la compactación de la información. El grado de compresión es limitado, y llegado a un punto no es posible reducir el tamaño original: en ese contexto comienzan a surgir los algoritmos que provocan pérdida de calidad en la imagen. Así como el **MP3** mejora la compresión omitiendo los sonidos no perceptibles, estos algoritmos trabajan sobre la limitación del ojo para captar características de las imágenes.

GOOGLE

Google acaba de proponer un formato de imagen propio, y según estimación alcanza alrededor de un 40 por ciento de lo que ocuparía una imagen jpeg tradicional. Podemos obtener más información en el sitio web oficial, que se encuentra en la dirección **http://code.google.com/intl/es-ES/speed/webp**.

FORMATOS DISPONIBLES

En el ambito web se destacan tres formatos de imagen como los más utilizados: **GIF** (*Graphical Interchange Format*), **PNG** (*Portable Network Graphics*) y **JPEG** (*Joint Photographic Experts Group*). Cada uno utiliza un tipo de compresión definido:

- **GIF** utiliza un algoritmo sin pérdida llamado **LZW**.
- **PNG** utiliza un algoritmo sin pérdida llamado **PNG**.
- **JPEG** utiliza un algoritmo con pérdida llamado **JPEG**.

Cada formato tiene sus ventajas y desventajas, e incluso conviven en un mismo sitio y se aplican según el tipo de imagen a exhibir en la página.

FORMATO	USO PRINCIPAL
GIF	Transparencias, animaciones, iconos, logos, banners y fotografías de hasta 256 colores.
PNG	Gráficos.
JPEG	Fotografías de más de 256 colores.

***Tabla 1**. Qué formato utilizar según el caso.*

JPEG

Logra archivos muy pequeños en tamaño pero si se comprime demasiado la pérdida de calidad se hace notoria a partir de la visualización de pixeles. Generalmente, se utiliza para fotografías y muchos dispositivos, como por ejemplo las cámaras fotográficas digitales o los teléfonos celulares, lo utilizan como formato por defecto.

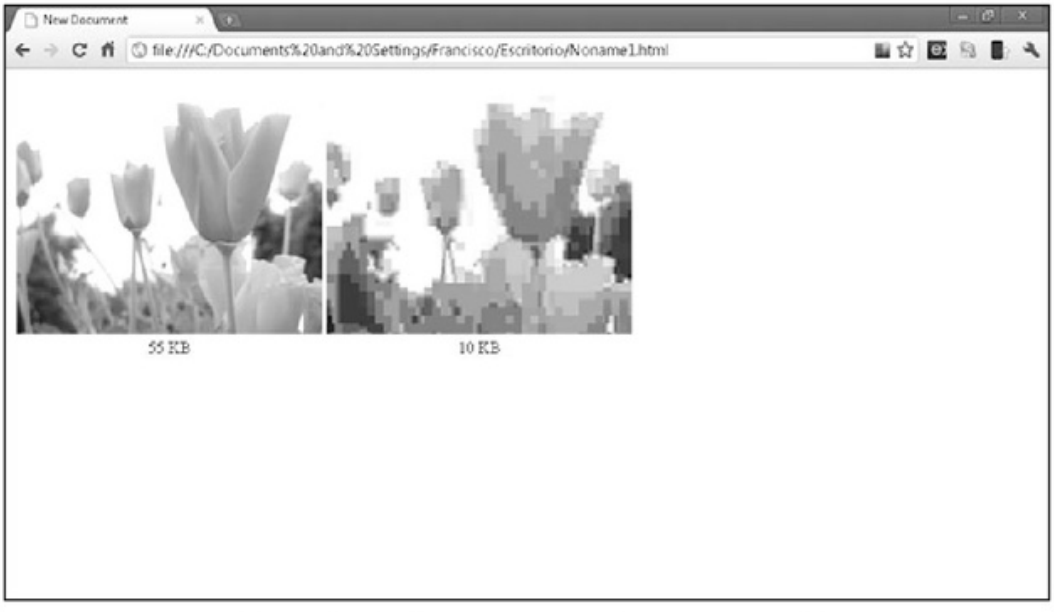


***Figura 1.** JPEG logra un nivel de compresión alto pero limitado según el tipo de imagen a tratar.*

Soporta la generación de imágenes con más de 16 millones de colores. Otra característica de este formato es que cada vez que se comprime (se aplica un grado mayor de compresión) y descomprime una imagen (se vuelve a un grado de compresión mayor), ésta pierde calidad, lo que produce una pérdida acumulativa entre proceso y proceso. El formato **JPEG 2000** fue diseñado por los mismos autores de JPEG y es considerado un mejoramiento del algoritmo original. Los archivos JPEG 2000 llevan la extensión **jp2**. Admiten un mayor nivel de compresión pero como contrapartida en la transformación se dejan de representar algunos detalles presentes en la imagen original. Este formato basado en un algoritmo sin pérdida no es soportado por todos los navegadores y su uso no está extendido. Podemos obtener más información en el sitio web oficial **www.jpeg.org/JPEG2000.htm**.

GIF

La cualidad principal que lo diferencia del resto es la posibilidad de generar animaciones, un recurso utilizado cuando no había herramientas como Macromedia Flash para generar películas de alto impacto. Dos aspectos conspiran contra su uso en la actualidad: por un lado está limitado a 256 colores (8 bits), muy reducido para las imágenes de alta calidad de hoy, y también está limitado por los problemas legales surgidos a partir del patentamiento que el algoritmo LZW sufrió durante cierto tiempo (veremos más acerca de esto en las próximas secciones).

***Figura 2.** La utilización de imágenes GIF está siendo reemplazada poco a poco por otras tecnologías u otros formatos.*

Pueden observarse en banners o anuncios publicitarios, y permiten almacenar múltiples imágenes en una sola, exhibiéndose en un orden preestablecido cada cierto

tiempo. Poco a poco están siendo reemplazados por películas Flash, que mediante el lenguaje ActionScript ofrece características avanzadas para controlar diversos comportamientos que van más allá de la simple muestra de imágenes (el entorno de desarrollo sirve para generar aplicaciones de propósito general). Otra característica de este formato es la trasparencia, que permite seleccionar un color en la aplicación utilizada para generar imágenes GIF, que será invisible dentro del entorno de una página web. Si contamos con una imagen que de forma predeterminada ubica como color trasparente al azul, y si ubicamos el archivo dentro de un elemento con color de fondo verde, al ver la imagen observaremos que visualmente el archivo reemplaza el color transparente (azul) por el de fondo (verde):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>imagenes </title>
</head>
<body>

 <table>
 <tr>
  <td><img src="123.gif" /></td>
  <td bgcolor="#FFFF00"><img src="123.gif" /></td>
 </tr>
 </table>

</body>
</html>
```

Figura 3. *Una de las características más importantes del formato GIF es el soporte para transparencias.*

Las imágenes GIF tienen la característica de cargarse de manera progresiva, lo que significa que inicialmente en el navegador se observa la imagen completa en baja calidad y a medida que progresa la descarga, la definición de ésta se actualiza hasta completarse.

PNG

Surgió en los últimos tiempos como la alternativa a GIF (PNG fue diseñado como reemplazo de GIF) fomentada por muchos sectores que aprueban el uso de tecnologías no propietarias. Soporta transparencias al igual que GIF (algunas versiones de **Internet Explorer** pueden no manejarlas correctamente) y se considera de suma utilidad para los gráficos y no tanto para las fotografías o ilustraciones. Al utilizar un algoritmo sin pérdida, los archivos de imagen que contengan muchos colores pueden no comprimirse tanto como uno quisiera. Se trata de un formato reciente que puede no ser visualizado correctamente por los navegadores web antiguos, casi en desuso.

***Figura 4.** La compresión lograda por PNG es limitada y no puede afectar la calidad de la imagen original.*

III JPEG 2000

Debemos tener en cuenta que la librería denominada **OpenJPEG** está escrita en **lenguaje C** y se encarga de permitir la creación de imágenes en formato **JPEG 2000**, el nuevo estándar de compresión de **Joint Photographic Experts Group** (**JPEG**). Es de código abierto y podemos descargarla de su sitio web oficial en **www.openjpeg.org**.

La idea de generar un formato similar a GIF surgió en 1996, cuando la empresa **Unisys** comenzó a hacer valer su patentamiento (que mantenía junto a **IBM**) sobre el algoritmo **LZW**. Esto podía afectar a aquellas aplicaciones que generaran archivos GIF. Las empresas Unisys e IBM mantuvieron durante algún tiempo la patente sobre el algoritmo de compresión LZW, utilizado para crear archivos GIF. En la práctica esto hizo que no existieran aplicaciones generadoras de archivos GIF libres, algo que propulsó a distintos organismos a optar por otras alternativas.

***Figura 5.** El sitio del proyecto GNU explica claramente con este caso en particular los inconvenientes que el patentamiento le trae al software libre.*

El texto completo puede ser revisado en el sitio web de **GNU**, en **www.gnu.org/philosophy/gif.es.html**. Este conflicto resultó fundamental para que el formato PNG se desarrollara y tomara un impulso que mantiene hasta hoy, aun cuando formalmente el algoritmo **LZW** puede ser utilizado por cualquier aplicación para generar archivos GIF (los derechos de propiedad vencieron en 2006).

III ORIGEN DE LZW

Es interesante saber que el algoritmo **LWZ** debe su nombres a las iniciales de los apellidos de sus creadores (**Abraham Lempel** y **Jakob Ziv**) y al de quien años más tarde se encargó de perfeccionarlo (**Terry Welch**). Este agoritmo fue creado inicialmente entre los años 1977 y 1978, y en 1984 se lograron avances notorios en su desarrollo.

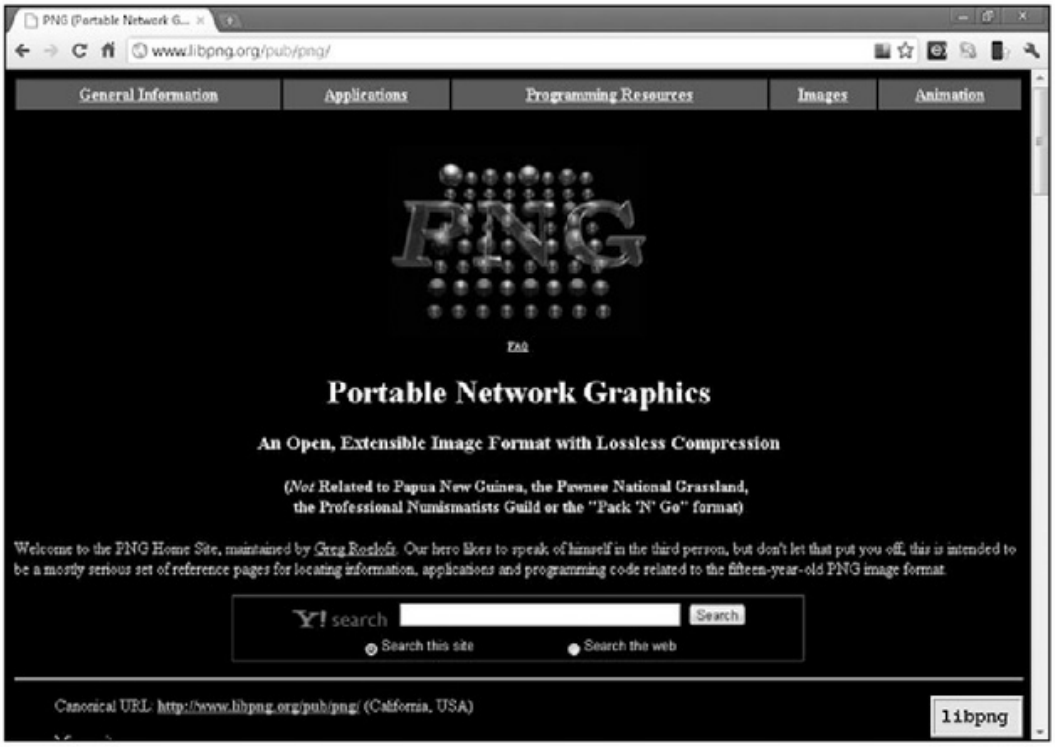


Figura 6. *PNG es un formato cada vez más popular que cuenta con el apoyo de los que fomentan el uso de software libre.*

PNG cuenta con su propio tipo **MIME**, un encabezado que indica el tipo de medio que representa el contenido de la página:

```
Content-Type: image/png
```

Existen dos subdivisiones para este formato: **PNG-8** (8 bits como GIF) y **PNG-24** (24 bits como JPEG). Estas imágenes utilizan la extensión PNG. Podemos obtener más información acerca de este formato en la dirección **www.libpng.org/pub/png**. En los siguientes sitios encontramos herramientas para procesar estas imágenes:

- **http://advsys.net/ken/utils.htm**.
- **http://pmt.sourceforge.net/pngcrush**.

W3C

Es importante saber que la organización conocida como W3C, que es la encargada de mantener un gran número de especificaciones relacionadas con diversas tecnologías abiertas, tiene en su sitio un apartado dedicado a PNG. Podemos visitarlo si accedemos a su sitio que se encuentra en la dirección **www.w3c.org/Graphics/PNG**.

TRATAMIENTO DE IMÁGENES

Un sitio web cuenta con dos tipos de imágenes, las que deben ser procesadas de manera dinámica y las que pueden ser tratadas previamente (antes de ser exhibidas) por algún programa de edición para reducir su peso sin perder calidad. En el primer caso, en general los usuarios cargan imágenes en un formulario, y éstas se deben procesar por una librería para amoldarlas a las necesidades del sitio. Todo esto se lleva a cabo de manera automatizada y sin la intervención de un operador / diseñador que examine y edite las imágenes originales. En el segundo caso, las personas encargadas de generar las páginas de un sitio utilizan programas de diseño para el tratamiento de las imágenes, y así pueden adecuarlas según sus necesidades.

Figura 7. *Según el servicio ofrecido, un sitio deberá publicar las imágenes cargadas por un usuario de manera inmediata.*

Es importante tener en cuenta que como norma general nunca es conveniente eliminar la imagen original, más allá de que la utilicemos directamente o no. A medida que aplicamos compresión a partir de determinados algoritmos, por ejemplo el utilizado por JPEG, la calidad inicial se va pierdiendo en forma progresiva, lo que llegado un punto impide la recuperación del original.

A continuación veremos algunos ejemplos de las dos alternativas mediante **phpThumb**, en primer lugar conoceremos una excelente librería basada en **GD** disponible mediante el lenguaje de programación PHP, posteriormente analizaremos el uso de **IrfanView**, se trata de una aplicación sencilla y gratuita que está diseñada para permitirnos visualizar y procesar imágenes.

PhpThumb

PHP provee una extensión para utilizar la librería **GD** (podemos obtener más información en el sitio **www.boutell.com/gd**) mediante funciones predeterminadas. En general, GD se utiliza para componer gráficos en tiempo de ejecución.

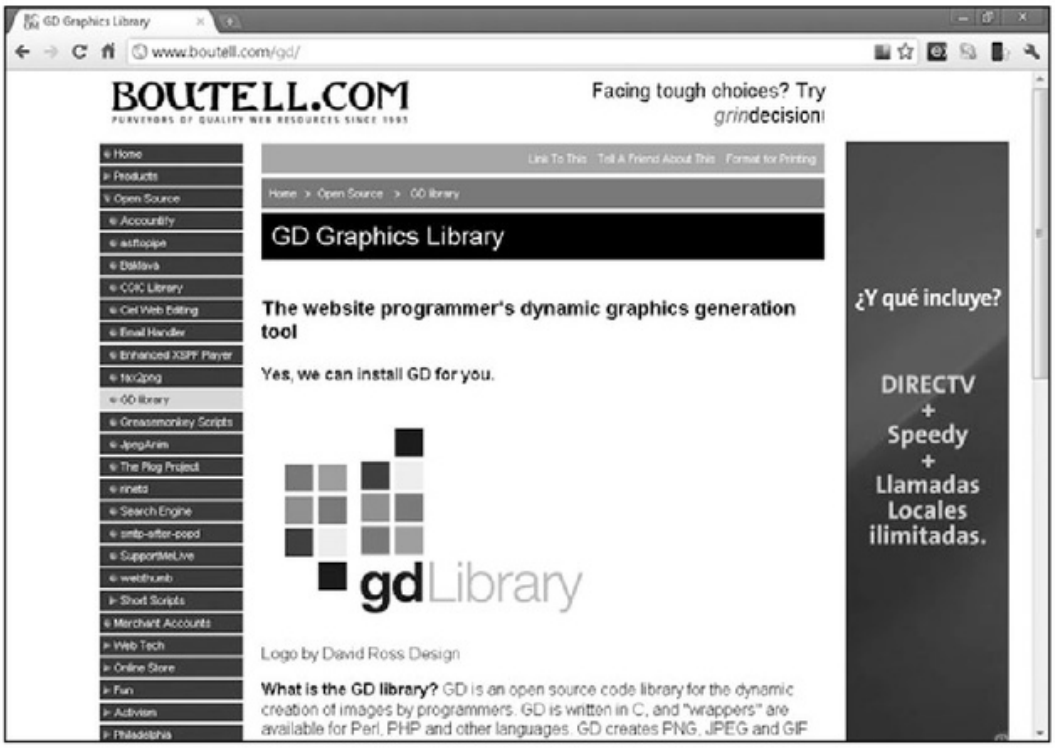


Figura 8. *GD es una biblioteca que permite la generación y el tratamiento de imágenes.*

Para su correcto funcionamiento, GD requiere que contemos con PHP versión 4.3.2 o superior (la librería viene incluida en la distribución estándar) y para comprobar si disponemos de la librería o no (normalmente, los servicios de alojamiento web dan soporte para su utilización), tenemos varias opciones, entre ellas:

- Utilizar la función **phpinfo**:

```
<?php phpinfo(); ?>
```

- Verificar que alguna de las funciones de extensión esté habilitada:

```
<?php

if (function_exists("gd_info")) {
   echo "GD esta disponible";
   echo '<pre>';
```

```
    print_r(gd_info());
    echo '</pre>';

} else {
    echo "GD no esta disponible !";
}

?>
```

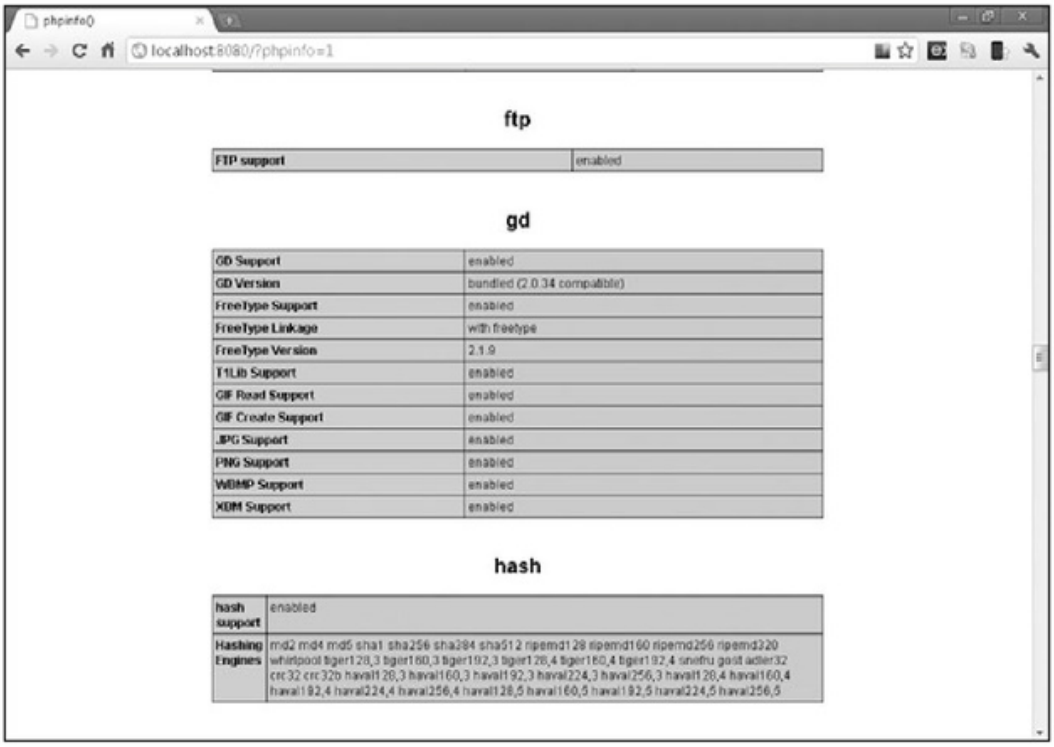


Figura 9. *La función* ***phpinfo()*** *nos brinda varios datos acerca de la configuración del servidor.*

Según la versión instalada, GD nos permitirá manipular distintos tipos de imágenes: las anteriores a la 1.6 admiten GIF, pero no PNG, y lo inverso sucede con las superiores (**phpThumb** trabaja con imágenes GIF aun cuando la versión de GD

III SVG

El organismo W3C ha desarrollado un formato de gráficos vectoriales llamado **SVG** (**Scalable Vector Graphics**). Aunque no es una novedad, está en pleno desarrollo y no son muchos los navegadores que lo soportan totalmente. Se trata de una alternativa para analizar y las posibilidades que obtenga dependerán de las empresas desarrolladoras de software.

instalada en el servidor no soporte este formato, con la clase **GIFutil**). Algunos de los formatos más populares soportados por GD son GIF, JPG y PNG.
La librería **phpThumb** se utiliza en las páginas desarrolladas en el lenguaje PHP para crear y manipular imágenes (GIF, PNG o JPEG) de forma dinámica (permite importar y exportar archivos de imagen en diferentes formatos).

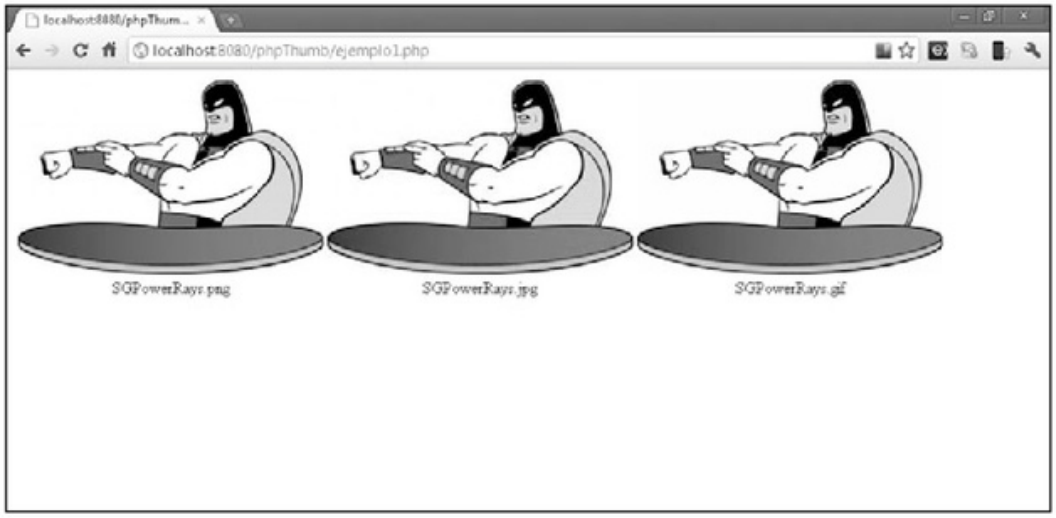


***Figura 10.** Con **phpThumb** podemos generar una misma imagen en diferentes formatos.*

Lo primero que debemos hacer luego de descargar la distribución y copiarla a un directorio del servidor será renombrar el archivo **phpThumb.config.php.default** a **phpThumb.config.php**. La forma de utilización de **phpThumb** es mediante una llamada al archivo **phpThumb.php**, que recibe una serie de parámetros y devuelve el resultado correspondiente, es decir, la imagen generada. El caso más común consiste en incluir la llamada dentro del atributo **src** del elemento **img**:

```
<img src="phpthumb.php?src=nombreImagen.jpg">
```

Entre los atributos más importantes puestos a disposición por **phpThumb**, se encuentran los siguientes:

- **src**: indica la dirección de la imagen a tratar (relacionada al archivo **phpthumb.php**).
- **w**: establece un ancho máximo en pixeles para la imagen.
- **h**: establece un alto máximo en pixeles para la imagen.
- **new**: permite crear una imagen nueva y recibe como valores un número hexadecimal (color) y opcionalmente un porcentaje de opacidad. Además, requiere que se definan los atributos **w** y **h**.

```
phpthumb/phpthumb.php?new=FFFF00&w=200&h=100
```

- **f**: establece un formato de salida para la imagen, que puede tomar como posibles valores a **JPEG**, **PNG** o **GIF**.

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&f=png
```

- **q**: define la calidad (nivel de compresión) de la imagen y solo se aplica al formato JPEG (1 para baja calidad, 95 para máxima calidad, 75 valor por defecto).

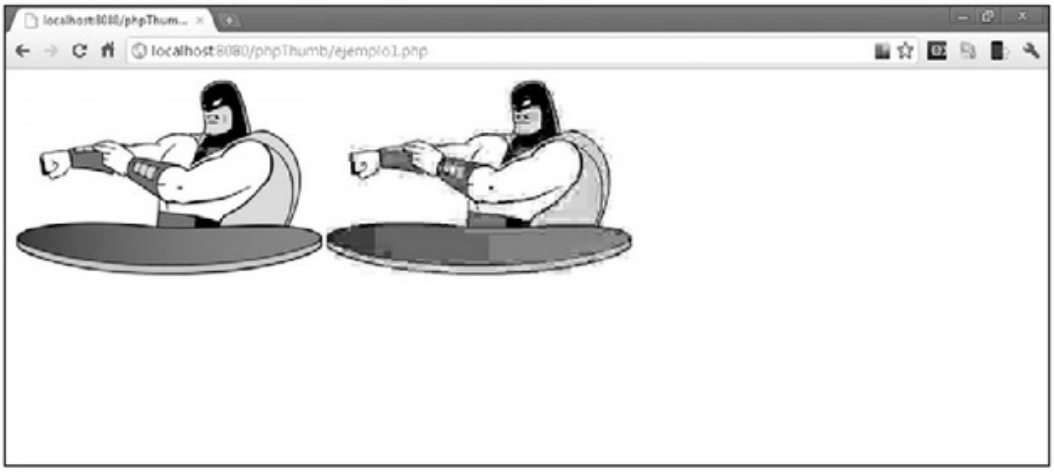


Figura 11. *La calidad de una imagen JPEG se puede tratar con* ***phpThumb*** *sin modificar el archivo original.*

- **iar** (*Ignore Aspect Ratio*): ignora las dimensiones originales de la imagen sin mantener su aspecto inicial.

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&w=400&h=100&iar=1
```

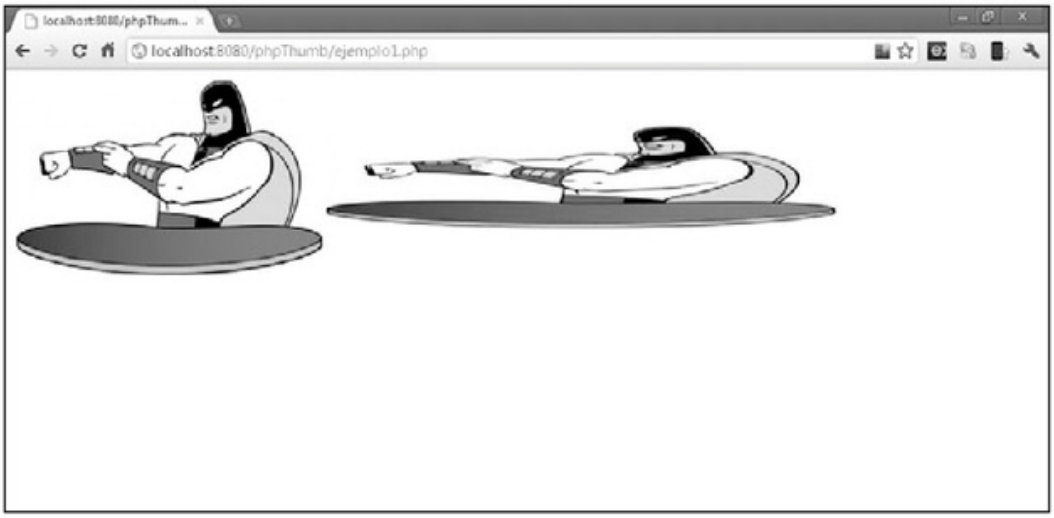


Figura 12. *La relación ancho / alto original puede ser modificada de manera dinámica con* ***phpThumb****.*

- **sia** (*Save Image As*): define el nombre que se le dará a la imagen cuando el usuario quiera guardarla. No es necesario incluir la extensión:

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&sia=nombre
```

- **maxb**: permite establecer un valor máximo en bytes para la imagen. En el ejemplo definimos el peso máximo a 40000 bytes (40kb):

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&maxb=40000
```

- **down**: permite forzar la descarga de una imagen y recibe como valor el nombre predeterminado con el que se guardará. La descarga comienza cuando accedemos a un enlace como el siguiente:

```
phpthumb.php?src=/phpthumb/imagen.jpg&f=png&down=nuevaImagen.png
```

Éstas son solo algunas de las múltiples alternativas que esta librería nos brinda, y podemos obtener más información en el sitio web oficial de la herramienta: **http://phpthumb.sourceforge.net**.

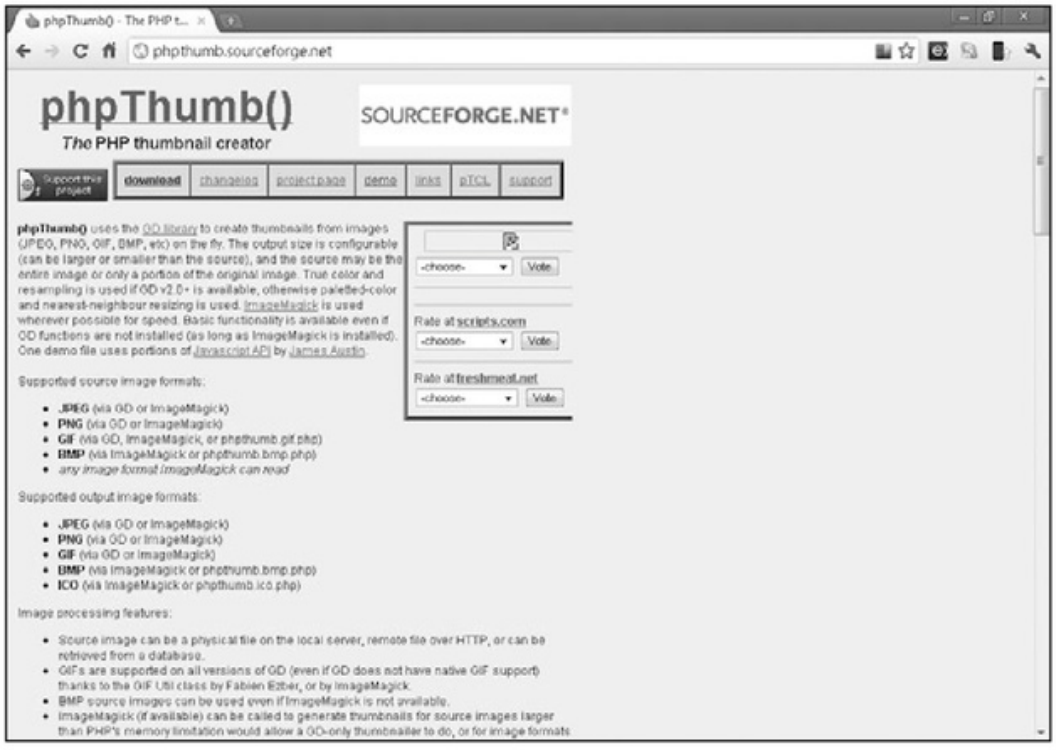


***Figura 13. phpThumb** es una aplicación muy utilizada en la actualidad gracias a su potencia y la facilidad ofrecida en cuanto a la implementación.*

phpThumb es una de las tantas opciones disponibles para el tratamiento de imágenes desde PHP. Las demás cumplen un papel similar y en su mayoría son muy profesionales y completas, por lo que nos serán de mucha utilidad.
Está disponible bajo la licencia **GNU GPL** (es posible encontrar más información en su sitio web: **www.gnu.org/copyleft/gpl.html**).

IrfanView

Se trata de una aplicación gratuita para uso personal no comercial que mantiene como características principales su simpleza y su potencia para el manejo de imágenes de distinto tipo. Está disponible para los sistemas operativos **Windows** (**9x / ME / NT / 2000 / XP / 2003 / Vista**) y podemos descargar la última versión del sitio web oficial, que se encuentra en **www.irfanview.com**.

***Figura 14.** La calidad de las imágenes JPEG se puede modificar con la aplicación **IrfanView**.*

Una de las características que hacen muy flexible el JPEG es el poder ajustar el grado de compresión. A mayor compresión, menor tamaño y menor calidad, y a

III EXTENSIÓN

A pesar de que el formato se denomine JPEG, las imágenes que lo utilizan pueden almacenarse con las extensiones **JPG** o **JPEG**. Esto se debe al hecho de que hay determinados sistemas operativos que solo aceptan tres caracteres para la extensión de los archivos. Por lo demás, debemos saber que ambas alterativas se consideran equivalentes.

menor compresión, mayor tamaño y una calidad similar a la de la imagen original. Las opciones que brinda **IrfanView** con el cuadro de diálogo al intentar guardar una imagen nos permiten definir ciertos aspectos relativos a su calidad, y para JPEG podemos establecer un valor entre 1 (calidad mínima) y 100 (calidad máxima). PNG por su parte admite valores entre 0 (sin compresión) y 9 (máxima compresión).

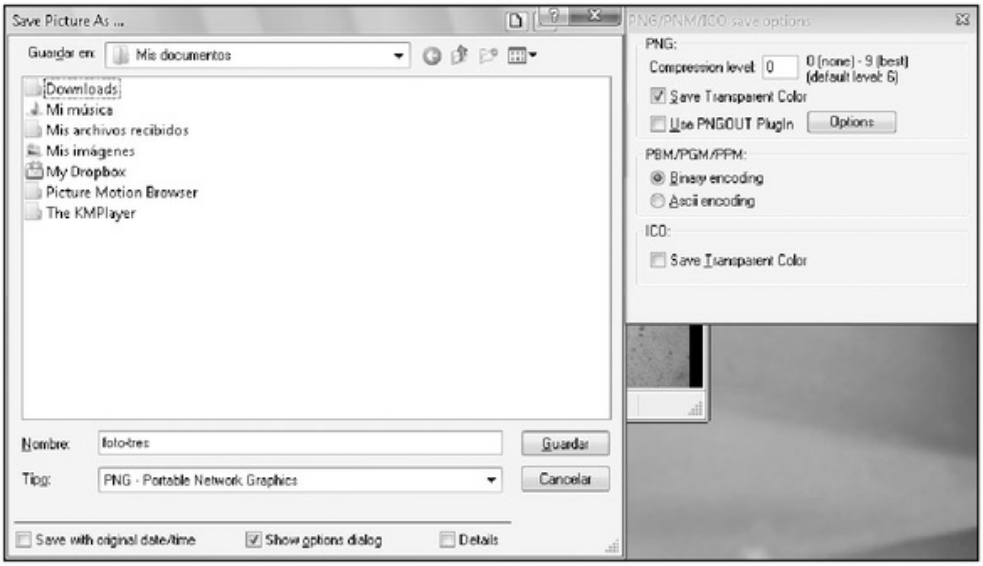


Figura 15. *Al momento de almacenar una imagen podemos definir diversos aspectos según se trate de un formato u otro.*

Además en irfanview podemos definir trasparencias para GIF y PNG, y desde **Options/JPG Loseless Operations** podemos aplicarle una optimización estándar a este tipo de imágenes de una manera fácil y práctica.

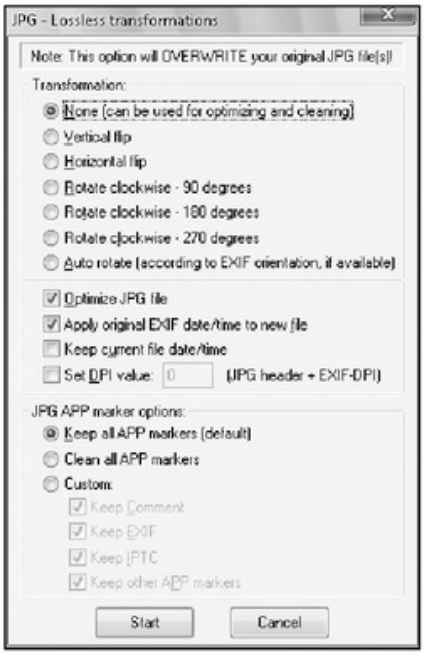


Figura 16. *La interfaz de* ***IrfanView*** *es intuitiva y a la vez nos brinda una gran cantidad de opciones.*

PRECARGA DE IMÁGENES

En ocasiones, una página requiere mostrar u ocultar imágenes durante la interacción con el usuario sin recargar la página ni generar esperas que hagan tediosa la navegación. La precarga sirve para inicialmente descargar imágenes y que éstas se encuentren disponibles en la memoria caché del navegador para cuando se las necesite.
Los posibles casos de uso pueden ser la necesidad de cambiar una imagen cuando el puntero del mouse le pasa por encima, o galerías que permiten avanzar a la posición siguiente o a la anterior. La opción por fuera de la precarga sería generar una petición **HTTP** al disparase un evento (**onMouseOver**, por ejemplo), algo que retardaría el procesamiento. En estos casos, algunos aspectos como el tiempo de espera lógico, el tamaño de las imágenes y el tipo de conexión del usuario pueden ser perjudiciales para el correcto funcionamiento de nuestras páginas.
Si bien en los navegadores es posible habilitar la memoria caché local, el cliente de la aplicación solo podrá aprovechar esta situación en las llamadas siguientes a la primera.
Hay distintas maneras de implementar la precarga de imágenes y la más popular es mediante la utilización del objeto **Image** del lenguaje de programación JavaScript:

```
var imagen = new Image(100, 100);
imagen.src = "ruta/a/nombreImagen.png";
```

Los dos argumentos dados a la función constructor corresponden al ancho y el alto de la imagen respectivamente, mientras que la propiedad **src** indica la ruta a la imagen que queremos precargar.
Algunos navegadores muy antiguos pueden no soportar este objeto. Para validar este hecho podemos incluir la siguiente expresión:

```
if (document.images) {

  var imagen = new Image(100, 100);
```

III ESTRUCTURA DE UNA PÁGINA

Es importante tener en cuenta que en algunos diseños es posible observar la utilización de imágenes para aplicarles márgenes a otros elementos. El caso típico es el de las imágenes trasparentes de un pixel de ancho por uno de alto. Más allá del peso de cada una, es recomendable evitar este tipo de acciones y utilizar estilos CSS, por ejemplo.

```
  imagen.src = “nombreImagen.png”;
}
```

En caso de que haya más de una imagen, debemos asignarles nombres diferentes:

```
if (document.images) {
  var imagen1 = new Image(100, 100);

  imagen1.src = “nombreImagen1.png”;

  var imagen2 = new Image(150, 10);
  imagen2.src = “nombreImagen2.jpg”;
}
```

El evento **onLoad** del objeto **Image** se dispara al momento de finalizar la carga. A conrinuación podemos ver un ejemplo:

```
if (document.images) {
  var imagen = new Image(100, 100);
  imagen.src = “nombreImagen.png”;

  imagen.onLoad = imagenCargada;
}

function imagenCargada () {
  alert(‘Imagen Cargada’);
}
```

Un caso clásico de utilización de la precarga es el del **RollOver** de imágenes, que al dispararse determinados eventos permite intercambiar una imagen por otra:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>precargar de imagenes </title>
  <script type="text/javascript">
```

```
  //<![CDATA[

  function precargar(nombreImagen) {
    if (document.images) {
      var objImage = new Image();
      objImage.src = nombreImagen;
      objImage.onload = function() { imagenPrecargada(nombreImagen); };
    }
  }

  function imagenPrecargada(nombreImagen) {
    alert('Imagen ' + nombreImagen + ' precargada');
  }

  function cargar(nombreImagen, elemento) {
    eval('document.' + elemento + '.src = "' + nombreImagen + '"');
  }

  //]]>
  </script>
</head>
<body onLoad="javascript:precargar('img1.jpg');cargar('img2.jpg', 'ima

     gen');">

  <a href="#"
  onMouseOver="javascript:cargar('img1.jpg', 'imagen');"
  onMouseOut="javascript:cargar('img2.jpg', 'imagen');">

  <img id="imagen" name="imagen" border="0" />
```

III IMÁGENES

Una de las cuestiones básicas y principales para tener en cuenta en relación con las imágenes contenidas en un sitio es la necesidad de utilizar la menor cantidad posible de archivos y reducir el tamaño / peso de cada uno lo máximo que podamos, siempre y cuando no afecte la correcta visualización. De esta forma lograremos páginas que se carguen mucho más rápido.

```
  </a>

</body>
</html>
```

En definitiva, la carga previa puede ser útil para equilibrar los recursos consumidos por el sitio. Es decir, precargar imágenes para páginas con mucho contenido gráfico desde otras con poco.

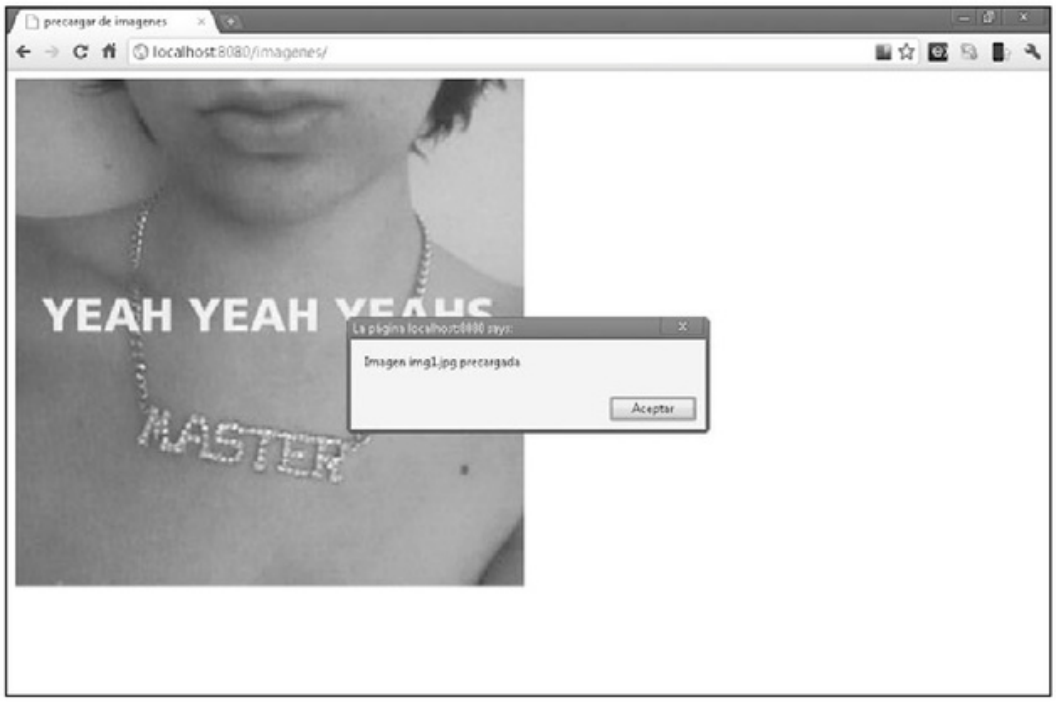


Figura 17. *La precarga puede ser una alternativa útil según la cantidad de imágenes a utilizar en una página.*

Una alternativa a la precarga de imágenes con JavaScript es utilizar CSS. Para esto es necesario incluir en la página un elemento como el siguiente:

```
<div class="contenedorImagenes"></div>
```

Y desde la hoja de estilos asociada al documento, establecer las dimensiones (alto y ancho) a cero para que no interfieran en el diseño original y utilizar la propiedad **background-image** para definir las imágenes:

```
.contenedorImagenes {
```

```
    width: 0px;
    height: 0px;

    display: inline;

    background-image: url(imagen1.png);
    background-image: url(imagen2.png);
    background-image: url(imagen3.png);
    background-image: url(imagenN.png);
    background-image: url();
}
```

Generalmente, la precarga demanda un tiempo mayor del habitual durante la carga inicial de las páginas, pero como contrapartida podemos disponer de las imágenes rápidamente y sin demoras.

... RESUMEN

La optimización de imágenes para aplicaciones web es un tema de importancia que afecta de manera directa el desempeño de nuestras páginas. En este capítulo observamos las posibilidades disponibles para generar archivos gráficos y las opciones relativas a la disminución del peso sin una pérdida de calidad notoria.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuáles son los formatos de imagen más utilizados?

2 ¿Cuáles son las características que los diferencian?

3 ¿Qué es un algoritmo de compresión?

4 ¿Qué tipos de algoritmos conoce? ¿En qué se diferencian?

5 Nombre 3 características de PNG.

6 Nombre 3 características de JPEG.

7 Nombre 3 características de GIF.

8 ¿Qué es LZW?

9 ¿Qué formato de imagen prefiere? ¿Por qué?

10 ¿Qué es JPEG 2000?

EJERCICIOS PRÁCTICOS

1 Comprima todas las imágenes de su sitio y calcule el peso total ahorrado.

2 Utilice IrfanView para guardar una imagen PNG con el máximo nivel compresión. ¿Qué diferencias nota con respecto a la imagen original?

3 Genere una función JavaScript para precargar imágenes.

4 Genere una galería de imágenes con PhpThumb.

Capítulo 6

Ajax

Sin dudas, Ajax produjo un quiebre en lo referido a la generación y el acceso a sitios web, y en este capítulo observaremos cómo puede ayudarnos a acelerar la interacción que los usuarios mantienen con las páginas de nuestros sistemas o aplicaciones web.

Características y tecnologías componentes 154
Cambios introducidos 157
Principios y origen del término 158
El objeto XMLHttpRequest 159
Compatibilidad 162
Desventajas en relación con el modelo tradicional 165
- JavaScript 165
- Internet Explorer 165
- Navegadores actuales 166
- CSS 166
- Seguridad 166
- Características del modelo 166

Implementación con Prototype 167
Disminución del tiempo de respuesta y uso del caché 173
Ajax y los lenguajes de programación 173
- Java 174
- .Net 176
- PHP 176

JavaScript 178
Resumen 179
Actividades 180

CARACTERÍSTICAS Y TECNOLOGÍAS COMPONENTES

La búsqueda de la reducción del tráfico es un hecho permanente y una de las técnicas más populares de los últimos tiempos que puede ayudarnos a lograr avances significativos al respecto es Ajax (*Asynchronous JavaScript And XML*). Éste nos proporciona un modelo en el que intervienen tecnologías ampliamente conocidas como **(X)HTML**, **CSS**, **JavaScript**, **DHTML**, **DOM**, **XML** y **XSLT**. Sin embargo, nada impide que utilicemos otras debido a que Ajax define conceptos acerca de la interacción de un usuario con una aplicación web que están por encima de las herramientas que se utilicen. Cada tecnología cumple un rol específico:

- La parte de presentación mediante XHTML y CSS.
- Acceso al contenido de la página mediante **DOM** y **DHTML**.
- Intercambio de datos con **XML** o **JSON**.
- Comunicación asincrónica con el servidor mediante **`XMLHttpRequest`**.

La **Sincronía** hace referencia a la dependencia entre procesos (cuando un navegador realiza una petición, la actividad del usuario se interrumpe hasta que se obtiene la respuesta). La **asincronía** puede definirse como la capacidad que tiene una aplicación de administrar procesos independientes unos de otros, es decir, que la actividad del usuario no se interrumpe en su totalidad. Esto se debe a que una de las cualidades de Ajax consiste en mantener la interfaz de usuario siempre visible y bloquear transitoriamente solo una parte, que se actualizará al momento de recuperar la respuesta del servidor. Esto es algo similar a lo que sucede con las aplicaciones de escritorio. Un cliente recurre al servidor solo cuando es absolutamente necesario.

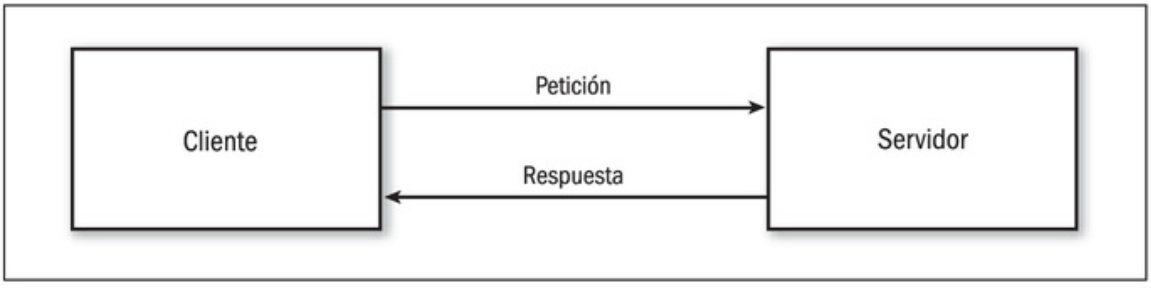


Figura 1. *El proceso cliente servidor tradicional requiere la recarga total de la página que realiza una petición.*

Dentro de la estructura definida a partir de XHTML y CSS se incluyen elementos (por ejemplo **`div`** y **`span`**), cuyo contenido se puede modificar de forma dinámica con DHTML. Para localizarlos y acceder a ellos contamos con **DOM**, que es accesible desde JavaScript, entre otros lenguajes. Tanto la recuperación de datos desde

el lado servidor como la actualización de la parte cliente quedan a cargo de funciones JavaScript. Mediante un objeto llamado **XMLHttpRequest** este lenguaje permite enviar y recibir datos entre un navegador y un servidor web de forma asincrónica, sin tener que actualizar la página completa. La recuperación de datos se produce a partir de fuentes externas como bases de datos, servicios web, archivos planos, etcétera. Todas las tecnologías componentes de Ajax se han propuesto con el objetivo de preservar el desarrollo en **capas** y separar aspectos como la **presentación** (interfaz de usuario), las **reglas** de negocio y los **datos**. En resumen, las tareas clásicas en una aplicación web basada en Ajax son las siguientes:

- Cargar la interfaz de usuario, en la que se incluyen tanto los aspectos relacionados con la presentación como las funciones JavaScript para actualizar los datos contenidos en ella, a los que se puede acceder mediante DOM.

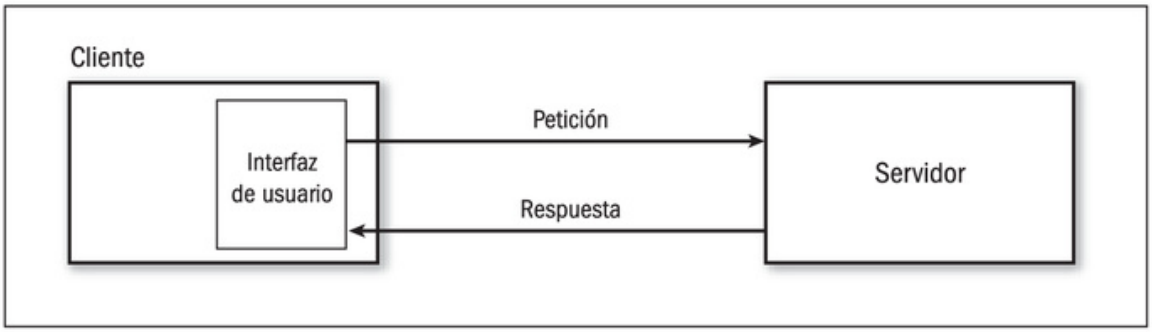


***Figura 2.** El papel de la interfaz de usuario en Ajax es de suma importancia dada su responsabilidad dentro del modelo.*

- Realizar las peticiones al servidor desde las funciones JavaScript mediante la implementación del objeto **XMLHttpRequest**. Normalmente, las peticiones se disparan mediante alguno de los eventos disponibles (**onClick**, **onMouseOver**, etcétera).

- Actualizar la interfaz con los datos recuperados desde el lado servidor con algún lenguaje de programación. Ajax no está ligado a ninguno en particular. De hecho, ni siquiera es obligatorio utilizar uno, aunque por supuesto la funcionalidad de la aplicación será reducida. Para citar algunos casos de uso común, podemos recu-

DOM

DOM (**Document Object Model**, Modelo de Objetos del Documento) equivale a una representación de los componentes de un documento estructurado (XHTML o XML, por ejemplo) como si se tratase de objetos, cada uno con sus propiedades, métodos y eventos asociados. Es un estándar mantenido por la W3C y podemos encontrar más información en **www.w3.org/DOM**.

perar información desde bases de datos, servicios web o generar salidas mediante lenguajes como por ejemplo **PHP**, **JSP** o **ASP.NET**, o cualquiera que esté soportado por el servidor. Una vez generada la respuesta, el resultado retorna a la aplicación cliente desde la cual se generó la petición.

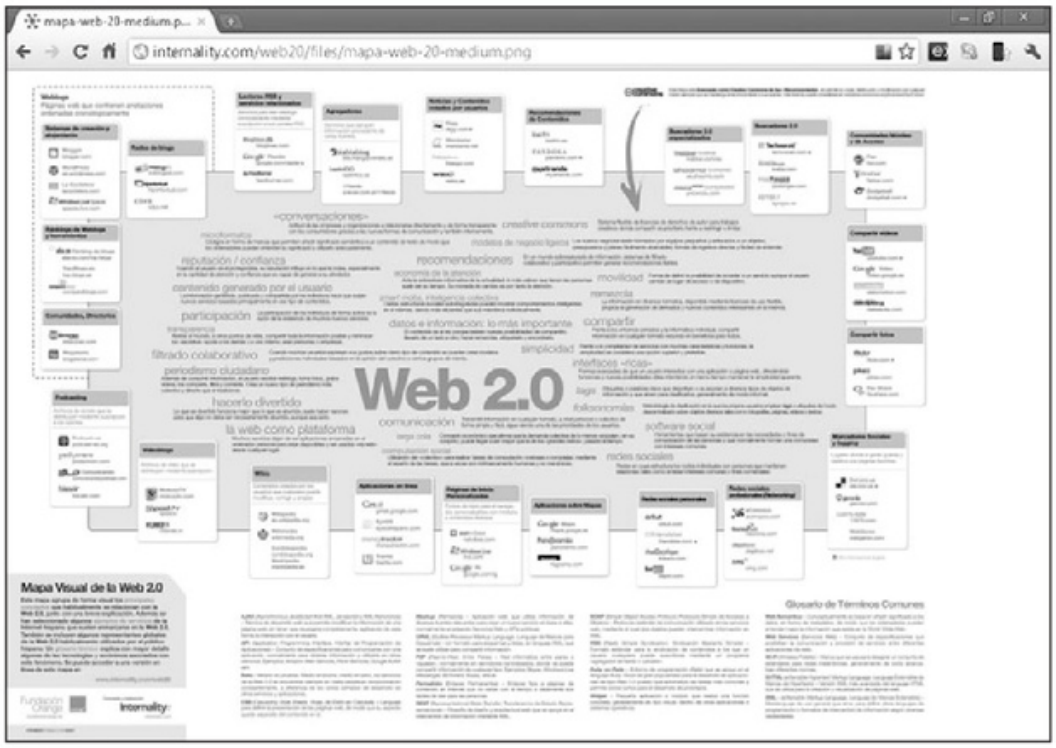


Figura 3. *Ajax es uno de los componentes del movimiento popularmente denominado* ***Web 2.0*** *(Fuente http://internality.com/web20).*

Una de las claves del modelo es que se sostiene a partir de tecnologías probadas, con una gran comunidad de usuarios a los cuales no les costará demasiado incorporar conceptos fundamentales del modelo y comenzar a desarrollar aplicaciones funcionales de manera inmediata. Existen frameworks y librerías especializadas que facilitan la tarea. Ajax no depende de una empresa u organización para tener a disposición más herramientas o controles. Si bien hay librerías para distintos lenguajes que facilitan tareas de desarrollo, son los usuarios los que tienen que generar aplicaciones a partir de conocimientos adquiridos en las tecnologías que rodean al modelo.

III ÉXITO

Una de las características más importantes del modelo consiste en que todas las herramientas componentes de Ajax están disponibles desde hace tiempo, y el éxito y la popularidad recientes del nuevo modelo tienen que ver con la experiencia lograda por parte de una gran cantidad de desarrolladores que pueden acceder a sus principios sin dificultades.

CAMBIOS INTRODUCIDOS

Según el modelo tradicional, en una aplicación web el cliente le envía una petición al servidor, que luego del procesamiento devuelve el resultado solicitado. El requerimiento desde el lado cliente depara que entre la petición y la respuesta se salta de una página a otra, o sea que es necesario actualizar el documento para ver reflejados los cambios correspondientes. Normalmente, solo una parte de la página necesita ser modificada. Sin embargo la recarga es total: se carga de nuevo la misma información, o sea, la que no cambia entre la página desde la cual se realiza la petición y la que contiene la respuesta. Esto supone en principio dos cuestiones, una es el tiempo perdido durante la actualización de la página origen (ya que el usuario deberá esperar cierto tiempo para poder visualizar una salida similar a la entrada) y otra es el ancho de banda que es utilizado en forma completamente innecesaria.

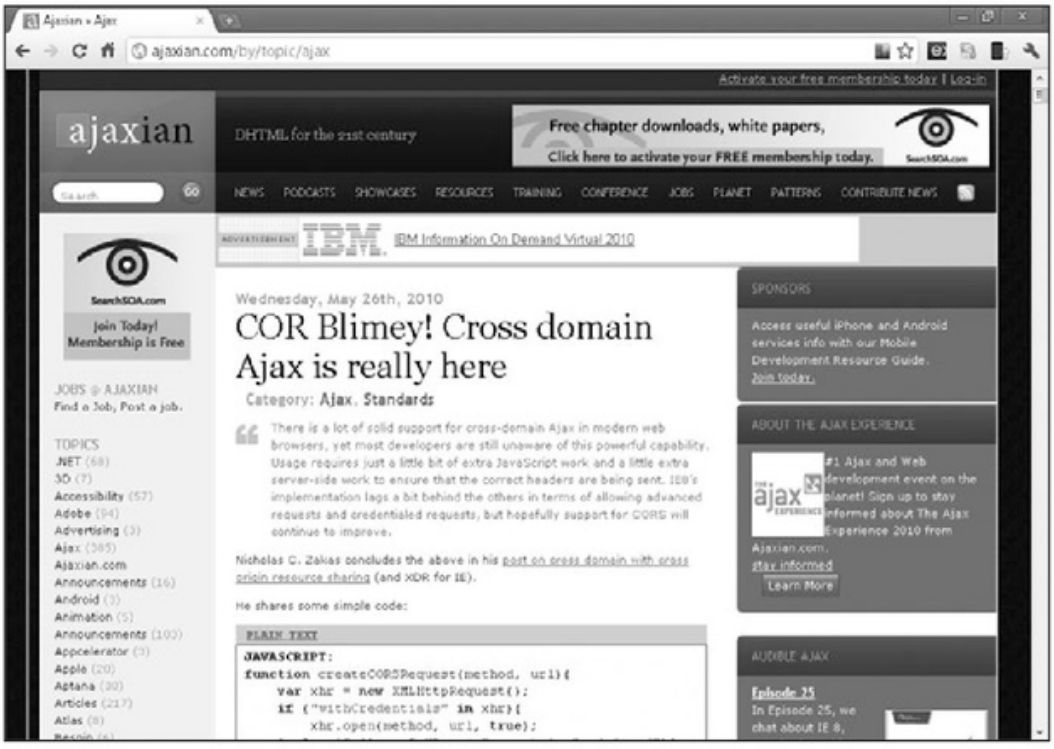


Figura 4. *La presencia de Ajax puede observarse cada día en más y más sitios. Esto afianza los alcances del proyecto.*

Mas allá de las diferencias entre las aplicaciones web tradicionales y las interfaces que son realizadas utilizando el potencial de Ajax, el modelo es aplicable a cualquier tipo de situación y no tiene por qué desplazar lo que ya hemos implementado. De esta forma podemos desarrollar determinadas secciones de una página utilizando la funcionalidad ofrecida por Ajax y dejar el resto del sitio sin realizar modificaciones. Además, el tipo de alcance del proyecto no es importante ya que desde las aplicaciones más simples hasta las más complejas pueden implementarse utilizando algunas de las características propuestas por Ajax.

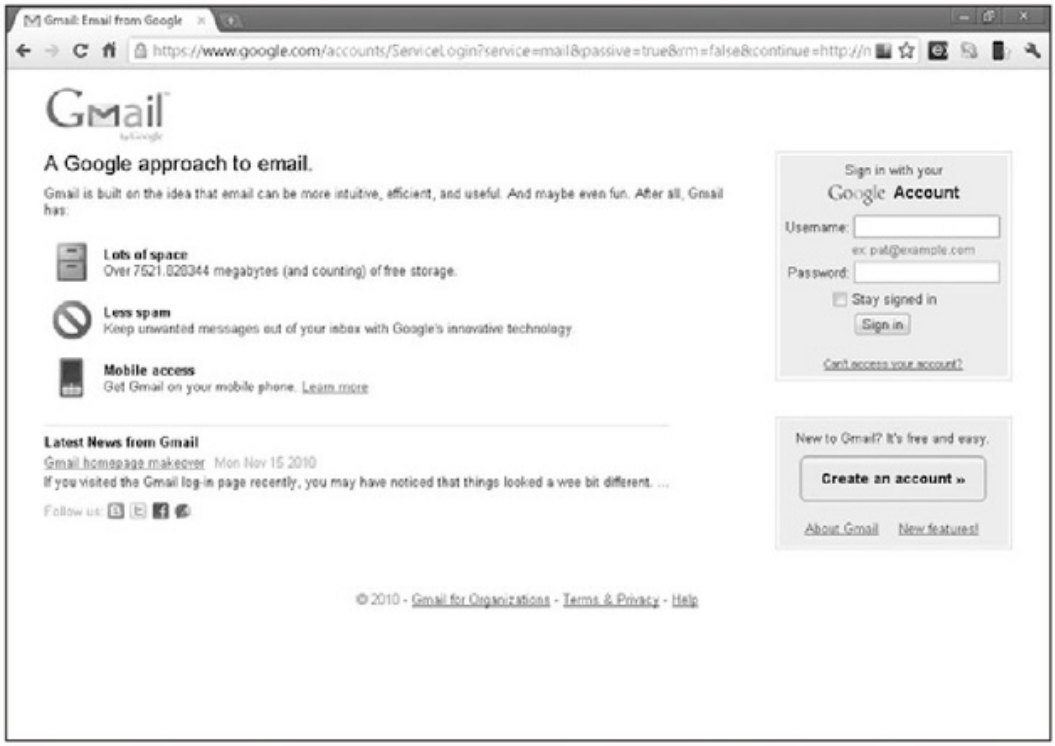


Figura 5. *El cliente del correo electrónico* ***GMail*** *fue pionero en incorporar Ajax a nivel profesional.*

PRINCIPIOS Y ORIGEN DEL TÉRMINO

El término fue concebido en un artículo publicado inicialmente en el sitio **Adaptive Path** en el año 2005, acerca del cual podemos obtener más información en **www.adaptivepath.com**. Allí se plantea que los usuarios de aplicaciones web actuales necesitan ir más allá de la simple navegación de páginas por enlaces, e intentar acercarse lo más posible a la sensación ofrecida por una aplicación de escritorio sin perder las posibilidades alcanzadas hoy en día por las aplicaciones web. También se hace referencia a cuestiones técnicas de implementación y se hace hincapié en que Ajax puede ser visto como una arquitectura en relación con las partes que intervienen en una aplicación web, pero también como un conjunto de herramientas

XML

Debemos tener en cuenta que si bien el metalenguaje XML forma parte del acrónimo Ajax como formato preferido para intercambiar datos con el servidor, no es la única alternativa disponible. Podemos utilizar opciones como **JSON**, texto plano, o cualquier alternativa que pueda ser comprendida tanto por el lado cliente como por el servidor.

específicas para implementarla. Según el artículo, Ajax es tan sólo una forma de acceder a los contenidos que se encuentran en un sitio y puede ser utilizado en reemplazo del modelo tradicional o bien en conjunto.

***Figura 6.** El artículo que introdujo la idea de Ajax tuvo un éxito inmediato que captó la atención de los creadores de sitios web.*

También cabe destacar que no se trata de una herramienta comercial ni tampoco requiere que abordemos un aprendizaje especial debido a que puede amoldarse a situaciones reales y solucionar inconvenientes cotidianos, y se puede incluir de manera progresiva en el desarrollo de un proyecto.

EL OBJETO XMLHTTPREQUEST

La función del objeto denominado **XMLHttpRequest** es la de realizar la recuperación de las respuestas desde el lado servidor sin que sea necesario actualizar de manera completa una página, y ubicar a un desarrollo basado en Ajax más cerca de una aplicación de escritorio que de una web tradicional.

Una vez que los datos han sido recuperados, mediante una comunicación entre el cliente y el servidor, y después de que se realiza la actualización de la interfaz de usuario, la persona que navega logrará notar una mayor rapidez que la usual en las aplicaciones web tradicionales y observará cómo parte del contenido de la página cambiará según las peticiones que ha efectuado.

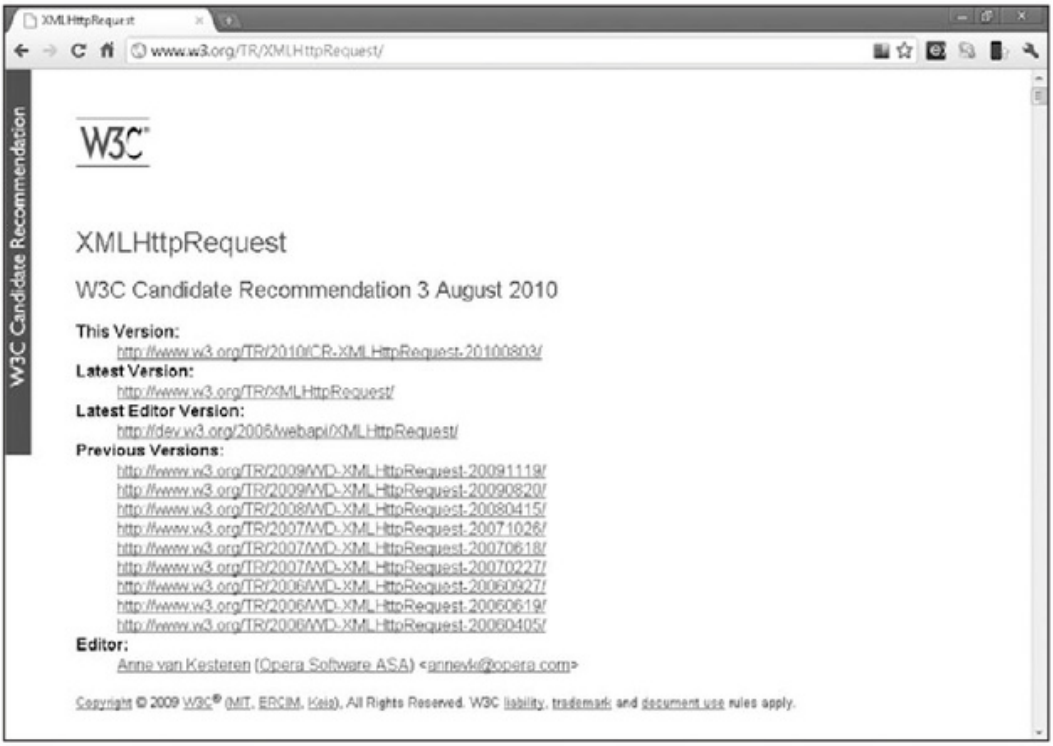


Figura 7. *Se espera que en breve la W3C lance la especificación oficial para el objeto* ***XMLHttpRequest****.*

Debemos tener en cuenta que la estructura de las peticiones y las respuestas, al igual que casi todo en el modelo, no está explícitamente definida ya que podemos utilizar fragmentos XML, texto sin formato, JSON, etcétera. Lo que sí está sobreentendido es que las respuestas recuperadas desde el servidor deberán ser pequeñas en tamaño, lo que deriva en que las actualizaciones sean instantáneas a la vista del usuario, de esta forma lograremos una navegación más fluida.

Una aplicación desarrollada con Ajax puede distinguirse de las tradicionales a simple vista por ejemplo cuando la interfaz, ante una determinada acción por parte del usuario, actualiza o refresca sus datos sin necesidad de recargar la página completa. Algunas herramientas como GMail, Google Suggest, Yahoo! o Flickr, entre muchas otras, utilizan la tecnología Ajax en sus respectivos sitios, así podemos darnos cuenta del potencial que esta herramienta pone a nuestra disposición.

III JSON

JSON (**JavaScript Object Notation**, Notación en Objetos de JavaScript) se utiliza masivamente como formato de intercambio de información entre aplicación y se postula como alternativa a XML. Su sintaxis se toma del lenguaje JavaScript, por lo que con frecuencia se utiliza en aplicaciones Ajax. Más información en el sitio que se encuentra en la dirección **www.json.org**.

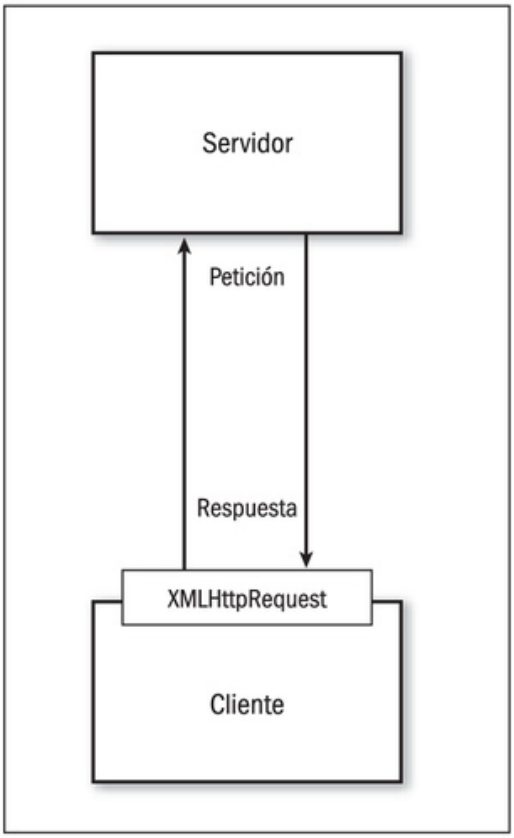


***Figura 8.** El objeto XMLHttpRequest hace de nexo entre los lados cliente y servidor de una aplicación.*

El objeto **XMLHttpRequest** se constituye actualmente como una pieza clave de la arquitectura propuesta por Ajax. Como todas los componentes del modelo, el objeto ya existía y estaba implementado en varios lenguajes, entre ellos **JScript**, **VBScript** y **JavaScript**. Dada la popularidad del lenguaje, la implementación **XMLHttpRequest** de JavaScript es la más utilizada en la actualidad. Su función básica consiste en crear un canal de conexión independiente, es decir, asincrónico, entre el cliente y el servidor. Esto permite enviar y recibir datos entre un navegador y un servidor web. El ciclo de vida típico que recorre este objeto es el siguiente:

1) se crea una instancia.
2) se prepara la petición.
3) se la envía al servidor.
4) se recibe una respuesta.

III REDUCIR DIFERENCIAS

El organismo encargado de mantener la mayoría de los estándares web, la W3C (**World Wide Web Consortium**), también se encarga de definir desde 2006 el **API** (**Application Programming Interface**) del objeto **XMLHttpRequest**. Sin dudas, esto proporcionará la disminución de las incompatibilidades entre los navegadores web utilizados.

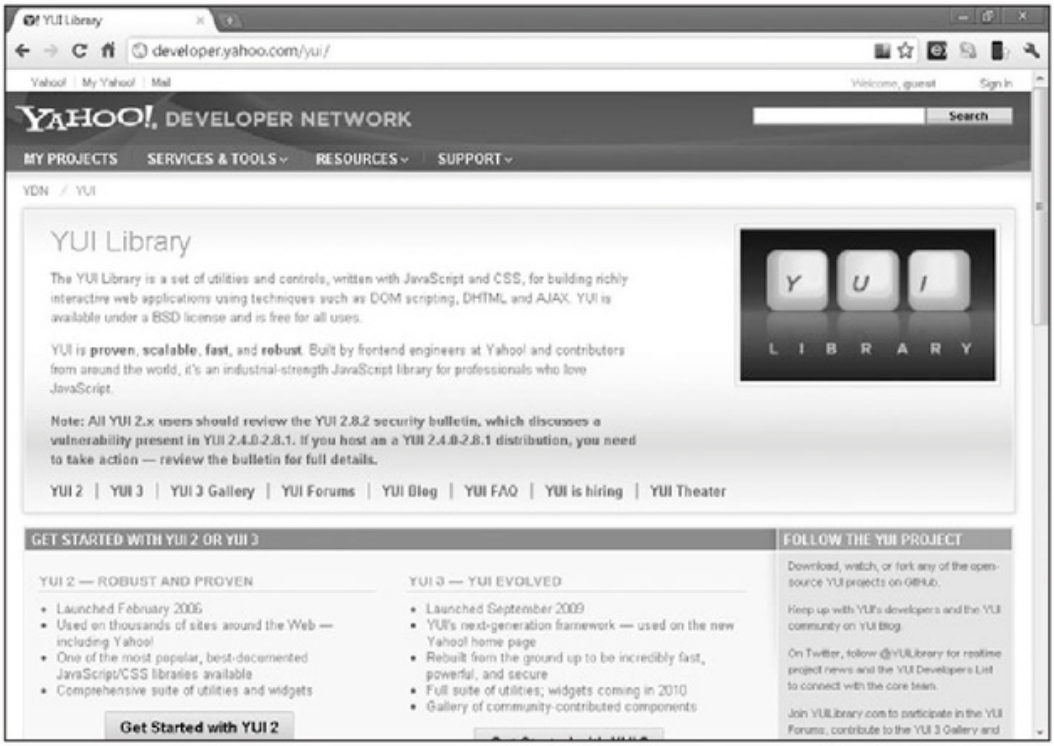


Figura 9. ***Yahoo!*** *es una de las empresas que mayor lugar le da a la generación de utilidades para implementar aplicaciones Ajax.*

La W3C ha publicado un documento para empezar a definir claramente los métodos, las propiedades y los eventos relacionados a **XMLHttpRequest** para estandarizar y documentar las funcionalidades, las características y las formas de acceso para facilitar el trabajo de los desarrolladores que deseen incorporarlo a sus proyectos y visualizar sus posibilidades en la práctica.

COMPATIBILIDAD

Al no ser un estándar oficial por el momento, los distintos navegadores mantienen sus implementaciones del objeto, incluso más de una por cada versión. Esta situación deriva en que el comportamiento de un mismo código puede variar entre navegadores si es que no se tienen en cuenta los aspectos particulares de cada una de las alternativas disponibles. El soporte ofrecido por los distintos navegadores con respecto a este objeto es el siguiente:

- **Internet Explorer** da soporte desde la versión 5, y desde la versión 7 incluye **XMLHttpRequest** como un objeto JavaScript nativo. En las anteriores lo hacía mediante un objeto **ActiveX** conocido como **XMLHTTP** y según la versión existe la posibilidad de utilizar distintos objetos ActiveX: los **Msxml2.XMLHTTP.X.X** y el **Microsoft.XMLHTTP**.

La funcionalidad del objeto nativo y el ActiveX es similar pero no idéntica ya que algunos métodos y algunas propiedades pueden variar en su comportamiento, presentando algunas diferencias. En este sentido debemos tener en cuenta que las mayores diferencias se dan en la velocidad de ejecución alcanzada.

***Figura 10. Yahoo!** es un ejemplo más de un proyecto web masivo que utiliza Ajax en diversas secciones, como por ejemplo la de registro de usuarios.*

- También podemos disfrutar de compatibilidad en los navegadores basados en el motor denominado Gecko, como ejemplo podemos mencionar los siguientes: Mozilla, Mozilla Firefox, Camino, Galeón y Netscape 7.1 y superiores, entre muchos otros que son menos reconocidos o con menos seguidores, los que han utilizado los fuentes de Mozilla en su desarrollo.
- **Konqueror** versión 3.2 y superiores.
- **Safari** de Apple versión 1.2 y superiores.

III EXTENSIONES

Es importante saber que una de las características de las aplicaciones Ajax es que no se necesitan **plugins** o controladores extras para poder interactuar con ellas incluso entre plataformas diferentes, a diferencia de lo que sucede con las aplicaciones **Flash** que eventualmente pueden cumplir una tarea similar dentro de un sitio web.

Figura 11. *El navegador Firefox es utilizado por más usuarios y mantiene un apego especial por los estándares web.*

- **Opera** versión 8.0 y superiores.
- **Chrome** y derivados.

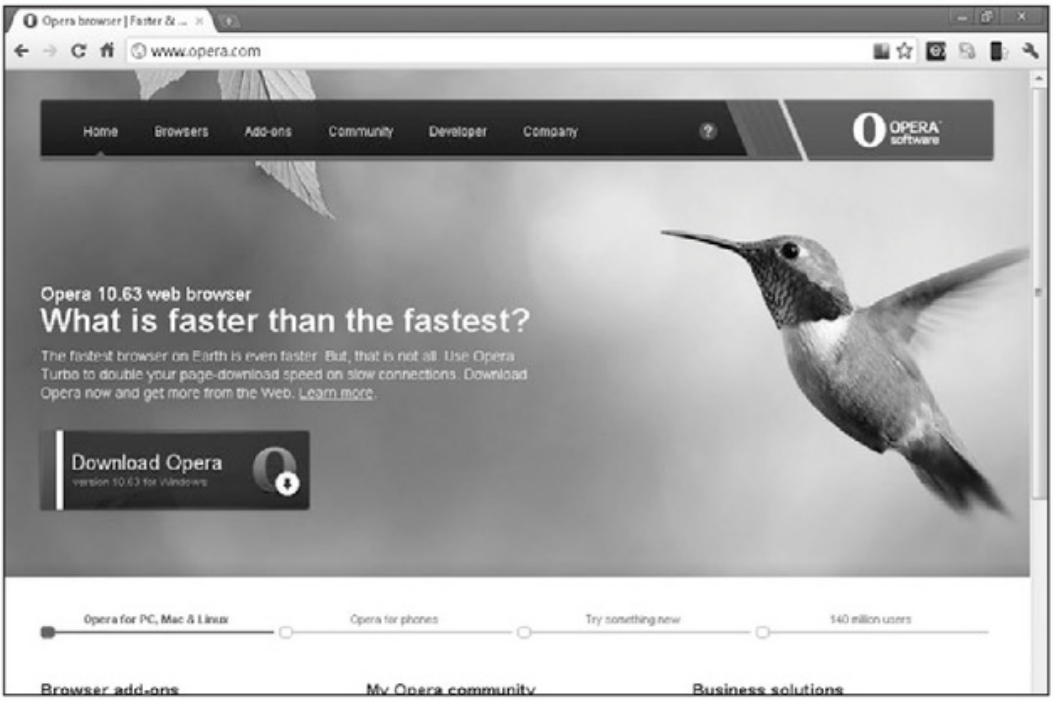


Figura 12. *El navegador Opera incorpora funcionalidades avanzadas en lo referido al trabajo con Ajax.*

Dada la popularidad del modelo, las distintas empresas trabajan constantemente en este aspecto para responder a las demandas de los desarrolladores de aplicaciones web.

DESVENTAJAS EN RELACIÓN CON EL MODELO TRADICIONAL

Como todo modelo en desarrollo, Ajax tiene limitaciones que deberemos prever antes de planificar nuestros sitios. Entre las más notables podemos citar las siguientes:

JavaScript

La mayoría de los navegadores actuales le permite al usuario desactivar total o parcialmente el soporte JavaScript. Si bien esta situación puede afectar a los sitios que no trabajan con Ajax, en este modelo se vuelve más determinante ya que es el objeto XMLHttpRequest el que permite la interacción entre el cliente y el servidor. Incluso algunos navegadores no tan nuevos no admiten la ejecución de código JavaScript.

Internet Explorer

En algunas versiones de Microsoft Internet Explorer es necesario tener activo el soporte para objetos ActiveX ya que la implementación de XMLHttpRequest así lo requiere.

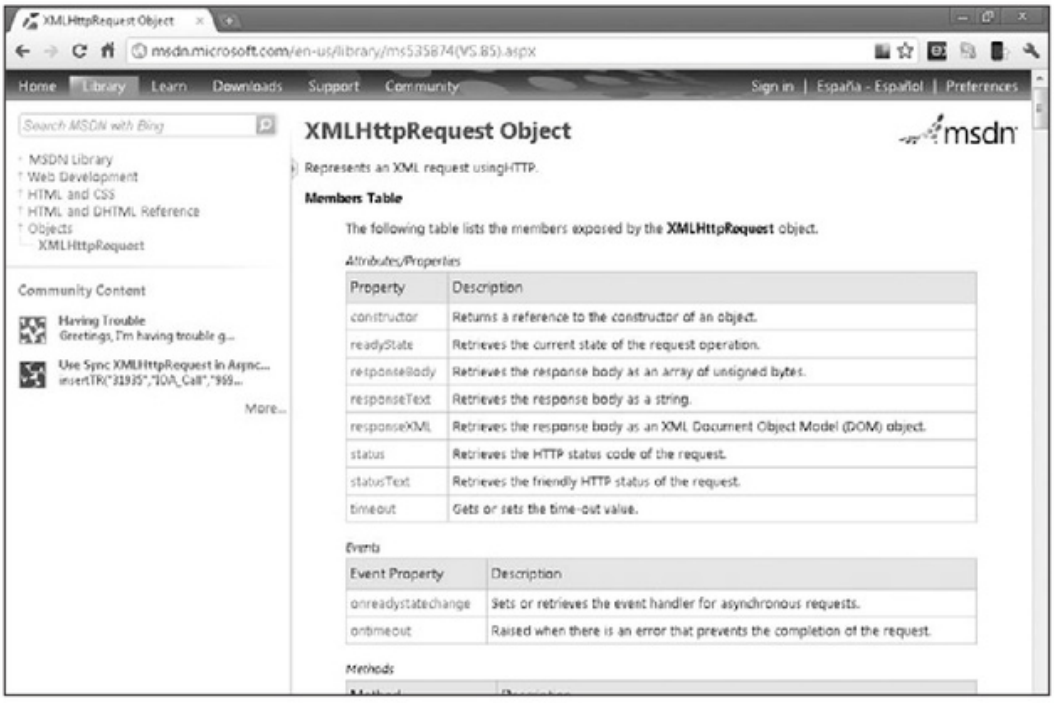


Figura 13. *Internet Explorer 7 incorpora un objeto XMLHttpRequest nativo.*

Navegadores actuales

Ajax deja en manos del lado cliente gran parte de la carga de trabajo de una aplicación, por lo que dependerá de las características del navegador que el usuario

utiliza para acceder al sitio en cuestión. Las últimas versiones de los navegadores más actuales admiten sin problemas a los sitios que trabajan con Ajax. Sin embargo, el inconveniente surge cuando el usuario, por motivos diversos, no cuenta con un navegador moderno que lo soporte.

CSS

Éste no es un inconveniente que tenga que ver directamente con Ajax, pero puede afectar el funcionamiento de las aplicaciones. El soporte brindado a CSS por parte de los navegadores difiere debido a que algunos interpretan ciertas características de una manera, otros de otra, y en algunos casos ni siquiera las interpretan. Este hecho puede afectar el diseño de la aplicación web en la que estemos trabajando.
Una solución es una función Javascript que se encarga de analizar el navegador web para implementar CSS de una manera u otra, según corresponda.

Seguridad

Acceder a datos almacenados en un servidor desde una aplicación cliente siempre supone una validación de datos y una autenticación de usuarios estrictas. Ésta no es ciertamente una limitación pero sí un aspecto de particular importancia, en base al lenguaje de programación que hayamos implementado en el servidor.

Características del modelo

Debemos tener en cuenta que en la utilización de Ajax es importante recordar algunas situaciones que no requieren atención cuando estamos trabajando en el modelo tradicional, por ejemplo la opción **Guardar Documento**, los botones de navegación **Anterior y Siguiente** o **Agregar a Marcadores**.
Cuando se actualizan solo algunas partes de una página, la URL del navegador permanece inalterable, por esta razón, al igual que lo sucedido en los sitios que han sido diseñados para trabajar con frames, la página tenida en cuenta por la aplicación no cambiará desde el punto de vista del navegador.

III WEB 2.0

Dadas sus características, el modelo Ajax suele enmarcarse dentro de la llamada **Web 2.0**, que mantiene como principio fundamental la fomentación de herramientas y tecnologías que permitan la participación activa por parte del usuario. De hecho, el movimiento Web 2.0 obtuvo mucha popularidad a partir de la aparición de aplicaciones Ajax.

Figura 14. *El desarrollo de aplicaciones Ajax demanda tener en cuenta nuevos aspectos en relación con las aplicaciones tradicionales.*

IMPLEMENTACIÓN CON PROTOTYPE

Una forma de interactuar con los objetos consiste en crear y acceder a los métodos y las propiedades de manera directa. La otra es enmascarar a XMLHttpRequest con librerías especializadas. En el siguiente ejemplo creamos el objeto de distinta manera, dependiendo del navegador desde el cual visualizamos la página:

```
<script type="text/javascript">
<!—
var XMLHttp=null
```

III INICIOS

Debemos saber que la empresa Microsoft fue la primera en introducir el objeto **XMLHTTP** (**XMLHttpRequest**), que es una pieza clave en el modelo. Luego de Microsoft fue Mozilla la que adoptó e implementó en sus propios navegadores soporte para este objeto, y así se acrecentó su popularidad. Esto propició la aparición de Ajax como modelo.

```
if (window.XMLHttpRequest) {
  XMLHttp=new XMLHttpRequest()
  alert('Mozilla / Safari / IE7 / etc')
} else if (window.ActiveXObject) {
  XMLHttp=new ActiveXObject("Microsoft.XMLHTTP")
  alert('Internet Explorer')
}

if (!XMLHttp) {

  alert('No se pudo crear la instancia');
}

//–>
</script>
```

Como ejemplo de la segunda opción tomaremos a Prototype, uno de los frameworks JavaScript que implementa Ajax. Es orientado a objetos y de propósito general, es decir que puede utilizarse incluso en proyectos en los que no intervenga Ajax, como hemos observado en el **Capítulo 4**.

Prototype introduce un objeto llamado **Ajax** que, como todos los demás objetos y todas las demás funciones busca eliminar las incompatibilidades entre los navegadores: en el caso de **Ajax**, las diferencias entre el objeto **XMLHttpRequest** de Mozilla Firefox y el objeto ActiveX de Microsoft Internet Explorer. Internamente, Prototype inicializa un objeto **XMLHttpRequest** de la siguiente manera:

```
...

var Ajax = {
  getTransport: function() {
    return Try.these(
      function() {return new XMLHttpRequest()},
      function() {return new ActiveXObject('Msxml2.XMLHTTP')},
      function() {return new ActiveXObject('Microsoft.XMLHTTP')}
    ) || false;
  },

  activeRequestCount: 0
```

```
};

...
```

Externamente, deberemos invocar los métodos necesarios para realizarle peticiones al servidor desde el lado cliente:

```
url = 'servidor.php';
var peticion = new Ajax.Request(

                url,
                {
                method: 'get',
                parameters: 'a=1&b=2',
                onComplete: funcionReceptora
                }
              );
```

El método **Request** recibe dos argumentos: la **URL** destino en donde se resolverá la petición (página, servicio web, archivo plano, etcétera) y una serie de datos en notación JSON. Algunos de los posibles son:

- **method**: método de comunicación (**POST** o **GET**).
- **parameters**: parámetros.
- **onComplete**: función que se invoca cuando se completa la transacción.
- **asynchronous**: comunicación asincrónica (**true** por defecto) o sincrónica (**false**).
- **onSuccess**: función por invocar cuando se complete satisfactoriamente la transacción.
- **onFailure**: función por invocar si surge algún inconveniente durante la transacción.
- **evalScripts**: **true** para evaluar el código JavaScript de la respuesta.

La función que tomará el control cuando se complete la transacción (conocida como **funcionReceptora**) automáticamente toma la respuesta como argumento:

```
function funcionReceptora (respuesta) {
  $('resultado').innerHTML = respuesta.responseText;
}
```

La respuesta admite las propiedades del objeto **XMLHttpRequest**, por ejemplo:

- **responseText** (respuesta devuelta por el servidor).
- **responseXML** (respuesta devuelta por el servidor, se valida como XML).
- **status** (estado de la respuesta).
- **statusText** (descripción del estado).

La propiedad **status** contiene un código numérico retornado por el servidor para representar el estado de la página solicitada. Algunos valores posibles son, por ejemplo, **404** ó **200**. En el mismo sentido, la propiedad **statusText** almacena una descripción de la propiedad **status** (por ejemplo, **Not Found** u **OK**).
En el siguiente ejemplo, recuperaremos la hora del servidor de manera asincrónica con Ajax. En el archivo **index.html**, incluiremos un vínculo a la librería Prototype y un botón para ejecutar la petición:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" src="prototype.js"></script>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 Event.observe(window, 'load', inicializar, true);

 function inicializar() {
  Event.observe($('btnObtenerHora'), 'click', realizarPeticion, true);
 }

 function realizarPeticion() {
  url = 'servidor.php';
  var peticion = new Ajax.Request(
   url,
   {
   onComplete: recibirRespuesta
   }
```

```
  );
 }

 function recibirRespuesta(respuesta) {
  $('contenedor').innerHTML = respuesta.responseText;
 }

//]]>
</script>

</head>
<body>

  <div id="contenedor"></div> <br />
  <input type="button" id="btnObtenerHora" value="Obtener Hora" />

</body>
</html>
```

Figura 15. *El archivo* ***prototype.js*** *puede descargarse del sitio oficial de la herramienta.*

El método **observe** de la clase **Event** nos permite ligar un evento de un objeto a una función, por ejemplo al cargar la página (evento **load** del objeto **Window**), se

invoca a **inicializar**. Al hacer clic en **btnObtenerHora**, se llama a **realizarPeticion**. Desde allí se ejecuta un pedido a **servidor.php** (en el ejemplo utilizaremos PHP como lenguaje, aunque puede ser intercambiado por cualquier otro), en donde se imprime la fecha y la hora actuales:

```
<?php echo date("d-m-Y . h:i:s A"); ?>
```

Una vez recuperada la respuesta, se pasa el control a la función **recibirRespuesta**, que se encarga de actualizar el contenido del elemento contenedor. Se trata de un ejemplo simple pero que sirve para ilustrar sin complicaciones innecesarias cómo funciona el proceso cliente servidor en Ajax y cómo se puede implementar de manera práctica con Prototype.

***Figura 16.** El desarrollo de aplicaciones con Prototype permite generar aplicaciones de manera rápida y consistente.*

El punto más importante de Ajax, más allá del aspecto visual que toman las aplicaciones desarrolladas bajo este modelo, es la manera en la que establece el diálogo entre el lado cliente y el lado servidor. El modelo Ajax divide al cliente en dos partes: la presentación (interfaz de usuario) y el motor Ajax (aplicación escrita en JavaScript). La interfaz de usuario se comunica con el motor Ajax mediante JavaScript (como vimos en el ejemplo con la función **realizarPeticion**), y éste le

RUBY ON RAILS

Entre los entornos de desarrollo de aplicaciones que popularizaron el desarrollo web con Ajax se encuentra **Ruby on Rails**, que utiliza el lenguaje **Ruby** e incorpora soporte para el modelo gracias a la incorporación de Prototype en su distribución oficial. Podemos obtener más información acerca de RoR en la dirección web **www.rubyonrails.org**.

envía una petición en segundo plano al servidor mediante el objeto **XMLHttpRequest** (en este caso generado por Prototype). Una vez que el servidor completó el procesamiento, le devuelve la respuesta (la salida de **servidor.php**) al motor Ajax, que a su vez actualiza datos en la interfaz (con **recibirRespuesta**) mediante DHTML y DOM (se actualiza la propiedad **innerHTML** del **div** contenedor). El resultado es que la interfaz se mantiene durante todo este proceso a la vista del usuario y solo se actualizan algunas partes. Entonces, el lado cliente toma un papel importante ya que le debe peticionar información al servidor, recuperarla, tratarla, adecuarla al contexto y actualizar elementos.

El método **Updater** es similar a **Request**, solo que se encarga de actualizar el contenido de un elemento (referenciado mediante su **id**). El método **PeriodicalUpdater** actualiza el contenido del elemento cada cierto período de tiempo (2 segundos por defecto, pero puede modificarse con la propiedad **frequency**).

DISMINUCIÓN DEL TIEMPO DE RESPUESTA Y USO DEL CACHÉ

Como mencionamos, idealmente las estructuras intercambiadas entre el lado cliente y el lado servidor de una aplicación Ajax no deben contener estilos explícitos ni ninguna clase de información relacionada con el formato, ya que de eso se encargará CSS luego. Esto es algo que forma parte de los fundamentos del modelo y que gracias a la división en capas logra reducir al máximo el tiempo de espera por parte de una aplicación.

En la utilización del caché, esto es algo que dependerá según el caso. Sin embargo, como principio básico podemos decir que con el lenguaje de programación seleccionado podemos hacer que las actualizaciones se lleven a cabo solo en los casos en los que sea necesario, es decir cuando hay algo que cambió desde la última petición. Por ejemplo, en una aplicación que exhiba mensajes de los usuarios, no hará falta actualizar la interfaz si no hubo nuevas entradas desde la última actualización.

III HERRAMIENTAS

Es importante destacar que Ajax es una nueva manera de utilizar tecnologías existentes, todas ellas conocidas y accesibles para los desarrolladores. Esto es algo que sin dudas facilita el acceso por parte de desarrolladores que ven en el modelo una manera de aplicar sus conocimientos en un sentido diferente, nuevo y con un futuro por delante.

AJAX Y LOS LENGUAJES DE PROGRAMACIÓN

En la actualidad, hay disponibles implementaciones Ajax para casi todos los lenguajes web, y la mayoría se distribuye sin costo. En las siguientes secciones, haremos un recorrido por algunos de los lenguajes y las plataformas más populares.

Java

Java es uno de los lenguajes más completos y uno de los primeros en incorporar opciones para implementar aplicaciones Ajax, entre las que podemos citar:

NOMBRE	DIRECCIÓN
ADL	http://eng.joyistar.com
Ajax4jsf	https://ajax4jsf.dev.java.net
AjaxAnywhere	http://ajaxanywhere.sourceforge.net
AJAXTags	http://ajaxtags.sourceforge.net
Backbase	www.backbase.com
BZByte EZAjax	www.bzbyte.com
DWR	http://directwebremoting.org/dwr
Echo 2	http://echo.nextapp.com/site
Google Web Toolkit	http://code.google.com/webtoolkit
Guise	www.guiseframework.com
ICEfaces	www.icesoft.com/products/icefaces.html
Java Web Objects	firestormws.sourceforge.net/design.html
Java Web Parts	http://javawebparts.sourceforge.net
Java2Script Pacemaker	http://j2s.sourceforge.net
JMaki	http://ajax.dev.java.net
JSON-RPC-Java	http://oss.metaparadigm.com/jsonrpc
JSP Controls	http://jspcontrols.sourceforge.net
JWic	http://www.jwic.de
Light Portal	http://light.dev.java.net
Moo.fx	http://moofx.mad4milk.net/#introduction

JSON

El formato de la respuesta retornada por una aplicación Ajax puede ser de cualquier tipo y dependerá de los requerimientos del sistema. Los más comunes son XML, texto plano, o **JSON**, una notación JavaScript de gran popularidad en la actualidad gracias a su versatilidad y facilidad de utilización con una amplia variedad de lenguajes de programación.

NOMBRE	DIRECCIÓN
Struts-Layout	http://struts.application-servers.com
SWATO	http://swato.dev.java.net
SweetDEV RIA	http://sweetdev-ria.sourceforge.net
Taconite	http://taconite.sourceforge.net
Tacos Tapestry	http://tacos.sourceforge.net
ThinWire	www.thinwire.com
WebORB	www.themidnightcoders.com/weborb
Wicket	http://wicket.apache.org
WidgetServer	http://www.c1-setcon.de/widgetserver
xulfaces	http://xulfaces.sourceforge.net
xWire	http://sourceforge.net/projects/xwireframework
ZK	www.zkoss.org

Tabla 1. *Algunas de las librerías y frameworks Ajax disponibles para Java.*

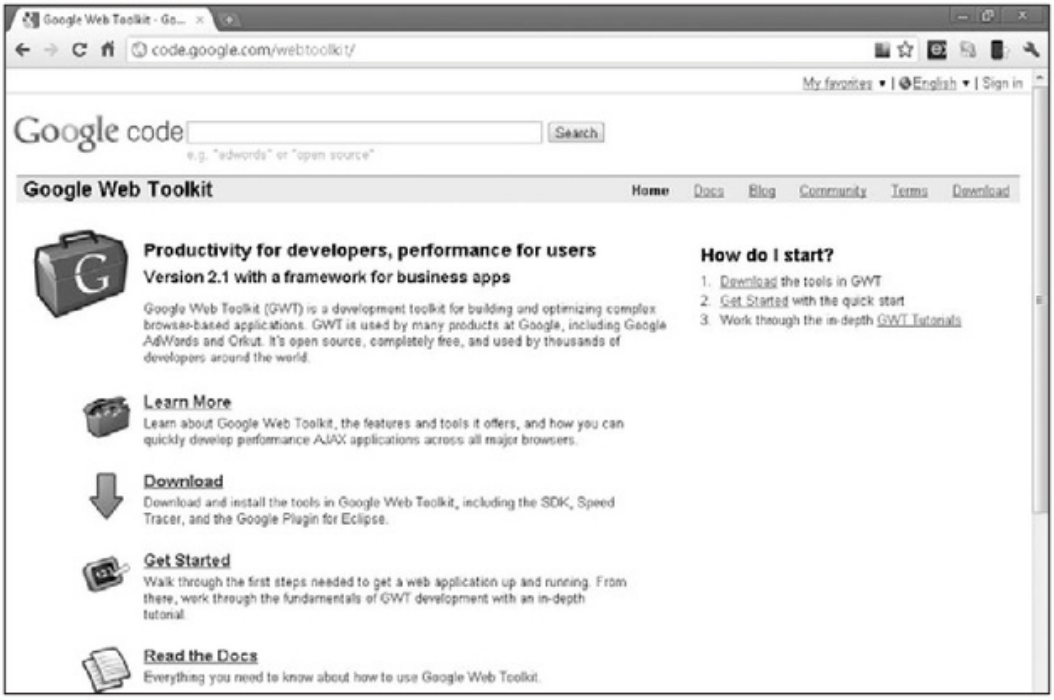


Figura 17. *El* ***Google Web Toolkit*** *es una alternativa válida a la hora de generar aplicaciones Ajax con lenguaje Java.*

III INFORMACIÓN ACERCA DE SITIOS

Veremos en los próximos capítulos como algunas herramientas nos permiten darnos cuenta si la página que estamos visitando utiliza Ajax. Además podremos conocer detalles como la url de la petición enviada y el contenido de la respuesta recibida, entre otros datos en información relevante para los desarrolladores y usuarios en general.

.Net

La plataforma **.Net** está compuesta por diversos lenguajes, por ejemplo, **C#**, **Visual Basic.Net** y **ASP.Net**. Entre los componentes más importantes para desarrollar aplicaciones Ajax, podemos nombrar a los siguientes:

NOMBRE	DIRECCIÓN
Ajax.NET	www.ajaxpro.info
AjaxAspects	www.mathertel.de/AJAXEngine
Anthem.Net	http://anthem-dot-net.sourceforge.net
Atlas	http://atlas.asp.net
Dart PowerWEB	http://dart.com/powerweb/livecontrols.asp
MagicAjax.NET	http://sourceforge.net/projects/magicajax
OutPost	http://csharpedge.blogspot.com
Telerik r.a.d. ajax	www.telerik.com
Visual WebGUI	www.visualwebgui.com
WebORB	www.themidnightcoders.com/weborb
ZumiPage	www.zumipage.com

***Tabla 2.** Algunas de las librerías y frameworks Ajax disponibles para la plataforma .Net.*

PHP

PHP es un tradicional lenguaje del lado servidor que provee múltiples opciones para implementar soluciones Ajax, entre ellas:

NOMBRE	DIRECCIÓN
AJASON	http://ajason.sourceforge.net
Ajax Agent	http://sourceforge.net/projects/ajaxagent
AjaxAC	http://ajax.zervaas.com.au
Cajax	http://sourceforge.net/projects/cajax
Claw	http://claw.tigris.org
DutchPIPE	www.dutchpipe.org
Flexible Ajax	http://sourceforge.net/projects/flxajax
Guava	http://guava.sf.net
HTML_AJAX (PEAR)	http://pear.php.net/package/HTML_AJAX
HTS	www.htsdesign.com
JPSpan	http://jpspan.sourceforge.net/wiki/doku.php
My-BIC	http://litfuel.net/mybic
NanoAjax	www.nanoajax.org
PAJAJ	http://pajaj.sourceforge.net
PAJAX	http://auberger.com/software/pajax

NOMBRE	DIRECCIÓN
phpAjaxTags	http://sourceforge.net/projects/phpajaxtags
PHPWebBuilder	http://groups.google.com/group/phpwebbuilder
Pipeline	http://livepipe.net
Qcodo	www.qcodo.com
Sajax	www.modernmethod.com/sajax
SimpleJax	http://simplejax.sourceforge.net
Stratos	www.stratosframework.com
Symfony	www.symfony-project.com
TinyAjax	http://codeigniter.com/wiki/TinyAjax
XAJAX	www.xajax-project.org
XOAD (NAJAX)	http://swik.net/NAJAX
Zephyr	http://zephyr-php.sourceforge.net

Tabla 3. *Algunas de las librerías y frameworks Ajax disponibles para PHP.*

Figura 18. Sajax *es una de las opciones más avanzadas en relación con el desarrollo de aplicaciones Ajax en PHP.*

III MODOS DE TRABAJO

Debemos tener en cuenta que, en el caso de las aplicaciones web tradicionales, cuando un navegador realiza una petición, la actividad del usuario se interrumpe hasta que se recupera la respuesta. Los procesos en el lado cliente y servidor son sincrónicos, es decir que un proceso depende del otro. En Ajax la comunicación es asincrónica.

JavaScript

Entre las librerías y los frameworks **JavaScript** de propósito general que incluyen soporte para la implementación del modelo Ajax se incluyen:

NOMBRE	DIRECCIÓN
ActiveWidgets	www.activewidgets.com
Adobe Spry	http://labs.adobe.com/technologies/spry
Ajax Toolbox	www.ajaxtoolbox.com
AjaxCaller	http://ajaxify.com/run/testAjaxCaller
AjaxGear	www.ajaxgear.com
AJFORM	http://sourceforge.net/projects/ajform
Backbase	www.backbase.com
Bindows	www.bindows.net
crossvision	www.softwareag.com
DOJO	http://dojotoolkit.org
Engine for Web Applications	www.imnmotion.com/projects/engine
Freja	www.csscripting.com/freja
HTMLHttpRequest	www.twinhelix.com/javascript/htmlhttprequest
Interactive Website Framework	http://sourceforge.net/projects/iwf
JackBe	www.jackbe.com
Javeline FrameWork	www.javeline.net
Javeline TelePort	www.javeline.org/modules/products/teleport.php
jQuery	http://jquery.com
JsLINB	www.linb.net
JSMX	http://libx.org/majax/index.html
JsRia	http://sharengo.org/Wiki?JsRIA
LibXMLHttpRequest	www.whitefrost.com
Lokris	www.ajaxbuch.de/lokris
Macao	http://macao.sourceforge.net
MAJAX	http://libx.org/majax/index.html
MochiKit	http://mochikit.com
Nitobi	www.nitobi.com
Open Rico	http://sourceforge.net/projects/openrico
Prototype	www.prototypejs.org
qooxdoo	http://qooxdoo.org
RSLite	www.ashleyit.com/rs/main.htm
Sack	www.twilightuniverse.com/?page_id=243
Script.aculo.us	http://script.aculo.us
SmartClient	www.smartclient.com
Subsys_JsHttpRequest	www.dklab.ru/lib/Subsys_JsHttpRequest
ThyAPI	www.sf.net/projects/thyapi

NOMBRE	DIRECCIÓN
TIBCO General Interface	www.tibco.com/devnet/gi
UI4W	http://ui4w.sourceforge.net/UI4W/website/index.html
vcXMLRPC	www.vcdn.org/Public/XMLRPC
XHConn	http://xkr.us/code/javascript/XHConn
Yahoo! User Interface Library	http://developer.yahoo.com/yui

Tabla 4. *Algunas de las librerías y frameworks Ajax disponibles para JavaScript.*

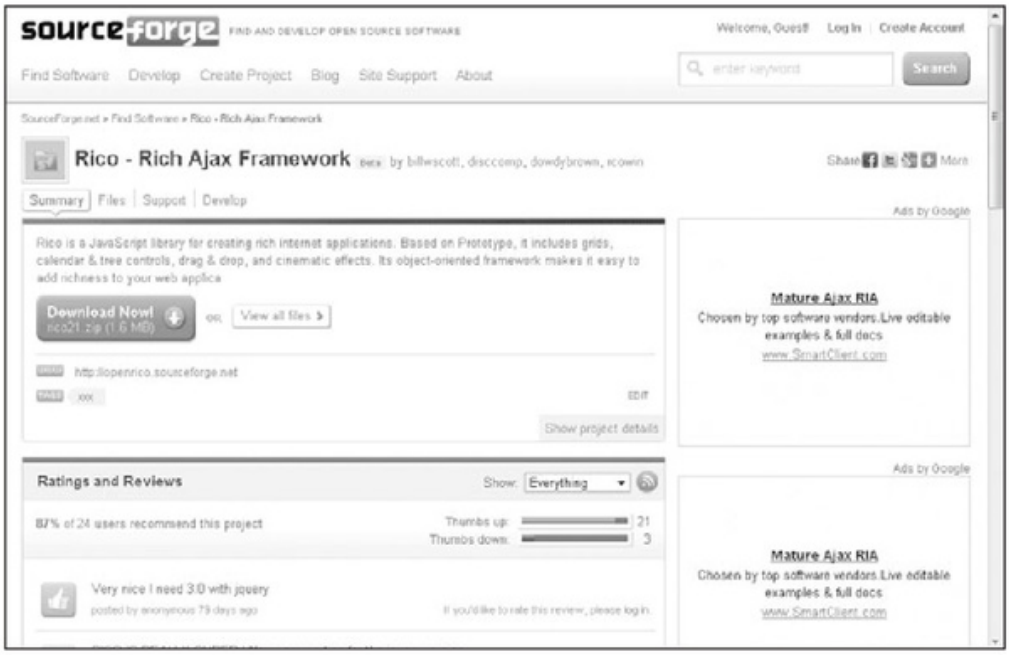


Figura 19. *En la actualidad,* ***Open Rico*** *se considera una de las alternativas más versátiles en cuanto a la implementación de controles y efectos Ajax.*

... RESUMEN

Ajax es una de las alternativas más celebradas por los autores de sitios web y sin dudas provocó un cambio en lo referente a la concepción del diseño de páginas. En este capítulo, observamos los principios generales de este modelo e incorporamos consejos prácticos de implementación. Además, introdujimos la librería denominada Prototype para acceder a ilustrar de manera práctica la generación de aplicaciones Ajax.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué significa Ajax?

2 ¿Cuáles son las diferencias en relación con el modelo tradicional?

3 ¿Cuáles son las tecnologías componentes de Ajax?

4 ¿Son las únicas? ¿Es posible reemplazarlas por otras?

5 ¿Qué es XMLHttpRequest?

6 ¿Cómo se desarrolla el ciclo de vida de una aplicación Ajax?

7 ¿En qué casos es conveniente utilizar Ajax?

8 ¿En qué casos no es recomendable utilizar Ajax?

9 ¿Qué es Prototype? ¿Para qué sirve?

10 ¿Qué entiende por lado servidor en una aplicación Ajax?

EJERCICIOS PRÁCTICOS

1 Nombre tres casos en los que utilizaría Ajax.

2 Nombre tres casos en los que no utilizaría Ajax.

3 Desarrolle una aplicación de registro de usuarios con Prototype.

4 Desarrolle una calculadora con las operaciones básicas con Prototype.

5 Desarrolle una función para enviar correos electrónicos con Prototype.

Capítulo 7

HTTP

En este capítulo observaremos algunas de las características accesibles mediante el protocolo HTTP que nos permitirán personalizar cuestiones referidas al acceso de documentos mediante un navegador web.

Introducción	**182**
Contenido comprimido	**184**
Reducción de peticiones HTTP	**189**
Contenido distribuido	**190**
Redes de distribución	**191**
Cabeceras	**191**
Posicionamiento en buscadores	194
El archivo .HTACCESS	**197**
Control de errores	199
Índices de directorio	201
Control de acceso por IP	202
Redirecciones	203
Fecha de expiración	203
Control de acceso por directorio	204
URLs cortas y amigables	205
Directivas PHP con php_value y php_flag	212
Herramientas de control	**213**
Fiddler	213
LiveHTTPHeaders	216
HttpWatch	216
Resumen	**217**
Actividades	**218**

INTRODUCCIÓN

HTTP (*HyperText Transfer Protocol*, Protocolo de transferencia de hipertexto) es el protocolo más utilizado para enviar y recibir datos por Internet. Un navegador web podría denominarse **cliente HTTP**, mientras que un servidor web sería un **servidor HTTP**. Este protocolo se impuso sobre **Gopher** a fines de 1993.
El modo general de trabajo utilizado por HTTP es el siguiente:

- un cliente HTTP abre una conexión y envía una petición.
- el servidor HTTP correspondiente recibe el mensaje y genera otro con la respuesta adecuada, y luego cierra la conexión.

Para identificar tanto a un cliente como a un servidor se utilizan direcciones **IP** (*Internet Protocol*, **xxx.xxx.xxx.xxx**) o **URLs**, que se encargan de enmascarar dichas direcciones (por ejemplo **www.servidor.com**).

A pesar de que en el mercado existen alternativas válidas, el servidor web Apache (**http://httpd.apache.org/**) se ha convertido en un estándar utilizado por un alto porcentaje de los servidores de alojamiento web.

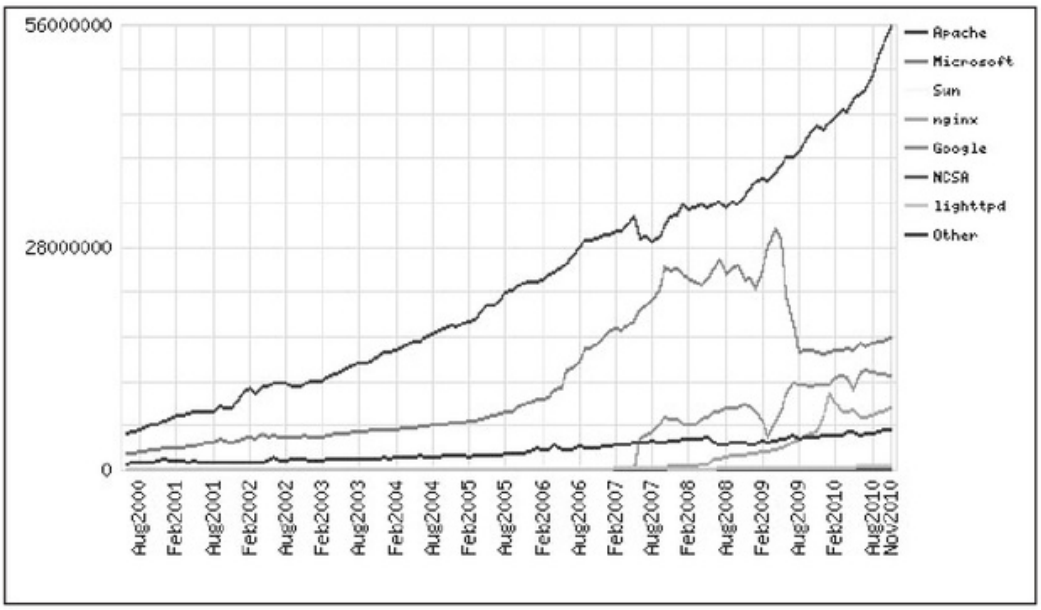


Figura 1. *Apache es el servidor web más utilizadoen la actualidad (**http://news.netcraft.com**).*

Algunos de los motivos para que esto suceda son la estabilidad de la herramienta, la compatibilidad con distintos sistemas operativos, y la facilidad de uso y mantenimiento que hacen que las cuestiones referidas a la administración de la aplicación sean simples de resolver para aquellos que se encargan del tema, sin olvidar también su gratuidad y tipo de licencia con que se ofrece en el mercado.

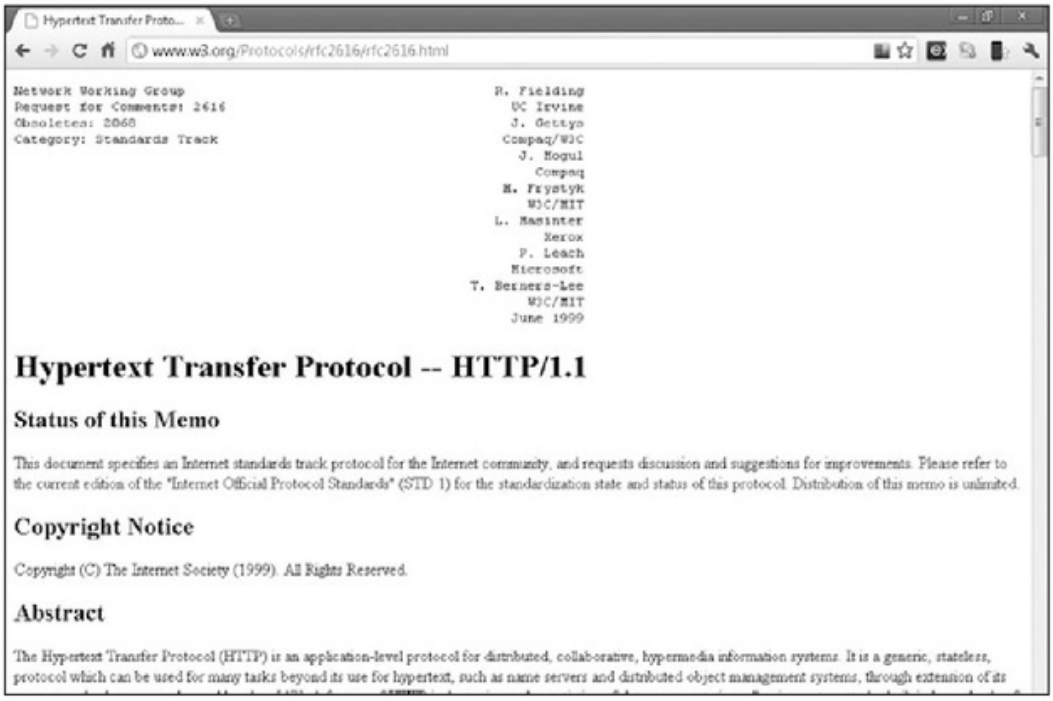


Figura 2. *El protocolo HTTP representa una de las bases más importantes del fenómeno Internet.*

Son pocos los usuarios que pueden tener a su cargo un servidor web para uso exclusivo con la posibilidad de modificar las configuraciones. Lo que en general sucede es que los servicios de alojamiento brindan una configuración estándar que cubre las necesidades generales de los clientes. Sin embargo, no siempre aprovechamos las oportunidades que este servidor nos brinda, aun cuando disponemos de ellas.

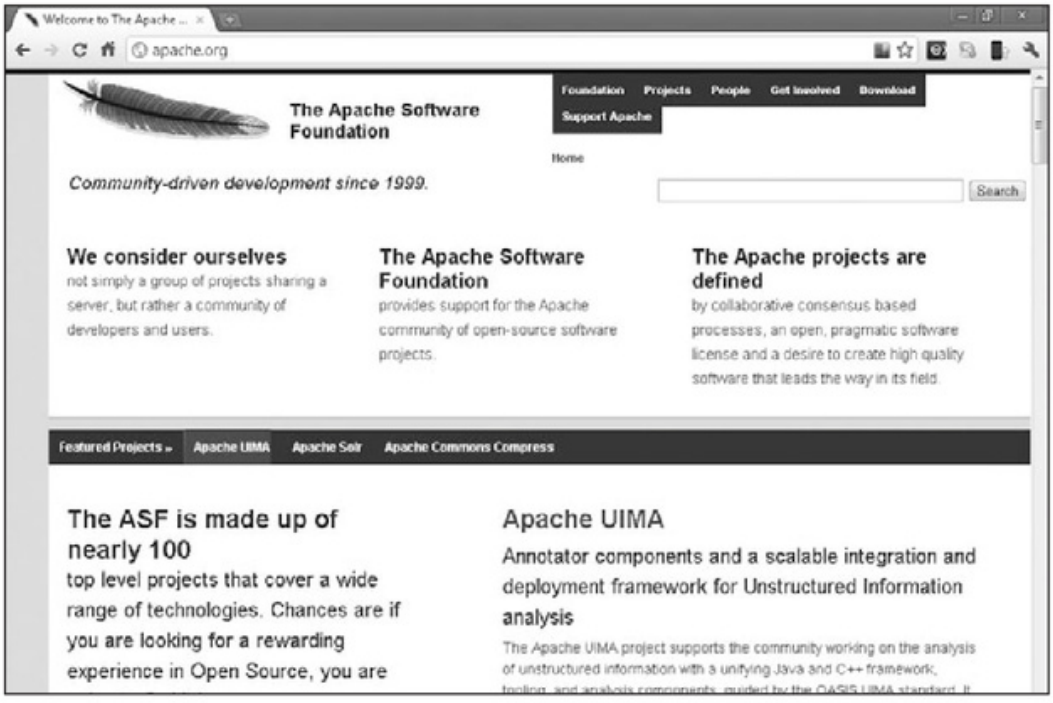


Figura 3. *El servidor Web Apache es multiplataforma y su código fuente está a disposición de quien quiera obtenerlo.*

En las siguientes secciones veremos cómo este servidor web puede ayudarnos a lograr mejoras notables en nuestras aplicaciones tanto en lo referido a la velocidad de acceso al contenido de las páginas como al posicionamiento y a la indexación de los documentos que componen nuestro sitio.

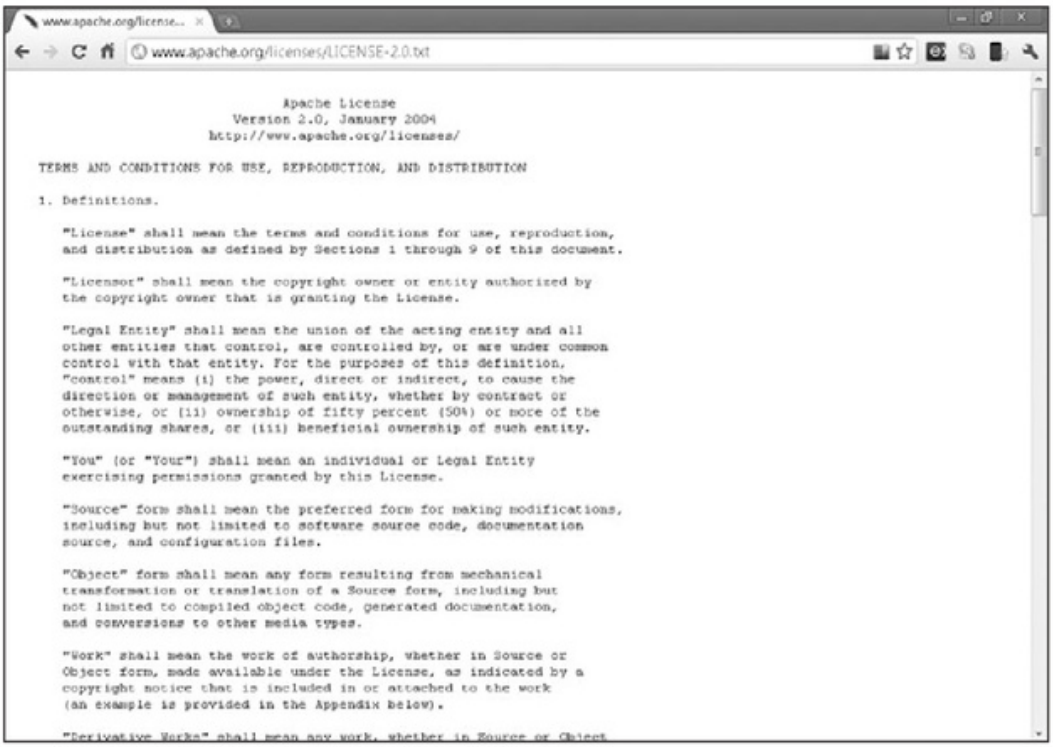


Figura 4. *El licenciamiento de Apache les permite a los servidores de alojamiento web contar con un servidor gratuito y profesional.*

CONTENIDO COMPRIMIDO

El tiempo que demora en llegar la respuesta enviada por parte del servidor depende de muchos factores, como por ejemplo la velocidad de conexión y el estado de la red. Estas cuestiones no son controlables por un desarrollador de sitios web. Algo que sí puede implementarse en este aspecto es la compresión de las respuestas HTTP mediante el método **Gzip** (más información en **www.ietf.org/rfc/rfc1952.txt**).
Gzip nos permite descargar una versión comprimida del sitio, sin tener que modificar en nada nuestro código. Si tanto el cliente como el servidor pueden descomprimir y comprimir contenidos, respectivamente, la velocidad de navegación se incrementará. Incluso ciertos motores de búsqueda como **Google** indexan contenidos comprimidos con mayor rapidez.

A pesar de que en general la utilización de los métodos de compresión no está en manos del diseñador de sitios, éstos permiten acelerar la descarga de contenidos y

brindarles un mejor servicio a los usuarios que posean conexiones lentas. Podemos encontrar más información en el sitio web oficial: **www.gzip.org**.
Apache cuenta con dos modos de compresión según utilicemos una versión u otra:

- **mod_gzip** para Apache 1.3.x y 2.0.x
- **mod_deflate** para Apache 2.0.x

Los algoritmos de compresión, como vimos en el capítulo dedicado al tratamiento de imágenes, en general poseen varios niveles. Esto también sucede en relación con la compresión de archivos a descargar mediante un navegador. A mayor nivel de compresión, más trabajo de procesamiento tendrá el servidor.

En general, no todos los tipos de archivo se comprimen antes de ser descargados al cliente. Los documentos **JavaScript**, **PDF** y los archivos de imagen son omitidos durante el procesamiento por dos motivos: para evitar problemas de compresión por parte de los navegadores y porque, en general, si se aplica compresión no se logrará reducir demasiado el tamaño original (incluso podría ser mayor). En cuanto al nivel de compresión, puede llegar a reducir hasta en un 70% el tamaño del archivo original, o más según el tipo de contenido.
Mas allá del algoritmo de compresión soportado, antes de devolver la respuesta comprimida el servidor deberá asegurarse de que el cliente, por ejemplo un navegador web, pueda comprender y exhibir el contenido devuelto. Para dar a conocer este hecho, se envía una cabecera extra en la petición, **Accept-encoding**, que puede tomar los valores:

- **Accept-encoding: gzip**
- **Accept-encoding: deflate**
- **Accept-encoding: gzip, deflate**

Una vez confirmada la compatibilidad (la gran mayoría de los navegadores la posee), el servidor aplica el algoritmo correspondiente y lo informa mediante la cabecera **Content-encoding**:

SITIO OFICIAL DE MOD_GZIP

Es importante tener en cuenta que podemos encontrar más información acerca de **mod_gzip**, la opción más utilizada por los que trabajan con el servidor Apache versión 1.3, en la página oficial del modulo, la cual se encuentra en la dirección **http://sourceforge. net/projects/mod-gzip**. De allí podemos descargar la última versión para instalarla en Apache.

- **Content-encoding: gzip**
- **Content-encoding: deflate**
- **Content-encoding: gzip, deflate**

Asimismo, se agrega la cabecera:

- **Content-Type: text/html**

Que sirve para indicarle al cliente que luego de la descarga el contenido deberá ser tratado como **text/html**. Si un cliente no puede manejar contenidos comprimidos, el servidor simplemente le enviará la respuesta original.
Con **mod_deflate** el radio de compresión es menor (**Gzip** comprime alrededor de un 6% más que **deflate**) pero se gana en velocidad de respuesta. En Apache versión 2 este módulo reemplaza a **mod_gzip**, que está disponible en la versión 1.3. Para disponer de alguno de los métodos de compresión, el módulo correspondiente deberá estar habilitado en el servidor:

```
LoadModule deflate_module modules/mod_deflate.so
LoadModule gzip_module modules/mod_gzip.so
```

Luego, desde el archivo **httpd.conf** nos será posible configurar los tipos de archivos a comprimir:

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css
```

Para definir todos los archivos de texto:

```
AddOutputFilterByType DEFLATE text/*
```

Otras opciones son:

```
<Location>
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE text/css
```

```
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/atom_xml
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/x-httpd-php
<Location />
```

Desde el archivo **httpd.conf** también podemos definir otras opciones, como por ejemplo especificar qué tipos de archivos no queremos comprimir (las primeras dos líneas indican que se comprimirá todo menos lo que excluyamos explícitamente luego):

```
SetOutputFilter DEFLATE
SetInputFilter DEFLATE

SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.(?:exe|t?gz|zip|bz2|sit|rar)$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.pdf$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.avi$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mov$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mp3$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.mp4$ no-gzip dont-vary
SetEnvIfNoCase Request_URI \.rm$ no-gzip dont-vary
```

Desde la directiva **DeflateCompressionLevel** podemos especificar el nivel de compresión entre 1 (menor compresión y procesamiento más rápido) y 9 (mayor compresión y procesamiento más lento). El valor por defecto es 6.

```
DeflateCompressionLevel 6
```

ELECCIÓN

En la versión 2.0 deApache contamos con la posibilidad de utilizar **deflate**, que es menos popular y mantiene un radio de compresión menor (es importante saber que **Gzip** logra llegar hasta niveles similares a un 70%). En este caso, Apache opta por niveles de compresión menores para lograr un mejor rendimiento, así la elección está en nuestras manos.

En cuanto a **mod_gzip**, para su utilización contamos con las directivas o funciones **mod_gzip_item_include** para incluir determinados tipos de archivo y **mod_gzip_item_exclude mime** para excluirlos:

```
mod_gzip_item_include mime ^text/.*
mod_gzip_item_include mime ^application/ms.*$
mod_gzip_item_exclude mime ^application/x-javascript$
mod_gzip_item_exclude mime ^application/pdf$
mod_gzip_item_exclude mime ^image/.*$
```

Para los sistemas operativos que necesitan configurar Apache para instalarlo, las líneas necesarias para **deflate** son las siguientes:

```
./configure
–enable-modules=all /
–enable-mods-shared=all /
–enable-deflate
```

Un sitio con muchos accesos debería privilegiar bajar el nivel de procesamiento por cada petición, por lo que tendríamos que utilizar **mod_deflate**. En el caso contrario (contar con pocas visitas), sería beneficioso utilizar más recursos y obtener mayores grados de compresión de la información.
Los servidores que soportan compresión HTTP, además de Apache, son:

- **Microsoft Internet Information Services**
- **Sun Java System**
- **Zeus**
- **Lighttpd**
- **RaidenHTTPD**
- **Serenity**

ISAPIZIP

Es una herramienta para verificar el tiempo de respuesta según los distintos tipos de conexión y la cantidad de bytes recibidos tanto si utilizáramos compresión como si no. Es una aplicación comercial y podemos descargar la versión de prueba o utilizarla desde la página **www.download3000.com/download-isapizip-count-reg-30158.html**.

A continuación, un listado con algunas herramientas online que nos permiten realizar experimentos y saber si un servidor o un cliente aceptan compresión:

- **www.pipeboost.com/report.asp** (*URL Compression Report*).
- **http://web-sniffer.net/** (nos permite definir el valor de **Accept-Encoding**).
- **www.whatsmyip.org/mod_gzip_test** (posee un espacio para verificar si tanto el cliente como el servidor poseen soporte para el módulo **mod_gzip**).

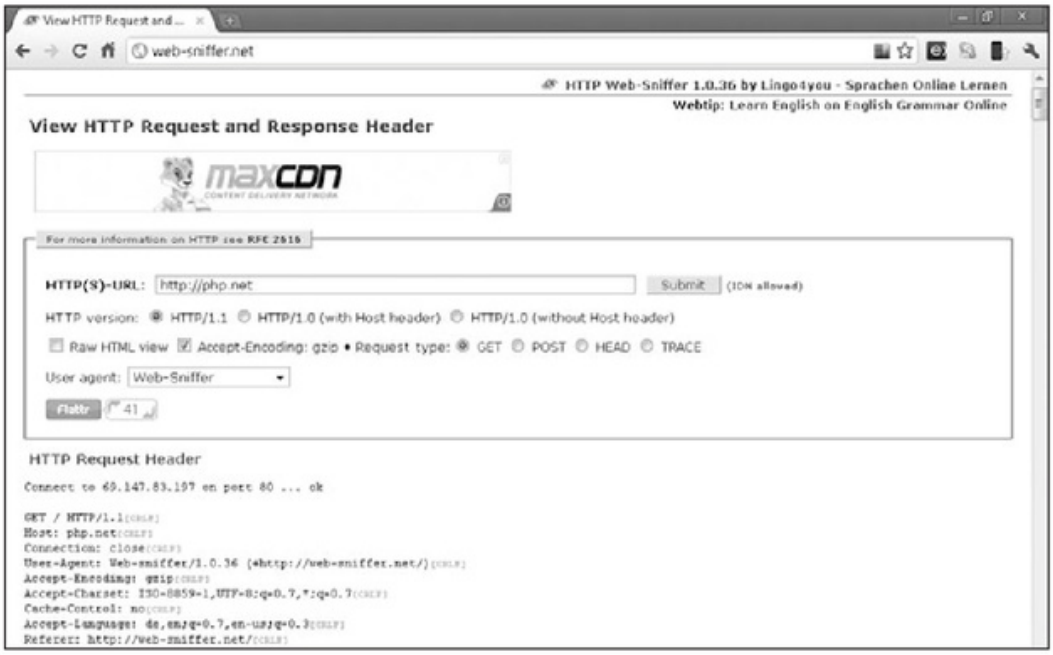


***Figura 5. Web Sniffer** es una de las alternativas disponibles para verificar el soporte de compresión por parte de los sitios web.*

REDUCCIÓN DE PETICIONES HTTP

El número de peticiones que se realizan desde una página tendrá relación directa con el tiempo de procesamiento requerido, lo que deriva en una mayor o menor espera por parte del usuario. Entre los puntos a tener en cuenta para disminuir la cantidad de peticiones por página lo más posible podemos citar los siguientes:

- ubicar en archivos externos el código JavaScript y los estilos CSS, y tratar de que sean la menor cantidad de documentos posible.
- es importante evitar el uso de marcos, ya que el navegador tendrá que descargar la página inicial y los frames referenciados en ella.
- evitar redirecciones que hacen inútil la petición a la primera página.
- incluir la menor cantidad de imágenes posible, por ejemplo mediante la utilización de mapas HTML o imágenes en conjunto.

CONTENIDO DISTRIBUIDO

El protocolo HTTP restringe la descarga simultánea de contenidos a dos por dominio. La solución implementada por las empresas que necesitan que sus sitios se visualicen lo más rápido posible en los navegadores de los usuarios es la utilización de otros dominios o incluso subdominios. Un mismo sitio se mantiene en distintos lugares. A la vista del usuario esto permite más descargas simultáneas. La ubicación de los contenidos dependerá de las necesidades de cada sitio, por ejemplo podrían almacenarse en un dominio los documentos HTML y en otro las imágenes.

Figura 6. *Yahoo! es una de las empresas que utiliza distintos dominios para almacenar su sitio.*

En el mismo sentido, una de las reglas de optimización recomienda mantener el número de peticiones DNS al mínimo. Esto se incrementa si utilizamos distintos dominios para almacenar contenidos.

DNS

Debemos tener en cuenta que los servidores denominados **DNS** (**Domain Name Server**, Servidor de nombres de dominio) se encargan de realizar la traducción del nombre de un servidor (por ejemplo **www.servidor.com**) a una dirección **IP** (**Internet Protocol**, que es un conjunto de números delimitados por puntos, por ejemplo **xxx.xxx.xxx.xxx**).

REDES DE DISTRIBUCIÓN

Al enviar una petición desde un navegador, internamente los distintos protocolos que componen Internet deberán encontrar el camino más corto o menos congestionado para obtener la información buscada, que reside en un servidor externo. **CDN** (*Content Delivery Network*) es una técnica que permite mantener copias de un mismo sitio en distintos servidores, cada uno ubicado geográficamente en lugares diferentes y alejados. Cuando se produce una petición, el sistema determina la ubicación más cercana y la selecciona para cargar los archivos desde allí.
Este tipo de servicio puede ser costoso para los sitios con poco tráfico, pero sin dudas es una alternativa muy utilizada por aquellos que requieren evitar la sobrecarga del servidor ante el elevado número de peticiones recibidas.

CABECERAS

Los elementos **meta** permiten incluir información descriptiva acerca de la página en cuestión. El número es ilimitado y se incluyen dentro del elemento **head**:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
  <meta name="keywords" content="ipod mp3 aac reproductor Apple" />
  <meta name="description" content="Bienvenido, aqui podra encontrar toda
     la informacion acerca de los reproductores iPod de Apple" />
 </head>
 <body>

...
 </body>
</html>
```

Podemos obtener más información acerca de las cabeceras HTTP existentes en las especificaciones oficiales para **HTTP/1.0** y **HTTP/1.1**:

- **www.ietf.org/rfc/rfc1945.txt**
- **www.ietf.org/rfc/rfc2068.txt**

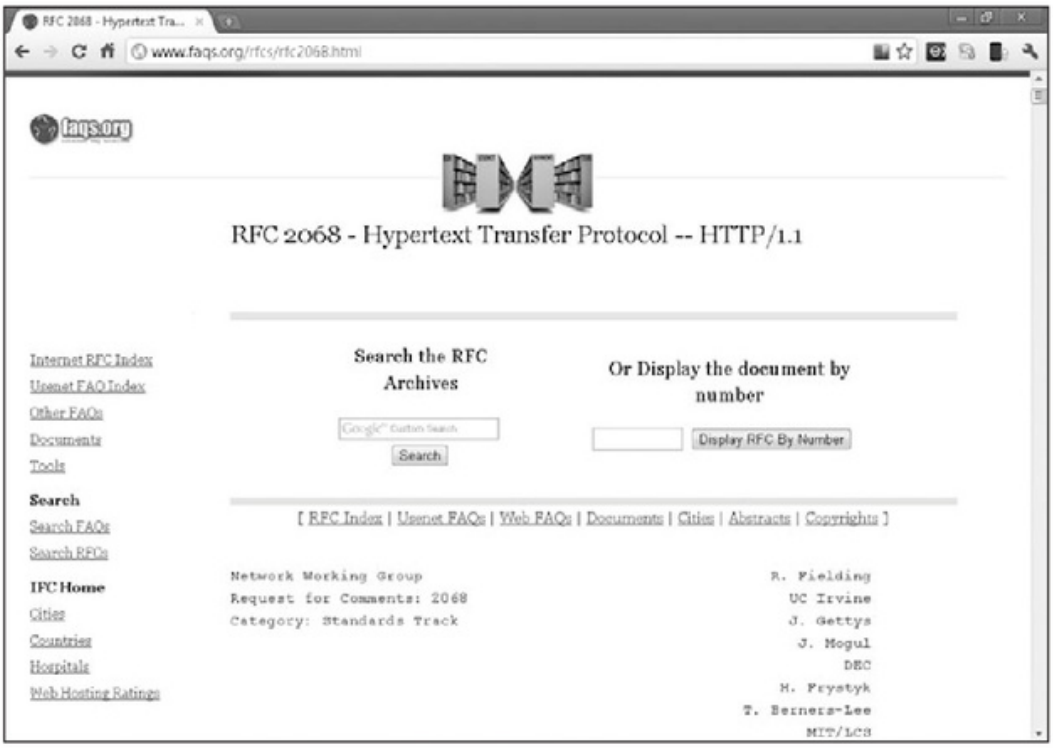


Figura 7. *Los* ***RFC*** *(****Request For Comments****) son documentos que refieren a diversos temas y dan detalles técnicos de suma utilidad.*

Debemos saber que existen dos tipos: uno para emular las cabeceras **HTTP** que normalmente el servidor le envía al navegador y otro para incorporar la información que no se incluya en el grupo anterior.
La sintaxis de la primera opción es la siguiente:

```
<meta http-equiv="nombre_de_la_cabecera" content="valor" />
```

Las etiquetas meta no afectan la visibilidad de la página. Algunas de las cabeceras más utilizadas son:

- **Refresh**, que redirige la acción a otra página luego de un número **N** de segundos:

```
<meta http-equiv="Refresh" content="5;URL=http://www.google.com/" />
```

Si no especificamos la dirección, la página se recargará a sí misma:

```
<meta http-equiv="Refresh" content="2" />
```

- **Content-Type**, para especificar el tipo de contenido y opcionalmente el juego de caracteres a utilizar, vemos un ejemplo a continuación:

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
```

- **Window-target**, se utiliza para definir la posición de la página. Por ejemplo, con la siguiente línea nos aseguramos de que no esté contenida en un marco (**frame** o **iframe**), a continuación vemos un ejemplo:

```
<meta http-equiv="Window-target" content="_top" />
```

Entre las del segundo grupo hay algunas muy importantes que están enfocadas más que nada al posicionamiento en los motores de búsqueda, pero en general no hay límite en la información a incluir. La sintaxis utilizada es:

```
<meta name="identificador" content="valor" />
```

- **Generator** es incluido por algunos editores de documentos y almacena el nombre de éstos, el siguiente código muestra su uso:

```
<meta name="Generator" content="EditPlus" />
```

- **Author** permite agregar el nombre del responsable del desarrollo de la página:

```
<meta name="Author" content="" />
```

- **Date** indica la fecha de creación de la página:

```
<meta name="Date" content="June 21, 1981" />
```

- **Copyright** define lo relacionado con los derechos intelectuales de la página:

```
<meta name="Copyright" content="2008 XXX. Todos los derechos
   reservados." />
```

• **Expires** define todo lo relacionado con la expiración de la página:

```
<meta name="expires" content="sat, 28 FEB 2009 23:15:00 EST" />
```

Para recargarla siempre y no utilizar la versión almacenada en el caché, optamos por el valor cero, vemos un ejemplo a continuación:

```
<meta name="expires" content="0" />
```

Otra opción similar es mediante la utilización de la cabecera **Cache-Control**:

```
<meta name="Cache-Control" content="must-revalidate" />
```

El valor **no-cache** evita que la página se incluya en el caché:

```
<meta name="Cache-Control" content="no-cache" />
```

Algunos navegadores utilizan **Pragma** en lugar de **Cache-Control**.

Posicionamiento en buscadores

Algunos motores de búsqueda (aunque cada vez son menos) se encargan de utilizar ciertas cabeceras para crear los índices correspondientes y devolver resultados a partir de las consultas de los usuarios, ya sea en relevamientos por palabra clave o por directorio. Si bien hoy en día los buscadores le otorgan más importancia al contenido de las páginas que a los tags **meta**, siempre es beneficioso conocer y hacer uso de las posibilidades que se encuentran a nuestro alcance. Algunas de las opciones relacionadas con este tema son las siguientes:

GOOGLE

Es importante que tengamos en cuenta que en el capítulo dedicado al motor de búsquedas Google veremos cuáles son los aspectos más importantes en relación al posicionamiento de las páginas en los listados de resultados y cómo esta herramienta logra indexar el contenido de nuestras páginas en busca de la información que sea más relevante.

- El identificador **keywords** permite incluir palabras clave y frases que definan y describan la información contenida en la página. La siguiente es la definición de la cabecera en el sitio de w3schools:

```
<meta name="keywords"
content="XML,tutorial,HTML,DHTML,CSS,XSL,XHTML,JavaScript,ASP,ADO,VBScript,D
     OM,W3C,authoring,programming,learning,beginner's
     guide,primer,lessons,school,howto,reference,free,examples,samples,sour
     ce code,demos,tips,links,FAQ,tag list,forms,frames,color
     tables,Cascading Style Sheets,Active Server Pages,Dynamic
     HTML,Internet database
     development,Webbuilder,Sitebuilder,Webmaster,HTMLGuide,SiteExpert"
     />MathML, markup, Amaya, Jigsaw, free, open source,
            software" />
```

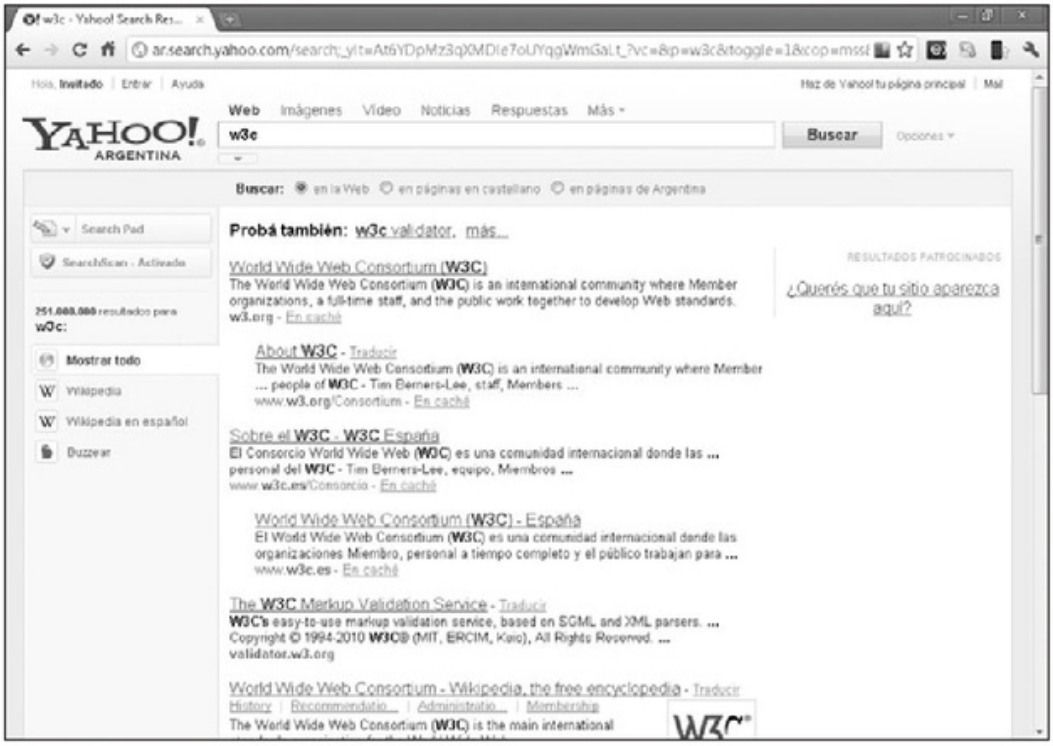


Figura 8. *Algunos motores de búsqueda utilizan las cabeceras disponibles en XHTML de manera activa.*

En la actualidad, los motores de búsqueda pueden reconocer si el autor de la página incluyó palabras populares no relacionadas con el contenido del sitio solo para ganar un mejor posicionamiento en los resultados. Por eso deberemos tener cuidado para restringir los valores de esta etiqueta y solo ingresar palabras representativas del servicio brindado. Es importante tener en cuenta que esto también vale para la siguiente cabecera: **description**, que analizamos a continuación.

- La palabra clave denominada **description** se encarga de contener la descripción del sitio que aparecerá en los listados de resultados de búsqueda. A continuación podemos ver un código de ejemplo para **description**:

```
<meta name="description" content="The World Wide Web Consortium (W3C) is
   an international consortium where Member organizations, a full-time
   staff, and the public work together to develop Web standards. W3C
   primarily pursues its mission through the creation of Web standards and
   guidelines designed to ensure long-term growth for the Web. Over 400
   organizations are Members of the Consortium. W3C is jointly run by the
   MIT Computer Science and Artificial Intelligence Laboratory (MIT CSAIL)
   in the USA, the European Research Consortium for Informatics and
   Mathematics (ERCIM) headquartered in France, Keio University in Japan,
   and has additional Offices worldwide." />
```

En la actualidad, los motores de búsqueda toman solo los primeros ciento cincuenta caracteres encontrados, por esta razón debemos tener en cuenta que la descripción debe ser precisa y no demasiado extensa ya que no se justifica ni tampoco ayudará a que aparezca en los buscadores.

- Los llamados **robots** son programas especiales que les permiten a los motores de búsqueda indexar páginas. La directiva **Robots** nos da la posibilidad de definir algunas cuestiones relacionadas con el tratamiento de la página por parte de los buscadores. Algunos de los valores posibles son:

  - **ALL** (indexar todo el contenido).
  - **INDEX** (indexa la página).
  - **NOINDEX** (no indexar la página).
  - **FOLLOW** (indexar los enlaces en la página).
  - **NOFOLLOW** (no indexar los enlaces en la página).
  - **NONE** (no indexar la página ni los enlaces).
  - **NOIMAGEINDEX** (solo indexar el texto, no las imágenes).
  - **NOIMAGECLICK** (no enlazar directamente las imágenes).
  - **NOARCHIVE** (no guardar una copia de la página en caché).

Es importante saber que la opción por defecto es la que vemos en el trozo de código que se muestra a continuación:

```
<meta name="Robots" content="INDEX, FOLLOW" />
```

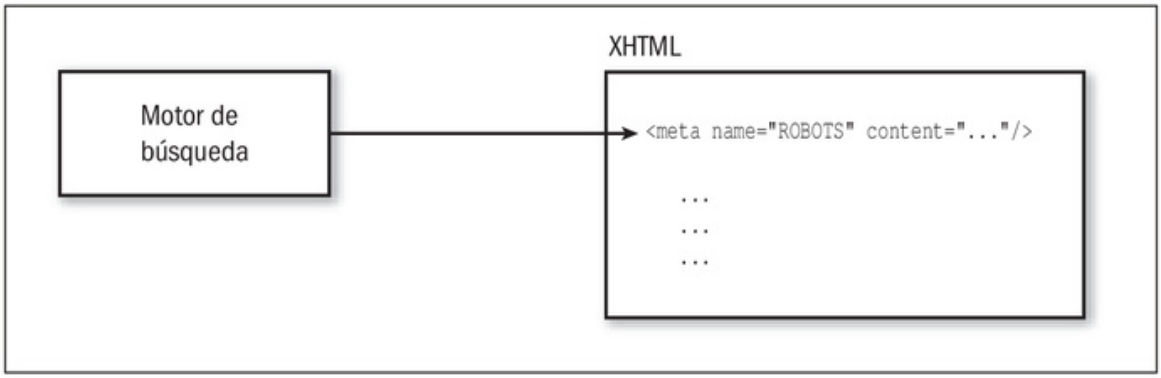


***Figura 9.** La cabecera Robots definirá el comportamiento de los motores de búsqueda frente a las páginas procesadas.*

Una alternativa disponible está dada por la utilización de los archivos **robots.txt** (en minúscula), que permiten funcionalidades similares a las de la cabecera vista anteriormente. Este archivo será procesado por los motores de búsqueda y deberá ubicarse en el directorio raíz del servidor.

EL ARCHIVO .HTACCESS

Un archivo **.HTACCESS** nos permite incluir directivas para modificar el comportamiento del servidor web. Son instrucciones simples y a la vez potentes que pueden ayudar a definir las decisiones predeterminadas por Apache.
Entre las opciones de configuración que esta clase de archivos permiten realizar podemos citar las mencionadas a continuación:

- protección (restricción) de directorios.
- evitar el listado de archivos dentro de un directorio.
- redirecciones automáticas.
- páginas de error personalizables.
- modificación en la extensión de los archivos.
- bloqueo según la dirección IP del visitante.
- modificar el documento de inicio de un directorio.

No todos los servicios de alojamiento web soportan la inclusión de archivos **.HTACCESS** por parte del usuario. En general, son aquellos que no cuentan con sistemas operativos **Unix** o **Linux**, o los que no trabajan con el servidor web Apache. Una forma de averiguar si el soporte está activo es subir un archivo **.HTACCESS** al servidor y verificar mediante un navegador si el sitio responde a la instrucciones ingresadas.
Podemos incorporar más de una directiva en un archivo y puede haber un **.HTACCESS** por directorio. Las instrucciones de un archivo tendrán validez en

el directorio actual y también en los subdirectorios correspondientes, a menos que en alguno de ellos contemos con otro archivo y de esta forma definamos la misma instrucción con un valor diferente.
Se trata ni más ni menos que de un documento de texto plano que puede ser creado con cualquier editor. Al momento de resolver las peticiones de los usuarios, el servidor buscará en cada uno de sus directorios (directiva **ServerRoot** de Apache) la aparición de un archivo de estas características. Si una misma instrucción o una nueva aparece en un directorio que está en un nivel inferior, sobrescribirá la anterior. Este proceso de búsqueda puede influir en la performance del servidor pero a cambio obtenemos un control pormenorizado de las opciones que pueden ser importantes para los proyectos que realicemos.
Algunos sistemas operativos consideran a **.HTACCESS** como nombre de archivo no válido. Generalmente, el formato requerido es un nombre seguido de un punto y una extensión, y prohíben su creación. Las posibles soluciones son guardar el archivo desde el cuadro de diálogo **Guardar Como** con el nombre **".htaccess"** (con las comillas) o almacenarlo con un nombre cualquiera, por ejemplo **txt**, subirlo al servidor y allí renombrarlo con algún cliente **FTP** (*File Transfer Protocol*). Los archivos que comienzan con un punto pueden ser considerados por estos programas como ocultos, por lo que deberemos configurarlo para que los liste.

***Figura 10.** Algunos de los clientes FTP ocultan de manera predeterminada los archivos que comienzan con un punto.*

Entre las categorías de directivas para los archivos **.HTACCESS** podemos citar:

CATEGORÍA	DESCRIPCIÓN
AuthConfig	Está referida a temas de seguridad para que los usuarios puedan proteger debidamente sus directorios y documentos.
FileInfo	Define la forma en que serán procesados los archivos.
Indexes	Afecta las directivas relacionadas con el listado de archivos en un directorio.
Limit	Es similar a AuthConfig y está relacionada con el control de acceso por parte de los usuarios.
Options	Para opciones que están por fuera de las categorías anteriormente listadas.

Tabla 1. *Categorías de directivas para los archivos* ***.HTACCESS****.*

Por cuestiones relativas a la seguridad del servicio, los proveedores de alojamiento web pueden editar el archivo **httpd.conf** de Apache, que se utiliza para configurar directivas del servidor. El objetivo es brindarle al usuario administrador del sitio la alternativa de incluir en sus archivos **.HTACCESS** solo las directivas de algunas de las categorías disponibles, y restringir así el campo de acción.

Si intentásemos incluir en un archivo **.HTACCESS** una directiva no permitida, toda petición a cualquier documento o directorio del sitio que esté al mismo nivel o inferior producirá un mensaje de error **500** (error interno del servidor).
El sitio oficial del servidor web Apache provee información importante acerca de cómo implementar y utilizar las directivas disponibles para incluirlas en un archivo **.HTACCESS**. Podemos obtener más información en las siguientes páginas:

- **www.apache.org**
- **www.apache.org/docs**
- **www.apache.org/foundation/preFAQ.html**

A continuación veremos algunos de los usos habituales de estos archivos.

Control de errores

Suele suceder que al intentar acceder a una determinada página de un sitio obtenemos un mensaje de error del tipo **404 - página no encontrada**. Con un archivo

CÓDIGOS

Debemos saber que conocer los códigos de respuesta que un servidor puede devolver ante una petición hecha desde el lado cliente es de suma importancia para encontrar y solucionar errores de manera rápida. Sin dudas, esto es de mucha importancia al momento de desarrollar sitios web, por lo tanto debemos tenerlo en cuenta.

.HTACCESS podemos personalizar el diseño y el contenido de esta página si utilizamos la directiva **ErrorDocument**:

```
ErrorDocument 404 /error.html
```

La instrucción debe estar seguida del número de error y la página a la que se pasará el control. Entre los códigos de error disponibles figuran los siguientes:

- **400** – Petición incorrecta.
- **401** – Autorización requerida.
- **403** – Acceso no permitido.
- **404** – Página no encontrada.
- **500** – Error interno del servidor.

***Figura 11.** Al momento de utilizar archivos **.HTACCESS**, los errores más típicos se deben normalmente a directivas mal escritas.*

Si la página no se encuentra en el directorio raíz del servidor podemos especificar la ruta de la siguiente manera:

III MOTORES DE BÚSQUEDA Y POSIBLES EXCLUSIONES

Los motores de búsqueda son aplicaciones inteligentes que prevén posibles intentos de aprovechar las opciones en las cabeceras de una página, como es el caso de **keywords**: no deben incluirse repeticiones de palabras ni tampoco términos que no estén relacionados con los temas tratados en la página o en el sitio que estamos desarrollando.

```
ErrorDocument numeroDeError /ruta/a/la/pagina.html
```

Otra opción es redirigir a una página de un sitio externo:

```
ErrorDocument numeroDeError http://www.sitio.com/pagina.html
```

Podemos especificar una página para cada tipo de error, por ejemplo:

```
ErrorDocument 400 /errores/peticion.html
ErrorDocument 401 /errores/autenticacion.html
ErrorDocument 403 /errores/prohibida.html
ErrorDocument 404 /errores/no_encontrada.html
ErrorDocument 500 /errores/servidor.html
```

Índices de directorio

Entre las múltiples configuraciones de un servidor web está la de definir cuáles serán los documentos índices de los distintos directorios. Por ejemplo en Apache, más precisamente en el archivo **httpd.conf**, podemos observar la directiva **DirectoryIndex**:

```
DirectoryIndex index.php index.html index.htm
```

Figura 12. *Apache cuenta con el archivo* ***httpd.conf*** *para administrar el valor de las directivas más importantes.*

En el ejemplo, si ingresamos mediante un navegador a cualquier directorio del sitio en cuestión sin especificar una página en concreto, el servidor buscará alguno de los archivos enumerados en **DirectoryIndex**: **index.php**, **index.html** o **index.htm**, en ese orden. En caso de que no encuentre ninguno, simplemente listará todos los documentos del directorio. Esto puede ser perjudicial en relación con la seguridad y el acceso a los contenidos del sitio. Podemos evitar este comportamiento con el archivo **.HTACCESS**, mediante la siguiente instrucción:

```
Options -Indexes
```

O también:

```
IndexIgnore *
```

***Figura 13.** La ausencia de índices en un directorio puede atentar contra la seguridad de los archivos contenidos.*

Control de acceso por IP

Otra directiva disponible con este archivo es la que permite negar o admitir el acceso a uno o varios visitantes según su dirección IP (*Internet Protocol*). Esta opción tiene sus limitaciones ya que muchos proveedores les brindan direcciones IP dinámicas a sus usuarios. Para impedir el acceso desde una dirección o un rango de direcciones IP, contamos con la siguiente instrucción:

```
deny from xxx.xxx.xxx.xxx
```

También contamos con una opción para negar el acceso totalmente:

```
deny from all
```

Para permitirles el acceso solo a determinados visitantes:

```
allow from xxx.xxx.xxx.xxx
```

En los casos en los que utilizamos IPs, si solo definimos el primer grupo, o el primero y el segundo, ampliaremos el rango de direcciones anuladas o permitidas.

Redirecciones

En cuanto a las redirecciones contamos con la directiva **Redirect**, que recibe como argumentos dos direcciones: la primera es la que el usuario ingresa desde la barra de direcciones del navegador, y la segunda es la ruta a la que se enviará el control. En el siguiente ejemplo, si intentamos acceder a la página **pagina.html** automáticamente se efectuará la redirección a **nuevaPagina.html**:

```
Redirect /pagina.html /nuevaPagina.html
```

Es posible redirigir a páginas de sitios externos o ubicados en subdirectorios o directorios diferentes al de la página origen. Incluso podemos aplicar esta opción a directorios completos, de la siguiente forma:

```
Redirect /nombreDirectorio /nombreNuevoDirectorio
```

El acceso a cada archivo se modificará y se buscará en el nuevo directorio indicado:

```
http://www.sitio.com/nombreDirectorio/pagina.html
http://www.sitio.com/nombreNuevoDirectorio/pagina.html
http://www.sitio.com/nombreDirectorio/imagen.png
http://www.sitio.com/nombreNuevoDirectorio/imagen.png
```

Las redirecciones duplican el número de peticiones, por lo que su uso debería regularse y en lo posible evitarse.

Fecha de expiración

Mediante directivas se puede definir cuándo usar el contenido de la memoria caché del navegador y cuándo realizar una nueva petición para actualizarlo.
Con **ExpiresActive**, que puede recibir los valores **on** / **off**, habilitamos o deshabilitamos la posibilidad de utilizar esta opción. Con **ExpiresDefault** especificamos el período durante el cual se acudirá al caché:

```
ExpiresActive On
ExpiresDefault “modification plus 2 years”
```

Lo anterior define como fecha de expiración dos años a partir de la copia del archivo al servidor. Al sobreescribir el archivo, los usuarios que accedan por primera vez verán la versión actualizada, los demás no, por lo que esta técnica no es útil en todos los casos.
Podemos obtener más información en el sitio oficial de Apache, más precisamente en la sección que habla del módulo **mod_expires**, en: **http://httpd.apache.org/docs/2.0/mod/mod_expires.html**.

Control de acceso por directorio

Otra alternativa de importancia es la referida a la protección de los directorios. En este caso deberemos agregar un archivo **.HTACCESS** dentro de cada directorio que queramos restringir, y cada uno tendrá que contar con las siguientes líneas:

```
AuthName “Nombre de la seccion”
AuthType Basic
AuthUserFile /ruta/al/archivo/.htpasswd
Require valid-user
```

- la primera indica el identificador de la sección a proteger, que puede ser cualquier nombre, por ejemplo “clientes”.

CONOCER LOS ERRORES EN DETALLE

Es importante tener en cuenta que una manera eficaz para realizar la verificación de los errores cometidos mediante la utilización de un documento **.HTACCESS** es visualizar el archivo de errores del servidor, que en Apache por defecto se denomina **error.log**. Los errores más típicos normalmente se deben a directivas que se encuentran mal escritas.

- la segunda define el tipo de autenticación.
- la tercera es la ruta al archivo **.HTPASSWD**.
- la cuarta implica que el acceso restringido se habilitará.

Deberemos generar nuestro propio archivo **.HTPASSWD**, que estará compuesto por líneas con los datos de acceso de cada usuario (una línea por usuario):

```
nombreDeUsuario:password
```

Cada vez que alguien intente ingresar a un directorio restringido por un archivo .HTACCESS, se pedirá un nombre de usuario y una contraseña. La contraseña deberá estar encriptada y en la actualidad existen múltiples alternativas para hacerlo:

- **www.z-host.com/scripts/ipasswd**.
- **http://tools.dynamicdrive.com/password**.
- **www.htaccesstools.com/htpasswd-generator**.

Es posible ingresar a un directorio restringido mediante una URL que deberá tener la siguiente sintaxis (con la contraseña sin encriptar):

```
http://nombreDeUsuario:password@www.sitio.com/directorio/
```

URLs cortas y amigables

Los sitios que recuperan información desde bases de datos y la procesan mediante algún lenguaje del lado servidor (**PHP**, **ASP.net**, **JSP**, o **Perl**, por ejemplo) para generar las páginas componentes, normalmente requieren pasar por URL los parámetros necesarios para obtener lo que se busca:

```
http://www.sitio.com/articulos.php?codigo=123&modo=2
```

Esta forma de acceso mantiene una serie de dificultades:

- las direcciones no son indexadas por algunos motores de búsqueda.
- las direcciones no son fáciles de recordar para un usuario.

El primer punto tiene que ver con algunos de los símbolos contenidos, por ejemplo el de pregunta (**?**), el de igualdad (**=**) o el ampersand (**&**). En cuanto a la dificultad

para memorizar una ruta, la persona que accede a una página en general no podrá retener por mucho tiempo el nombre del archivo (**articulos.php**) y el **query string** con los argumentos y sus respectivos valores (**codigo=123&modo=2**).
El archivo **.HTACCESS** provee, utilizado en conjunto con algún lenguaje de programación, diversas alternativas para lograr facilitar y reducir las URLs por defecto. En los siguientes ejemplos utilizaremos el lenguaje de programación **PHP** (**www.php.net**) por su extenso uso y la simpleza de su sintaxis, pero podríamos emular su funcionalidad con cualquiera de las demás alternativas existentes hoy en día.

Lo que intentaremos lograr es pasar de una dirección como ésta:

```
http://www.sitio.com/articulos.php?codigo=123&modo=2
```

A otra más sencilla y agradable como la siguiente:

```
http://www.sitio.com/articulos/123/2
```

El primer método utiliza la directiva **ErrorDocument**, para redirigir cualquier ingreso a un documento no encontrado hacia una determinada página:

```
ErrorDocument 404 /control.php
```

Si el usuario ingresa una dirección como la siguiente:

```
http://www.sitio.com/noticias/2008/11
```

Mediante el archivo **.HTACCESS** se pasará el control a la página **control.php**, en donde podremos recuperar los argumentos incluidos mediante algunas funcionalidades

EXPRESIONES REGULARES

Es importante saber que las llamadas expresiones regulares permiten tomar como entrada una cadena de caracteres, buscar patrones específicos (por ejemplo, secuencias de números o determinados signos o palabras) y posteriormente recuperarlos o reemplazarlos por otros. Apache cuenta con soporte para el manejo de expresiones regulares.

del lenguaje. Una consecuencia de utilizar este método consiste en que las peticiones serán tomadas por Apache como errores de acceso y agregados al archivo de errores, por defecto **error.log**.

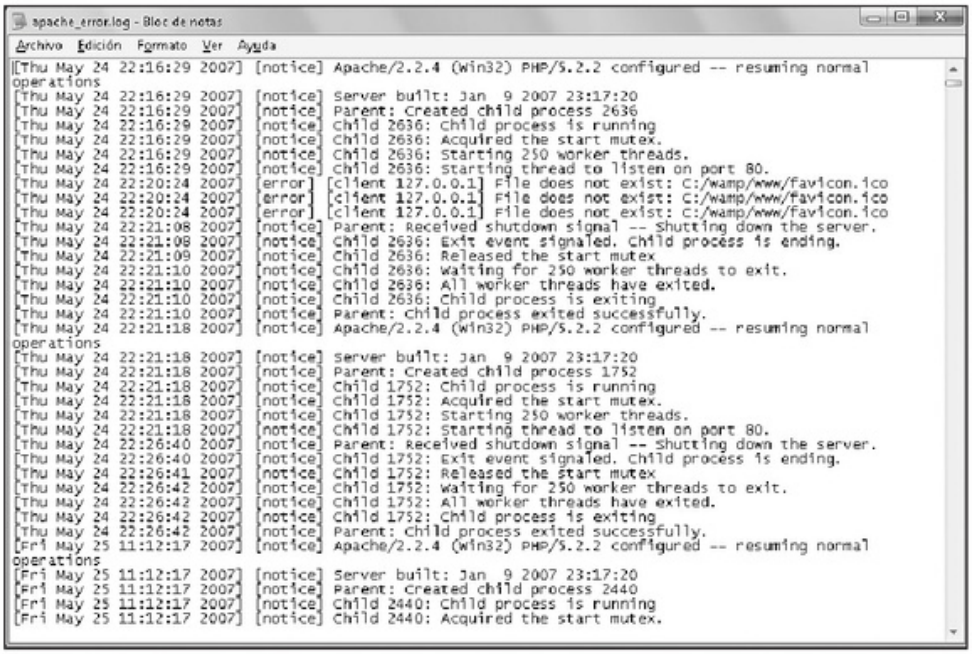


Figura 14. *El registro de Apache puede ayudarnos a obtener información detallada acerca de los errores cometidos en un documento.*

El contenido del archivo **control.php** es el siguiente:

```
<?php

header("HTTP/1.1 200 OK");

$uri = explode("/", $_SERVER[REQUEST_URI]);

$seccion = $uri[1];
$_GET['anio'] = $uri[2];
$_GET['mes'] = $uri[3];

if ($seccion == 'noticias') {
  include "noticias.php";
} else {
  echo '<h1>Pagina no encontrada !</h1>';
}

?>
```

La primera línea le indica al servidor que reconozca a la página como correcta, y no como no encontrada (error **404**):

```
header("HTTP/1.1 200 OK");
```

El array especial **$_SERVER** está disponible en PHP y nos permite acceder a información relacionada con el servidor, como la dirección invocada luego del dominio (**REQUEST_URI**):

```
$uri = explode("/", $_SERVER[REQUEST_URI]);
```

***Figura 15.** Con un archivo **.HTACCESS** podemos redirigir las peticiones y sus respectivos argumentos.*

La función **explode** permite generar un array a partir de un delimitador (primer argumento) y una cadena de caracteres (segundo argumento). A continuación almacenamos la sección, el año y el mes ingresados en variables:

```
$seccion = $uri[1];
$_GET['anio'] = $uri[2];
$_GET['mes'] = $uri[3];
```

Si la sección es igual a **noticias**, cargamos la página **noticias.php**. De lo contrario, indicamos el error con un mensaje:

```
if ($seccion == 'noticias') {
  include "noticias.php";
```

```
} else {
  echo '<h1>Página no encontrada !</h1>';
}
```

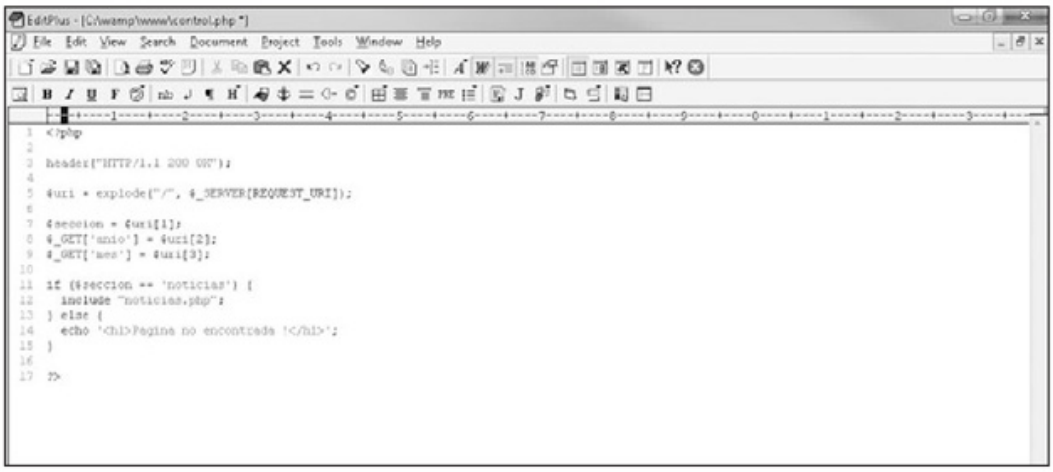


Figura 16. *Es posible centralizar el control para cada petición en un solo archivo dinámico.*

Ya en **noticias.php** podremos recibir las variables del array especial **$_GET** y trabajar normalmente como en cualquier página:

```
<?php

echo '<h1>La pagina actual es noticias.php </h1>';
echo '<pre>';
print_r($_GET);
echo '</pre>';

?>
```

Figura 17. *El archivo* ***.HTACCESS*** *puede ayudarnos a mejorar la interacción con los usuarios de nuestros sitios.*

En el ejemplo anterior suponemos que todos los archivos se encuentran en el directorio raíz del servidor web, en caso contrario deberemos modificar los archivos **.HTACCESS** y **control.php**, según corresponda:

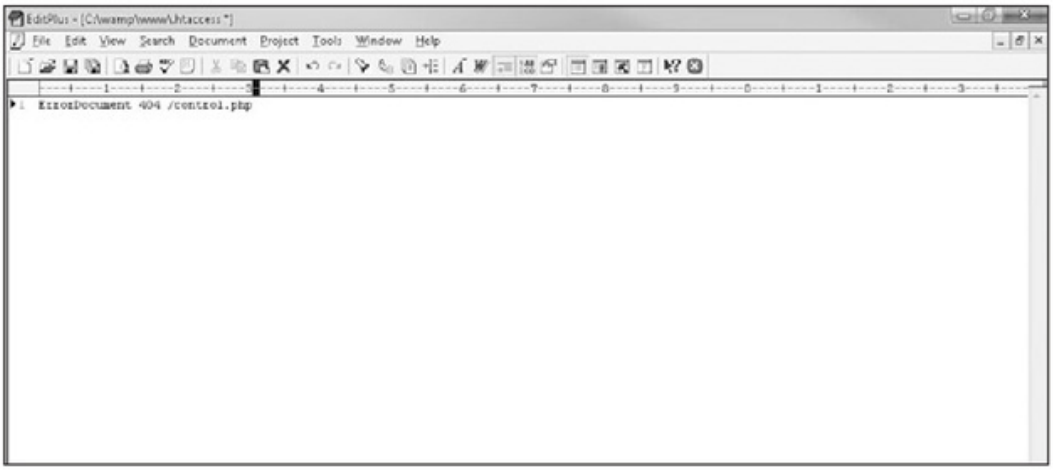


Figura 18. *El archivo* ***.HTACCESS*** *redirige la acción a* ***/control.php****, que deberá estar ubicado en el directorio inicial del servidor.*

Otra opción para reducir la longitud y la dificultad de las direcciones se implementa al hacer uso de una característica de Apache que consiste en tomar la dirección escrita y buscar de derecha a izquierda la existencia de un archivo o un directorio, por ejemplo si tomamos en cuenta la siguiente URL:

```
http://www.sitio.com/a/b/c/d
```

Apache intentará encontrar una aparición siguiendo el orden preestablecido y emitirá un mensaje de error **404** si no encuentra ninguno:

```
http://www.sitio.com/a/b/c/d
http://www.sitio.com/a/b/c
http://www.sitio.com/a/b
http://www.sitio.com/a
```

La idea de este método es que por lo menos el primer argumento exista (**a**) y los demás se tomen como argumentos:

```
http://www.sitio.com/productos/televisores/sanyo
http://www.sitio.com/productos/televisores
http://www.sitio.com/productos
```

En este caso podemos crear un archivo llamado **productos** y establecer que Apache lo interprete como un determinado tipo de documento aun cuando no tenga extensión. Esto se puede lograr, entre otras alternativas, con el archivo **.HTACCESS**:

```
<Files productos>
ForceType application/x-httpd-php
</Files>
```

Así, el archivo **productos** (sin extensión) será procesado por Apache como si se tratase de **productos.php**, y el contenido es el siguiente:

```
<?php

 $uri = explode("/", $_SERVER[PATH_INFO]);
 $articulo = $uri[1];
 $marca = $uri[2];
  echo "<li>$articulo";
  echo "<li>$marca";

?>
```

El array especial **$_SERVER** está disponible en PHP y nos permite acceder a información relacionada con el servidor, como la ruta definida en la URL tomando como referencia el archivo actual (**PATH_INFO**). A partir de esa información podemos obtener los valores de los argumentos:

```
$uri = explode("/", $_SERVER[PATH_INFO]);
```

***Figura 19.** Los lenguajes del lado servidor mantienen funcionalidades para acceder y manipular la URL de la página actual.*

A diferencia del método anterior, aquí no se toman los accesos como errores, por lo que no se agregan entradas al archivo **error.log** de Apache. Por otra parte, pueden crearse múltiples referencias en cuanto al manejo de archivos en el **.HTACCESS**:

```
<Files productos>
ForceType application/x-httpd-php
</Files>
<Files noticias>
ForceType application/x-httpd-php
</Files>
```

Éstas son solo algunas de las funcionalidades ofrecidas por los archivos de configuración provistos por Apache para ser modificados por los responsables de los sitios web.

Directivas PHP con php_value y php_flag

Con el archivo **.HTACCESS** también es posible modificar directivas de configuración del lenguaje de programación PHP. Debido a que es un aspecto sensible para el funcionamiento de un sitio, deberemos contar con los privilegios necesarios para hacerlo:

```
AllowOverride Options
AllowOverride All
```

- con **php_value** podemos modificar la directiva dada como primer argumento, asignándole el valor pasado como segundo argumento. Veamos algunos ejemplos:

```
php_value error_log /ruta/a/directorio/errores.log
php_value error_reporting 1
```

Se utiliza **none** para borrar valores previos y no asignar ninguno.

- **php_flag** es similar a **php_value**, solo que se utiliza cuando las directivas poseen valores booleanos (**on** u **off**). Veamos algunos ejemplos:

```
php_flag display_startup_errors off
php_flag display_errors on
php_flag html_errors off
```

Tanto **php_value** como **php_flag** pueden trabajar con directivas que estén categorizadas bajo **PHP_INI_ALL** y **PHP_INI_PERDIR**. Podemos obtener un listado de las disponibles en cada caso si visitamos el sitio web que se encuentra en la dirección **www.php.net/manual/es/ini.php**.

HERRAMIENTAS DE CONTROL

El flujo de peticiones y respuestas enviadas mediante el protocolo HTTP puede ser controlado y monitoreado por medio de algunas aplicaciones específicas para tal fin. En los siguientes apartados veremos cómo **Fiddler**, **LiveHTTPHeaders** y **HttpWatch** pueden ayudarnos en nuestros proyectos.

Fiddler

Esta herramienta nos permite inspeccionar el tráfico HTTP entre un cliente y un servidor, y es posible visualizar la información de una manera ordenada y clara.

***Figura 20.** El sitio web oficial ofrece información técnica para utilizar los servicios brindados por la aplicación.*

Fiddler se descarga de manera gratuita de **www.fiddlertool.com/Fiddler2/version.asp** y debemos tener el **Microsoft .NET Framework version 2** instalado previamente. Por defecto, monitorea el tráfico generado por medio de los navegadores Internet Explorer

y Mozilla Firefox, pero podemos personalizar la petición hacia un determinado servidor desde dentro de la aplicación mediante la pestaña **Request Builder**.

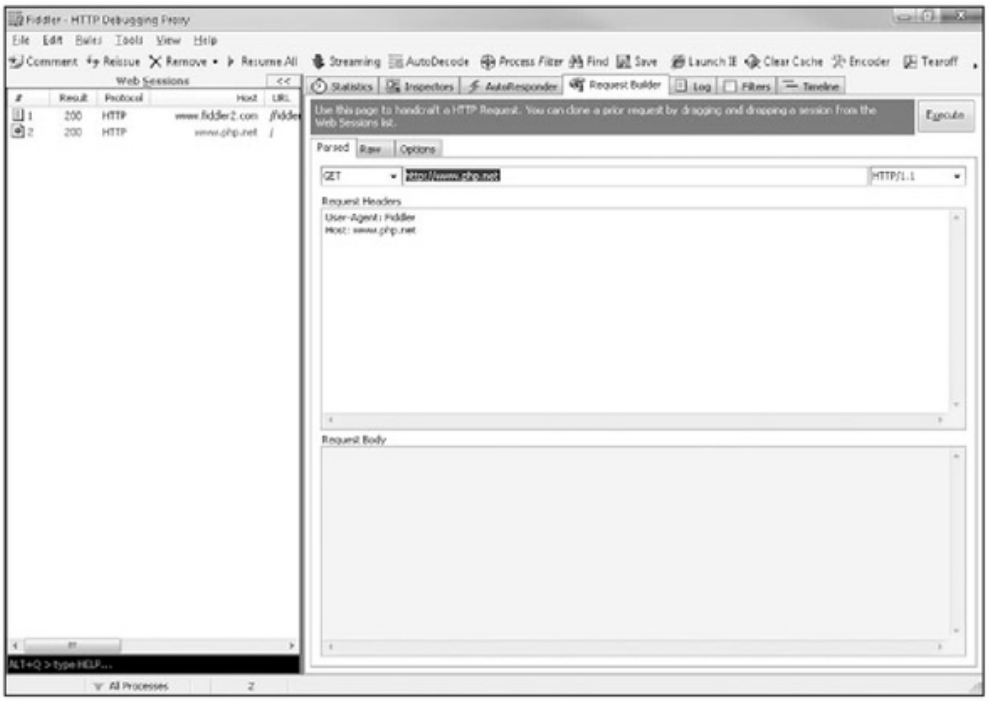


Figura 21. *Podemos especificar manualmente el servidor sobre el que queramos trabajar.*

Una vez realizada la petición y si la seleccionamos en el panel izquierdo, podemos obtener estadísticas acerca de la respuesta (solapa **Statistics**), como por ejemplo bytes enviados, bytes recibidos (dividido entre cabeceras HTTP y texto), tiempo de respuesta y tipo de respuesta, entre otras.

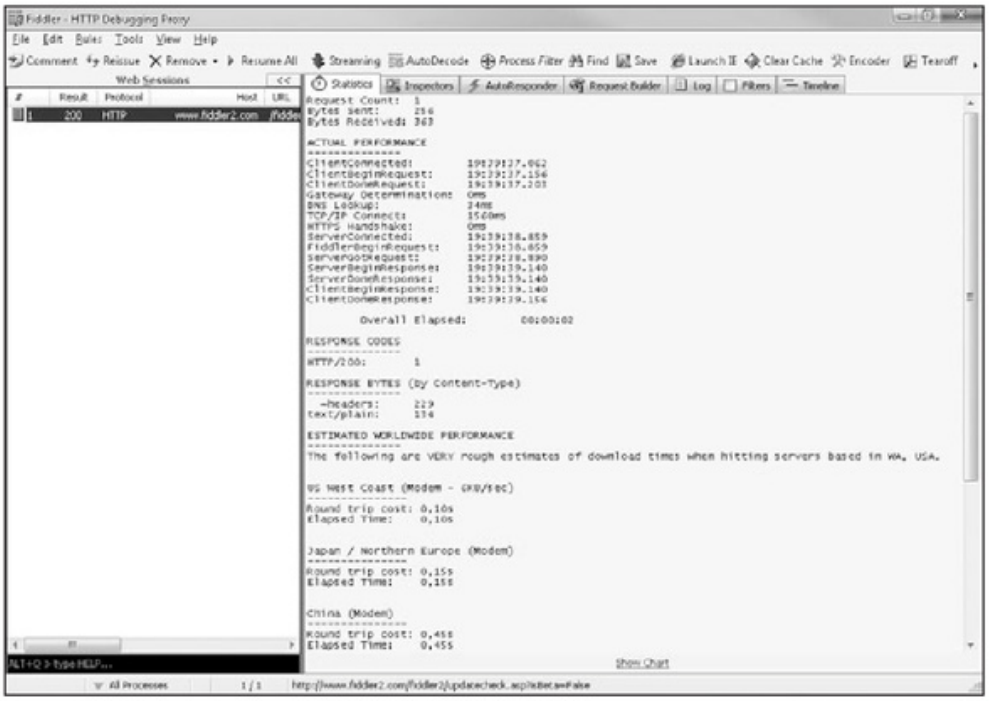


Figura 22. *Las estadísticas obtenidas pueden observarse de manera clara y comprensible.*

Desde **Session Inspector** podemos visualizar el tamaño de la respuesta (sección **Transformer**) según algunas de las opciones disponibles (sin comprimir, comprimida con **Gzip** o comprimida con **Deflate**).

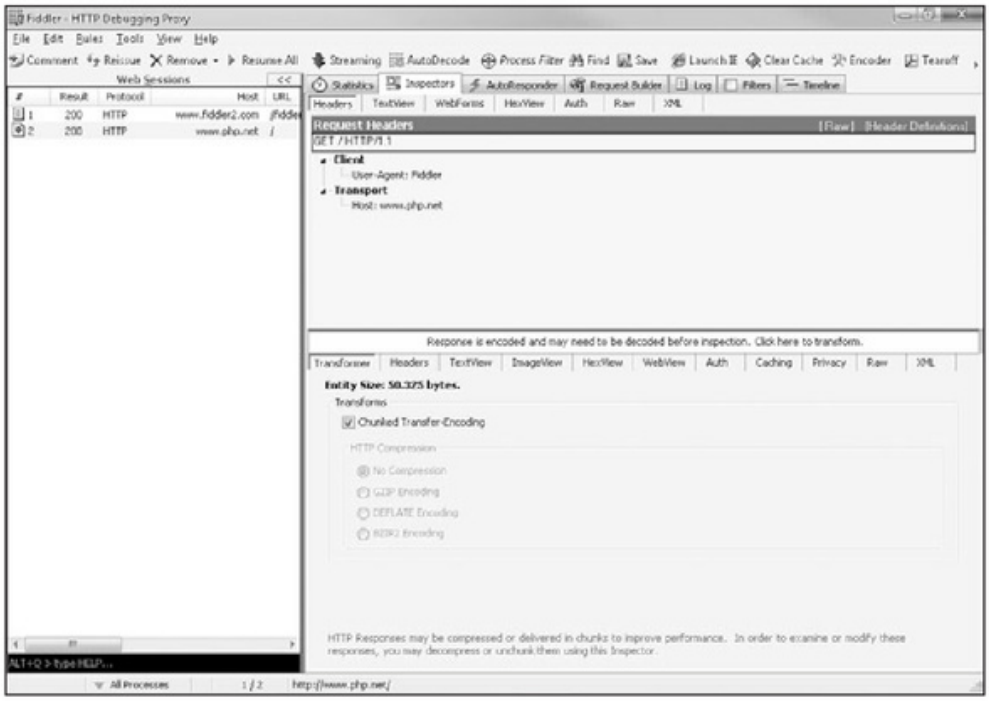


Figura 23. *Podemos obtener el peso total de la respuesta obtenida con alguno de los métodos de compresión disponibles.*

Las cabeceras recibidas pueden observarse en **Headers** de **Session Inspector**.

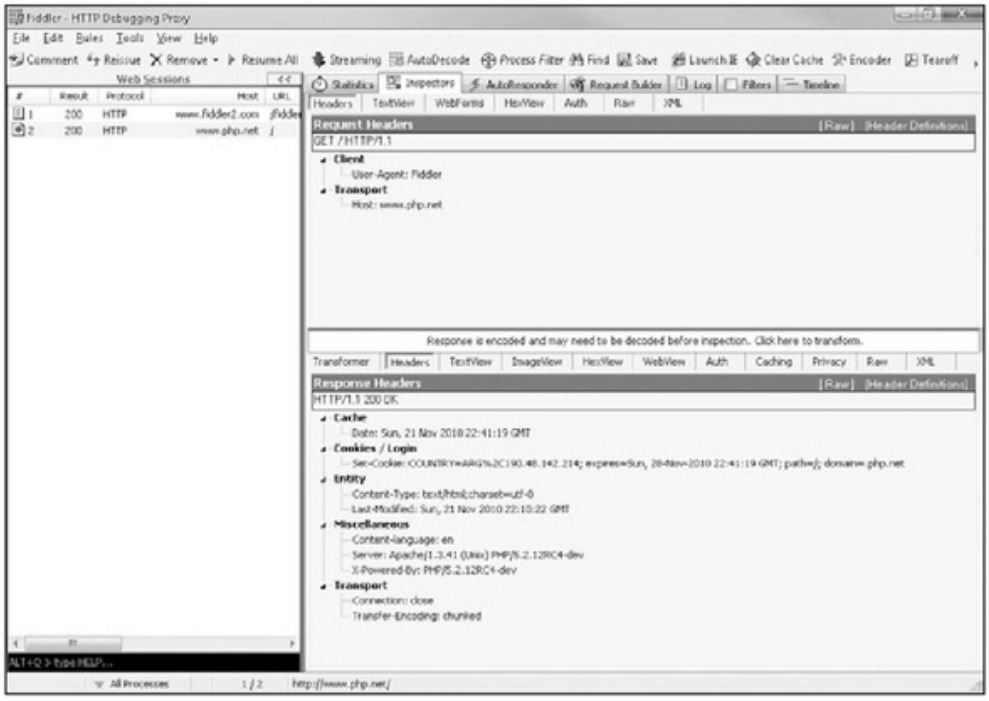


Figura 24. *Fiddler proporciona una interfaz con un listado de la información intercambiada entre un cliente y un servidor.*

Fiddler brinda otras posibilidades adicionales como incluir puntos de quiebre para seguir en detalle el comportamiento de una aplicación o filtrar el tráfico según distintos tópicos como el tipo de respuesta, por ejemplo. Además, este posee un gran número de extensiones para complementar la funcionalidad principal de la herramienta. Podemos obtener más información acerca de esto en su sitio web oficial: **www.fiddler2.com/fiddler2/extensions.asp**.

LiveHTTPHeaders

Esta extensión para Firefox permite visualizar las cabeceras enviadas y recibidas durante la comunicación. Para instalarla debemos ingresar en **http://livehttpheaders.mozdev.org/** y optar por la versión más actual.

Una vez instalada la extensión y después de haber reiniciado el navegador, podemos acceder a la funcionalidad de la aplicación de dos maneras: mediante la obtención de la información de la página actual (botón derecho, **Ver información de la página**, solapa **Cabeceras**).

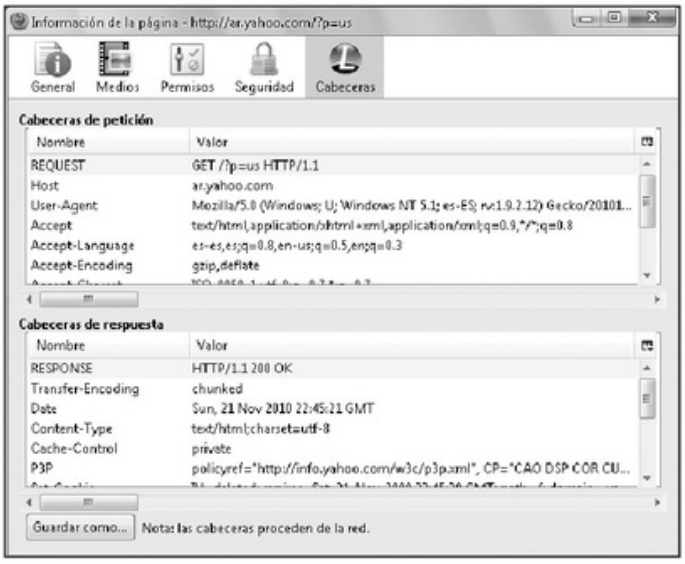


***Figura 25.** Esta extensión se complementa con los servicios ofrecidos por la versión base de Firefox.*

O desde el menú **Herramientas/Live HTTP Headers**, que muestra el tráfico actual en tiempo real de todas las páginas que están siendo visitadas.

Desde esta última opción, y si presionamos el botón **Repetir**, podemos modificar los parámetros enviados al servidor.

HttpWatch

Esta herramienta funciona bajo Internet Explorer y está disponible en dos versiones: una básica, que es gratuita y limitada, y una profesional con licencia comercial.

Podemos descargarla del sitio **www.httpwatch.com/download**. Monitorea tanto el protocolo HTTP como **HTTPS** (**secure HTTP**).
Una vez instalada, abrimos Internet Explorer y accedemos al menú **Ver/Barra del Explorador/HttpWatch** (o si presionamos **SHIFT + F2**). Podemos observar la aplicación en la parte inferior de la página o en una nueva ventana. Para comenzar a monitorear las peticiones y las respuestas debemos presionar el botón **Record** (**Stop** para finalizar). Desde **Summary** podemos observar el tiempo transcurrido, el número de peticiones, el contenido descargado, el peso ahorrado con la compresión HTTP, etcétera. Luego podemos filtrar la información exhibida y exportar todo a un archivo (**XML**, **CSV** o **HWL**, un formato propio de HttpWatch).

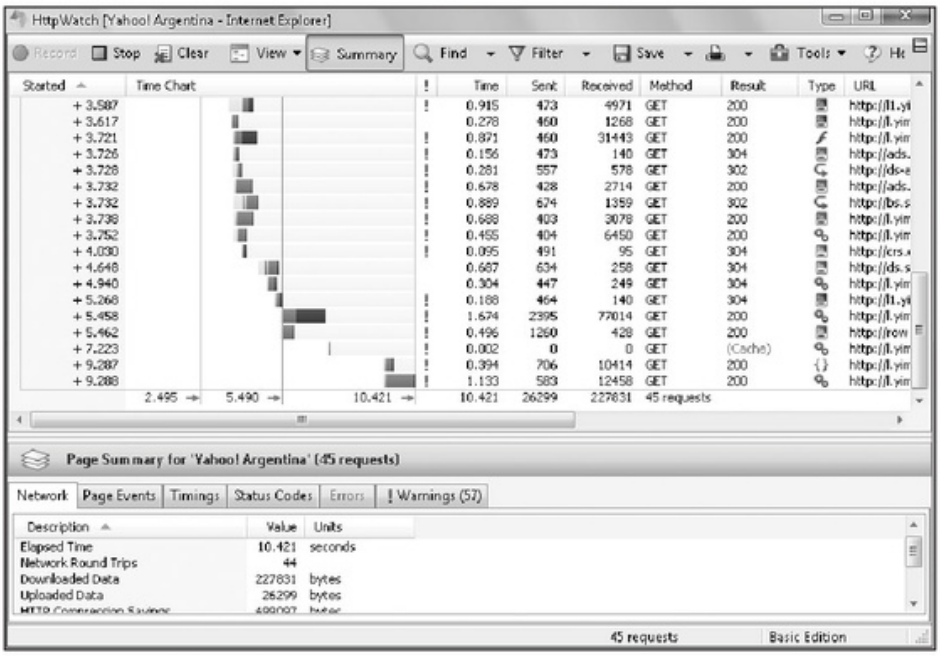


***Figura 26. HttpWatch** es una opción profesional para mantener un control pormenorizado del tráfico HTTP.*

En definitiva, se trata de una herramienta potente y de fácil utilización, acerca de la cual podemos obtener más información en su sitio web oficial: **www.httpwatch.com**.

... RESUMEN

En este capítulo observamos algunas de las opciones disponibles para agilizar las peticiones HTTP y permitir que el acceso por parte de los usuarios sea lo más simple y claro posible. Entre otras variantes, vimos cómo trabajar con URLs y realizamos un recorrido por las posibilidades ofrecidas por el archivo .HTACCESS.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es un archivo .HTACCESS?

2 ¿Para qué sirve la directiva ErrorDocument?

3 ¿Para qué sirve la directiva DirectoryIndex?

4 ¿Cómo se implementa la protección de directorios con .HTACCESS?

5 ¿Qué se almacena en el archivo error.log?

6 ¿Cuál es la utilidad que nos brinda la directiva ForceType?

7 ¿Para qué sirve el archivo httpd.conf?

8 ¿Qué es el error 404?

9 ¿Para qué se utilizan los archivos .HTPASSWD?

EJERCICIOS PRÁCTICOS

1 Genere la aplicación de noticias con otro lenguaje en lugar de PHP.

2 Implemente una solución para acortar URLs en alguno de sus sitios.

3 Nombre tres casos de uso para la directiva Redirect.

4 Nombre tres casos de uso para las directivas Allow y Deny.

Capítulo 8

Herramientas complementarias

En ocasiones necesitamos visualizar las características de una página de manera simple y rápida mediante un navegador, sin complicaciones ni pérdidas de tiempo innecesarias. En este capítulo haremos un recorrido por algunas de las opciones más renombradas.

Introducción	**220**
Firebug	220
Web Developer	247
Web Inspector	257
Opera Dragonfly	258
XRAY	260
DebugBar	260
Companion.JS	263
Internet Explorer Developer Toolbar	264
Resumen	**265**
Actividades	**266**

INTRODUCCIÓN

El ámbito natural de una aplicación web es evidentemente un navegador web, y dada la situación en los últimos tiempos surgieron iniciativas para extender las funcionalidades de estos programas para añadirles la posibilidad de probar el funcionamiento de las páginas, obtener información útil acerca del tiempo de carga y el peso de cada elemento externo (imágenes, hojas de estilos, documentos JavaScript, etcétera). A continuación repasamos las herramientas y extensiones más destacadas:

- **Firebug**
- **YSlow**
- **Web Developer**
- **Web Inspector**
- **Opera Developer Console**
- **XRAY**
- **DebugBar**
- **Companion.JS**
- **Internet Explorer Developer Toolbar**

Firebug

Firebug es una extensión disponible mediante el navegador Mozilla Firefox (**www.mozilla.com/firefox**), que nos permite obtener información referida a la ejecución de documentos de distintas tecnologías y el tiempo de carga de cada elemento. Es sumamente útil para detectar errores y verificar cómo trabaja la página internamente. Podemos obtener más información acerca de Firebug en **www.getfirebug.com**.

***Figura 1.** La extensión Firebug es una de las tantas alternativas brindadas por el navegador Firefox.*

Para instalar la herramienta, debemos ingresar al sitio oficial por Firefox y acceder a la opción correspondiente para comenzar el proceso. Si todo funcionó correctamente, una vez reiniciado el navegador podremos acceder a **Herramientas/Complementos** para verificar que la extensión se encuentre habilitada.

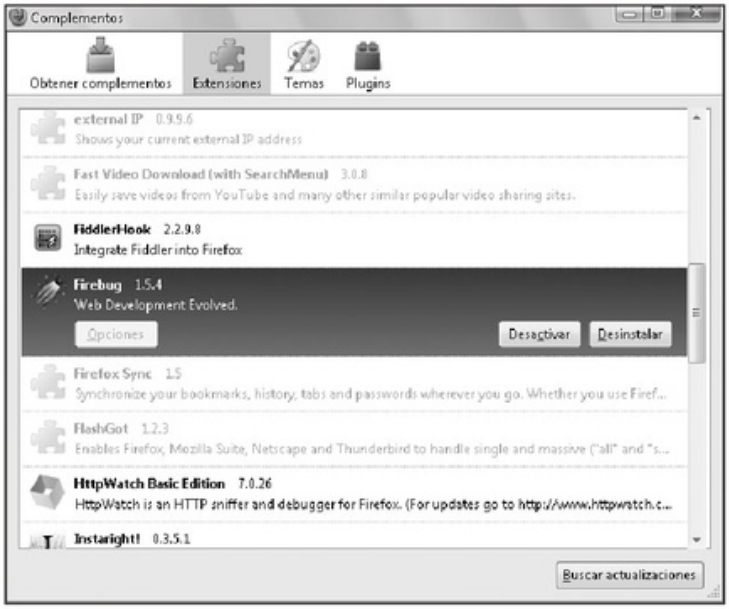


***Figura 2.** El panel de complementos de Firefox permite habilitar y deshabilitar las extensiones instaladas.*

Otra manera es observar si en la parte inferior derecha figura el icono de la aplicación, desde el que podemos iniciar o detener su ejecución. Para conocer las opciones de la extensión, primero debemos activarla. Esto se logra al hacer clic sobre el icono de la barra de estado / opción **Enable Firebug**. La pantalla principal se abre en la misma ventana del navegador, que constituye el comportamiento por defecto, o en una nueva (clic derecho del mouse y opción **Open Firebug in New Window**).

***Figura 3.** Para comenzar a trabajar con Firebug, debemos inicializar el servicio con la opción correspondiente.*

Algunos entornos de desarrollo como **Microsoft Visual Studio**, por ejemplo, dan la opción de ejecutar comandos o visualizar las líneas exactas en donde se producen errores. Esta actividad se conoce como **debugging**. En su línea de comandos, Firebug nos provee una manera rápida para probar códigos y acceder a partes de un documento. Podemos utilizar esta alternativa accediendo a la solapa **Consola** y haciendo clic en el botón ubicado en el margen inferior derecho.

***Figura 4.** La línea de comandos de Firebug nos permite ejecutar instrucciones JavaScript.*

Dentro del cuadro de texto que se despliega, podremos escribir instrucciones Javascript. Observamos también tres botones al final: **Correr** (ejecutar los comandos ingresados), **Limpiar** y **Copiar**.

***Figura 5.** El modo de múltiples líneas nos permite ingresar más de un comando JavaScript, si separamos cada uno con un punto y coma.*

Podemos obtener más información acerca de todas las opciones disponibles que tenemos desde la línea de comandos en el sitio web oficial de Firebug, ingresando en la dirección: **http://getfirebug.com/wiki/index.php/Command_Line**.

En definitiva, Firebug se ha convertido en una herramienta indispensable para el diseño y el desarrollo web, que ofrece alternativas avanzadas de control y edición de sitios. Aquí se listan solo algunas de las opciones utilizables, ya que abarcar cada funcionalidad es imposible.

Secciones disponibles

La interfaz de Firebug está dividida en solapas o secciones, y cada una mantiene una funcionalidad específica:

- **Consola**: desde aquí podemos ingresar instrucciones JavaScript para ejecutarlas sobre el documento actual. En las próximas secciones, veremos más acerca de la consola de Firebug.
- **HTML**: si accedemos a esta solapa, podemos visualizar el código fuente actualizado de la página. Esto es útil en caso de que, luego de la carga inicial las sentencias JavaScript, hayan modificado el contenido de la página. Además, es capaz de resaltar el código modificado por los eventos que disparan código JavaScript, por ejemplo **onMouseOver**, **onClick**, etcétera.

Es posible editar código en la vista previa y visualizar los cambios de manera inmediata. Si hacemos clic sobre un atributo o sobre un valor (de un elemento o de un atributo), se hace posible la edición. Si hacemos clic con el botón derecho del mouse sobre un elemento, tenemos la opción de editar el HTML, agregar atributos o eliminarlos, entre otras posibilidades.

Podemos presionar el botón **Inspect** y ubicar el puntero del mouse sobre una parte de la página para acceder a la sección correspondiente sin tener que recorrer el código fuente completo. En la esquina superior derecha, también contamos con un cuadro de texto para ingresar palabras y buscarlas dentro del código.

III MODO DE USO

La extensión denominada Firebug puede ser utilizada para acceder a páginas ubicadas en servidores web externos, locales e incluso documentos abiertos desde el sistema de archivos local de la computadora. Sin dudas, esto habla de la versatilidad y las posibilidades que esta herramienta les ofrece a los desarrolladores y usuarios interesados.

Figura 6. *Firebug nos da la posibilidad de acceder a partes específicas de una página de manera intuitiva y rápida.*

- **CSS**: nos permite acceder a las hojas de estilo asociadas con el documento y editarlas para visualizar los cambios en el momento. La utilidad de esta funcionalidad consiste en que instantáneamente podemos observar las repercusiones de las modificaciones.

Figura 7. *La edición CSS es sin dudas uno de los puntos fuertes de la extensión Firebug.*

- **SCRIPT**: está compuesto por dos paneles. El primero contiene el código a evaluar, que podrá ser modificado, y el segundo se subdivide en dos: **Observar** y **Puntos de ruptura**.

Al lado de **Inspect** aparece un listado de archivos, y en caso de tener referencias a uno o más scripts externos, aparecerán aquí y podremos seleccionar uno en particular para acceder a su código y visualizarlo en el editor.
Para seleccionar los puntos en los que queramos que la aplicación se detenga (**Breakpoint**, punto de quiebre) para observar su ejecución parcial debemos hacer clic sobre el número de línea, tal cual se hace en **Visual Studio**, por ejemplo:

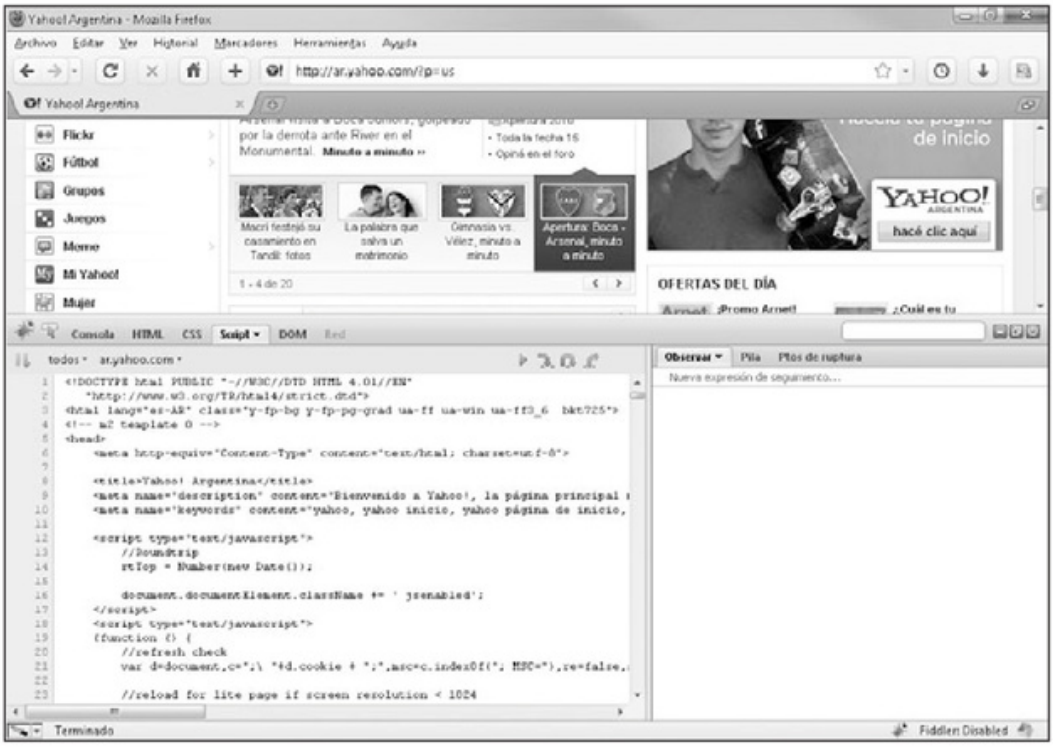


***Figura 8.** La inclusión de puntos de referencia en scripts nos permite mantener un control profundo sobre la ejecución de una aplicación.*

Una vez hecho esto, ante cada **Breakpoint** se detendrá la ejecución. En ese momento, la solapa **Watch** se cargará con los valores de las variables en ese preciso instante y se activará una barra de iconos al lado del cuadro de texto para búsquedas

III JAVASCRIPT

Debemos tener en cuenta que la mayoría de las aplicaciones vistas en este capítulo hace uso del lenguaje de programación JavaScript para permitir realizar el acceso, obtención de información e incluso modificación de las características de cada elemento de cada página encontrada. Sin dudas, esto habla del poderío y la versatilidad del lenguaje.

que nos permitirá seguir con la ejecución, recomenzarla, retroceder o avanzar una línea. Por su parte, **Breakpoint** lista todos los puntos disponibles, con la posibilidad de activarlos o desactivarlos.

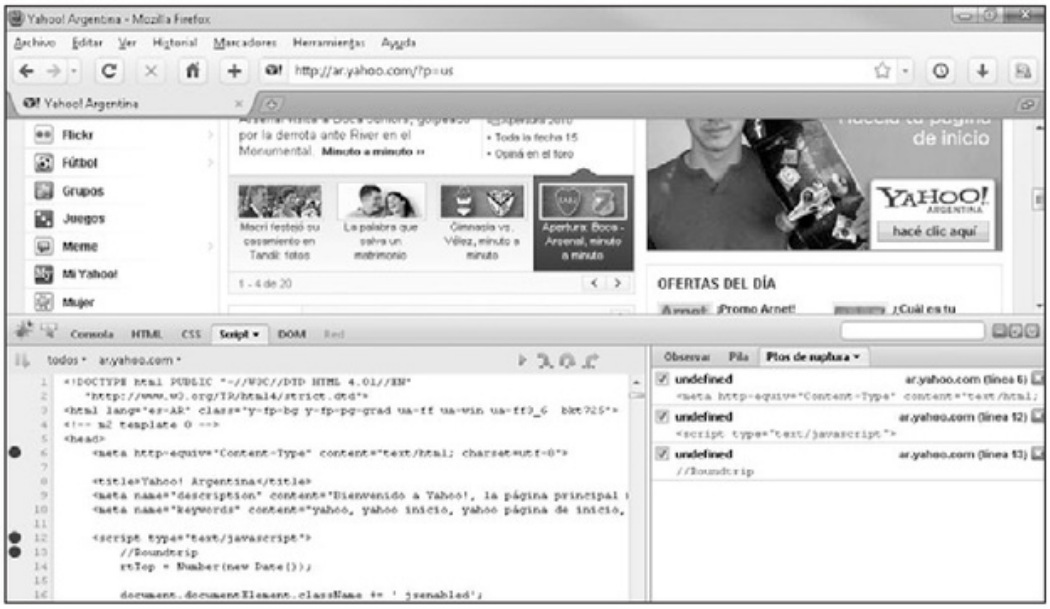


***Figura 9.** La ejecución de un script se detiene al encontrar un **breakpoint**, y da opciones para continuar cuando lo creamos conveniente.*

- **DOM**: nos permite observar el modelo de objetos del documento o un elemento en particular con el listado de todas sus características (propiedades y métodos).

***Figura 10.** El modelo de objetos del documento nos permite acceder al detalle de cada uno de los elementos encontrados.*

- **NET**: esta opción es de mucha importancia ya que nos da la posibilidad de observar gráficamente el tiempo de carga de cada elemento (documento HTML, hojas de estilo CSS, archivos JavaScript, peticiones con **XMLHttpRequest**, imágenes con vista previa incluida y objetos **Flash**). Incluso podemos filtrar por tipo si utilizamos la barra de opciones superior.

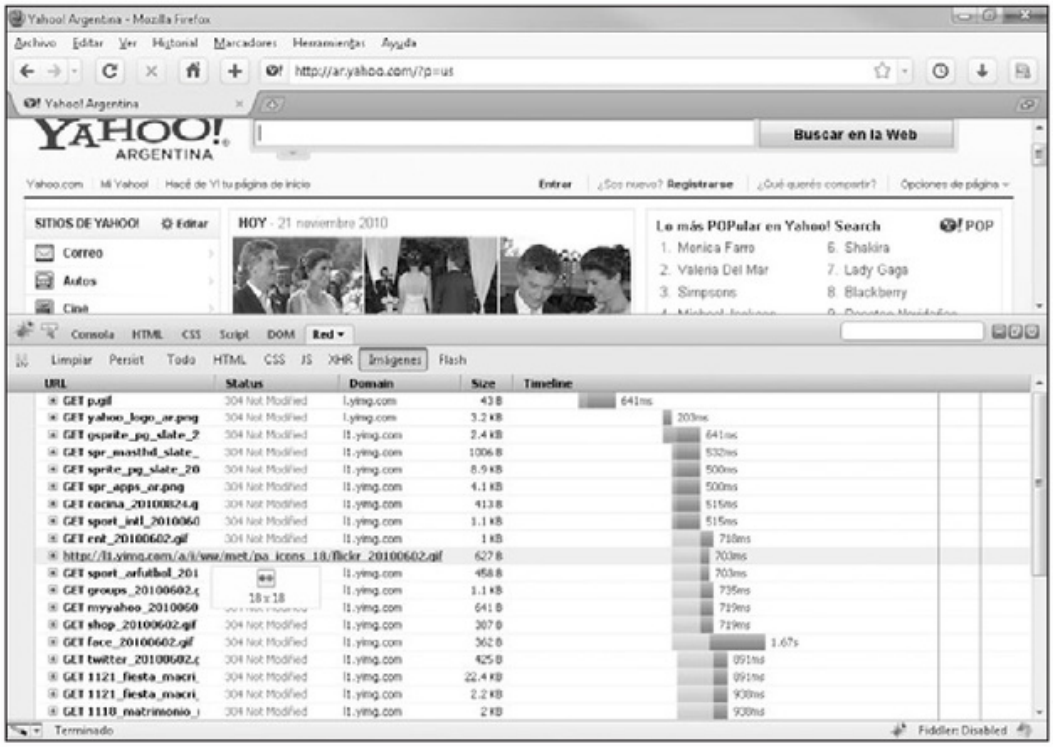


Figura 11. *Conocer los recursos consumidos por una página nos será de gran ayuda para controlar aspectos como la velocidad de carga y el peso de cada objeto insertado.*

Comandos especiales

Desde la línea de comandos podemos ingresar y ejecutar sentencias JavaScript tradicionales o aprovechar algunas funcionalidades especiales ofrecidas por Firebug. Entre las disponibles se encuentran:

- el comando **$(id)** le devuelve el elemento correspondiente al identificador pasado como argumento, por ejemplo si tenemos en cuenta el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Firebug </title>
</head>
<body>
```

```
  <input id="nombre" type="text" />
  <input id="apellido" type="text" />
  <input id="telefono" type="text" />
</body>
</html>
```

y desde la línea de comando ingresamos:

```
$('apellido')
```

y por último debemos presionar el botón **Run**. Con esto podremos observar cómo en la parte izquierda del programa aparece la definición completa del elemento. Si hacemos clic en la respuesta, obtenemos el código completo con el elemento resaltado y, si es posible, la vista previa correspondiente. Podríamos ingresar códigos como vemos en el ejemplo a continuación:

```
var apellido = $('apellido');
apellido.value = 'Firebug';
apellido.style.color = 'green';
```

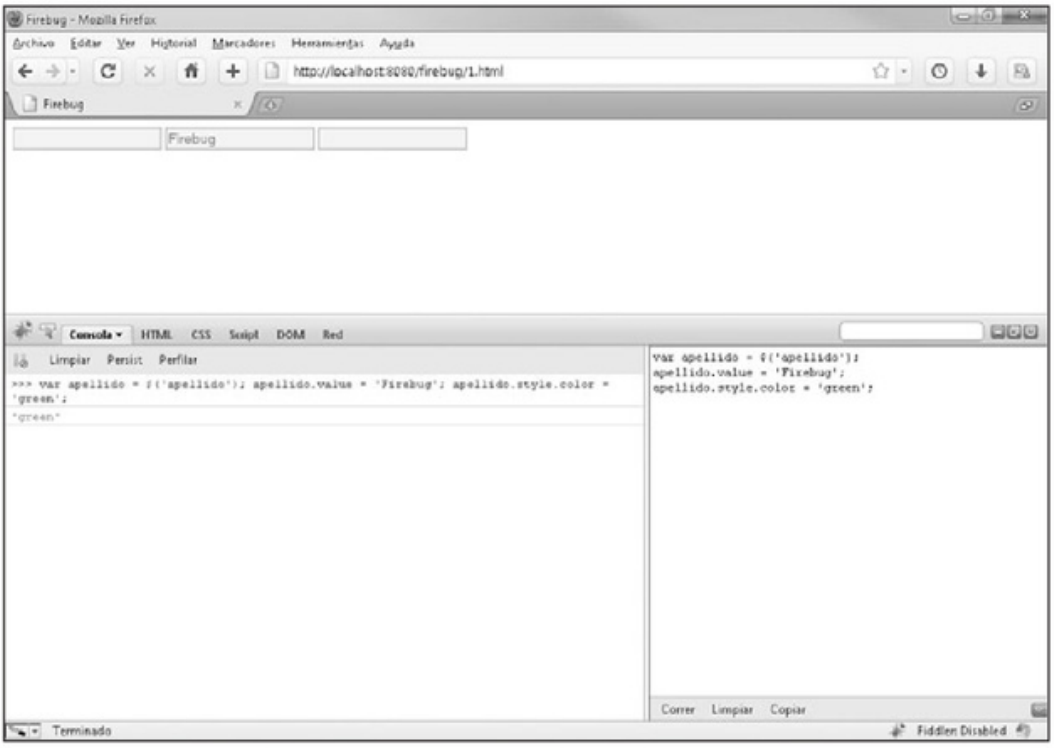


Figura 12. *La línea de comandos nos permite ejecutar instrucciones que tendrán efecto inmediato sobre el documento actual.*

- para recuperar todos los elementos que pertenecen a un determinado selector CSS contamos con la sintaxis **$$(nombreSelector)**. Veamos un ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head>
  <title>Firebug </title>
  <style type="text/css">
  /*<![CDATA[ */

  .nombreEstilo {
    border: 1px solid #000;
  }

  /*]]> */
  </style>
</head>
<body>
  <input class="nombreEstilo" id="nombre" type="text" />
  <input id="apellido" type="text" />
  <input class="nombreEstilo" id="telefono" type="text" />
</body>
</html>
```

Si ejecutamos el código que vemos podremos darnos cuenta de que obtendremos el array correspondiente a los elementos denominados **nombre** y **telefono**:

```
$$('.nombreEstilo');
```

III INSTALACIÓN

La extensión llamada Firebug puede ser descargada desde la página de extensiones del navegador Firefox, más precisamente visitando el sitio que se encuentra en la dirección: **https://addons.mozilla.org/en-US/firefox/addon/1843**. La instalación se realiza de manera automática accediendo al enlace que apunta al archivo **xpi** correspondiente.

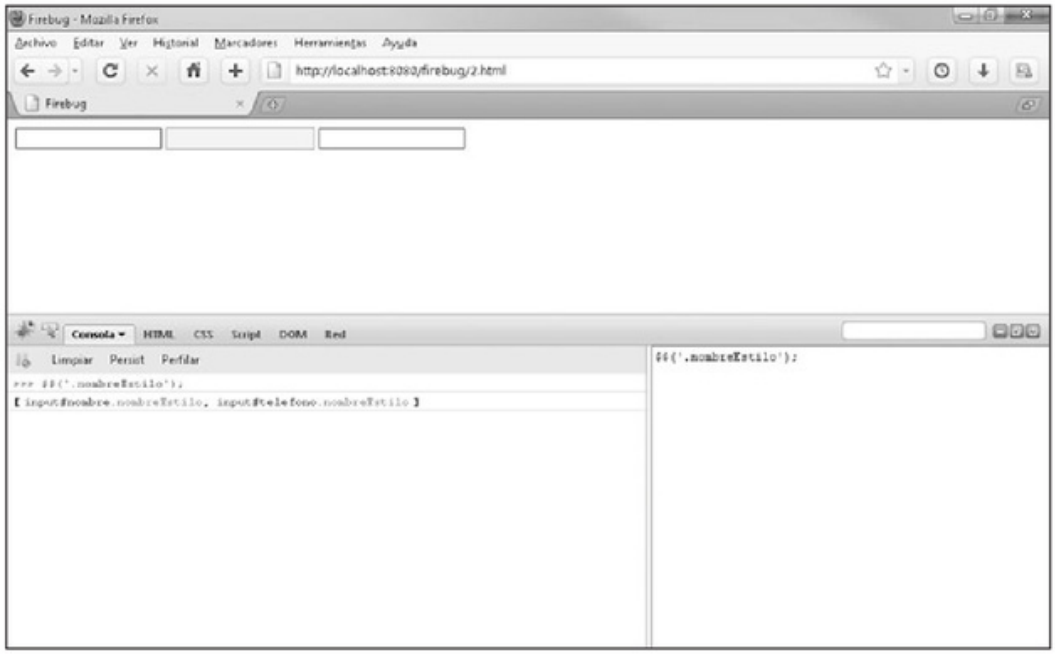


Figura 13. *Firebug nos da la posibilidad de agrupar elementos según el selector CSS especificado.*

- Otra manera de acceder a elementos es mediante **XPath**. Podemos obtener más información acerca de esta técnica en el sitio web **www.w3schools.com/xpath**.

```
$x(xpath);
```

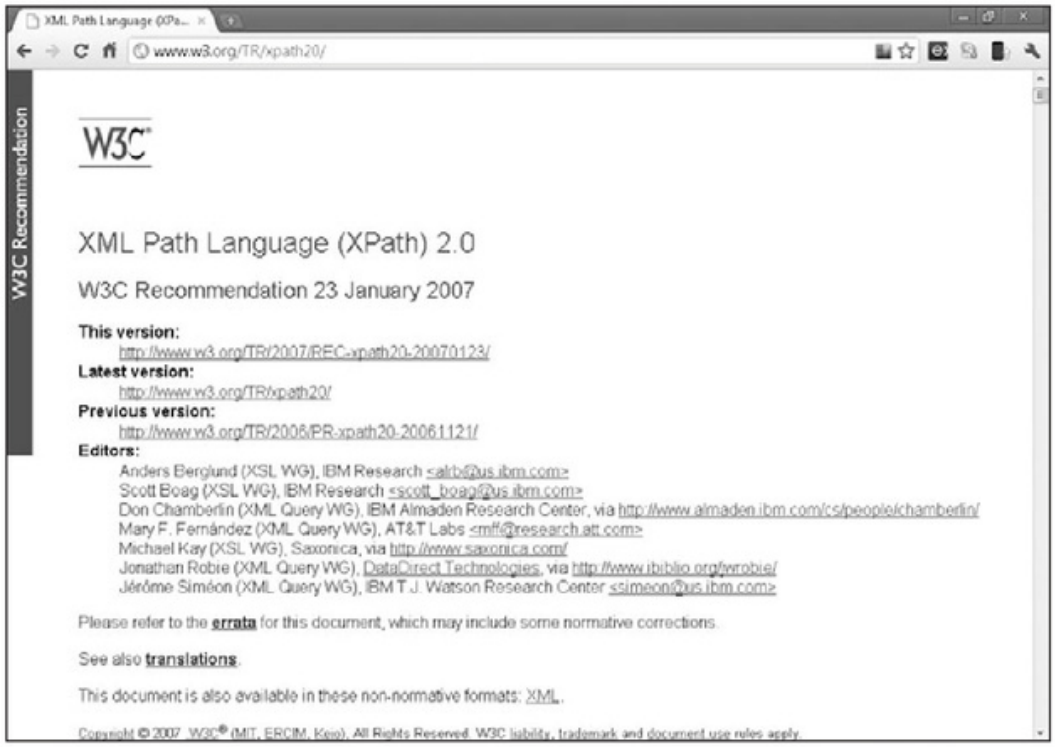


Figura 14. *XPath es un lenguaje que nos permite acceder al contenido de documentos estructurados en tags, como por ejemplo XML o XHTML.*

- es importante señalar que por su parte, la función **dir** recibe como argumento una referencia a un objeto y lista todas sus propiedades. Tomemos como caso el código que se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Firebug </title>
  <style type="text/css">
  /*<![CDATA[ */

  .nombreEstilo {
    border: 1px solid #000;
  }

  /*]]> */
  </style>
</head>
<body>

<div id="seccion1" class="nombreEstilo"> texto </div>

</body>
</html>
```

La instrucción que mostramos a continuación se encarga de devolver el listado de propiedades correspondientes a **seccion1**:

```
dir($('seccion1'));
```

III EXTENSIONES

Firefox se caracteriza por ser un navegador personalizable y expandible mediante las extensiones disponibles en el sitio que se encuentra en la dirección **https://addons.mozilla.org**. Es interesante saber que allí podemos acceder a un listado enorme de opciones gratuitas divididas en categorías y listas para ser utilizadas por los usuarios.

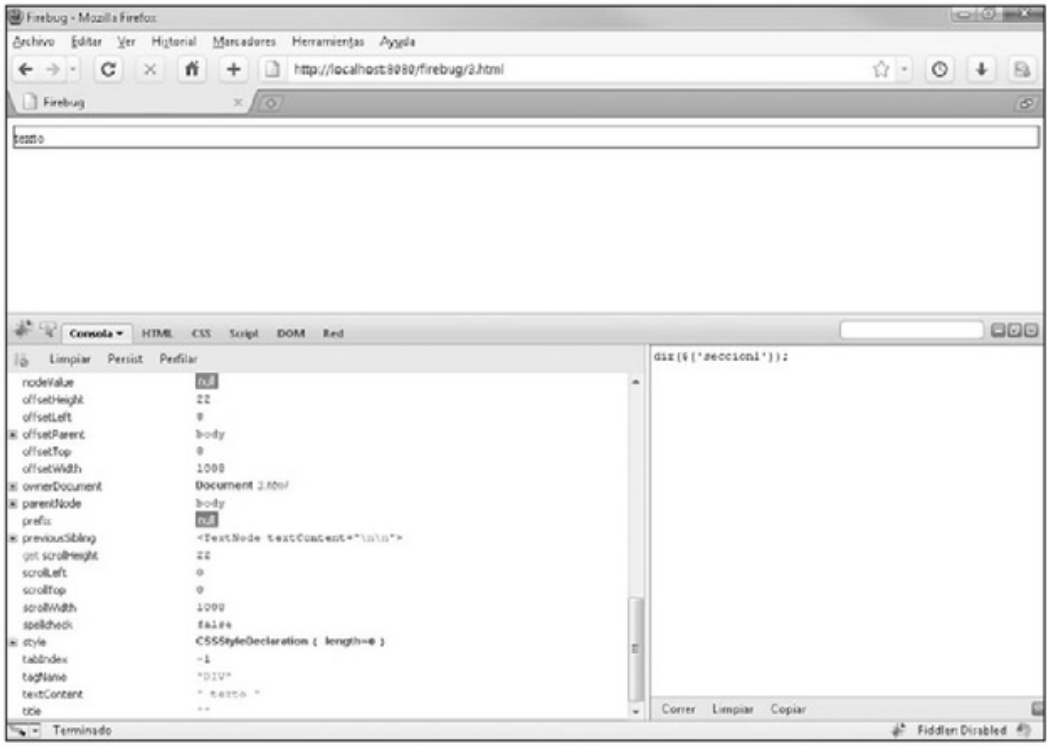


Figura 15. *Conocer las propiedades y características de un elemento se vuelve sencillo si utilizamos la extensión Firebug.*

- la función **dirxml** imprime el contenido de un nodo XML y sus descendientes. En el siguiente ejemplo obtenemos la información contenida en el elemento **b**:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Firebug </title>
  <style type="text/css">
  /*<![CDATA[ */

  .nombreEstilo {
    border: 1px solid #000;
  }

  /*]]> */
  </style>
</head>
<body>

<div id="a">
```

```
  <div id="b">
    <div id="c"> xxx </div>
  </div>
</div>

</body>
</html>
```

Con la instrucción que se muestra a continuación:

```
dirxml($('b'))
```

El resultado que obtenemos es el siguiente:

```
<div id="b">
  <div id="c"> xxx </div>
</div>
```

- con el comando **clear** limpiamos la consola.

```
clear();
```

- el comando **inspect** recibe como argumentos una referencia a un objeto y opcionalmente un tipo de exhibición (la interfaz de Firebug está dividida en tabs o solapas, entre las que se encuentran **HTML**, **CSS**, **SCRIPT** y **DOM**):

```
inspect($('b'), 'HTML');
```

III CASOS DE USO

Es muy importante saber que la utilización de algunas herramientas para el control de aplicaciones web puede ser de mucha utilidad para que podamos observar de cerca el comportamiento de cada parte del sistema y de esta forma lograr desarrollos libres de error y probadamente confiables. Sin dudas, esto será valorado por los usuarios.

- la función **keys** devuelve un array con las propiedades del objeto argumentado:

```
keys($('b'));
```

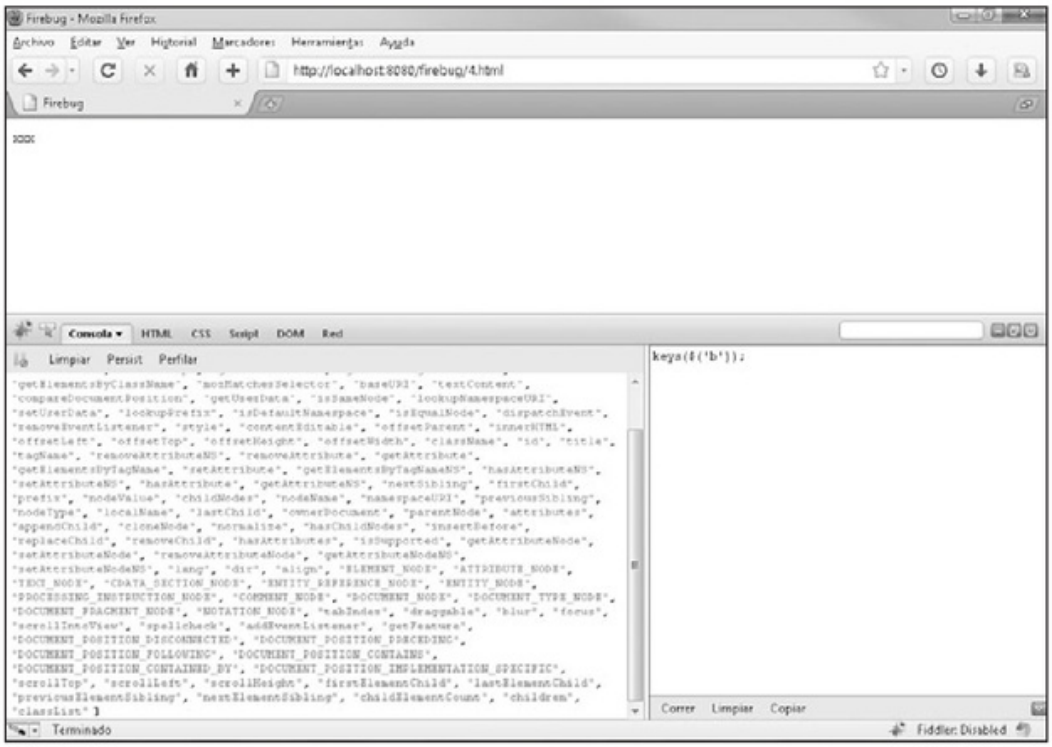


Figura 16. *Las funciones Firebug facilitan el acceso a los elementos de un documento mediante JavaScript.*

- debemos tener en cuenta que la función denominada **values** se encarga de devolver un array que contiene los valores correspondientes a las propiedades del objeto que hemos pasado como argumento:

```
values($('b'));
```

- la función **monitor** recibe como argumento el nombre de alguna función JavaScript definida en el archivo que analiza y ante cada llamada se agregará una línea en la salida de la consola Firebug. Para dejar de monitorear una función determinada contamos con **unmonitor**. Veamos un ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```

```
  <title>Firebug </title>
  <script type="text/javascript">
  //<![CDATA[

  function a() {
    //definicion ...
  }

  function b() {
    //definicion ...
  }

  function c() {
    a();
  }

  //]]>
  </script>
</head>
<body>

<input type="button" onclick="javascript:a();" value="Funcion a" />
<input type="button" onclick="javascript:b();" value="Funcion b" />
<input type="button" onclick="javascript:c();" value="Funcion c" />

</body>
</html>
```

Si ingresamos el código que mostramos en la línea de comandos y posteriormente esperamos algunos segundos, podremos darnos cuenta de que aparecerá el siguiente mensaje de salida en la consola de Firebug:

```
monitor(a);
monitor(b);
monitor(c);
```

Si nos encargamos de hacer clic sobre alguno de los botones que se presentan en el archivo HTML podremos ver cómo se agregan las entradas a la consola, tal como se muestra en la imagen que se presenta a continuación:

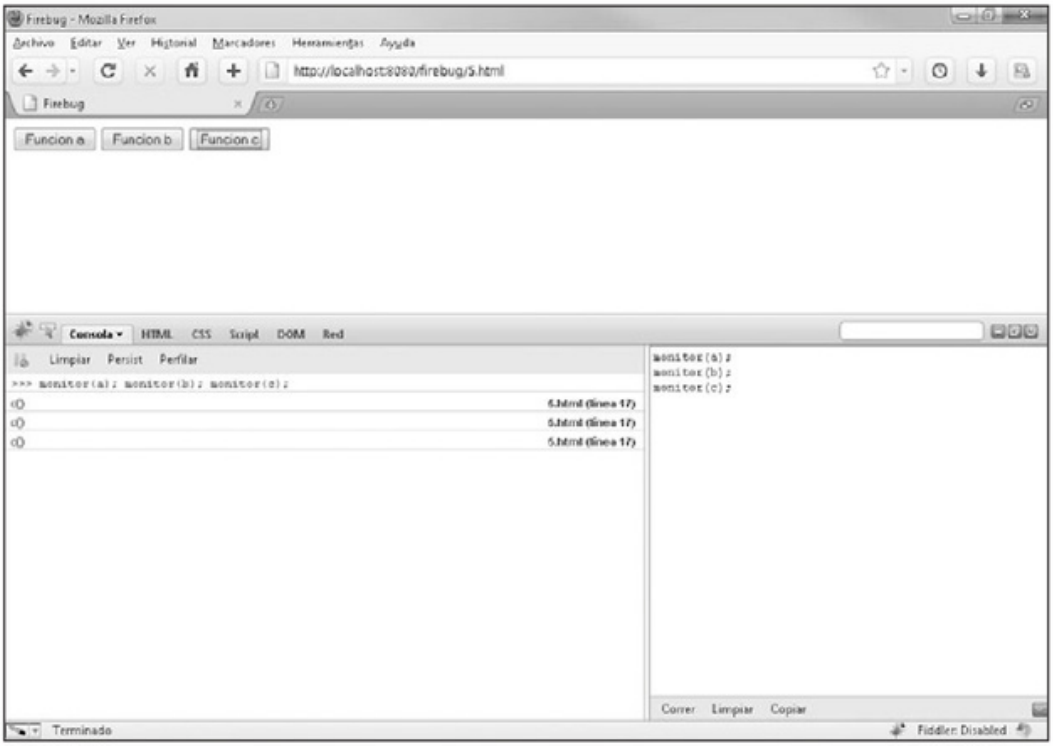


Figura 17. *Firebug nos permite mantener un control acerca de la ejecución de funciones de un script.*

Para quitar el control de una determinada función utilizamos **unmonitor**, que también puede tardar algunos segundos en surtir efecto:

```
unmonitor(b);
```

También están disponibles **debug** y **undebug**. Ambas reciben como argumento un nombre de función y permiten marcar un punto de quiebre en la primera línea. Al llegar a este punto, el script se detendrá e informará sobre la situación actual (valores correspondientes a variables, por ejemplo).

Consola

En algunos navegadores está disponible el objeto **console**, que nos permite ubicar comentarios en distintas partes del código JavaScript para mantener un control más profundo sobre lo que acontece durante la ejecución del script:

```
if(window.console){
  console.log("Texto");
}
```

Firebug admite los siguientes tipos de mensajes:

```
<script type="text/javascript">
//<![CDATA[
console.log('Decripcion del comentario aqui');
console.debug('Decripcion aqui');
console.error('Decripcion del error aqui');
console.info('Decripcion de la informacion aqui');
console.warn('Decripcion de la advertencia aqui');
//]]>
</script>
```

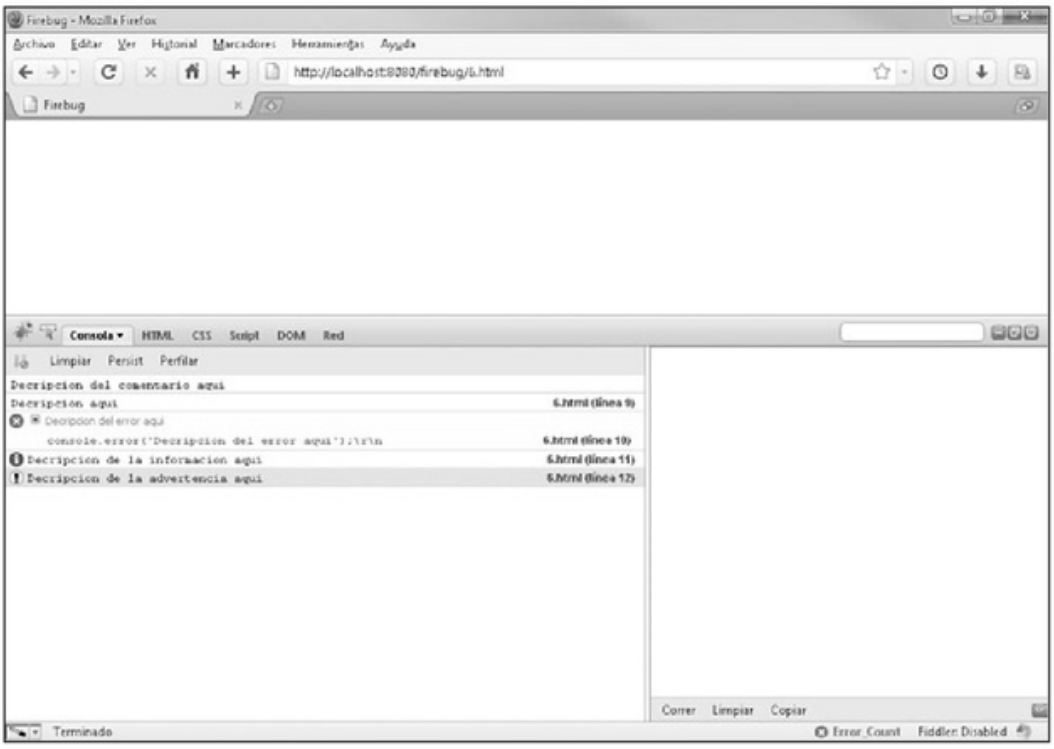


Figura 18. *El control de errores se vuelve más amigable si utilizamos la consola Firebug.*

Otras utilidades de **console** son las siguientes:

- **console.dir** recibe la referencia a un objeto, y devuelve sus propiedades y métodos.
- **console.dirxml** imprime el contenido de un nodo del documento.
- **console.assert** sirve para evaluar si una expresión es verdadera o falsa:

```
<script type="text/javascript">
//<![CDATA[

var x = 10;
```

```
var y = 10;

console.assert(x == y);

//]]>
</script>
```

- **console.trace** nos permite obtener el camino recorrido por un script hasta un determinado punto. En el código que se muestra a continuación es posible que veamos el ejemplo correspondiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head>
  <title>Firebug </title>
  <script type="text/javascript">
  //<![CDATA[

  function a(x) {
    b(2);
  }

  function b(y) {
    c(3);
  }

  function c(z) {
    console.trace();
    //definicion ...
  }

  //]]>
  </script>
</head>
<body>

<input type="button" onclick="javascript:a(1);" value="Funcion a" />
```

```
</body>
</html>
```

El proceso hasta invocar la función **c** es clic en el botón, llamada a **a(1)**, llamada a **b(2)** y llamada a **c(3)**.

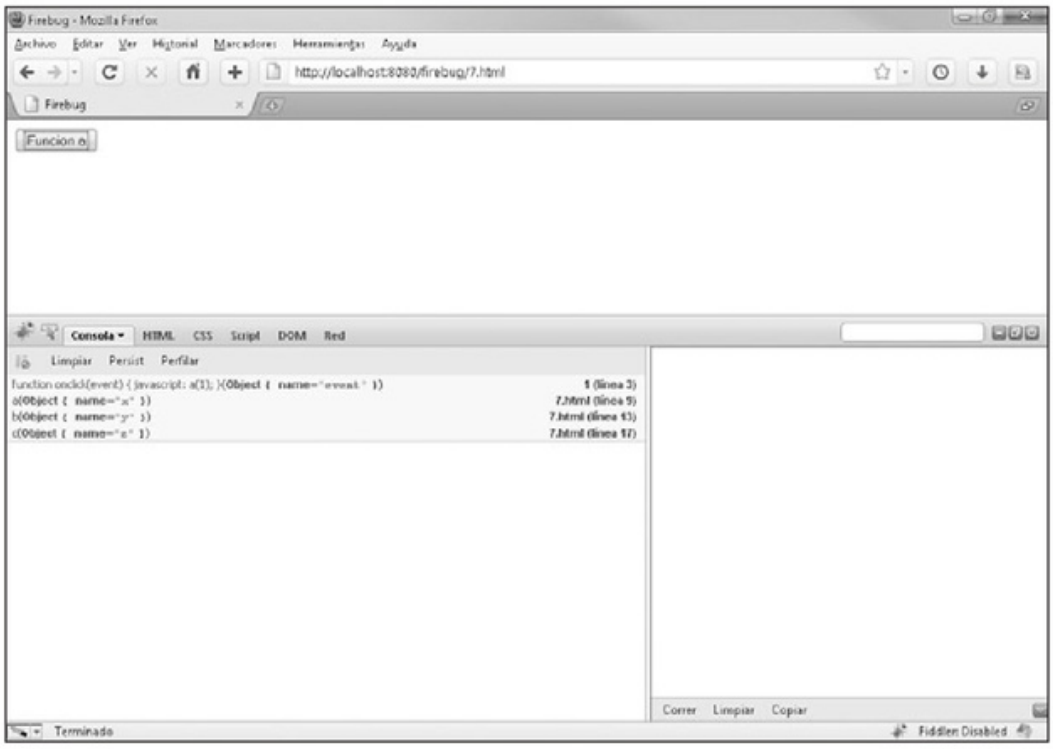


Figura 19. *Hallar el camino que finaliza en la ejecución de una determinada función puede sernos de utilidad para controlar nuestros scripts.*

- podemos calcular el tiempo transcurrido entre un punto (denominado **console.time**) y otro (conocido como **console.timeEnd**):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head>
  <title>Firebug </title>

  <script type="text/javascript">
  //<![CDATA[

  var tiempo = 'Tiempo transcurrido';
```

```
    console.time(tiempo);
    for(var i=0;i<1000000;i++){
      //...
    }
    console.timeEnd(tiempo);

    //]]>
    </script>

</head>
<body> ... </body>
</html>
```

Debemos tener en cuenta que el resultado que obtendremos será algo parecido a lo que se muestra a continuación:

```
Tiempo transcurrido: 20ms
```

- **console.profile** nos da la posibilidad de generar informes acerca de la performance de un script, incluidos los recursos consumidos por cada función. Pueden generarse distintos reportes por medio de la sintaxis:

```
console.profile('nombrePrueba');
```

Es similar a **console.time**, solo que da más detalles acerca de la ejecución de las porciones de código delimitadas. Para utilizar esta alternativa, contamos con una línea de apertura (**console.profile**) y otra de cierre (**console.profileEnd**):

```
//código JavaScript
console.profile('Fragmento número 1');
//código JavaScript
console.profileEnd();
//código JavaScript
```

De manera general, podemos obtener perfiles si presionamos el botón denominado **Profile** disponible en la barra de menús de Firebug (una vez para iniciarlo y otra para detenerlo y visualizar los resultados obtenidos).

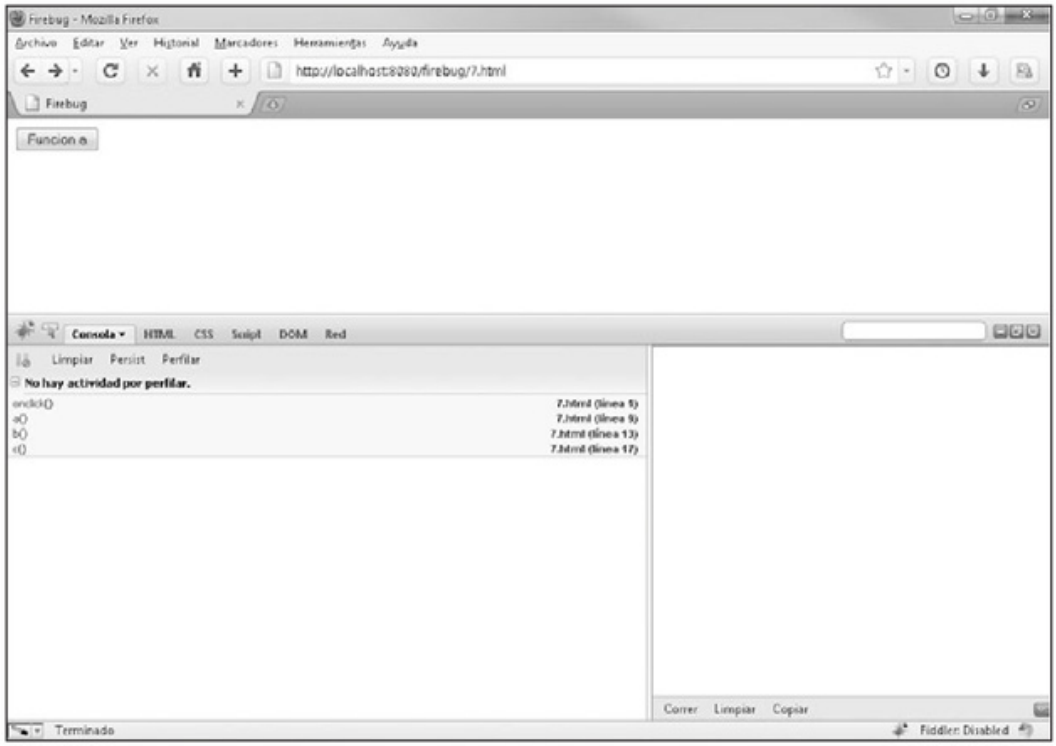


Figura 20. *La generación de informes es uno de los puntos fuertes de la extensión Firebug.*

Otra opción es con la línea de comandos, mediante la utilización de la sintaxis:

```
profile('nombrePrueba');
profileEnd();
```

Entre los datos devueltos se encuentran los siguientes:

COLUMNA	DESCRIPCIÓN
Function	Nombre de la función ejecutada
Call	Número de llamadas a la función
Percent	Porcentaje de tiempo consumido
Own Time	Duración de la función, sin contar las llamadas internas a otras funciones
Time	Duración de la función, contando las llamadas internas a otras funciones
Avg	Own Time / Call
Min	Duración de la ejecución que menos duró
Max	Duración de la ejecución que más duró
File	Nombre del archivo

Tabla 1. *Datos devueltos por la opción* ***Profile****.*

Podemos obtener más información acerca de las utilidades de la consola en Firebug, en la dirección **http://getfirebug.com/console.html**.

Firebug Lite

Aunque Firebug nació inicialmente como una extensión para el navegador Mozilla Firefox, ha surgido **Firebug Lite** para utilizar la herramienta desde otras opciones como Internet Explorer, Opera o Safari. Podemos obtener más información en el sitio **http://getfirebug.com/firebuglite**. Una vez descargada, la descomprimimos y referenciamos al archivo **firebug.js** desde el documento que queramos analizar:

```
<script language="javascript" type="text/javascript"
   src="ruta/a/firebug.js"></script>
```

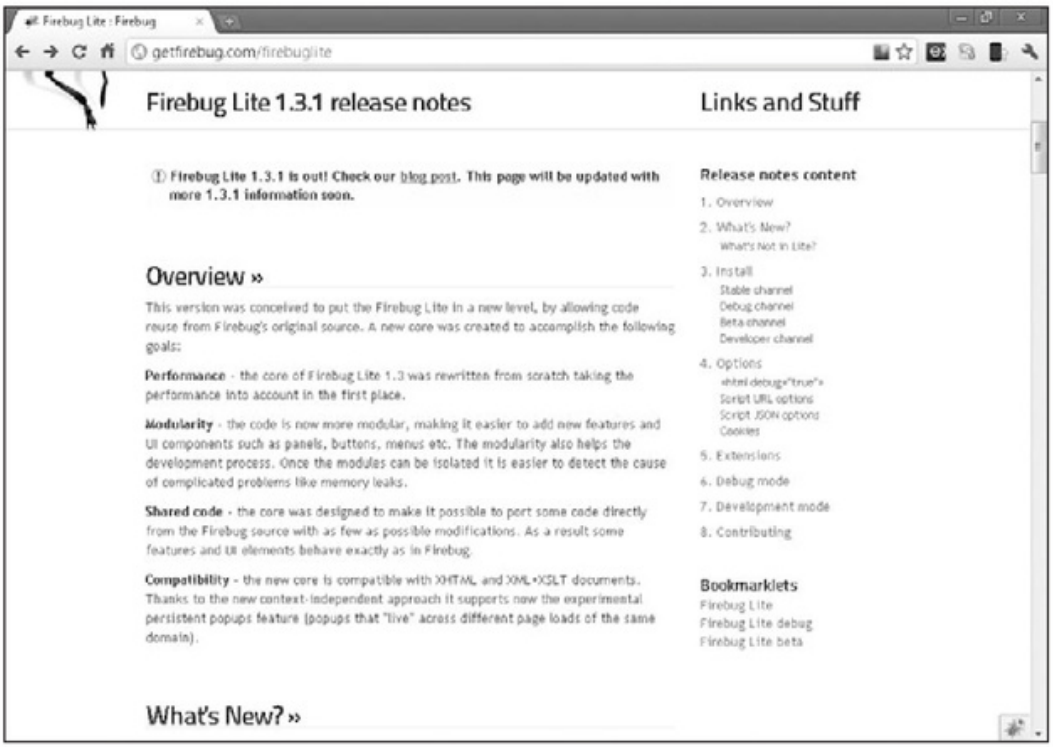


Figura 21. *El proyecto para migrar Firebug a otros navegadores está tomando impulso gracias a la aplicación Firebug Lite.*

Luego abrimos la página con el navegador y presionamos la tecla **F12** para iniciar la consola. Para que inicie de manera automática, podemos añadirle el atributo **debug** al elemento **html** y asignarle el valor **true**:

```
<html debug="true" xmlns="http://www.w3.org/1999/xhtml">
```

Así, con Firebug Lite podemos acceder a las funcionalidades de la consola del Firebug original. Existe un plugin para incorporar Firebug Lite a las aplicaciones desarrolladas en **Ruby on Rails**. Podemos obtener más información en **http://source.collectiveidea.com/public/rails/plugins/firebug**.

Cabe destacar que Firebug también está disponible para Google Chrome. Encontraremos más información en la dirección **https://chrome.google.com/extensions/detail/bmagokdooijbeehmkpknfglimnifench**.

Control de errores

Firebug puede configurarse para seleccionar qué errores y advertencias queremos visualizar a medida que se ejecuta el script actual. Entre las opciones disponibles del botón **Options**, se encuentran las siguientes:

OPCIÓN	DESCRIPCIÓN
Show Javascript Error	Errores Javascript
Show Javascript Warnings	Advertencias Javascript
Show CSS Errors	Errores en hojas de estilos CSS asociadas al documento
Show XML Errors	Errores XML
Show XMLHttpRequests	Peticiones con este objeto

***Tabla 2.** Configuración de los mensajes de error exhibidos.*

Esta última opción es útil cuando verificamos el funcionamiento de una aplicación Ajax, que generalmente realiza peticiones en segundo plano no perceptibles a simple vista. Entre la información que se muestra, figura el tiempo de ejecución de la petición, los parámetros pasados como argumento (con **POST** o **QueryString**), las cabeceras **HTTP** y la respuesta recibida.

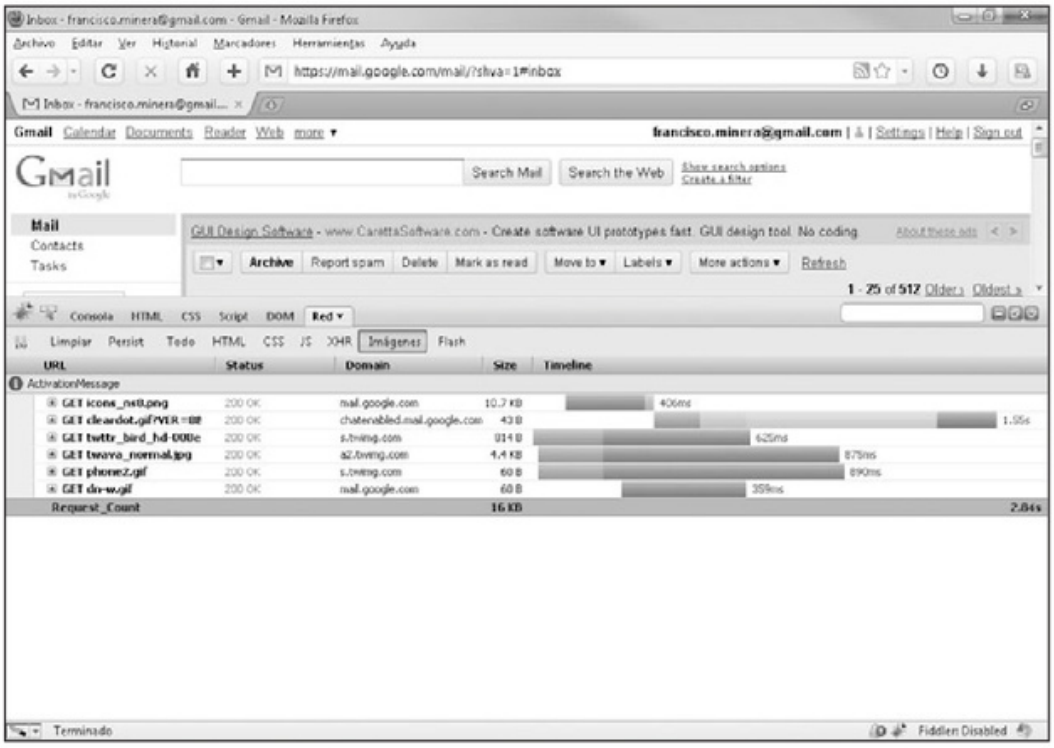


***Figura 22.** El control de las peticiones y las respuestas en una aplicación Ajax es de suma importancia.*

YSlow

Yahoo! desarrolló una herramienta llamada **YSlow** que trabaja en conjunto con Firebug (deberá estar instalada previamente). Para su instalación, puede descargarse del sitio de extensiones de Firefox, **https://addons.mozilla.org/en-US/firefox/addon/5369**. Aquí aparecerá una solapa extra llamada **YSlow**, y si la seleccionamos en la barra superior se agregarán las opciones **Grade**, **Components**, **Statics**, **Tools** y **Help** (ayuda de la extensión). Podemos acceder a YSlow de dos maneras: desde el icono de Firebug o desde el generado por YSlow, que se puede deshabilitar para que no ocupe espacio en la barra de estado del navegador, y luego habilitarse desde la opción **Options/Hide Status Bar Info** de Firebug.

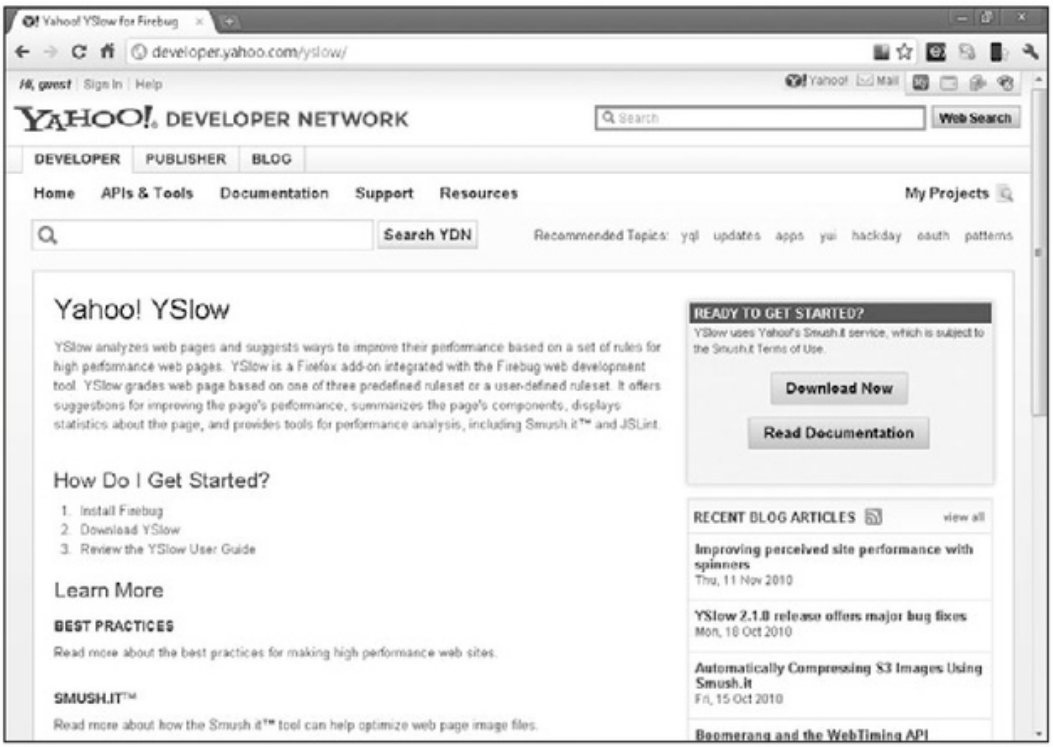


Figura 23. *YSlow se ha convertido en el complemento ideal para la extensión Firebug.*

- en la solapa **Grade**, observamos las consideraciones hechas por YSlow a partir de las reglas para la optimización de sitios de Yahoo! (**http://developer.yahoo.com/performance/rules.html**). Cada una contará con una graduación o un puntaje (de

III TIEMPO DE CARGA Y PESO TOTAL

Es importante tener en cuenta que si, activamos la opción denominada Autorun, YSlow se ejecutará ante las cargas de las páginas y nos informará cuánto tiempo tardó (en segundos) y el peso total (en Kb). Para habilitar esta opción, debemos acceder a las propiedades correspondientes a la extensión (**Herramientas/Complementos/YSlow**).

A hasta **F**, es decir desde las que más atención requieren en nuestra página hasta las menos relevantes, y **n/a** para las sin importancia). Cada regla tiene un enlace que apunta a la descripción oficial en el sitio web de Yahoo! y algunas poseen un contenido con comentarios particulares, por ejemplo el número de documentos JavaScript cargados, la posición de los enlaces a hojas de estilo CSS, etcétera. Esto nos ayudará a obtener una idea precisa acerca de los puntos que debemos solucionar.

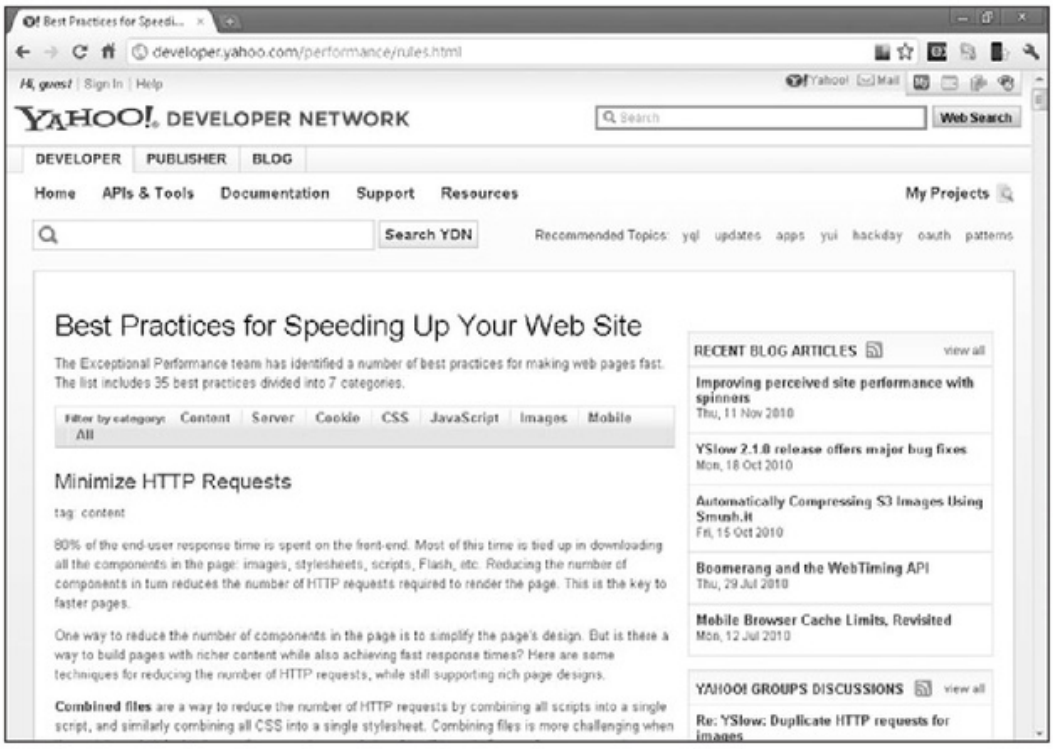


Figura 24. *Las reglas de optimización de sitios son la base esencial para las recomendaciones propuestas por YSlow.*

- en la solapa **Statistics** se exhiben estadísticas de la página, como por ejemplo el tamaño de cada uno de los componentes cargados, el tamaño total del documento, el número de peticiones HTTP realizadas o las cookies recibidas mediante éstas. En la primera columna, **Empty Cache**, se muestra la situación sin utilizar el caché del navegador, y en la segunda, **Primed Cache**, cómo cambiaría en caso de utilizarlo para cargar los objetos del documento.

III INTEGRACIÓN CON FIREBUG LITE

La extensión conocida como **YSlow** todavía no está disponible para ser utilizada junto con Firebug Lite, que es la opción de Firebug para los navegadores diferentes a Firefox, por ejemplo Microsoft Internet Explorer. Firebug Lite es una herramienta en pleno desarrollo y se esperan cambios importantes para las próximas versiones.

Figura 25. *Una de las reglas indica la necesidad de reducir al máximo el número de peticiones HTTP por página.*

- en **Components** podemos observar datos acerca de la carga de componentes y documentos HTML y hojas de estilo CSS entre otros. También es posible divisar el tipo de archivo, la dirección y la fecha de expiración, entre muchos otros.

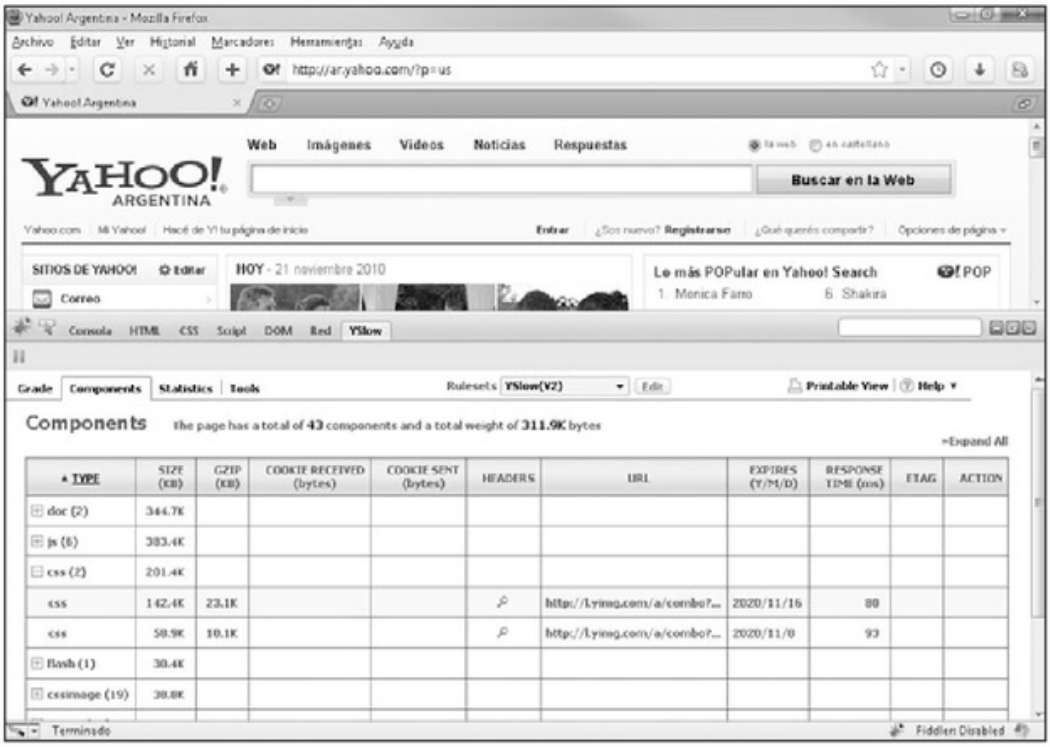


Figura 26. *Conocer los recursos consumidos por cada elemento es importante para hacer que nuestras páginas se ejecuten más rápidamente.*

- en **Tools**, disponemos de alternativas para generar reportes sobre el comportamiento de la página. **JSLint** exhibe los errores JavaScript, **All JS** devuelve todos los documentos JavaScript, **All CSS** devuelve todas las hojas estilos y **Printable Version**, una versión de impresión de las estadísticas obtenidas por YSlow.

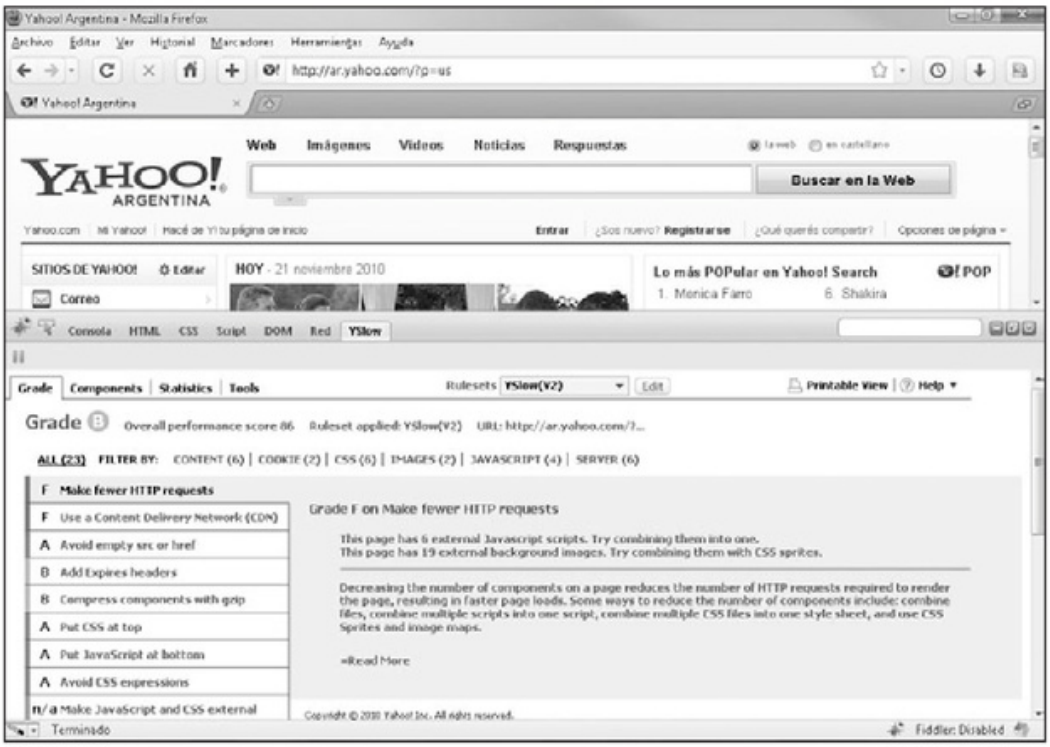


***Figura 27.** YSlow es una herramienta de control y a su vez permite generar reportes con los resultados obtenidos.*

- por último, la solapa **Help** nos redirige a distintas fuentes de información acerca de las estadísticas que YSlow nos brindó.

Web Developer

Otra gran utilidad que se encuentra disponible con el navegador web Mozilla Firefox es la llamada **Web Developer**. Se trata de una interesante extensión que hará más fácil el trabajo de los desarrolladores. Una vez que hayamos instalado esta extensión, es posible acceder a su funcionalidad mediante una barra de herramientas especial o también desde el menú contextual.

Está disponible bajo la licencia **GNU** (**www.gnu.org/licenses/gpl.txt**), y podemos descargar la última versión disponible del sitio web de Firefox: **https://addons.mozilla.org/es-ES/firefox/addon/60**, o de la página oficial de la herramienta: **http://chrispederick.com/work/web-developer/**. La forma de instalación es la usual. Debemos acceder al archivo XPI correspondiente, que es el utilizado por las extensiones Firefox, y seguir las instrucciones. Una vez reiniciado el navegador, podemos comenzar a utilizar Web Developer.

Figura 28. *El menú contextual de Web Developer facilita y agiliza el acceso a las opciones disponibles en la extensión.*

Cabe destacar que Web Developer también está disponible para Google Chrome. Más información en la dirección **https://chrome.google.com/extensions/detail/bfbameneiokkgbdmiekhjnmfkcnldhhm**.

Figura 29. *La barra de herramientas de Web Developer está dividida en solapas, cada una con un propósito específico.*

Contamos con múltiples opciones divididas en las siguientes solapas:

- **Disable**: permite deshabilitar ciertas características de la página, como por ejemplo el caché, el soporte Java, JavaScript, las redirecciones, los colores, etcétera.
- **Cookies**: desde esta opción podemos deshabilitar las cookies, eliminar las existentes para la sesión actual, ver información acerca de cada una o agregar nuevas.

***Figura 30.** La generación de cookies es de gran ayuda para emular escenarios reales en los que podría participar nuestra aplicación.*

- **CSS**: a partir de aquí tenemos la opción de deshabilitar los estilos de la página con la posibilidad de discriminar los externos, los internos, los en línea u hojas de estilo en particular; mostrar la visualización al seleccionar un tipo de dispositivo; recorrer los elementos de la página. Así es posible observar en una nueva barra a qué estilo pertenecen, añadir nuevas hojas, e incluso editar los estilos existentes y ver cómo cambia la página en tiempo real.

III INFORMACIÓN

La mayoría de las herramientas que hemos analizado en este capítulo nos da la posibilidad de generar reportes variados acerca de las características de las páginas revisadas y analizadas. Éste es un punto de gran importancia para analizar qué aspectos pueden ser mejorados para lograr un comportamiento óptimo en nuestros sistemas.

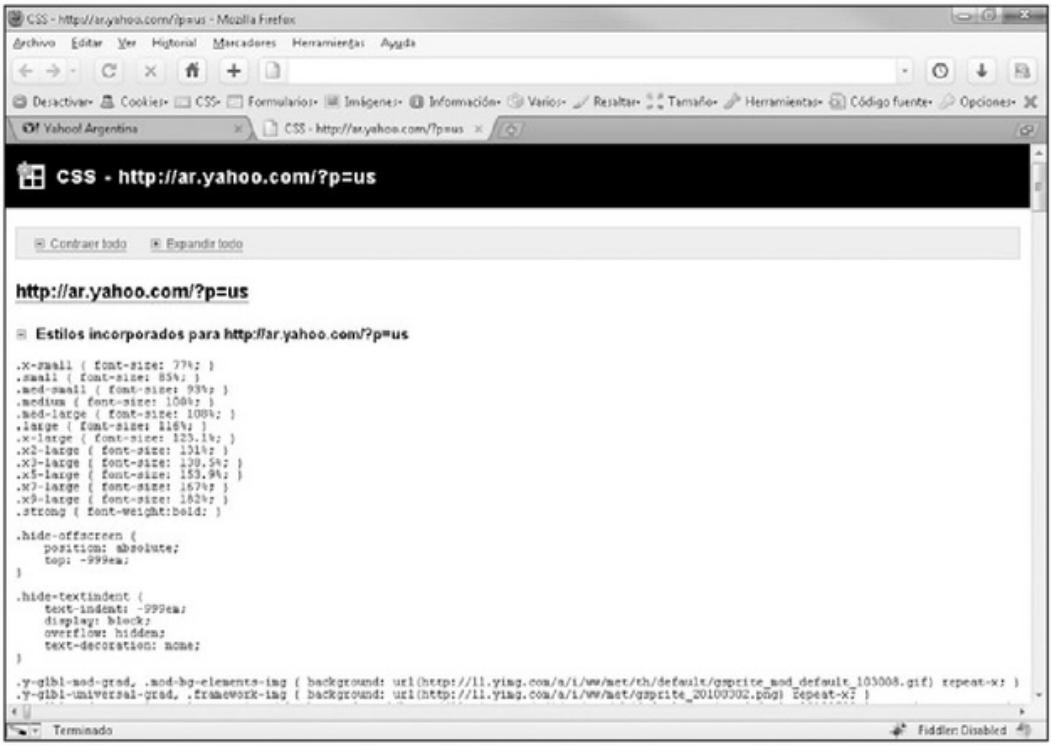


Figura 31. *La navegación de elementos y la referencia a los selectores CSS relacionados es uno de los puntos fuertes de la extensión.*

El editor permite introducir estilos desde archivos externos, modificar los ya existentes, seleccionar y trabajar sobre una determinada hoja si es que existen varias y guardar los cambios realizados.

Figura 32. *La edición CSS nos permite visualizar los resultados de las modificaciones en tiempo real.*

- **Forms**: es necesario tener en cuenta que las opciones que se encuentran disponibles en cuanto a formularios son las que mencionamos a continuación: destacar los detalles de cada uno, proceder a exhibir las contraseñas, ver información detallada de los forms y sus respectivos elementos, modificar los métodos utilizados (conocida como **POST** y **GET**), habilitar autocompletar, limpiar los botones radio y completar los campos con valores aleatorios.

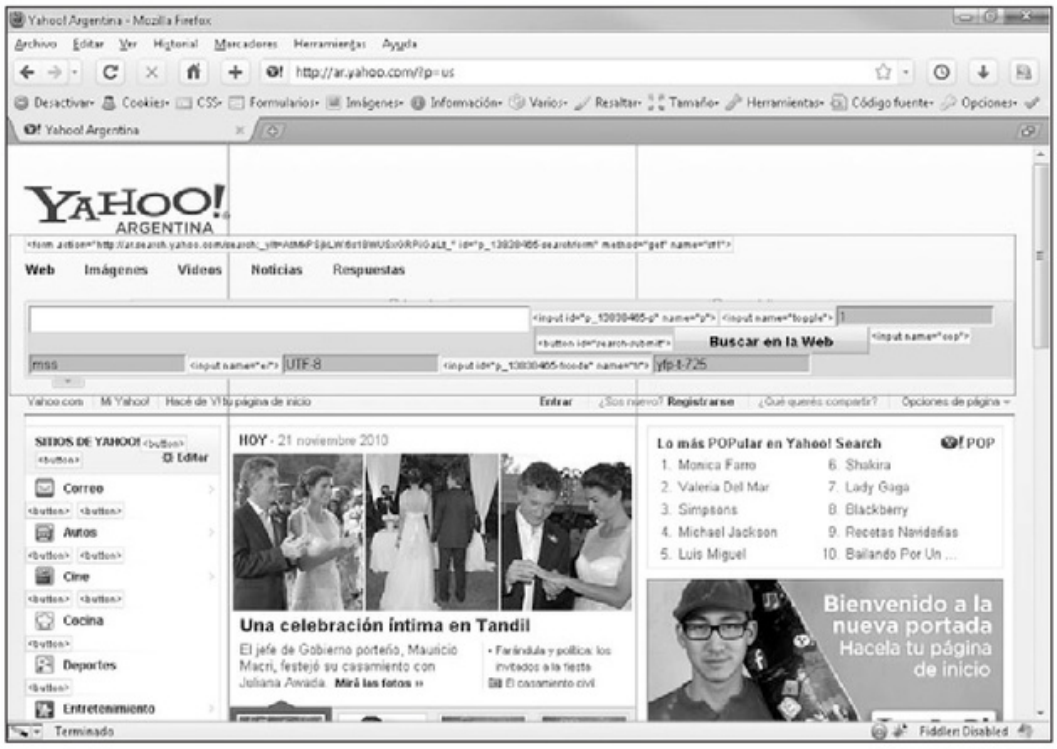


***Figura 33.** El detalle de cada formulario nos evita tener que acceder al código fuente de la página.*

- **Images**: desde aquí es posible que accedamos a deshabilitar imágenes (podemos seleccionar entre todas, solo las externas o solo las animaciones), exhibir los atributos denominados **alt**, las dimensiones de cada una, la ruta correspondiente (atributo llamado **src**), buscar y listar las imágenes mal formadas o rotas, generar un reporte con información acerca de cada imagen y ocultar imágenes de fondo, entre otras interesantes opciones disponibles.

III ORÍGENES

La extensión Web Developer se nutre de otros desarrollos para su funcionamiento, y entre los más importantes se encuentra la aplicación **PNH Developer Toolbar** (podemos encontrarla en la dirección web **www.placenamehere.com/pnhtoolbar**), que permite realizar la modificación de los aspectos relacionados con el diseño de las páginas.

Figura 34. *Acceder a una página sin cargar sus imágenes puede sernos de ayuda para los casos en los que por una u otra razón necesitamos una carga rápida solo del texto disponible.*

Para mostrar distinta información acerca de cada imagen, Web Developer incorpora a la página original etiquetas resaltadas.

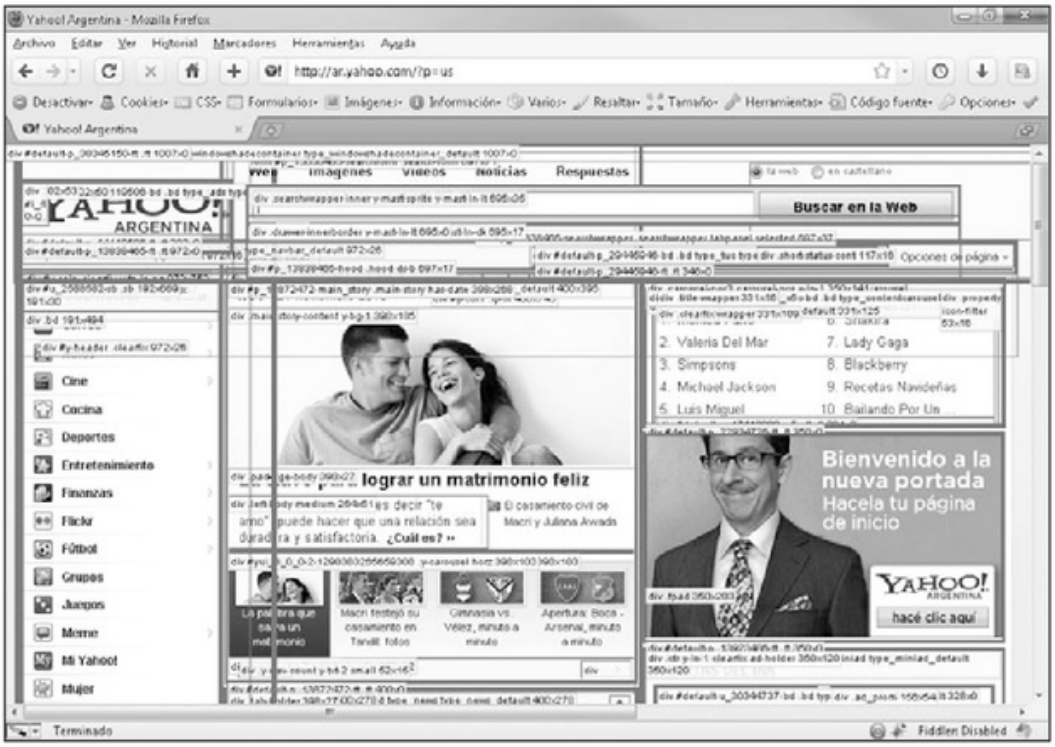


Figura 35. *Acceder al tamaño de cada elemento es sencillo y rápido gracias a Web Developer.*

- **Information**: desde aquí podemos resaltar las teclas de acceso para cada uno de los elementos, los enlaces, los tamaños de bloque, los elementos **div** y su orden dentro del documento, la información acerca de cada elemento. Para esto simplemente tenemos que hacer clic sobre uno de los elementos en particular, y podremos observar las clases y los atributos **id** de cada ítem, el detalle de los enlaces, la información de los objetos incrustados en éste, los distintos niveles de tablas, los atributos **title**, los reportes (enlaces encontrados, colores utilizados, etcétera), un informe acerca de los archivos JavaScript utilizados, los elementos **Meta** encontrados y las cabeceras de respuesta, entre otras opciones disponibles.

Figura 36. *Web Developer muestra las características de cada elemento de la página de manera visual.*

Además, tenemos la posibilidad de recuperar estadísticas acerca del documento, por ejemplo el peso de la página, el detalle de las imágenes contenidas, los objetos incluidos como una película o un archivo de sonido, los scripts y las hojas estilo.

III NAVEGADORES

Es muy importante mencionar que hoy en día existe en el mercado un gran número de navegadores web disponibles, y esta diversidad hace que sea absolutamente necesario probar y visualizar nuestras aplicaciones en los más utilizados. Las diferencias no se han podido resolver y ese tema queda en manos de los desarrolladores.

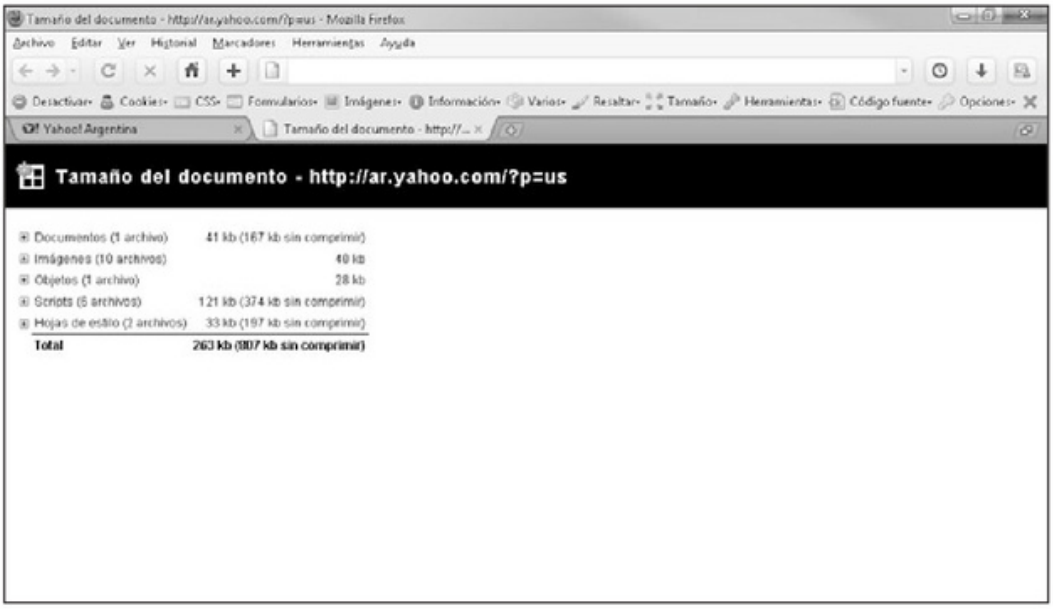


Figura 37. *Web Developer ofrece un acceso detallado a las características de los elementos que componen una página mediante sus herramientas de reporte.*

- **Miscellaneus**: algunas de las opciones son limpiar la información privada, utilizar una cuadrícula, mostrar comentarios HTML, ampliar la página, visualizar en modo pantalla reducida, editar el código HTML y mostrar los enlaces visitados.

Figura 38. *Web Developer ofrece un ampliador similar al que podemos utilizar desde los sistemas operativos Windows.*

- **Outline**: nos permite resaltar marcos, enlaces, encabezados, tablas, elementos desactualizados, o simplemente navegar y seleccionar el elemento debajo del puntero.

Figura 39. *La navegación de las partes de una página simplifica el camino a elementos no accesibles fácilmente.*

- **Resize**: podemos obtener las dimensiones actuales de la ventana, editar su tamaño y aplicar acercamientos o alejamientos con la opción **Zoom**.
- **Tools**: nos da la posibilidad de enviar la página actual a distintos servicios de validación (**CSS**, **Feed**, **HTML**, **Links**, etcétera), ver un reporte acerca de la velocidad de carga y acceso al inspector **DOM**.

Figura 40. *El sitio* ***www.websiteoptimization.com*** *brinda un informe detallado de la velocidad de carga de las páginas de un sitio.*

- **View Source**: si bien Firefox brinda la posibilidad de acceder al código fuente de la página, con esta opción Web Developer nos permite visualizar el código final generado, que puede variar del original por medio de funciones JavaScript, por ejemplo. También es posible editarlo con herramientas externas.

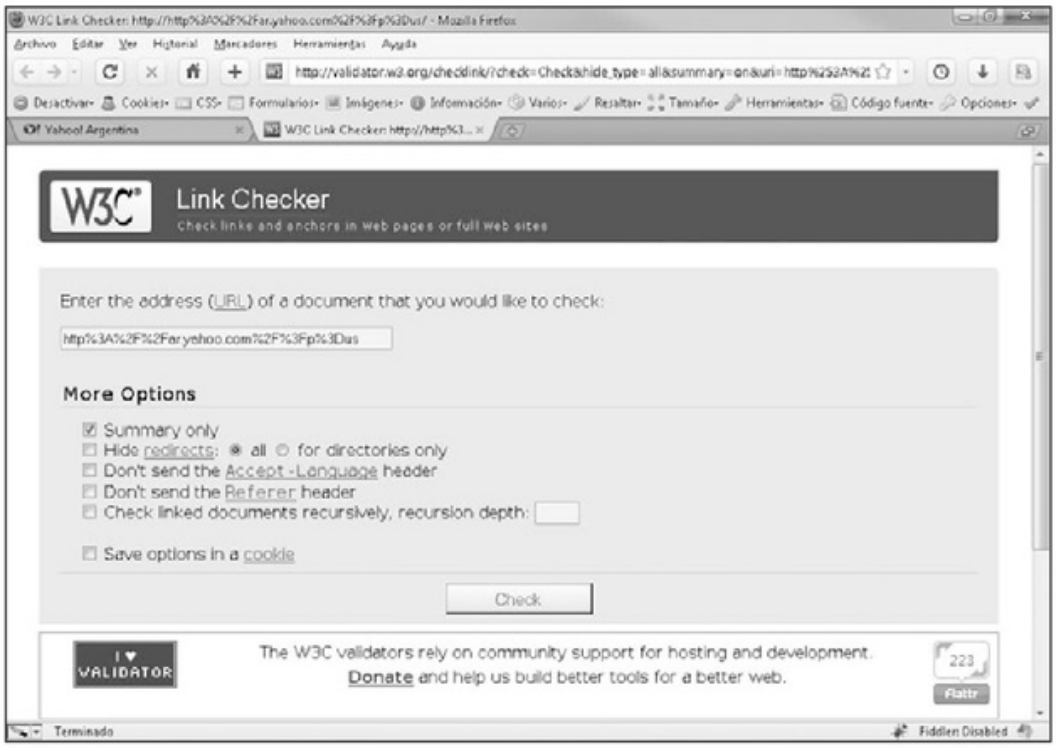


***Figura 41.** Web Developer se complementa con otras aplicaciones para que el usuario pueda personalizar el servicio según sus necesidades.*

Web Developer se encarga de guardar las configuraciones para la página actual, es decir que podríamos, por ejemplo, visualizar la página original en una pestaña y, en otra, la misma página modificada por las opciones disponibles en la extensión.
Ademas del navegador web Firefox, Web Developer está disponible para los navegadores **Flock** (se encuentra en el sitio web **www.flock.com**) y **Seamonkey** (que podemos encontrar visitando la dirección web **www.mozilla.org/projects/seamonkey**), y se puede utilizar desde cualquier sistema operativo que los admita, por ejemplo Windows, Mac OS X y Linux, entre otros.

III EL AVANCE DE SAFARI

Safari es un excelente navegador web que desde hace algún tiempo viene ganando terreno de manera notoria frente a otras alternativas presentes en el mercado. Siempre fue muy popular en los sistemas operativos Mac OS, pero la llegada de la versión para PC con sistemas operativos Windows hizo que lograra más adeptos.

Web Inspector

El navegador Safari (**www.apple.com/safari**) está disponible para múltiples sistemas operativos, entre los que se encuentran Mac OS X y Windows. **Web Inspector** es una herramienta que puede trabajar en conjunto y permite analizar páginas web mediante la obtención de información relevante, con la posibilidad incluso de modificar algunas de sus características para observar el comportamiento y obtener los resultados esperados. Web Inspector cuenta puntualmente con las siguientes características:

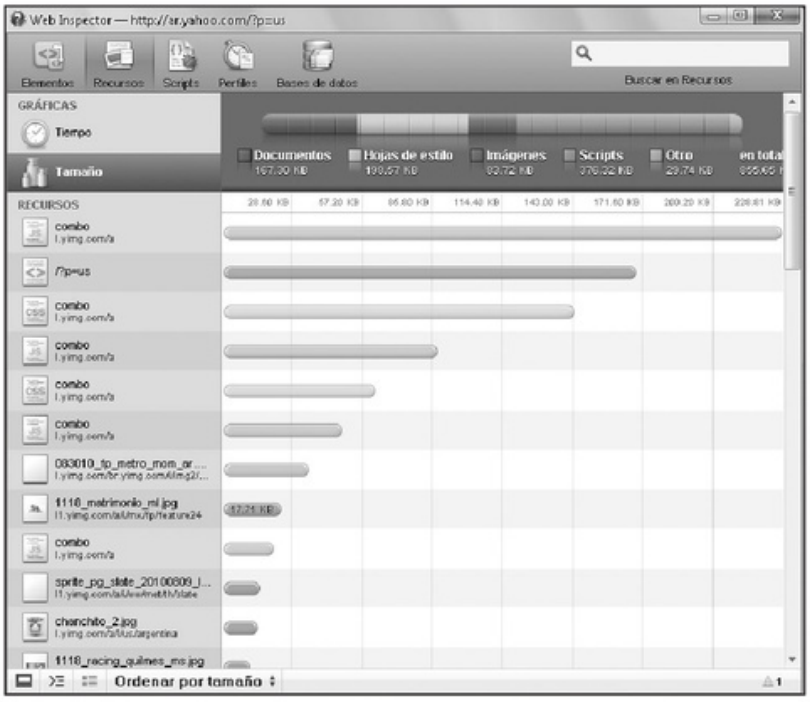


Figura 42. *Web Inspector es una de las herramientas más valoradas por los usuarios del navegador Safari.*

- Edición de hojas de estilo CSS en tiempo real.
- Soporte de las nuevas propiedades CSS.
- Existe también la posibilidad de almacenar bases de datos en la parte cliente y de esta forma acceder mediante JavaScript.
- Fuentes descargables con la directiva **@font-face**.
- Navegación del DOM de la página.

III CARACTERÍSTICAS

Es interesante tener en cuenta que una de las mayores virtudes de muchas de las aplicaciones vistas a lo largo de este capítulo consiste en la posibilidad de acceder a realizar la **edición** de distintos aspectos o partes de la página con la que estamos trabajando y también **visualizar** los cambios de manera inmediata sin modificar los archivos originales.

- Consola JavaScript para visualizar errores y advertencias, y además evaluar sentencias.
- Tiempo de carga y peso de la página.
- Listado de las cabeceras HTTP de petición y respuesta.
- Detecta errores HTML.

***Figura 43.** Web Inspector forma parte del proyecto **WebKit**, que a su vez es incorporado por múltiples herramientas en la actualidad.*

Web Inspector es una herramienta multiplataforma y forma parte de WebKit (**http://webkit.org/**), un motor de navegación web de código abierto utilizado por muchas aplicaciones del sistema operativo Mac OS X como **Safari**, **Dashboard** y **Mail**. Podemos descargar la última versión disponible del sitio **http://nightly.webkit.org/**.

Opera Dragonfly

A partir de la versión 9.5, Opera cuenta con esta herramienta que nos permite conocer aspectos relacionados con el modelo de objetos del documento (DOM), así como también todo lo asociado con los archivos JavaScript y CSS.

III UN NAVEGADOR CON HISTORIA

Opera es un navegador web que se destaca entre sus competidores por estar siempre a la vanguardia de los últimos adelantos tecnológicos. Si bien siempre se distribuyó de manera gratuita, durante las primeras versiones mantenía variantes pagas y con publicidades. En la actualidad, su uso se ofrece a quienes lo requieran en forma totalmente gratuita.

Figura 44. *El navegador Opera lleva muchos años en el mercado y provee herramientas útiles para el mantenimiento y el control de sitios.*

Para abrir el panel de Opera Dragonfly, deberemos acceder al menú **Tools/Advanced/Opera Dragonfly**. Al principio podemos observar siete pestañas o **tabs**:

- **DOM**: permite la inspección de los elementos del documento actual.
- **Scripts**: nos da la posibilidad de visualizar y monitorear los scripts referenciados dentro del documento actual.
- **Network**: se exhiben estadísticas de la página, como por ejemplo el tamaño de cada componente cargado, el tiempo de carga individual, el tamaño total del documento y el número de peticiones.
- **Storage**: lista los archivos que el sitio almaceno en nuestro sistema (cookies recibidas).
- **Error Console**: listado de errores encontrados (CSS y JavaScript).
- **Utilities**: Selector de color.
- **Settings**: Nos permite configurar en detalle las demás opciones disponibles.

III OPERA

Este navegador se caracteriza por estar a la vanguardia en lo que a funcionalidades se refiere. Aun cuando su cuota de mercado en PCs es baja, la competencia suele incorporar tiempo después ciertas características que Opera ofrecía en sus versiones anteriores. Por esta razón se trata de una alternativa que debemos tener en cuenta.

Para acceder rápidamente a la propiedades de un elemento de la pagina que estamos visitando, podemos hacer clic con el botón derecho sobre el elemento y luego ir a la opción **Inspect Element**. Esto abrirá la ventana principal de la herramienta.

XRAY

Esta herramienta consta de un inspector DOM y funciona en Internet Explorer 6 y superiores, los navegadores que utilizan WebKit, por ejemplo Safari, y los basados en Mozilla, como Firefox, Camino, etcétera. No requiere instalación, simplemente debemos acceder al sitio oficial de la herramienta: **http://westciv.com/xray**, y agregar a los marcadores el enlace correspondiente:

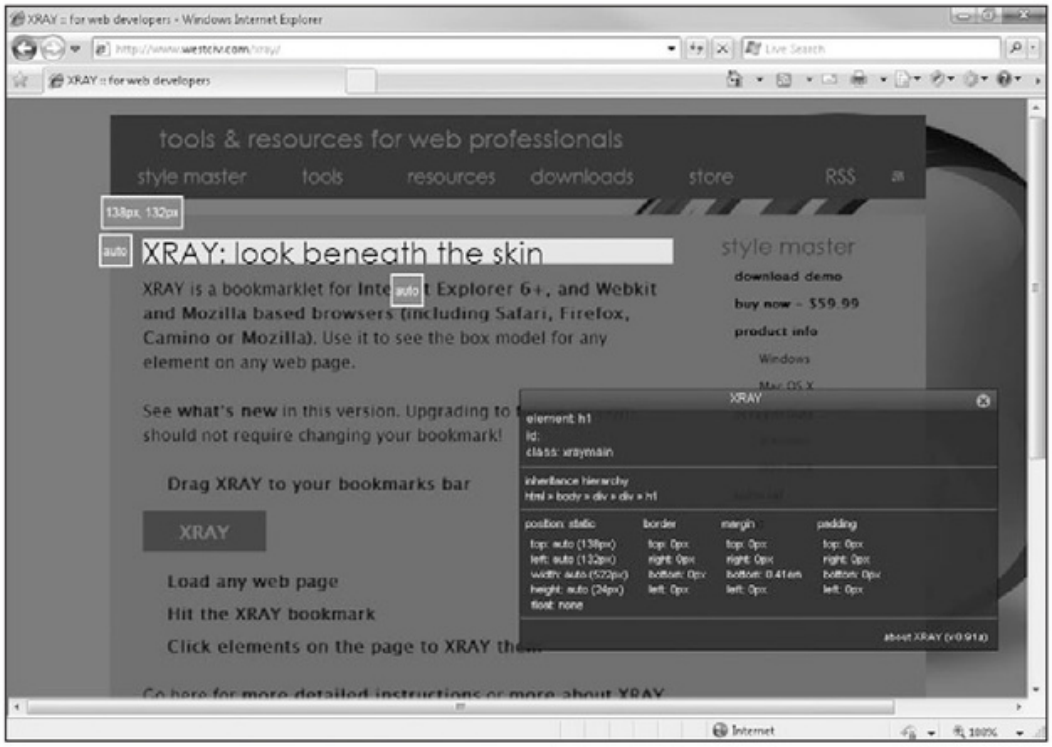


Figura 45. *XRAY es una curiosa y útil herramienta para inspeccionar sitios web de manera rápida y sin complicaciones.*

Para comenzar a utilizarlo, debemos acceder a la página que queremos inspeccionar y, luego, acceder al enlace guardado en los marcadores. Esto abrirá una capa por encima de la página actual. Si hacemos clic sobre un elemento, podemos ver sus propiedades y atributos. Para desactivar **XRAY** cerramos la capa o simplemente refrescamos la página.

DebugBar

Disponible para el navegador Internet Explorer, **DebugBar** es otro inspector de páginas web. Podemos descargar la última versión disponible de **www.debugbar.com/download.php?langage=en**. Una vez concretada la instalación podemos visualizar

las opciones disponibles si vamos a menú **Ver/Barras de Herramientas/DebugBar**. DebugBar cuenta con una barra horizontal y una vertical, que es la barra de debug o de desarrollo. Dentro de la primera contamos con las siguientes opciones:

- personalización de DebugBar y actualizaciones disponibles.
- mostrar u ocultar la barra de desarrollo.
- mostrar errores JavaScript.
- enviar captura de la página actual por correo electrónico.
- selector de color.
- cambiar la resolución de la página.
- zoom para ampliar o reducir la visión de la página.
- ver código fuente de la página actual, el original y el final.
- ver código fuente de alguna de las páginas abiertas.

Figura 46. *DebugBar funciona bajo Internet Explorer y permite la inspección de las características de una página.*

Por su parte, la barra de desarrollo de DebugBar está dividida en solapas y admite las siguientes opciones listadas a continuación:

- **DOM**: desde aquí podemos navegar el modelo de objetos del documento, visualizar el listado de enlaces (**Links**, resaltados), las imágenes (**Images**, resaltadas), los formularios (**Forms**, resaltados), el código JavaScript (**Scripts**) y las hojas de estilo (**Style Sheets**, con la posibilidad de editar las propiedades de cada elemento y visualizar la vista previa con los cambios).

- **HTTP(S)**: desde aquí podemos obtener un detalle acerca de las peticiones realizadas al servidor, por ejemplo:

```
GET /us.yimg.com/i/ww/news/2008/01/28/sagawards.jpg HTTP/1.1
Accept: */*
Referer: http://www.yahoo.com/
Accept-Language: es-ar
Accept-Encoding: gzip, deflate

HTTP/1.1 200 OK
Content-Length: 9915
Content-Type: image/jpeg
Expires: Thu, 25 Jan 2018 17:02:28 GMT
Last-Modified: Mon, 28 Jan 2008 15:16:14 GMT

Request start time: Mon, 28 Jan 2008 14:07:42
All request retrieved from cache (no server request)

URL: http://us.i1.yimg.com/us.yimg.com/i/ww/news/2008/01/28/sagawards.jpg
Content Type: image/jpeg
Headers size (bytes): 318
Data size (bytes): 9915
Total size (bytes): 10233
Transferred data size (bytes): 0

Cached data: Yes
Cache File: C:\DOCUME~1\abc\CONFIG~1\Temp\Archivos temporales de
   Internet\Content.IE5\AZGHKZKZ\sagawards[1].jpg
Cache expires: Thu, 25 Jan 2018 14:02:28
Cache last modified: Mon, 28 Jan 2008 12:16:14
```

- **Script**: contiene un listado de todas las funciones JavaScript cargadas y también es posible obtener su definición. Al pie del panel, contamos con un espacio para ejecutar instrucciones JavaScript personalizadas.
- **HTML Check**: esta opción nos permite validar el documento actual con la herramienta **Tidy** (que hemos repasado en capítulos anteriores) o **CSE Validator** (**www.htmlvalidator.com**).
- **Info**: brinda estadísticas de la página, como el nombre, el título, el tamaño, el número de imágenes, el tamaño ocupado por las imágenes, el tiempo de descarga, el juego de caracteres **Mime**, el protocolo y el modo de visualización.

DebugBar mantiene una versión gratuita para uso personal y una paga para obtener las funcionalidades de la versión comercial. Podemos obtener más información en el sitio **www.debugbar.com**.

Companion.JS

Esta opción funciona bajo Internet Explorer y se complementa con DebugBar (ambas están desarrolladas por la misma compañía). Es un debugger para JavaScript y entre sus características principales figuran un reporte de errores detallado y una consola al estilo Firebug. Uno de los requisitos de **Companion.JS** es contar con la aplicación **Microsoft Script Debugger** previamente instalada. En el sitio oficial hay un enlace para descargarla. Una vez que instalamos Companion.JS, debemos acceder a las opciones de Internet Explorer (menú **Herramientas/Opciones de Internet**) y quitar la selección de las opciones:

- deshabilitar la depuración de las secuencias de comandos (Internet Explorer).
- deshabilitar la depuración de las secuencias de comandos (Otros).

Para visualizar la consola debemos acceder desde Internet Explorer al menú **Ver/Barra del Explorador/Companion.JS**. Así podemos observar en la parte inferior de la pantalla una opción para acceder al código fuente de la página y otro para ingresar instrucciones JavaScript.

Figura 47. *La consola de Companion.JS nos permite obtener un acceso rápido a las características de la página.*

Podemos obtener más información en el sitio web oficial de la herramienta: **www.my-debugbar.com/wiki/CcompanionJS/HomePage**.

Internet Explorer Developer Toolbar

Como su nombre lo indica, esta aplicación es una barra de herramientas que funciona bajo Internet Explorer, y da opciones para acceder y modificar las características de las páginas web exploradas. Para descargar este programa debemos ingresar al sitio de **Microsoft** (**www.microsoft.com/downloads**) y buscar el enlace correspondiente, o a **http://bit.ly/aTvrh2**. Una vez concluida la instalación, para visualizar la barra debemos acceder desde Internet Explorer al menú **Ver/Barra del Explorador/IE Developer Toolbar** o utilizar el icono ubicado en la barra de herramientas con los botones estándares o presionando la tecla F12:

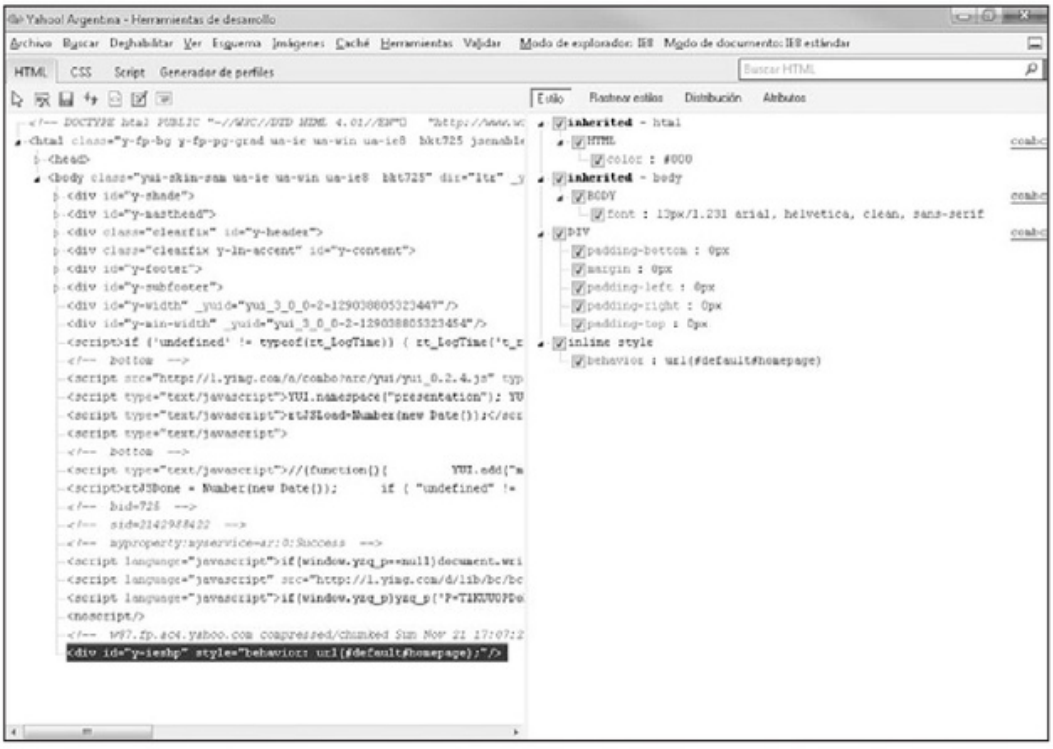


Figura 48. *Internet Explorer Developer Toolbar es una de las opciones más utilizadas por los usuarios de este popular navegador.*

Debemos tener en cuenta que inicialmente, contamos con un visualizador del DOM correspondiente a la página que se encuentra activa, desde el cual podemos seleccionar algunos elementos en particular. Para hacer lo mismo de manera gráfica disponemos del primer icono ubicado en la barra, **Select Element By Click**, que activado permite seleccionar un elemento si hacemos clic sobre éste. A la derecha del árbol aparecen dos paneles más: uno para visualizar las propiedades del elemento seleccionado, con la posibilidad de editar sus valores o ingresar nuevas propiedades, y otro para ver a qué estilo CSS pertenecen.

Además de **Select Element By Click**, contamos con iconos para refrescar la página, limpiar el caché del navegador y visualizar solo el código del elemento seleccionado. Entre los menús tenemos los siguientes:

- **File**: se encarga de deshacer cambios.
- **Find**: buscar elementos por nombre o navegar el documento.
- **Disable**: deshabilitar scripts, ventanas emergentes u hojas de estilo CSS.
- **View**: visualizar los atributos **Class** y **Id** de cada elemento, las rutas a cada uno, un reporte con los enlaces disponibles en la página, el orden de los elementos, las teclas de acceso de cada uno, el código fuente de la página y el número de usos de cada estilo definido en la hoja de estilos.
- **Outline**: exhibir los bordes de las celdas, de las tablas, elementos **div**, imágenes, algún elemento en particular y elementos según su posicionamiento.
- **Images**: deshabilitar las imágenes del documento, mostrar las dimensiones de cada una, sus nombres, las rutas a los archivos, las etiquetas **Alt**, y un reporte acerca de las imágenes incluidas.
- **Cache**: siempre actualizar la página, limpiar el caché, limpiar el caché solo para este dominio, deshabilitar cookies, eliminar cookies, eliminar cookies solo para este dominio y ver información adicional.
- **Tools**: ésta opción nos permite modificar el tamaño de la resolución de la página, mostrar regla, utilizar el selector de color.
- **Validate**: validar HTML, CSS, Feed, Links, HTML y CSS locales, verificar la accesibilidad y aplicar validaciones múltiples.

En definitiva, **Internet Explorer Developer Toolbar** es una herramienta muy completa similar en algún punto a la extensión Web Developer de Firefox. A diferencia de las demás opciones disponibles, Internet Explorer Developer Toolbar es un proyecto oficial de la empresa Microsoft.

RESUMEN

En este capítulo observamos algunas de las herramientas disponibles para modificar el diseño y la estructura de los documentos desde el entorno mismo de un navegador y verificar las estadísticas relacionadas con el tiempo de carga y el peso de los elementos. Además, vimos cómo depurar código JavaScript y generar reportes acerca de las características de cada página.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Cuál cree que es la principal ventaja de Firebug?

2 ¿Qué programa utiliza para depurar sus scripts?

3 ¿Cómo prueba sus aplicaciones Ajax?

4 ¿De qué manera edita sus hojas de estilo?

5 ¿En qué casos utilizaría la línea de comandos de Firebug?

6 ¿Qué utilidad tiene la opción Inspect de Firebug?

7 ¿Qué es WebKit?

8 ¿Cuál es la opción para visualizar los comentarios HTML con Web Developer?

9 ¿Cuál es la utilidad de editar hojas de estilo con Firebug?

10 ¿Cuál es la utilidad de editar documentos HTML con Web Developer?

EJERCICIOS PRÁCTICOS

1 ¿En qué casos aplicaría la opción View Source de Web Developer?

2 Desarrolle una aplicación Ajax y verifique su comportamiento con Firebug.

3 Ingrese al sitio del diario Clarín y verifique las características de los formularios de la página inicial con Web Developer.

4 Calcule el peso de un sitio con imágenes y sin éstas.

5 Calcule el número de peticiones a una determinada función JavaScript con las funciones monitor y unmonitor de Firebug.

Posicionar sitios en Google

En este capítulo observaremos algunas de las características más destacadas de Google, que es el motor de búsquedas más popular de la actualidad, e intentaremos orientar la estructura de nuestros sitios para lograr mejores posicionamientos en los listados de resultados de las búsquedas.

Introducción	268
Herramientas disponibles	269
Sitios de calidad	271
Rastreo	273
Googlebot	275
Índices	276
Sitemaps	278
Resumen	279
Actividades	280

INTRODUCCIÓN

Es indudable que sin los **motores de búsqueda** el acceso a Internet sería muy diferente, y lograr encontrar información acerca de un determinado tema sería complicado. En consecuencia, existiría la necesidad de mantener una serie de innumerables directorios organizados por temas con direcciones a los enlaces correspondientes, que es justamente una de las tareas que cumplen los buscadores. Pero la gran ventaja de estas aplicaciones es que logran, mediante mecanismos complejos, relevar una enorme cantidad de páginas cuyo contenido luego podrá ser comparado con un criterio de búsqueda que haya sido ingresado por el usuario.
Cada cierto período de tiempo **Google** recorre las páginas de la red y revisa su contenido. De allí recupera cierta información (palabras, frases, valores de algunos elementos y atributos XHTML) para generar un **índice de contenidos**. Al realizar una búsqueda, los resultados devueltos por el motor de Google serán seleccionados del índice teniendo en cuenta múltiples factores, entre los cuales uno de los más importantes es el contenido de la página y también su relación con el criterio de búsqueda que haya sido definido por el usuario.
Otro factor que aumenta la aparición e incluso la ubicación dentro del listado de resultados está dado por la cantidad de sitios que incluyen enlaces a nuestras páginas.
Podemos resumir las principales características de un motor de búsqueda básicamente en los puntos que se mencionan a continuación:

- rastreo de páginas.
- actualización periódica de los índices.
- sistema de posicionamiento de resultados.
- gran velocidad de respuesta.

Si bien Google intenta no dar demasiados detalles técnicos acerca del funcionamiento de su buscador para evitar el **spam** y la manipulación de los resultados por parte de los usuarios avanzados, en los siguientes apartados resumiremos de manera general y comprensible cómo enfocar nuestros esfuerzos para lograr ubicaciones de privilegio en los listados de búsqueda.

III ALTERNATIVAS

Google no es el único buscador existente, aunque si podemos afirmar que se trata del más popular en la actualidad. Otros directorios, como por ejemplo **Open Directory Project** y **Yahoo!** son muy populares y es absolutamente recomendable hacer saber de la existencia de nuestro sitio a éstas y otras empresas relacionadas con la catalogación de sitios en la red.

HERRAMIENTAS DISPONIBLES

Google reúne las opciones para visualizar y modificar los aspectos relacionados a nuestro sitio en lo referido al motor de búsquedas en su sección para **webmasters** (autores de sitios). Éstas son una ayuda invalorable para nosotros.

***Figura 1.** Con sus herramientas, Google nos da la posibilidad de controlar de cerca el desempeño de los sitios web.*

Lo primero que haremos será iniciar sesión con una cuenta Google. Podemos abrir una nueva en **www.google.com/accounts/NewAccount**. Luego registremos nuestro sitio en la dirección **www.google.com/webmasters/tools?hl=es** con la opción **Añadir sitio**. Debemos incluir la URL completa, por ejemplo:

```
http://www.nombre-sitio.com/
```

III LOCACIONES

Google se encarga de mantener los llamados centros de datos, los cuales son los encargados de resolver las búsquedas que han sido realizadas por los usuarios. Existen múltiples centros ubicados geográficamente en lugares diferentes, y es posible que cada uno actualice sus índices en distintos momentos, así es posible obtener resultados que varían según el lugar.

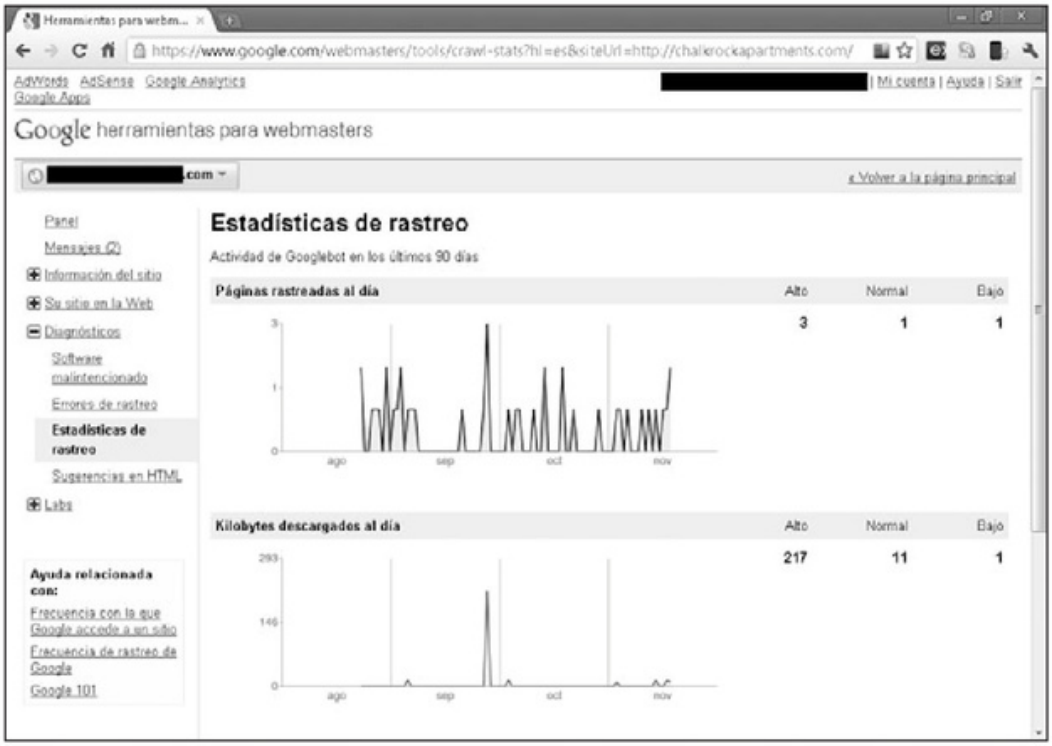


Figura 2. *Google provee un servicio de estadísticas gratuito para nuestros sitios.*

El paso siguiente será comprobar la propiedad del sitio. Este mecanismo es necesario para que Google pueda validar que somos los responsables de la dirección registrada. Junto al nombre del sitio aparecerá el enlace **Verificar**, y si hacemos clic podremos observar las opciones disponibles para la confirmación:

- agregarle una etiqueta **meta** a la página principal del sitio.
- cargar un archivo HTML.

En el primer caso, Google nos asignará una cabecera para agregar a la página inicial de nuestro sitio, por ejemplo **index.html**, y un valor para asignarle. La otra alternativa consiste en subir un archivo cuyo nombre será definido por el buscador Google al directorio raíz del servidor correspondiente.
Si por algún motivo otras personas han registrado nuestro sitio, podemos ingresar a **Herramientas/Administrar la verificación del sitio web** y obtener un listado con las cabeceras o los archivos definidos por Google para cada responsable. Para eliminar sus derechos debemos editar el índice para quitar la cabecera o borrar el archivo HTML, según corresponda. Cada cierto período de tiempo Google vuelve a verificar la propiedad del sitio con alguno de los métodos seleccionados.
Una vez terminado el proceso, podemos conocer de manera profunda cómo Google recorre e indexa nuestro sitio, y si surgieron inconvenientes durante el proceso. Además, se listan las consultas habituales que devuelven a nuestro sitio como parte del resultado de búsqueda. Esto nos permitirá conocer cómo llegan los usuarios a conocernos y cuáles son sus requerimientos.

Podemos obtener más información acerca de las últimas novedades, los artículos y las opiniones acerca de diversos temas relacionados en el **Centro para Webmasters de Google**, disponible en **http://googlewebmastercentral.blogspot.com**. Asimismo, existe un grupo de discusión y ayuda para los autores de sitios en **www.google.com/support/forum/p/Webmasters?hl=en**.

Figura 3. *Google mantiene un blog especialmente dedicado al intercambio de opiniones entre los responsables de sitios web.*

SITIOS DE CALIDAD

Google pone a disposición de los responsables de los sitios una serie de directivas o consejos que tienden a facilitar el trabajo del motor de búsqueda y a que las páginas aparezcan en posiciones de privilegio en los listados. Algunos consejos prácticos son:

- ligar páginas entre sí mediante enlaces para que Google las rastree. Los vínculos deben ser de texto estático (no en **Flash** ni con imágenes). En este sentido contamos con dos opciones: **Páginas con enlaces externos** y **Páginas con enlaces internos**.
- relacionado con lo anterior, debemos tratar de usar texto para contenidos importantes que deban ser indexados por Google.
- utilizar **sitemaps** (mapas del sitio, veremos más sobre ellos en las próximas secciones).
- incluir en el contenido de las páginas las palabras que los usuarios normalmente utilizarían para localizar un sitio como el nuestro.

- incluir el elemento **title** y los atributos **alt** con contenido descriptivo.
- no incluir demasiados vínculos dentro de una página (no más de 100).
- para páginas dinámicas tratar de que las direcciones sean en lo posible cortas, tomando en cuenta lo que sigue al carácter **?** (**?idArticulo=1&modo=a&...**).
- utilizar archivos **robots.txt** para restringir el acceso.
- tratar de que los enlaces no incluyan identificadores de sesión.

***Figura 4.** Podemos obtener el listado completo de las recomendaciones y sugerencias por parte de Google en su Centro de Asistencia para webmasters.*

Luego de cumplir estos pasos, Google recomienda intentar que otros sitios de calidad, es decir, los que respetan las reglas recomendadas por el buscador, mantengan enlaces al nuestro. Esta clase de observaciones son apreciadas por Google. En las próximas secciones veremos cómo esto es importante para el motor de búsquedas. Una de las cuestiones que pueden hacer que Google elimine un sitio de los resultados de búsqueda es la inclusión de texto accesible por los motores, pero no por los

III CONTENIDO EN FORMATO TEXTO

Es muy importante tener en cuenta que los motores de búsqueda se encuentran basados en texto, por esta razón a lo que deberemos prestar atención es a la utilización de tecnologías como **JavaScript**, **DHTML**, imágenes o incluso archivos **Flash**, ya que los analizadores de Google no podrán indexar su contenido en forma correcta.

usuarios, es decir en general todas las acciones que intenten engañar a los motores de búsqueda para lograr mejores posiciones en los resultados.
Google plantea todas sus reglas y todos sus consejos en una dirección: que nuestras páginas tengan un contenido original y de calidad, algo que derivará en la atención de los usuarios y de los demás sitios relacionados, que incluirán enlaces al nuestro.

RASTREO

La cabecera **keywords**, disponible desde hace tiempo en XHTML, les proporcionaba a los diseñadores de páginas web la posibilidad de incluir palabras clave para que fueran tomadas por los motores de búsqueda en la confección de las páginas de resultados. Sin embargo, esta herramienta dejó de ser tomada en cuenta por la mayoría de los buscadores, entre otros motivos porque la alternativa de dejar en manos del autor la responsabilidad de definir las características de sus páginas era poco viable, en especial porque para aumentar sus visitas podrían llegar a incluirse términos que no se correspondían con el contenido real. En lugar de esta cabecera, los motores de búsqueda comenzaron a indexar el contenido de los sitios.
Al momento de intentar resolver una búsqueda, el rastreador de Google realiza un relevamiento sobre las páginas indexadas y devuelve los resultados más relevantes. Una de las ideas detrás de esto es que, mientras más detallado sea el contenido de un sitio, más probabilidades habrá de que contenga alguna de las palabras ingresadas por el usuario. Por otra parte, indexar páginas hace que los resultados tengan correspondencia con el contenido de los documentos, es decir con lo que realmente contienen y no con lo que el autor de la página afirma.
Por lo tanto, Google sostiene que un sitio aparecerá en un listado de resultados si alguno de los términos de búsqueda ingresados por el usuario forma parte de alguna de las páginas que han sido indexadas.

Google nos provee herramientas que pueden sernos de utilidad para saber cómo el contenido de nuestro sitio es visto por parte de los usuarios. Existe una sección en

III PROGRESO

En la actualidad, el buscador Google es una de las empresas más importantes a nivel general, por fuera del ámbito informático. Comenzó como un proyecto universitario y, gracias a la calidad lograda por sus autores, logró expandirse. Hoy en día, mantiene muchas otras aplicaciones de propósito general además del buscador, las cuales están a disposición de los usuarios.

el panel de control que lista las 20 consultas más habituales en las que aparecen nuestras páginas (opción **Consultas de búsqueda más habituales**), y otra en la que se registran los 20 criterios de búsqueda a partir de los cuales los usuarios ingresaron a nuestro sitio (opción **Clics en las búsquedas más habituales**). Además, contamos con la posibilidad de conocer qué sitios enlazan al nuestro y cómo lo hacen, es decir qué frases utilizan para relacionarse con nosotros.
Otra opción disponible es la de verificar la frecuencia con la que Google rastreó las páginas en los últimos 90 días. Para conocer estos valores (solo en las páginas del directorio raíz) debemos ingresar al menú **Herramientas/Establecer frecuencia de rastreo**. Podemos conocer el numero de páginas rastreadas, el tiempo promedio utilizado en el procesamiento y los kilobytes descargados. Desde aquí incluso podremos definir de manera personalizada la frecuencia de rastreo.

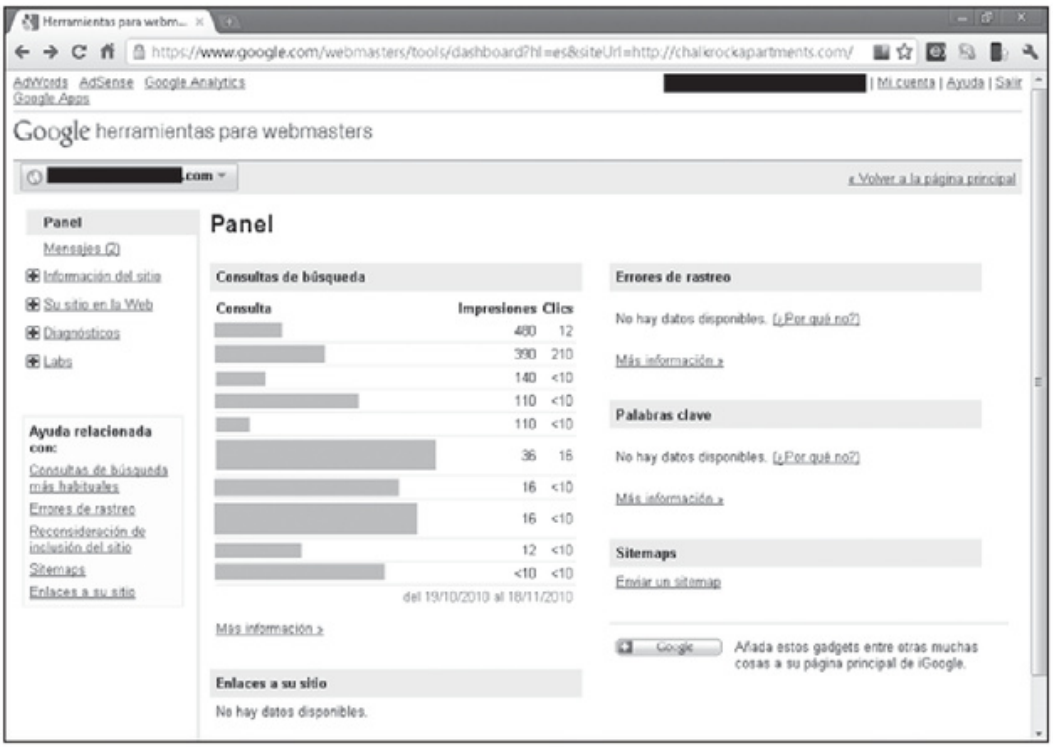


***Figura 5.** Desde el panel de control de **Google Webmaster Tools** podemos observar cómo acceden los usuarios a nuestros sitios.*

Además, una de las claves que le permite a una página escalar posiciones dentro de los resultados de búsqueda es contar con la mayor cantidad de enlaces posibles en otros sitios considerados por Google como **de calidad**. La lógica es que mientras más sitios enlacen, más importancia le dan y más valioso creen que es para los usuarios. Este sistema de posicionamiento conocido como **PageRank** es la base de Google, que considera un voto por cada enlace de un sitio a otro. No es el único criterio pero sí es uno de los más importantes. Una manera rápida de conocer qué sitios enlazan nuestras páginas es con el operador **link**. Para utilizarlo debemos escribir en el cuadro de búsquedas de Google lo siguiente:

```
link:www.nombre-sitio.com
```

En donde el texto **www.nombre-sitio.com** corresponde a la dirección de nuestro sitio web; luego de esto veremos los resultados relacionados.
En las secciones **PageRank de sus páginas en Google** y **Su página con el PageRank más elevado**, podemos obtener detalles acerca del posicionamiento de nuestras páginas. Además del PageRank hay otros factores que influyen en la ubicación que ocupa un sitio en una página de resultados, como veremos más adelante.
Podemos obtener más información acerca de cómo Google clasifica las páginas en **www.google.com/corporate/tech.html**.

GOOGLEBOT

La aplicación que se encarga de agregar nuevas páginas y actualizar las ya existentes en el índice se denomina **Googlebot**. Las listas de páginas que se rastrean se obtienen a partir de enlaces encontrados en las páginas visitadas, en las listas anteriores y en los **sitemaps**. A partir de este procesamiento, el índice se actualiza teniendo en cuenta los nuevos sitios encontrados, los cambios en los que ya estaban agregados y aquellos que ya no están accesibles.
Debemos tener en cuenta que puede ocurrir que Googlebot intente rastrear una página, pero que por algún motivo no pueda realizar esta acción y, por lo tanto, no la agregue al índice. Esta clase de errores se pueden tratar a partir de las descripciones ubicadas en la sección de herramientas para los autores de sitios de Google que encontramos en la siguiente ruta: **Diagnóstico/Rastreo web**, y también en **Diagnóstico/Rastreo móvil**. Allí se listan los inconvenientes surgidos y los consejos para solucionarlos, en los casos en los que sea posible. Una vez hecho esto, Googlebot intentará visitar las páginas nuevamente.
Como vimos en los capítulos anteriores, el archivo **robots.txt** nos permite regular el acceso a nuestras páginas por parte de los buscadores. Éste puede ser el motivo por el que alguna de nuestras páginas no aparezca listada en las búsquedas. Google mantiene un analizador disponible para esta clase de archivos, que nos dará la posibilidad de conocer de manera práctica qué páginas o directorios están bloqueados a Googlebot. Es posible obtener más información acerca de su funcionalidad si visitamos el sitio web que podemos encontrar en la siguiente dirección **www.google.com/support/webmasters/bin/answer.py?hl=es&answer=156449**.
Desde el enlace **Estadísticas detectadas por Googlebot**, podemos obtener detalles acerca de cómo Googlebot analiza nuestras páginas (las palabras y frases más habituales, y el tipo de contenido y juego de caracteres de cada una, entre otros datos).

***Figura 6**. Si bien Google analiza la web de manera automática en busca de nuevas apariciones o modificaciones, podemos agregar nuestros sitios manualmente si completamos un formulario.*

En resumen, Google mantiene una serie de procesos que finalizan en la muestra de resultados al usuario luego de una búsqueda:

- Rastreo y recopilación de páginas.
- Generación del índice.
- Publicación de resultados.

Es importante aclarar que, más allá de los enlaces patrocinados, Google no interfiere de manera arbitraria en las posiciones que las páginas ocupan en los resultados.

ÍNDICES

Luego de rastrear y visitar cada página, Google genera un índice que contiene todas las palabras encontradas (más los valores de los elementos **title** y los atributos **alt**). Esto permite que sean utilizados los criterios de búsqueda ingresados por el usuario para que se pueda devolver un resultado concreto.
El índice del contenido de los sitios mantenidos por Google evoluciona de manera permanente, se actualiza de forma periódica y tiene repercusión en los resultados de las búsquedas. Sin dudas, este dinamismo es una de las claves del éxito para este motor.
Para verificar que nuestro sitio está incluido en el índice debemos escribir lo siguiente en el cuadro de búsquedas de Google:

```
site:www.nombre-sitio.com
```

En donde **www.nombre-sitio.com** es la dirección en cuestión. Además, desde este enlace se nos informa acerca de la última vez que se accedió a nuestra página principal. Debemos recordar que Google actualiza periódicamente sus índices, por lo que es posible que, si un sitio es demasiado reciente, se incluya en el próximo rastreo.

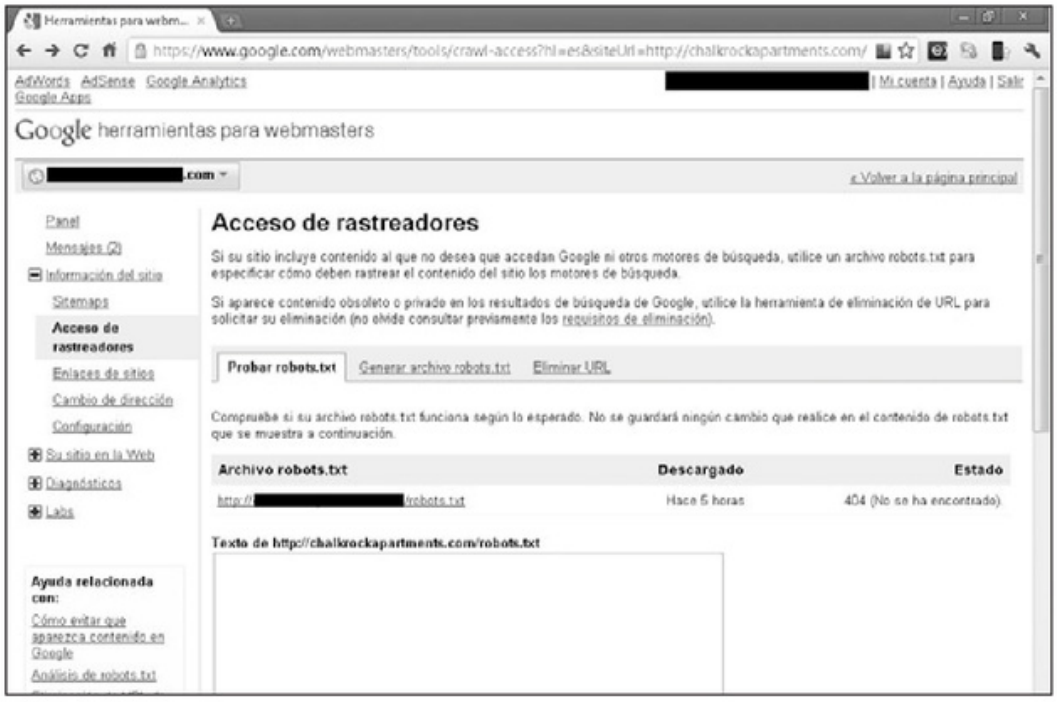


Figura 7. *Google nos permite observar los últimos accesos a nuestras páginas en detalle mediante el panel de control.*

Google pone a disposición de los usuarios una serie de operadores avanzados (**www.google.com/support/websearch/bin/answer.py?hl=en&answer=136861**) para obtener información acerca de nuestro sitio, y **site** es uno de ellos. Otros son las páginas externas que nos enlazan (el ya visto **link**), la versión caché de nuestra página principal (**cache**), la información general (**info**), y las páginas que Google considera similares (**related**). Todos tienen la misma sintaxis: **operador:www.nombre-sitio.com**.

III MÉTODOS

Aunque la oferta de soluciones mágicas abunde en el mercado, los pasos y las acciones que nos permitirán ubicar a nuestras páginas en la cima de los listados de resultados de Google, son conocidas y están al alcance de cualquiera: solo se trata de brindar la información justa, detallada y de valor acerca de un tema en particular.

SITEMAPS

Estos archivos permiten darle a conocer a Google qué páginas de nuestro sitio consideramos de más importancia, más allá de que se encuentren en el índice o no. En la práctica, se trata de documentos que pueden ser enviados a Google y que contienen un listado con páginas de nuestro sitio. Debemos saber que un **sitemap** puede contener hasta 50.000 direcciones y pesar hasta 10 MB. Además, un sitemap puede contener referencias a otros sitemaps.

En las aplicaciones web actuales es usual encontrar páginas para los llamados mapas del sitio (**sitemaps**), que son documentos que listan a los enlaces más importantes y que sirven como guía para que el usuario visualice la estructura del sitio. Este término no debe confundirse con el utilizado por Google.

Son muy útiles para que Googlebot verifique páginas que utilizan **Ajax** o películas **Flash**, a las que el motor de búsqueda no puede acceder fácilmente. Otra razón para utilizarlo es cuando nuestro sitio es muy reciente y contiene pocos vínculos en páginas de otros sitios, y todavía no ha sido visitado e indexado por Google. Además, pueden sernos de utilidad para trabajar con contenidos dinámicos.

Como mencionamos antes, una de las técnicas que Googlebot utiliza es la de rastrear enlaces en las páginas y agregarlos a una lista para su posterior indexación. Por este motivo, si nuestras páginas no están ligadas unas a otras es posible que Google tenga problemas para analizarlas todas, y en este caso los sitemaps pueden ayudar a que Google conozca las páginas componentes del sitio.

***Figura 8.** Los sitemaps le permiten al autor de sitios demandarle a Google el registro de la información contenida en una página.*

Otras funcionalidades disponibles con la utilización de sitemaps son definir la frecuencia de actualización de cada página por parte de Googlebot y la fecha de la última modificación de cada una de ellas, y categorizar las páginas internamente por nivel de importancia, lo que nos permite personalizar el sitio según nuestro criterio. Hay opciones para las páginas y también para los contenidos específicos como:

- sitemaps de video (**Google Video**).
- sitemaps para móviles (**Google Mobile**).
- sitemaps de noticias (**Google News**).
- sitemaps para búsquedas de código (**Google Code Search**).

En resumen, la utilización de sitemaps es de suma utilidad para forzar la indexación de documentos por parte de Google en los casos en que por un motivo u otro esto no ocurra naturalmente. Podemos obtener más información acerca de los sitemaps en:

- **www.sitemaps.org/es**.
- **www.google.com/webmasters/sitemaps**.
- **http://code.google.com/p/googlesitemapgenerator**.

RESUMEN

En este capítulo observamos todo lo referente a cómo Google se relaciona con los autores o desarrolladores de páginas web para trabajar en conjunto, y pone a disposición de ellos herramientas especializadas para conocer las estadísticas de acceso y manipular las opciones relativas a la interacción con el motor de búsquedas.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 ¿Qué es un sitio de calidad para Google?

2 ¿Qué factores principales influyen en el posicionamiento de los sitios en los resultados de búsqueda?

3 ¿Qué relación hay entre el PageRank y los sitios de calidad?

4 ¿Qué es Google Webmaster Tools?

5 ¿Qué entiende por indexación de contenido?

6 ¿Cuál es la función de Googlebot?

7 ¿En qué casos utilizaría sitemaps?

EJERCICIOS PRÁCTICOS

1 Verifique el PageRank de su sitio.

2 Verifique las páginas externas que enlazan a su sitio.

3 Verifique las páginas que Google considera similares a su sitio.

4 Genere un sitemap para su sitio.

5 Verifique las consultas más habituales en relación con su sitio.

Anexo A

Referencia JavaScript

Con el correr del tiempo JavaScript se ha convertido en el lenguaje por excelencia para desarrollar tareas o funciones de propósito general en el ámbito de una página web.

En este capítulo veremos las principales características y posibilidades de este lenguaje para comenzar a entender el por qué de su éxito.

Introducción	**282**
Sintaxis básica	**284**
Comentarios	285
Variables	285
Operadores disponibles	288
Tipos de datos	289
Estructuras de control	291
Funciones	296
Implementación DOM	301
Eventos	305
Manejo y control de errores	308

INTRODUCCIÓN

El lenguaje de programación JavaScript se utiliza principalmente para desarrollar páginas web (tiene otros usos además de éste), y se complementa con otras tecnologías como **XHTML**, **CSS** y otros lenguajes de programación del lado servidor como **PHP**, **ASP.net**, **JSP** o **PERL**, por ejemplo.

Es considerado como un lenguaje que actúa en el lado cliente, lo que significa que el código es resuelto por el navegador o por la aplicación que se utilice para acceder a documentos web, sin interacción con el servidor.

Se trata de un lenguaje interpretado, es decir, que los scripts no necesitan ser compilados, y que para funcionar precisan contar con un intérprete, que es incluido por la gran mayoría de los navegadores actuales.

La primera versión de JavaScript, desarrollada por **Brendan Eich**, fue incluida junto al navegador **Netscape** en 1995. El lenguaje inicialmente se denominó **LiveScript**, y luego esta empresa, en alianza con **Sun Microsystems** que comenzó a participar en el desarrollo, cambiaron el nombre por el de **JavaScript**. Al poco tiempo, **Microsoft** lanzó junto a **Internet Explorer 3** un lenguaje similar: **JScript**.

Para propulsar el lenguaje, Netscape decidió estandarizar JavaScript, es decir, hacerlo independiente de cualquier empresa y darle la posibilidad a toda persona que incluya sus propias implementaciones. El organismo **ECMA** (*European Computer Manufacturers Association*) fue el encargado de definir las bases del lenguaje denominado oficialmente **ECMAScript** (**ECMA-262**).

***Figura 1.** ECMA es una de las organizaciones con más historia dentro del ámbito de las empresas informáticas.*

En la actualidad, tanto JavaScript (**Mozilla Firefox**, **Konqueror**, **Opera** y **Safari**, entre otros) como JScript (**Internet Explorer**) podrían definirse como implementaciones del estándar ECMAScript. A pesar de su denominación, JavaScript no tiene relación directa con el lenguaje de programación **Java**, aunque su sintaxis es similar (también se la relaciona con la del lenguaje C).
Los documentos JavaScript poseen la extensión **js** y pueden ser creados y editados con cualquier editor de texto, como **Notepad** o **Vi** de **Windows** y **Unix** respectivamente. En los primeros capítulos de este libro observamos las diferentes alternativas para incluir código JavaScript en nuestros documentos XHTML.
Si bien los navegadores de Internet actuales pueden interpretar el código del lenguaje JavaScript (hasta la tercera edición del estándar ECMA-262), esto puede ser desactivado desde el panel de configuración del navegador, por los usuarios. Por su parte, es posible que los navegadores muy antiguos no den soporte. Para estos casos existe la etiqueta **noscript**, cuyo contenido es mostrado por el navegador cuando por un motivo u otro no es posible invocar al intérprete JavaScript:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <noscript>
 <b> texto de muestra ... </b>
 </noscript>

 </body>
</html>
```

III CREADOR DE JAVASCRIPT

Brendan Eich ha pasado por varias empresas de *Sillicon Valley* y participó de varios proyectos anteriores al desarrollo del lenguaje *JavaScript*. Él comenzó su carrera como programador en **Silicon Graphics**, luego en **MicroUnity Systems** entre otras, hasta llegar a **Netscape** donde comenzó el desarrollo de un lenguaje llamado en un principio *Mocha* y luego *LiveScripts*.

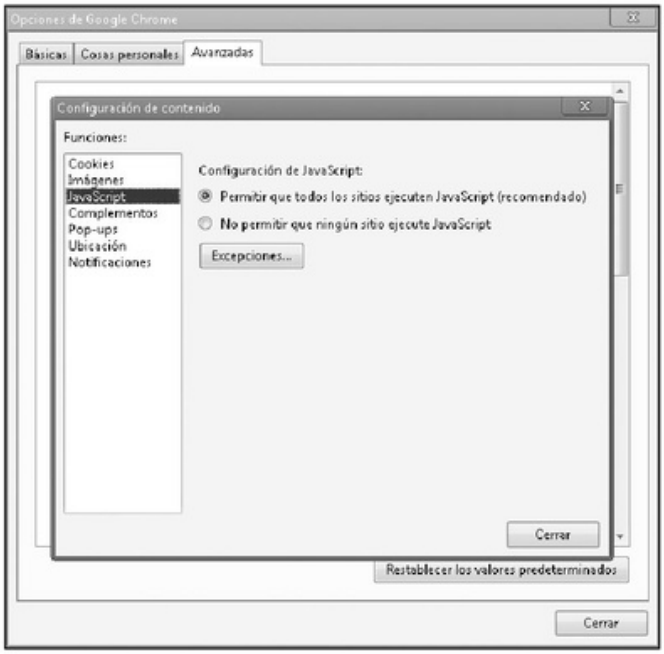


Figura 2. *En la actualidad, los navegadores le dan la posibilidad al usuario de deshabilitar la ejecución de código JavaScript, ya sea de manera parcial o incluso total.*

SINTAXIS BÁSICA

Si bien JavaScript es un lenguaje orientado a objetos, en los siguientes apartados incluiremos las nociones básicas para desarrollar scripts sencillos, sin ahondar en cuestiones relacionadas con el paradigma en sí mismo. Algunas características del lenguaje:

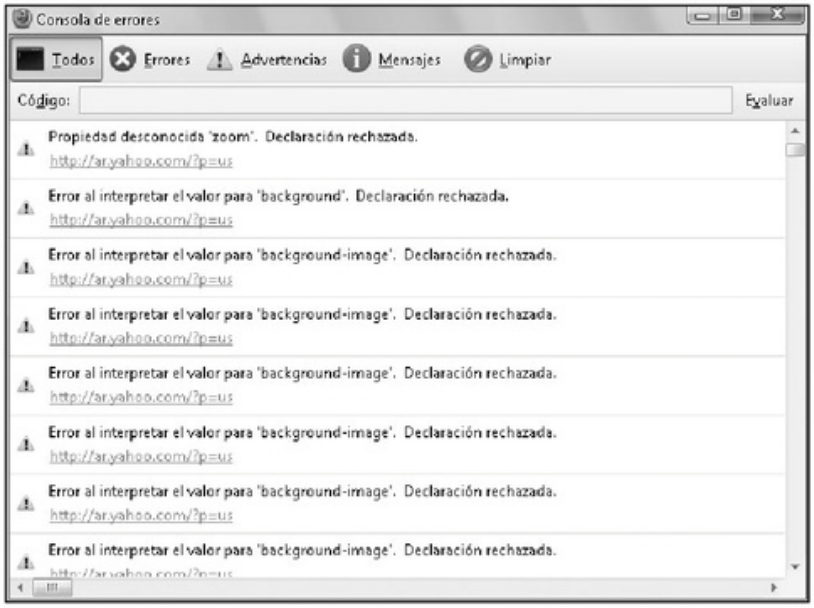


Figura 3. *Algunos navegadores como Mozilla Firefox incluyen herramientas especiales para interactuar con código JavaScript.*

- es tipado, es decir que diferencia entre mayúsculas y minúsculas.
- no es necesario indicar el tipo de dato de una variable.

Comentarios

En JavaScript hay disponibles básicamente dos tipos de comentarios:

- los de una línea,

```
// comentario de un linea
```

- y los de múltiples líneas.

```
/*

comentario
de
multiples
lineas

*/
```

Variables

Las variables en JavaScript se declaran al anteponer la palabra reservada **var**. Veamos un ejemplo:

```
var a;
var b;
```

III ESTILO DE LENGUAJE

JavaScript es un lenguaje de programación interpretado, basado en el estilo de sintaxis del lenguaje **Java** y el lenguaje **C**. Éste es orientado a objetos y soporta herencia, la cual se basa en el paradigma de programación basada en prototipos. Las nuevas clases se generan clonando cualquiera de sus clases bases y luego expandiendo su funcionalidad.

```
var c;
a = 5;
b = 6;
c = a * b;
```

Es posible asignar valores al momento de la declaración:

```
var a = 5;
var b = 6;
var c = a * b;
```

Un nombre de variable solo puede contener letras, números y símbolos **$** y _. El primer carácter no puede ser numérico.

```
var pi = 3.14;
var $dia = "Lunes";
var mes_4 = 'Abril';
var anioActual = 2008;
var ok = true;
var no = false;
```

Además, es conveniente no utilizar ningún nombre o ninguna palabra reservada del lenguaje, tanto para las variables como para los nombres de las funciones:

PALABRAS RESERVADAS		
abstract	float	public
boolean	for	return
break	function	short
byte	goto	static
case	if	super
catch	implements	switch
char	import	synchronized
class	in	this
const	instanceof	throw
continue	int	throws
default	interface	transient
do	long	true
double	native	try

PALABRAS RESERVADAS		
else	new	var
extends	null	void
false	package	while
final	private	with
finally	protected	

Tabla 1. *Listado de palabras reservadas en JavaScript.*

No es obligatorio declarar variables, un estilo últimamente utilizado en muchos lenguajes, aunque es una buena costumbre hacerlo cuando sea necesario. Si una variable no se declara, se considerará como global. En el siguiente ejemplo dentro de **sumar** no declaramos la variable **a**, por lo que se considera como global:

```
var a = 10;
alert(a);
sumar();
alert(a);

function sumar() {
  a = a + 1;
}
```

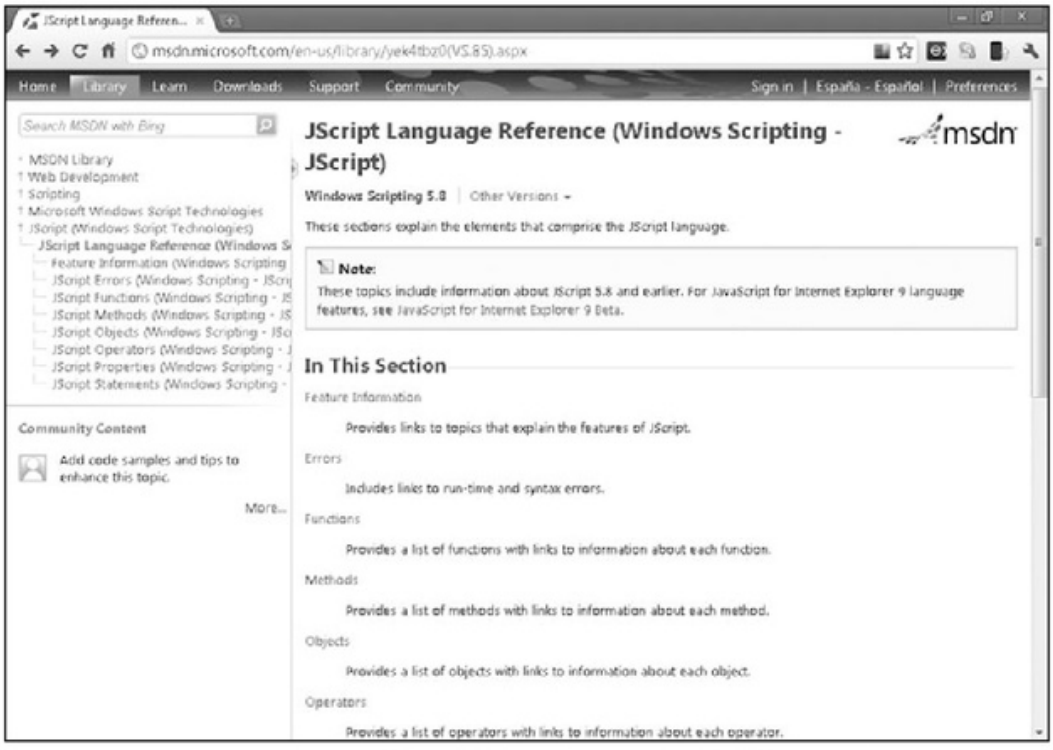


Figura 4. *Microsoft mantiene su propia implementación de la especificación ECMAScript.*

En el siguiente código nos encargamos de realizar la definición de una variable **a** específica para **sumar**, que es local y no afecta a las demás:

```
var a = 10;
alert(a);
sumar();
alert(a);
function sumar() {
  var a = a + 1;
}
```

Operadores disponibles

El operador = implica asignarle un valor a una variable. También existen otros operadores válidos como los que se muestran a continuación:

• incremento (++)

```
var x = 3;
var y;
alert(x + x++); //6 (3+3), evalua e incrementa
alert(x + ++x); //9 (4+5), incrementa y evalua
```

• decremento (—)

```
var x = 3;
var y;
alert(x + x–); //6 (3+3), evalua y decrementa
alert(x + –x); //3 (2+1), decrementa y evalua
```

• negación (!)

```
var x = true;
alert(!x); //true si x es distinto de true, false si x es true
```

Debemos tener en cuenta que para los números y las cadenas, **0** (cero) y **""** (vacío) son respectivamente equivalentes a **false**.

• y (**and**)

```
var x = true;
var y = true;
alert(x && y); //true si ambos son verdaderos
```

• alternativa (**or**)

```
var x = true;
var y = true;
alert(x || y); //true si al menos un es verdadero
```

• mayor que (**>**)
• menor que (**<**)
• mayor o igual que (**>=**)
• menor o igual que (**<=**)
• igual que (**==**)
• distinto que (**!=**)

Entre los operadores matemáticos se encuentran:

• suma (**+**).
• resta (**-**).
• multiplicación (*****).
• división (**/**).
• módulo (**%**).

En el contexto de las cadenas de caracteres el operador de suma se utiliza para realizar la concatenación, como vemos a continuación:

```
var a = "uno";
var b = "dos";
var c = a + " " + c; //uno dos
```

Tipos de datos

Dentro del lenguaje Javascript encontramos cuatro tipos de datos básicos: números, cadenas de texto, **booleanos** y **arrays**.

• Los números pueden ser enteros o decimales (el punto es el separador).

MÉTODO	DESCRIPCIÓN
isNaN	Devuelve verdadero si se aplica sobre un valor no numérico (NaN: Not a Number).
toFixed	Redondea el número y deja una cierta cantidad de posiciones decimales (definidos en el argumento pasado).

***Tabla 2.** Algunos de los métodos disponibles para el trabajo con valores numéricos.*

• Las cadenas se encierran entre comillas (simples o dobles).

MÉTODO	DESCRIPCIÓN
charAt	Devuelve el carácter ubicado en la posición pasada como argumento.
concat	Concatenación de cadenas de caracteres. Se aplica sobre una variable y se le da como argumento a la cadena a concatenar.
indexOf	Devuelve la posición correspondiente a la primera aparición del carácter pasado como argumento (-1 si no ha sido encontrado).
lastIndexOf	Devuelve la posición correspondiente a la última aparición del carácter pasado como argumento (-1 si no ha sido encontrado).
length	Longitud de la cadena de texto.
split	Recibe como argumento un separador y devuelve un array. Cada vez que se encuentra un separador, se crea una nueva posición para el array.
substring	Recibe como argumentos la posición inicial y final, y recupera esa parte de la cadena.
toLowerCase	Convierte la cadena en minúsculas.
toUpperCase	Convierte la cadena en mayúsculas.

***Tabla 3.** Algunos de los métodos disponibles para el trabajo con cadenas de caracteres.*

• Los booleanos pueden tomar dos valores: **true** y **false**.
• JavaScript, al igual que la mayoría de los lenguajes, permite incluir arrays o colecciones de variables.

```
var meses = ["Enero", "Abril", "Agosto"];
```

Un array puede contener valores de cualquier tipo (cadenas de caracteres, números, valores booleanos, otros arrays) y para acceder a cada posición (comenzando a partir de cero) debemos utilizar la sintaxis:

```
alert(meses[1]); //Abril
```

En el siguiente ejemplo accedemos a un array que forma parte de otro:

```
var dias2 = ["Jueves", "Viernes"];
var dias1 = ["Lunes", "Martes", "Miercoles", dias2];
alert(dias1[3][1]); //Viernes
```

MÉTODO	DESCRIPCIÓN
concat	Recibe como argumento un array y lo concatena al original.
join	Une todos los elementos del array y forma una cadena de texto. Como argumento recibe un separador para delimitar las cadenas.
length	Calcula la longitud del array (número de posiciones).
pop	Elimina el último elemento del array y lo devuelve.
push	Agrega un elemento, cuyo valor se pasa como argumento, al final del array.
reverse	Invierte las posiciones del array original.
shift	Elimina el primer elemento del array y lo devuelve.
unshift	Agrega un elemento, cuyo valor se pasa como argumento, al principio del array.

***Tabla 4.** Algunos de los métodos disponibles para el trabajo con arrays.*

Estructuras de control

En cuanto a las estructuras de control, en JavaScript están disponibles las siguientes:

- **if**. El código contenido se ejecuta si la condición es verdadera.

```
if (condicion) {
  //...
} else if (condicion2) {
  //...
} else if (condicion3) {
  //...
} else if (condicionN) {
  //...
} else {
  //...
}
```

Tanto **else if** como **else** son opcionales. Si ninguna es verdadera se ejecuta la del **else**, si es que está definida. Veamos un ejemplo:

```
var a = 10;
```

```
if (a > 100) {
  alert('Mas de cien');
} else if (a < 100) {
  alert('Menos de cien');
} else {
  alert('Cien');
}
```

Es posible evaluar más de una condición con los operadores disponibles:

```
if (!a && (x == 100 || b % 2)) {
  //...
}
```

- **for**. Se trata de una estructura repetitiva cuya sintaxis es la siguiente:

```
for (var c=0; c<10; c++) {
  alert(c);
}
```

Primero, la variable **c** toma el valor 0 (esto sucede la primera vez). Si c es menor que 10, entonces se ejecutan las instrucciones contenidas dentro de la estructura. Luego de la primera ejecución, el valor de **c** se incrementa en 1 y se vuelve a evaluar si **c** es menor que 10. El proceso termina cuando **c** sea mayor o igual a 10. Podemos utilizar cualquier operador tanto para la condición como para la actualización del índice:

```
for (var i=20; i>=10; i–)
{
  alert(i);
}
```

- **for in**. Se utiliza para recorrer estructuras de datos como arrays u objetos. Su sintaxis básica es la siguiente:

```
var dias = ["Lunes", "Martes", "Miercoles", "Jueves", "Viernes",
   "Sabado", "Domingo"];
```

```
for(c in dias) {
  alert(dias[c]);
}
```

En cada iteración, la variable **c** toma el valor del índice correspondiente (inicialmente toma el primero, luego el segundo, y así hasta el último disponible).
Tanto **for** como **for in** admiten la terminación y el salto manual de las iteraciones con las construcciones **break** y **continue**. La sentencia **break** nos permite discontinuar un bucle de manera definitiva:

```
for (var c=1; c<=10; c++) {
  alert(c);
  if (c == 9) {
    alert("Termina");
    break;
  }
}
```

En el ejemplo, cuando **c** llega a 9 se muestra un mensaje y el ciclo termina. En cuanto a **continue**, podemos utilizarla para omitir determinadas vueltas:

```
for (var c=1; c<=10; c++) {
  if (c >= 5 && c <= 7) {
    alert("No Cuenta");
    continue;
  } else {
    alert(c);
  }
}
```

MODELO DE OBJETOS

Para evitar estas incompatibilidades, la W3C (World Wide Web Consortium) ha diseñado el estándar **Document Object Model** (**DOM**, o Modelo de Objetos del Documento), que incorporan Konqueror, las versiones 6 y superiores de Internet Explorer y Netscape Navigator, Opera versión 7, Mozilla desde su primera versión y Firefox por supuesto.

Si **c** está entre 5 y 7, el ciclo se omite.

- **while**. Se utiliza para ejecutar instrucciones mientras se cumple una sentencia:

```
var x = 1;

while (x<10) {
  x++;
}
```

Un ciclo de este tipo puede no ejecutarse nunca:

```
var x = false;

while (x) {
  alert("Nunca");
}
```

- **do while**. Es similar a la anterior solo que garantiza al menos una ejecución. Se utiliza para ejecutar instrucciones mientras se cumple una sentencia:

```
var x = false;

do {
  alert("Nunca ?");
} while (x)
```

Las instrucciones dentro del bucle se ejecutan y luego de la primera vuelta se evalúa la condición. Si se cumple se repite el ciclo, de lo contrario, no.

III JAVASCRIPT NO OBSTRUCTIVO

Es un paradigma floreciente en el uso del lenguaje de programación **JavaScript**, utilizado en la Web. El término no se define formalmente, sus principios incluyen la separación de la funcionalidad JavaScript de las capas de **estructura/contenido** y de presentación de un página, y el uso de buenas prácticas para evitar incompatibilida de código entre navegadores.

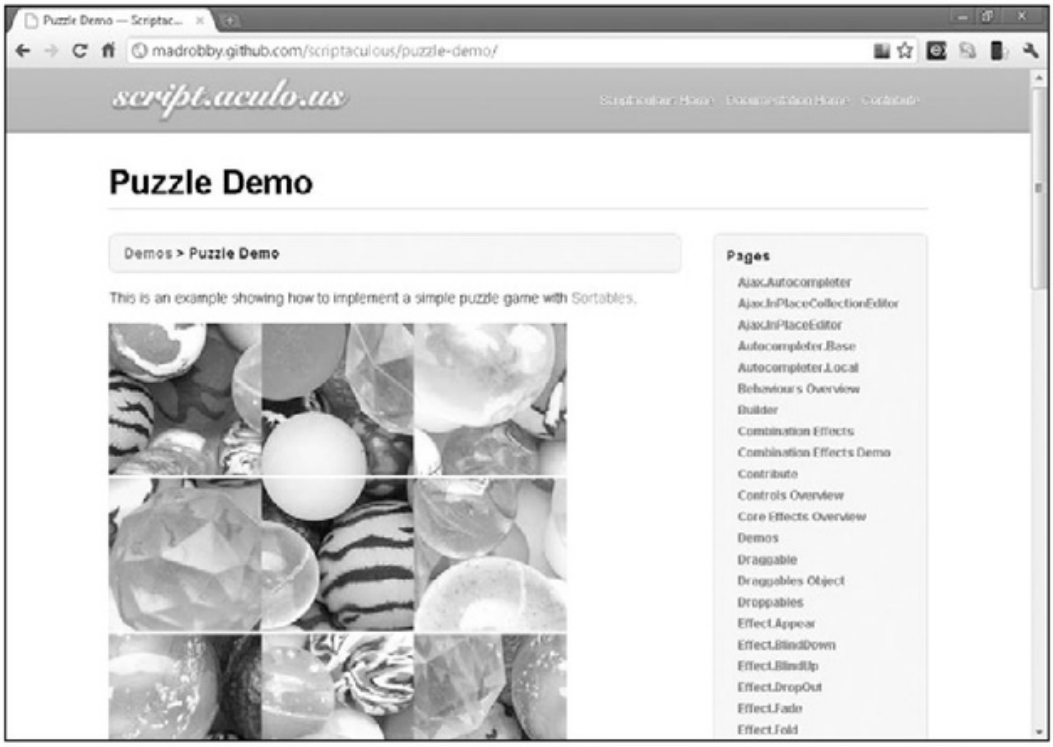


Figura 5. Prototype y script.aculo.us son dos herramientas basadas en JavaScript que han generado una revalorización del lenguaje.

- **switch**. Cuando tenemos acciones que se tienen que evaluar dependiendo del valor de una variable, podemos usar esta opción en lugar de **if else if**. Por ejemplo:

```
var x = 5;

switch(x) {
  case 1:
    //...
    //...
    //...
    break;
  case 2:
    //...
    //...
    //...
    break;
  case 3:
    //...
    //...
    //...
    break;
```

```
  case 8:
    //...
    //...
    //...
    break;
  default:
    //...
    //...
    //...
    break;
}
```

Si la variable no corresponde a ningún caso, se evalúa el contenido de la sección **default**, cuya inclusión es opcional.

Funciones

Las funciones nos sirven para aislar fragmentos de código que utilizaremos más de una vez a lo largo del script, o simplemente para clarificar la programación dividiendo una aplicación en módulos.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 var a = 10;
 alert("El doble de a es " + a*2);

 var b = 11;
 alert("El doble de b es " + b*2);
```

```
 var c = 33;
 alert("El doble de c es " + c*2);

 var d = 1;
 alert("El doble de d es " + d*2);

 //]]>
 </script>

 </body>
</html>
```

Con funciones:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 var a = 10;
 calcularDoble('a', a);

 var b = 11;
 calcularDoble('b', b);

 var c = 33;
 calcularDoble('c', c);

 var d = 1;
 calcularDoble('d', d);

 function calcularDoble(variable, valor) {
```

```
    alert("El doble de " + variable + " es " + valor*2);
  }

  //]]>
  </script>

  </body>
</html>
```

Una función puede recibir argumentos o no:

```
function nombreFuncion(arg1, arg2, argN) {
  //...
}
```

Siempre se debe respetar el número y el orden de definición de los argumentos:

```
var nombre = "Francisco";
var edad = 28;

nombreFuncion(nombre, edad);
//no nombreFuncion(edad, nombre);

function nombreFuncion(nombre, edad) {
  //...
}
```

Los nombres de los argumentos no tienen necesariamente que coincidir:

```
var n = "Francisco";
var e = 28;

nombreFuncion(n, e);

function nombreFuncion(nombre, edad) {
  //...
}
```

Para devolver valores utilizamos la palabra reservada **return**:

```
var radio = 21;

var sup = superficie(radio);

alert(sup);

function superficie(radio) {
  var resultado = 3.14 * radio * 2;
  return resultado;
}
```

Las funciones son de utilidad para conocer la diferencia entre las variables locales y las globales. Si reformulamos el ejemplo anterior:

```
var radio = 21;

var sup;

superficie(radio);

alert(sup);

function superficie(radio) {
  sup = 3.14 * radio * 2;
}
```

podemos observar que la variable **sup** se declara (**var**) en el script pero no dentro de **superficie**, por lo que allí dentro se asume a **sup** como global y se modifica su valor. En cambio, en el siguiente ejemplo:

```
var radio = 21;

var sup;

superficie(radio);
```

```
alert(sup);

function superficie(radio) {
  var sup = 3.14 * radio * 2;
}
```

la variable **sup** se declara dentro de la función **superficie** y de alguna manera se independiza, se convierte en local y no interfiere con la variable **sup** declarada por fuera de la función. Las variables definidas por fuera de las funciones son consideradas globales. Por otra parte, las variables que son declaradas dentro de una función no pueden ser utilizadas desde fuera de ella, por ejemplo:

```
x();

alert(p);

function x() {
  var p = 11;
}
```

Figura 6. *Si probamos la vigencia del lenguaje, cada vez son más las librerías y bibliotecas que eligen a JavaScript como herramienta base.* ***jQuery*** *es una de ellas (****http://jquery.com****).*

Implementación DOM

El **DOM** (*Document Object Model*, Modelo de Objetos del Documento) es un estándar que permite visualizar y tratar a un documento XHTML mediante las jerarquías internas formadas por los elementos que lo componen. En sus orígenes fue pensado para su uso con documentos XML en general, pero su mayor éxito fue logrado al trabajar en conjunto con XHTML. En este contexto de implementación se incluyen distintos tipos de elementos o **nodos**:

- **Document** (nodo raíz, o sea **HTML**).
- **Element** (etiquetas XHTML).
- **Attr** (atributos de elementos).
- **Text** (contenido de los elementos).
- **Comment** (comentarios).

Una gran variedad de lenguajes implementa el DOM, y JavaScript es uno de ellos. A continuación veremos algunas de las funciones más importantes diseñadas para acceder al DOM de un documento:

- **getElementsByTagName**: recibe como argumento el nombre del elemento y devuelve un array DOM con los resultados.

```
var d = document.getElementsByTagName("div");
```

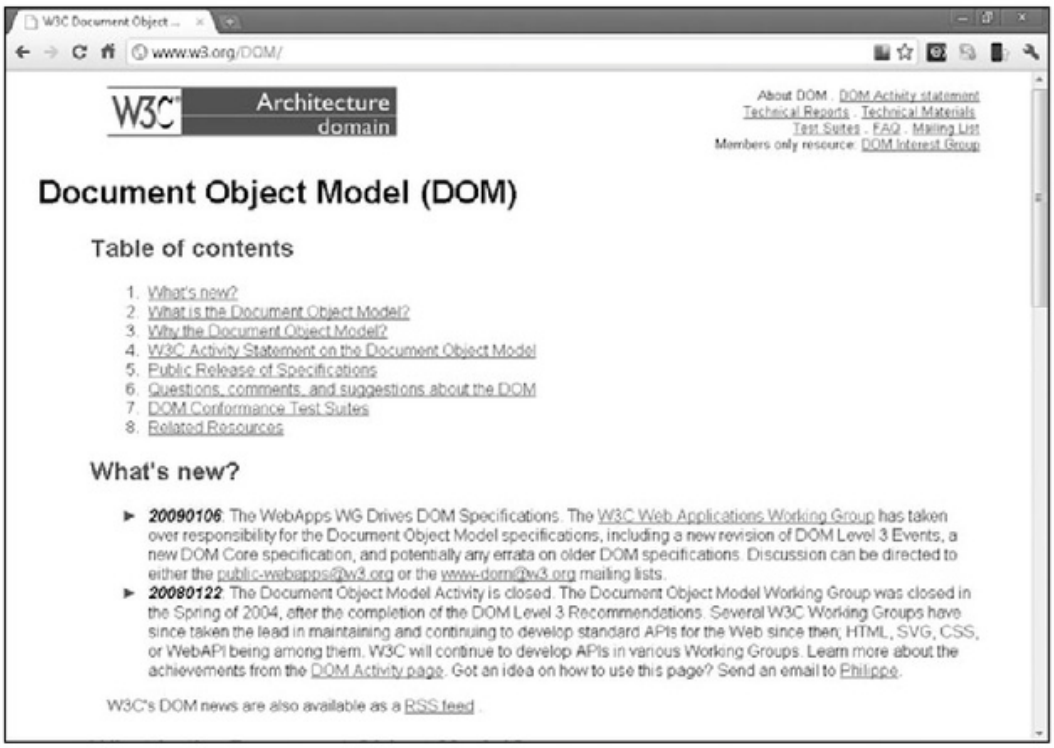


***Figura 7.** DOM es un estándar muy popular mantenido por la W3C.*

Con lo anterior se recuperan todos los elementos **DIV** que descienden directamente del elemento **HTML**, es decir de la raíz del DOM.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <div> texto 1 </div>
 <div> texto 2 </div>
 <div> texto 3 </div>

 <script type="text/javascript">
 //<![CDATA[

 var d = document.getElementsByTagName("div");
 alert(d[1].innerHTML);

 //]]>
 </script>

 </body>
</html>
```

- **getElementsByName**: recupera un array DOM con los elementos cuyo atributo **name** coincida con el argumento dado.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>
```

```
 <div name="uno"> texto 1 </div>
 <div name="dos"> texto 2 </div>
 <div name="tres"> texto 3 </div>

 <script type="text/javascript">
 //<![CDATA[

 var d = document.getElementsByName("uno");
 alert(d[0].innerHTML);

 //]]>
 </script>

 </body>
</html>
```

- **getElementById**: esta función recibe un identificador y devuelve una referencia al elemento cuyo atributo **id** coincida.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <div id="uno"> texto 1 </div>
 <div id="dos"> texto 2 </div>
 <div id="tres"> texto 3 </div>

 <script type="text/javascript">
 //<![CDATA[

 var d = document.getElementById("dos");
 alert(d.innerHTML);
```

```
 //]]>
 </script>

 </body>
</html>
```

- **createElement**: permite crear un elemento (**Element**) y recibe como argumento el nombre de la etiqueta XHTML.
- **createTextNode**: permite crear un contenido (**Text**) recibido como argumento, que luego podrá ser asignado a un elemento.
- **appendChild**: se encarga de añadir un nodo previamente creado al documento correspondiente. Se aplica sobre el nodo padre.
- **removeChild**: elimina un nodo previamente creado en el documento. Se debe especificar el nodo padre, como podemos observar en el siguiente ejemplo:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 var div = document.createElement("div");
 var contenido = document.createTextNode("texto 1");
 div.appendChild(contenido);
```

III USOS

Debemos saber que cada vez son más las librerías y las bibliotecas que son desarrolladas en JavaScript, pensadas y diseñadas para llevar a cabo la implementación de soluciones diversas para utilizar en cualquier tipo de aplicaciones. Prototype, Script.aculo.us y Mochikit son solo algunas de ellas, ya que existen muchas opciones disponibles.

```
  document.body.appendChild(div);

  var d = document.getElementsByTagName("div");
  alert(d[0].innerHTML);

  document.body.removeChild(d[0]);

  //]]>
  </script>

  </body>
</html>
```

Existen muchos otros métodos disponibles para acceder al DOM de un documento y todos los detalles se pueden revisar si se visita el sitio web oficial de la especificación: **www.w3.org/DOM**. Allí también podremos estar al tanto de las últimas novedades.

Eventos

Podemos definir a un evento como una serie de instrucciones que se desatan ante una determinada acción, generalmente por parte del usuario (presionar un botón, pasar por encima de un elemento, etcétera). Existen distintos eventos, entre los que se encuentran los siguientes. Estas acciones nos permiten mantener un control detallado acerca de los elementos contenidos en nuestra página.

- **onblur** (deseleccionar).
- **onchange** (deseleccionar un elemento modificado).
- **onclick** (clic).
- **ondblclick** (doble clic).
- **onfocus** (obtener foco).
- **onkeydown** (mantener presionada una tecla).
- **onkeypress** (presionar una tecla).
- **onkeyup** (soltar una tecla).
- **onload** (carga completa).
- **onmousedown** (mantener presionado el botón del mouse).
- **onmousemove** (mover el cursor del mouse).
- **onmouseout** (quitar el cursor del elemento).
- **onmouseover** (pasar por encima del elemento).
- **onmouseup** (soltar el botón del mouse).
- **onreset** (reinicializar las selecciones del formulario).

- **onresize** (modificar el tamaño de la ventana).
- **onselect** (seleccionar un texto).
- **onsubmit** (enviar un formulario).
- **onunload** (cerrar la ventana).

Podemos obtener más información acerca de cuáles son los elementos que disponen de cada uno de los eventos en el capítulo dedicado a XHTML.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 function mensaje() {
   alert("Texto del mensaje");
 }

 //]]>
 </script>

 <div onclick="mensaje();">Haga Clic Aqui</div>

 </body>
</html>
```

III UNA BATALLA ETERNA

La guerra de navegadores que existió entre Netscape, Microsoft y otras compañías menores, ha causado y les continúa causando hoy por hoy grandes problemas a los programadores o desarrolladores de sitios web debido a la complicación de implementación del código Javascript. Muchas veces se debe recomendar un navegador específico para visualizar el sitio.

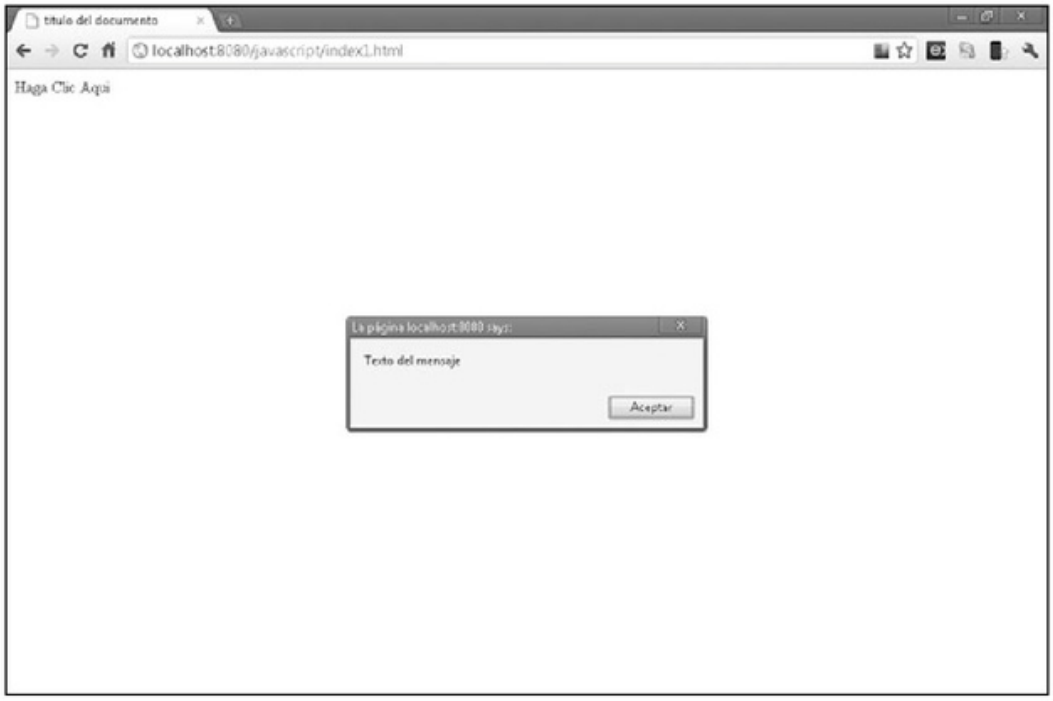


Figura 8. *La captura de eventos con JavaScript nos da la posibilidad de mantener un control más profundo acerca de las acciones realizadas por los usuarios.*

También es posible ligar eventos a elementos de la siguiente manera:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

 <script type="text/javascript">
 //<![CDATA[

 window.onload = function () { ligarEvento(); };

 function ligarEvento() {
   document.getElementById("div1").onclick = mensaje;
 }

 function mensaje() {
   alert("Texto del mensaje");
```

```
  }

  //]]>
  </script>

  <div id="div1">Haga Clic Aqui</div>

  </body>
</html>
```

Manejo y control de errores

En cuanto al control y la corrección de errores, en el capítulo dedicado a las **herramientas complementarias** para la optimización de los sitios podemos observar y conocer algunas de las alternativas más populares. Asimismo, cada navegador incluye notificadores de errores JavaScript en sus versiones estándares.

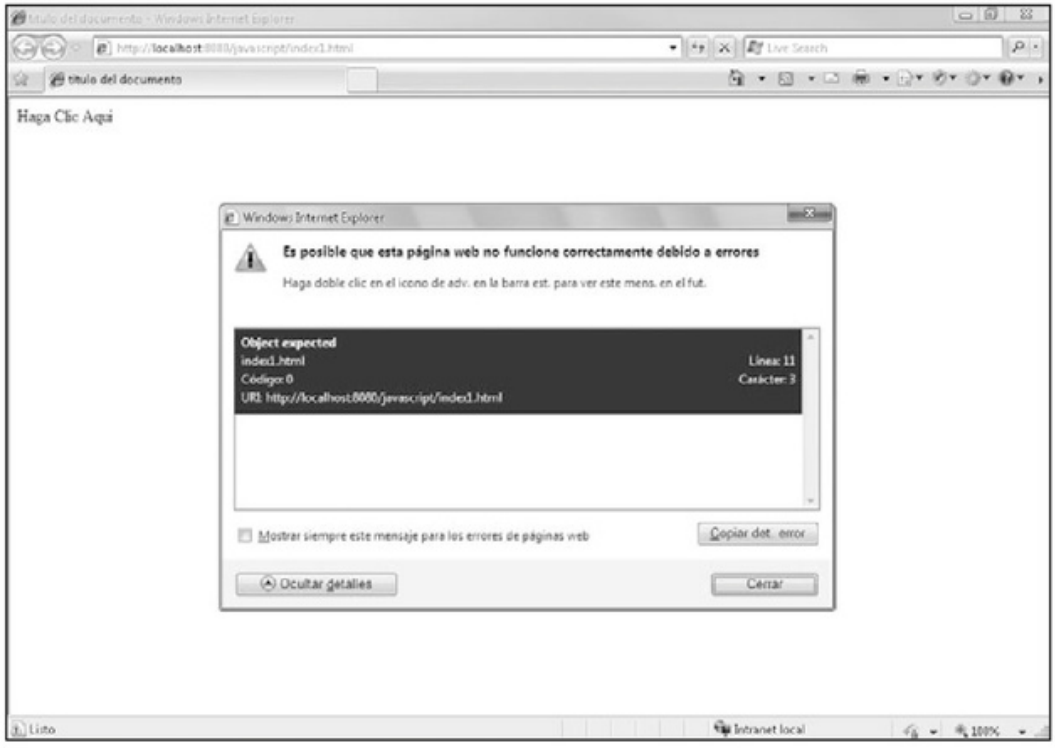


Figura 9. *Internet Explorer, al igual que la gran mayoría de los navegadores, incluye en sus versiones un visor de errores JavaScript.*

Anexo B

Referencia CSS 2

CSS es una tecnología que permite aplicar estilos sobre lenguajes de marcado como HTML y XML. En este apartado haremos un recorrido por las diferentes maneras de asociar estilos con documentos estructurados.

Introducción	**310**
Opciones disponibles	**311**
Comentarios	311
Selectores	311
Pseudo clases y elementos	320
Contadores	322
Unidades de medida	325
Colores	326
Cursores	328
Textos	329
Fuentes	331
Tablas	333
Fondos	333
Bordes	334
Dimensiones	336
Listas	336
Márgenes	337
Posicionamiento	338
Visibilidad	339

INTRODUCCIÓN

Uno de los objetivos de XHTML que resuelve CSS (*Cascading Style Sheets*) es separar por un lado el contenido de un documento y por el otro su diseño y su formato. Entre las ventajas que esto conlleva se encuentran:

- la posibilidad de, a partir del documento que contiene la información, obtener diferentes salidas aplicándole diferentes hojas de estilo.
- centralizar el diseño de los componentes comunes a las páginas de manera que su modificación sea rápida y sencilla.

A pesar de que CSS se utiliza desde hace mucho tiempo en conjunto con HTML, y les añade estilos a los elementos componentes del lenguaje, esto continúa siendo válido para XHTML, ya que mantiene la mayoría de los elementos soportados por las versiones anteriores. Estos elementos son tratados por los navegadores web de una manera predeterminada, es decir, estas aplicaciones les asignan un formato estándar. Es importante tener en cuenta que una de las tareas fundamentales de las hojas de estilo consiste en permitir especificar de manera detallada y personalizada cómo se mostrarán los componentes de un documento.

Figura 1. *CSS es uno de los múltiples estándares mantenidos por la organización* ***W3C*** *(World Wide Web Consortium).*

Cabe recordar que en el sitio oficial de la W3C se incluyen referencias actualizadas con las últimas novedades en relación con distintas tecnologías utilizadas en Internet, entre ellas CSS. Podemos obtener más información en el sitio oficial de W3C: **www.w3.org/Style/CSS/** y en **www.w3.org/MarkUp/Guide**.

OPCIONES DISPONIBLES

A continuación, una referencia por las características más importantes acerca de **CSS versión 2** (**www.w3.org/TR/REC-CSS2/cover.html**) y su modo de interacción con documentos estructurados, como XHTML por ejemplo.

Comentarios

Debemos saber que es posible incorporar comentarios dentro de las hojas de estilo. Los comentarios tienen la siguiente sintaxis:

```
/* esto es un comentario */
```

Si queremos ahorrar espacio y no demorar la carga de la página, debemos tener en cuenta que los comentarios innecesarios suelen eliminarse.

Selectores

En cada hoja de estilo podemos encontrar las llamadas reglas de visualización o de estilo, que pueden ligarse a elementos del documento XHTML para lograr la visualización buscada. Cada regla está formada por dos componentes: un **selector** y un conjunto de **declaraciones** contenidas dentro de éste.

```
nombreSelector {
  propiedad-1: valor;
  propiedad-2: valor;
  ...
  ...
  propiedad-N: valor
}
```

Las declaraciones dentro de un selector están compuestas por propiedades, cada una con valor asociado. Cada declaración está separada por un punto y coma que se ubica al final (en la última declaración esto es opcional). Si el valor que es tomado por una propiedad está compuesto por más de una palabra será necesario encerrar ese valor entre comillas:

```
body{
```

```
  font-family: "lucida grande", helvetica;
  text-align: center;
  color: #333;
  font-size: 76%;
}
```

Es necesario recordar que estos detalles son importantes ya que la generación de una regla mal escrita dará como que sea ignorada por completo. En este sentido, no les está permitido a los usuarios implementar o crear sus propias propiedades ni tampoco sus propios valores, así se minimizan las posibilidades de error.

Figura 2. *Se espera que los navegadores incorporen todas las funcionalidades de la versión 3 de CSS en breve.*

Existe la posibilidad de agrupar selectores que compartan estilos. Los selectores pueden agruparse si se ubican sus nombres separados por comas:

```
nombreSelector1, nombreSelector2, nombreSelector3 {
  propiedad-1: valor;
  propiedad-2: valor;
  ...
  ...
  propiedad-N: valor
}
```

Un estilo luego podrá ampliarse o modificarse en relación con su estado original, es decir que algo como lo siguiente es válido:

```
nombreSelector1, nombreSelector2, nombreSelector3 {
  propiedad-1: valor;
  propiedad-2: valor;
  ...
  ...
  propiedad-N: valor
}

nombreSelector2 {
  propiedad-2: otroValor;
}
```

La última propiedad definida tiene preponderancia sobre las anteriores.

Selectores de elementos

Cualquier elemento que se encuentre disponible dentro de la especificación XHTML que pueda definirse en el elemento denominado **body** (incluyéndolo) puede ser vinculado a un estilo. Para hacerlo solo será necesario que realicemos la tarea de incluir el nombre del elemento seguido de las propiedades que queramos definir, vemos un ejemplo a continuación:

```
h1{
  color: #003d4c;
  margin: 5px;
  font-size: 180%;
}

td {
  color: #F00;
}
```

Debemos recordar que como las reglas de estilo más específicas se aplican con precedencia a las reglas más genéricas, de esta forma si un elemento denominado B se encuentra dentro de un elemento conocido como A, entonces B heredará el formato de A, a menos que B redefina todas las propiedades incluidas en A, reemplazando los valores correspondientes con otros.

Selector ID

El atributo **id** nos permite ligar elementos de un documento XHTML con un determinado selector en la hoja de estilos. Éste se utiliza para representar de manera única (no debería haber más de un mismo valor **id** en un documento) a cada elemento disponible en un archivo XHTML y luego es posible tomar ese valor e incluirlo dentro de la hoja de estilos:

```
<div id="titulo">Texto del título</div>
<div id="subtitulo">Texto del subtítulo</div>
```

Para declarar una regla en la hoja de estilos que hace referencia a un atributo **id** se utiliza la sintaxis:

```
#titulo {
  font-size: 10pt;
  font-family: verdana, arial, helvetica;
  color: #000000;
  background-color: #ffffff;
  margin: 3 0 0 0px;
  text-align: center;
}
```

Por motivos de compatibilidad y de definición o convenio en muchos lenguajes, el nombre de un atributo **id** no debe comenzar con números. Los nombres dados a los atributos tipo **id** son sensibles a mayúsculas y minúsculas.

Selector Class

El atributo denominado **class** suele utilizarse en casos en los que más de un elemento usa el mismo selector. A diferencia de **id**, más de un elemento puede contar con el mismo valor para **class**, vemos un ejemplo a continuación:

III REGLAS EN CSS2

Es importante que como desarrolladores tengamos siempre presente que una regla correspondiente CSS consta de dos partes principales: un selector y una declaración. La declaración tiene dos partes: una propiedad y un valor. Combinado con otras hojas de estilo, esto determinará la presentación final del documento.

```
<div class="titulo">Texto del título</div>
<div class="subtitulo">Texto del subtítulo 1</div>
<div class="subtitulo">Texto del subtítulo 2</div>
```

Para declarar una regla en la hoja de estilos que hace referencia a un atributo **class** se utiliza la sintaxis:

```
div.titulo {
  font-size: 10pt;
  font-family: verdana, arial, helvetica;
  color: #000000;
  background-color: #ffffff;
  margin: 3 0 0 0px;
  text-align: center;
}
```

o también:

```
.titulo {
  font-size: 10pt;
  font-family: verdana, arial, helvetica;
  color: #000000;
  background-color: #ffffff;
  margin: 3 0 0 0px;
  text-align: center;
}
```

En la primera se aplicará el formato a los elementos **div** que pertenezcan a la clase **titulo**;en la segunda a cualquier elemento que pertenezca a la clase **titulo**.

III RECOMENDACIONES

Para una mayor flexibilidad, siempre se les recomienda a los desarrolladores que especifiquen hojas de estilo externas. Éstas pueden cambiarse sin necesidad de editar el documento HTML o el que fuere, y a su vez, éstas pueden compartirse entre varios documentos. Para vincular una hoja de estilo externa comúnmente se utiliza el elemento **LINK**.

Figura 3. *Los elementos del estándar XHTML se pueden ligar a estilos con CSS.*

En general, cada elemento podrá contar solo con un atributo **class**. Solo algunos navegadores soportan la asignación de más de una clase a un elemento:

```
<p class="clase1 clase2 clase3"> ... </p>
```

Debemos saber que por motivos de compatibilidad, el nombre de un atributo **class** nunca debe comenzar con números. Los nombres dados a los atributos tipo **class** son sensibles a mayúsculas y minúsculas.

Selectores de atributos

Es posible aplicar estilos a un elemento que tenga un determinado atributo e incluso especificar que el atributo contenga algún valor específico:

```
elemento[atributo] { estilo }
elemento[atributo="valor"] { estilo }
```

Por ejemplo, la siguiente regla solo se aplicaría a elementos **img** cuyo atributo **title** fuera igual a **thumb**, vemos su sintaxis a continuación:

```
img[title='thumb'] {
  cursor: pointer;
}
```

Si quisiéramos aplicar la regla para las imágenes cuyo título tuviera la palabra **min**:

```
img[title~='min'] {
  cursor: pointer;
}
```

Y para todas las imágenes que simplemente contarán con el atributo **title**, sin importar su valor, analizamos el código a continuación:

```
img[title] {
  cursor: pointer;
}
```

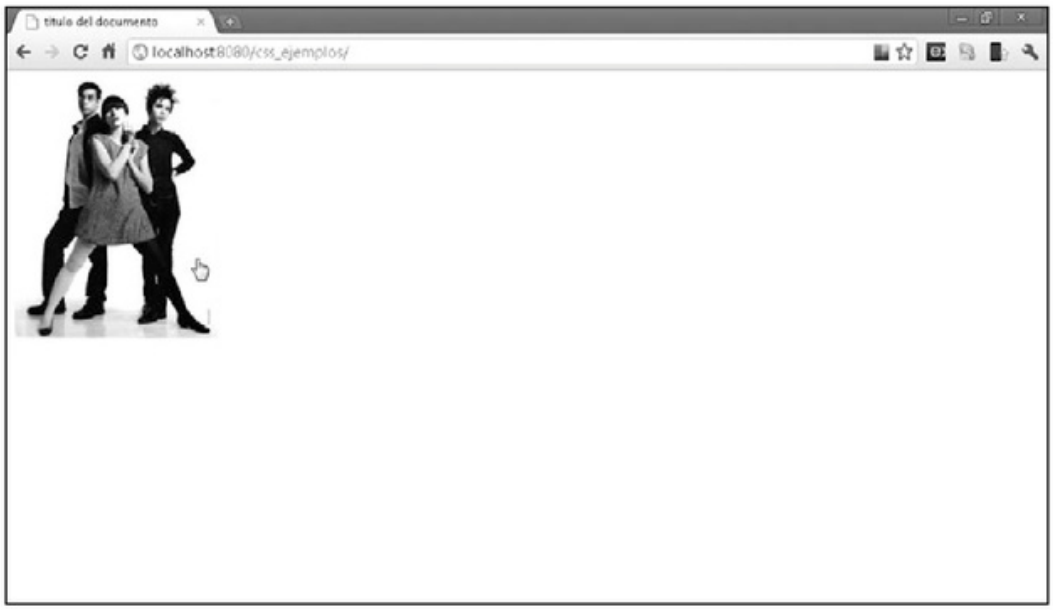


***Figura 4.** Los selectores de atributos permiten personalizar los estilos de los elementos según sus propiedades.*

En la actualidad, no todos los navegadores aceptan selectores de atributos.

Selector universal

Con el selector universal es posible referenciar a todos los elementos del documento. El signo utilizado para esta representación es el asterisco:

```
* {
  color: red;
}
```

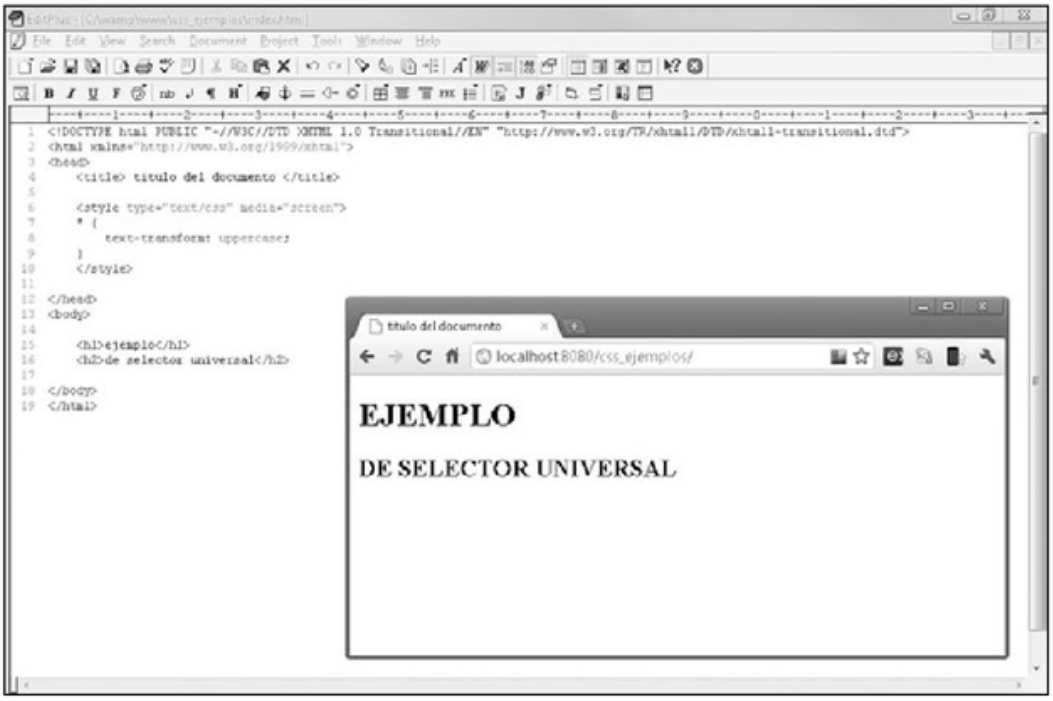


Figura 5. *CSS provee distintos tipos de selectores que pueden interactuar entre sí dentro de una misma hoja de estilos.*

El selector universal puede utilizarse junto a otros tipos de selectores.

Selectores contextuales

Hay diversos selectores contextuales, entre ellos los llamados selectores descendientes. En el siguiente ejemplo, el estilo se aplica a los enlaces dentro de párrafos:

```
p a {
  text-decoration : none;
  color : #3333cc;
}
```

Así definiremos un estilo para enlaces en párrafos y de la misma forma crearemos otro para los enlaces en las celdas correspondientes:

```
p a {
  text-decoration : none;
  color : #3333cc;
}
td a {
  text-decoration : none;
  color : #fff;
```

```
  cursor: pointer;
}
```

No hay límite para el número de niveles y cada uno se separa con un espacio en blanco. En un selector descendiente pueden combinarse selectores identificados con atributos **id**, **class** o nombres de elementos. Otro tipo de selector contextual es el de los **selectores hijos**: mientras que con los selectores descendientes es posible aplicar un determinado estilo a elementos que desciendan de otros (hijos, nietos, etcétera), con los selectores hijos es posible aplicar estilos solo a elementos que sean descendientes directos (hijos) de otros. La relación se define con el símbolo **mayor que**:

```
b > c {
  text-decoration: underline;
}
```

En los selectores hijos pueden combinarse selectores identificados con atributos **id**, **class** o nombres de elementos. Por último, dentro de los selectores contextuales se encuentran los **selectores adyacentes**, que permiten aplicar estilos a elementos que son próximos a otros. Se consideran próximos cuando no hay elementos intermedios. Para representar adyacencia se utiliza el símbolo suma (+).

```
elemento1 + elemento2 { background-color: #FF0000; }
```

En el ejemplo que acabamos de ver, el estilo es aplicado al denominado **elemento2** siempre y cuando fuera adyacente (es decie, que apareciera después) al conocido como **elemento1**. Debemos tener en cuenta que en los selectores adyacentes pueden combinarse selectores identificados mediante los atributos **id**, **class** o también con nombres de elementos, entre otros ejemplos.

Estilos en línea

También existe la posibilidad de formatear elementos dentro del documento con el atributo **style**, un método conocido como **inline**:

```
<p style="text-decoration: underline;"> ... </p>
```

El atributo **style** sobrescribe estilos definidos en un archivo CSS en caso de que exista un conflicto, como que se intente darle formato a un mismo elemento desde

ambas opciones. Puede ser utilizado en cualquier elemento contenido en **body**, incluyéndolo (no están habilitados: **basefont**, **param** y **script**).

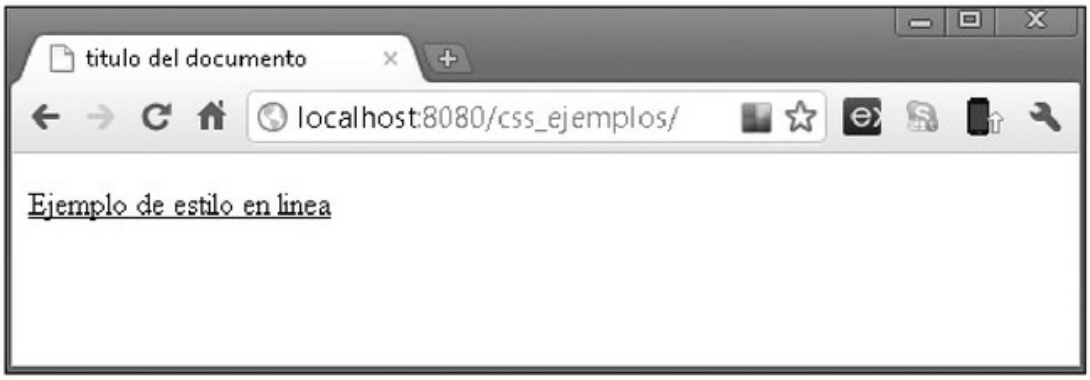


***Figura 6.** Los estilos en línea están siendo desplazados y no cumplen con algunos de los principios básicos de XML.*

Para habilitar esta opción en los navegadores, debemos incluir la cabecera correspondiente dentro de **head**, aunque en la mayoría de los casos no es necesario:

```
<META HTTP-EQUIV="Content-Style-Type" CONTENT="text/css">
```

Debemos tener en cuenta que se trata de un método en desuso y por lo tanto poco recomendable ya que no aprovecha en absoluto las ventajas de trabajar con las hojas de estilo externas que provee CSS.

Pseudo clases y elementos

Estas opciones permiten hacer referencia desde las hojas de estilo a determinados elementos según su posición con respecto a otros o en base a su contenido o estado actual. Los pseudo elementos disponibles en CSS versión 2 son:

- **:first-letter**, se trata de una opción que permite asignarle estilos al primer carácter contenido en un determinado elemento.

```
p:first-letter {
  font-size : 10pt;
  font-family : verdana,arial,helvetica;
  font-weight : bold;
}
```

Si tomamos el estilo anterior y lo aplicamos a un párrafo, la salida sería:

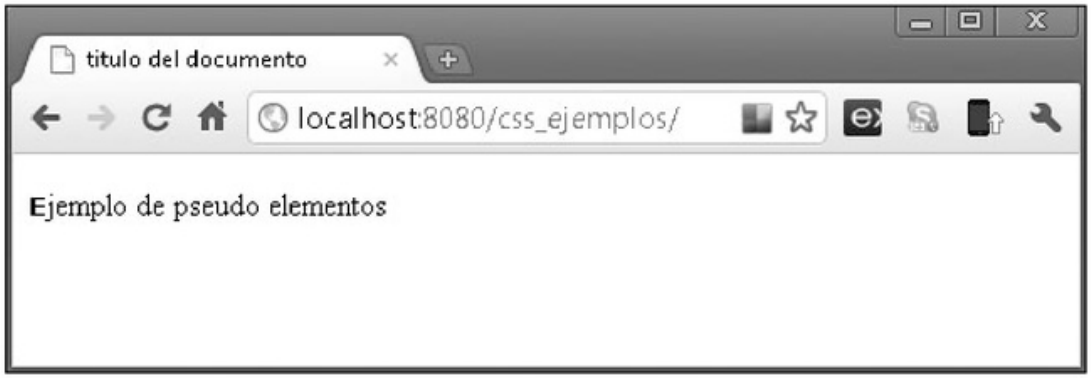


***Figura 7.** El uso de pseudo elementos puede ser de mucha utilidad y nos permite un control más profundo sobre diversos aspectos del documento.*

- **:first-line**, permite asignar estilos a la primera línea contenida en un elemento.

```
p:first-line {
  color: red;
}
```

- **:before**, es un pseudo elemento que agrega el contenido ubicado dentro de las llaves antes del selector. Por ejemplo, en el caso que veremos a continuación se imprime la letra **J** antes de cada párrafo:

```
p:before {
  content: "J";
}
```

- **:after**, que es similar a **:before** solo que imprime el contenido después de la aparición del elemento.

```
p:after {
  content: "...";
}
```

En cuanto a las pseudo clases disponibles en CSS podemos nombrar:

- **:first-child**, para representar al primer hijo del elemento.

```
p:first-child { background-color: #00FF00 }
```

- **:link**, enlace que no ha sido visitado por el usuario.

```
a:link              { color: yellow; }
```

- **:visited**, enlace que ha sido visitado por el usuario.

```
a:visited   { color: red;  }
```

- **:hover**, enlace que tiene sobre él al puntero del mouse.

```
a:hover             { color: green;       }
```

- **:active**, enlace seleccionado con el mouse.

```
a:active    { color: blue;        }
```

- **:focus**, enlace que tiene el foco.
- **:lang**, que identifica el idioma utilizado. Por ejemplo **lang(en)** equivale al idioma Inglés, **lang(es)** corresponde al español, etcétera.

Contadores

Con hojas de estilo CSS es posible incorporar contadores asociados con determinados elementos. En este ejemplo contamos el siguiente documento XHTML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
 </head>
 <body>
    <h1>Primera seccion</h1>
```

```
      <p>Primer parrafo</p>
      <p>Segundo parrafo</p>
      <p>Tercer parrafo</p>

      <h1>Segunda seccion</h1>
      <p>Primer parrafo</p>
      <p>Segundo parrafo</p>

      <h1>Tercera seccion</h1>
      <p>Primer parrafo</p>
</body>
</html>
```

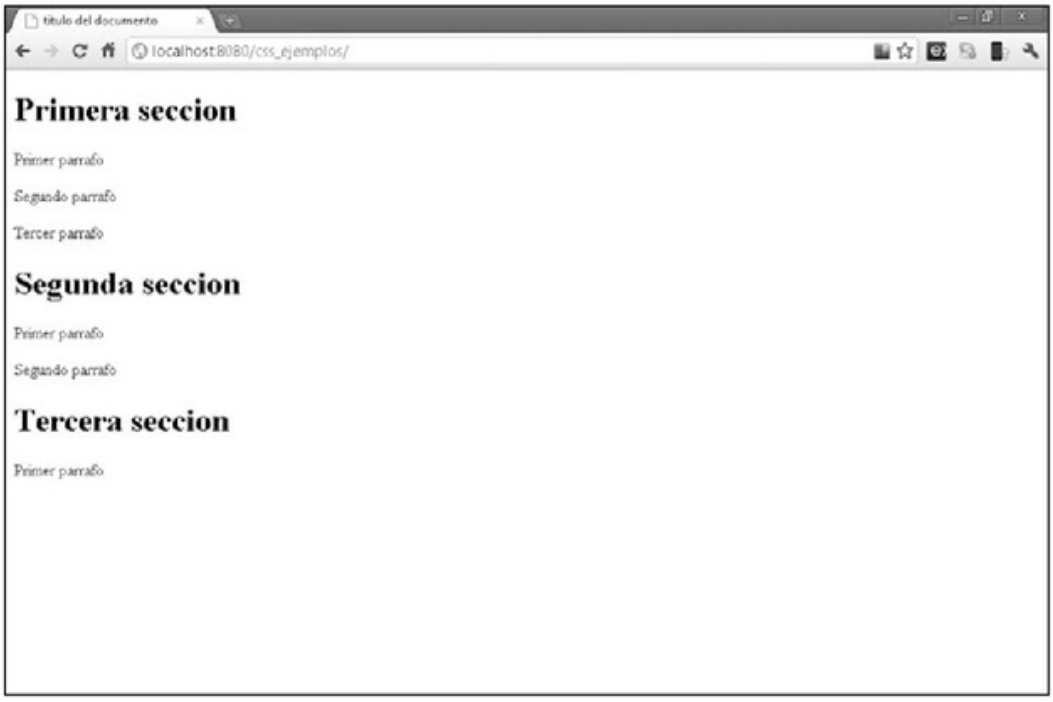


Figura 8. *Los contadores no son soportados actualmente por todos los navegadores, aunque se espera que esto cambie en un futuro no tan lejano.*

La idea es que, con los contadores CSS, se numeren los párrafos dentro de cada sección. Veamos la hoja de estilos:

```
H1 {
  counter-reset: contador 0;
}

p {
```

```
  counter-increment: contador 1;
}

p:before {
  content: counter(contador) ". ";
}
```

Figura 9. *La utilización de contadores evita tareas repetitivas como visualizar la posición de un elemento en relación con el entorno.*

Debemos saber que ante cada aparición del elemento denominado **H1** el contador **contador** toma el valor 0 con **counter-reset**:

```
H1 {
  counter-reset: contador 0;
}
```

Cuando se encuentra un elemento **p**, el contador incrementa su valor en una unidad:

```
p {
  counter-increment: contador 1;
}
```

Con la pseudo clase denominada **:before**, el contenido de cada párrafo será precedido por un punto y un espacio, como lo vemos a continuación:

```
p:before {
  content: counter(contador) ". ";
}
```

La función denominada **counter** devuelve el valor actual del contador y puede recibir como segundo argumento una constante para modificar el formato de salida. Algunas posibilidades son las que se muestran a continuación:

CONSTANTE	DESCRIPCIÓN
decimal	Valor por defecto. 1, 2, 3, etcétera
decimal-leading-zero	01, 02, 03, etcétera
lower-roman	Números romanos en minúscula
upper-roman	Números romanos en mayúscula
lower-alpha	a, b, c, etcétera
upper-alpha	A, B, C, etcétera
lower-latin	Latín en minúscula
upper-latin	Latín en mayúscula
lower-greek	Griego en minúscula
upper-greek	Griego en mayúscula

***Tabla 1.** Constantes disponibles para counter.*

Puede incluirse más de un contador en una misma hoja de estilo. Los contadores no son soportados por la totalidad de los navegadores.

Unidades de medida

Algunas propiedades necesitan que sus valores sean representados con un número seguido de una unidad de medida. Hay dos tipos de unidades: **absolutas** y **relativas**.

III COMBINACIÓN DE ESTÁNDARES

Junto con el desarrollo de XHTML se incorporaron los estándares CSS y las consideraciones de Accesibilidad WAI AAA y Sección 508, como así también temas relativos a la Usabilidad y Arquitectura de Información. Siempre que se desarrolla una tecnología que previamente tuvo afinidad con otra, se busca combinar ambas y avanzar a la par en sus mejoras.

UNIDAD	TIPO	DESCRIPCIÓN	EJEMPLO
in	Absoluta	Pulgada. Una pulgada equivale a 2,54 cm.	10in
cm	Absoluta	Centímetro.	10cm
mm	Absoluta	Milímetro.	10mm
pt	Absoluta	Punto. Un punto equivale a 1/72 de pulgada.	10pt
pc	Absoluta	Pica. Una pica equivale a 12 puntos.	10pc
em	Relativa	Altura de la fuente que se utiliza.	10em
ex	Relativa	Altura de la letra x minúscula de la fuente utilizada.	10ex
px	Relativa	Pixel. La cantidad de pixeles dependerá de las características en la máquina cliente (por ejemplo 800 pixeles horizontales por 600 verticales ó 1024 pixeles horizontales por 768 verticales).	10px
%	Relativa	Porcentaje.	10%

***Tabla 2.** Unidades de medida disponibles en CSS.*

Es interesante saber que por motivos de compatibilidad no debe haber espacios entre el número y la unidad de medida utilizada.

Colores

Dentro de CSS existen distintas maneras de definir el color de un elemento, entre las que podemos citar las siguientes:

- mediante el valor **RGB** (**Red**, **Green**, **Blue**) se disponen tres números separados por comas que pueden representar porcentajes:

```
rgb(100%,100%,100%) /* color blanco */
```

```
rgb(0%,0%,0%) /* color blanco */
```

o valores decimales entre 0 y 255:

```
rgb(255, 0, 32)
```

- mediante valores en hexadecimal, por ejemplo **#3333cc**. Una opción es utilizar solo tres dígitos siempre y cuando los pares que indican el rojo, el verde y el azul sean iguales (según el ejemplo anterior el equivalente sería **#33c**).
- hay determinados colores que pueden representarse mediante constantes. Entre los 16 disponibles se encuentran los siguientes:

Aqua, **Black**, **Blue**, **Fuchsia**, **Gray**, **Green**, **Lime**, **Maroon**, **Navy**, **Olive**, **Purple**, **Red**, **Silver**, **Teal**, **White** y **Yellow**.

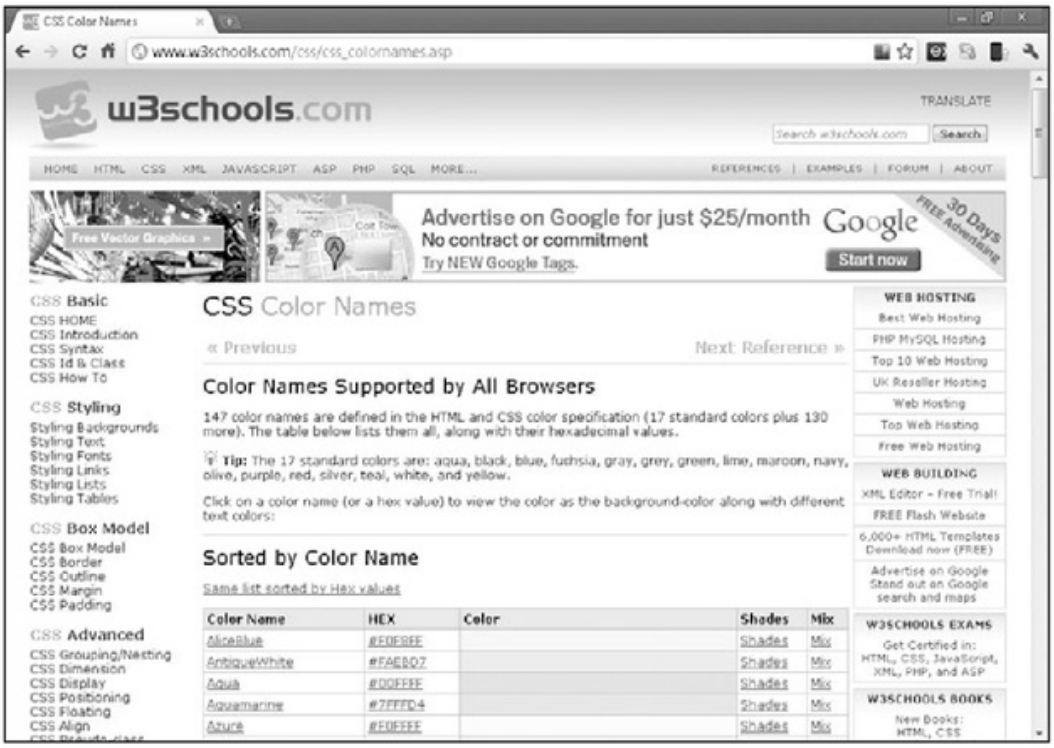


Figura 10. *CSS provee una serie de constantes para definir colores de manera rápida e intuitiva.*

- otras constantes ahora relacionadas con elementos del sistema de la máquina cliente son las que listamos a continuación:
  - **ActiveBorder** (borde de una ventana activa).
  - **Background** (color de fondo del escritorio).
  - **ButtonHighlight** (botón seleccionado).
  - **ButtonShadow** (sombra de los botones).
  - **ButtonText** (color del texto de los botones).
  - **Highlight** (elementos seleccionados).
  - **HighlightText** (texto seleccionado).
  - **InactiveBorder** (borde inactivo)
  - **InactiveCaption** (color de un título inactivo).
  - **InactiveCaptionText** (color del texto de un título inactivo).
  - **InfoBackground** (color de fondo de la ayuda).
  - **InfoText** (color del texto de la ayuda).
  - **Menu** (color del menú).
  - **MenuText** (color del texto del menú).
  - **Scrollbar** (color de la barra de scroll).
  - **ThreeDDarkShadow** (sombra oscura en 3D).
  - **ThreeDFace** (cara principal en 3D).

- **ThreeDHighlight** (cara de iluminación en 3D).
- **ThreeDLightShadow** (sombra clara 3D).
- **ThreeDShadow** (sombra 3D).
- **Window** (color de la ventana).
- **WindowFrame** (color del frame de la ventana).
- **WindowText** (color del texto de la ventana).

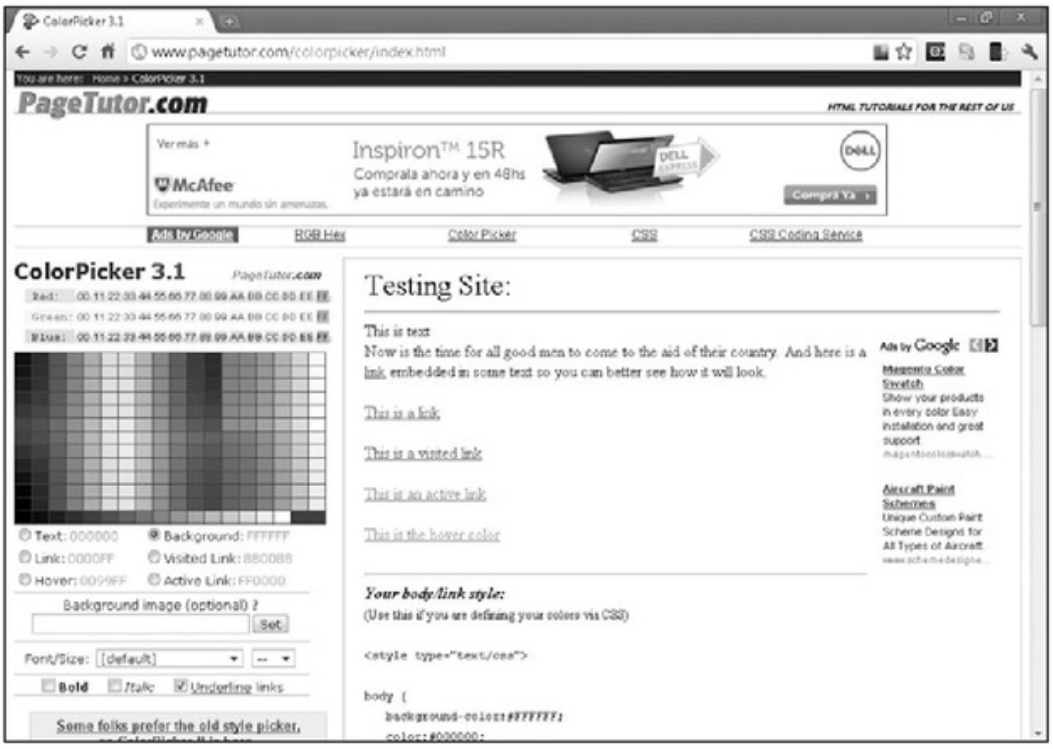


***Figura 11.** Con CSS es posible acceder a los colores predefinidos en el sistema operativo de la máquina cliente.*

Cursores

Con CSS es posible asociar una determinada apariencia a un puntero según pase sobre un elemento u otro. Las opciones prefijadas son: **auto**, **crosshair**, **default**, **pointer** (o **hand**), **move**, **e-resize**, **ne-resize**, **nw-resize**, **n-resize**, **se-resize**, **sw-resize**, **w-resize**, **s-resize**, **text**, **wait**, **progress** y **help**.

III CONSIDERACIONES DE USO

Como es de esperarse, en la mayoría de tecnologías web utilizadas hoy en día y en la variedad de navegadores web que se disponen en el mercado, siempre hay diferencias en la utilización de cursores entre un navegador y otro. Hasta su versión 1.5 Firefox no admitía los cursores animados dentro de cualquier página web que visualizaba.

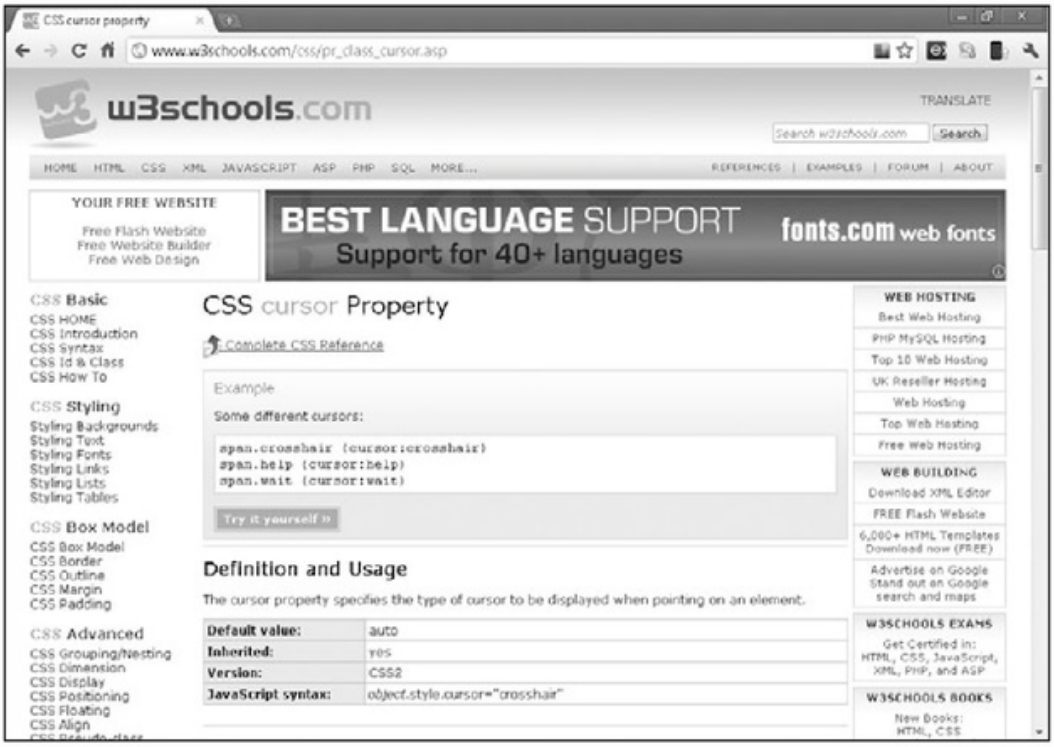


Figura 12. *CSS provee cursores predefinidos y además la posibilidad de que el diseñador utilice los propios.*

En el ejemplo a continuación, veremos la sintaxis que se utiliza para definir un cursor determinado para cada elemento:

```
elemento { cursor: help }
```

El soporte para estas opciones dependerá del navegador que se utilice. Desde CSS versión 2 es posible definir cursores propios si se indica la ruta al gráfico a utilizar:

```
elemento { cursor: url("nombreCursor") }
```

Normalmente se trata de un archivo con extensión **CUR**, **ANI** o **CSR**.

Textos

El contenido textual de los elementos puede tratarse mediante diversas propiedades especialmente dispuestas, por ejemplo:

- **direction** puede tomar dos valores: **rtl** (*right to left*, derecha a izquierda), que imprime el texto sin dejar margen derecho o **ltr** (*left to right*, izquierda a derecha), que imprime el texto sin dejar margen izquierdo. Ésta última es la opción por defecto.

- **letter-spacing** nos permite definir el espaciado entre caracteres. Admite los valores **normal** o unidades de longitud, que pueden ser negativas.
- **text-align** se utiliza para alinear textos. La alineación se realiza en referencia a un contenedor y los valores que puede tomar son: **left**, que alinea a la izquierda, **right**, que alinea a la derecha, **justify**, que es justificado y **center**, que alinea al centro.
- **text-decoration** decora un texto y puede tomar los siguientes valores: **underline** (línea por debajo del texto), **overline** (línea por encima del texto), **line through** (línea por medio del texto), **blink** (parpadeo) y **none** (ninguno).
- **text-indent** se utiliza para tabular la primera línea de un texto ubicado dentro de un contenedor. Admite valores representados con cualquier unidad de medida.
- **text-shadow** especifica el color y el tamaño de la sombra de un texto. Recibe tres argumentos y el formato es:

```
text-shadow: color distancia-h-texto, distancia-v-texto;
```

donde **distancia-h-texto** es la distancia horizontal de la sombra con respecto al texto y **distancia-v-texto** es la distancia vertical de la sombra con respecto al texto.

- **text-transform** se encarga de admitir los siguientes valores: **uppercase** (permite realizar la transformación en mayúsculas), **lowercase** (transforma en minúscula), **capitalize** (convierte en mayúscula solo el primer carácter de cada palabra) y **none** (se trata del valor seleccionado por defecto).

Figura 13. *CSS provee funcionalidades avanzadas para el tratamiento de texto desde hojas de estilos.*

- **white-space** se utiliza para mantener un cierto control sobre los espacios en blanco. Admite como valores **normal**, que traduce más de un espacio consecutivo a solo uno y **pre**, que no traduce más de un espacio consecutivo a solo uno, toma los que están. Éste actúa como la etiqueta **pre** de HTML. También admite otros valores como **nowrap**, que no incluye saltos de línea a menos que se utilice una etiqueta **br**.
- **word-spacing** permite definir el espaciado entre las palabras. Admite los valores **normal** o las unidades de longitud. Los valores pueden ser negativos.

Fuentes

Las tipografías en CSS cumplen un papel importante y entre las propiedades más destacadas se incluyen las siguientes:

- **font-family** identifica el nombre de la fuente a aplicar. Es posible indicar múltiples fuentes si se las separa por comas, y se optará por la primera que esté disponible. Si ninguna de las fuentes incluidas está disponible en la máquina cliente, se tomará una fuente genérica (**serif**, **sans-serif**, **cursive**, **fantasy** o **monospace**), que puede especificarse al final de la lista. Además, existe la posibilidad de definir la ubicación de la fuente, por si no está disponible en la máquina cliente, con la regla **@font-face**:

```
@font-face {
  font-family: nombreDeLaFuente;
  src: url(rutaAlaFuente);
}
```

- **font-size** se encarga de definir el tamaño de la fuente que utilizaremos. En este sentido puede asignársele un valor específico o también alguna de las constantes que mencionamos a continuación:
  - **xx-small** (extra pequeña).
  - **x-small** (muy pequeña, alias de **smaller**).
  - **small** (pequeña).

III CSS1 Y LAS FUENTES

En la versión 1 de CSS, se entendía que por defecto, todas las fuentes estaban presentes en el sistema cliente y eran identificadas solamente por su nombre. Se permitía especificar fuentes alternativas por medio de propiedades, pero más allá de eso, las aplicaciones del usuario no tenían ninguna manera de proponerle otra fuente al usuario más allá de las predeterminadas.

- **medium** (media, valor por defecto).
- **large** (grande).
- **x-large** (muy grande, alias de **larger**).
- **xx-large** (extra grande).

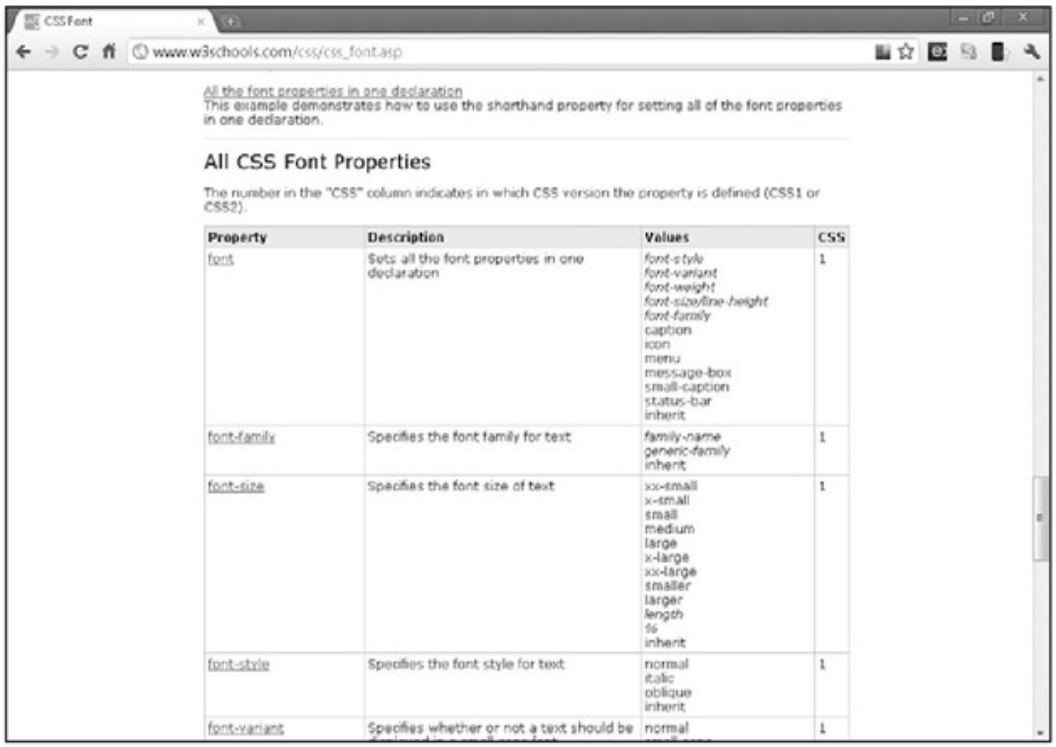


Figura 14. *Con CSS podemos especificar el tamaño de visualización de una fuente mediante distintas alternativas.*

Con **smaller** y **larger** es posible aplicar un tamaño de fuente relativa o predeterminada. Estos valores hacen que la fuente sea más pequeña o más grande comparada con el valor que tenía inicialmente.

- **font-size-adjust** sirve para ajustar el tamaño genérico de la fuente y estabiliza la visión del tamaño.
- **font-stretch** define la expansión o la compresión de la fuente. Los valores posibles son: **ultra-condensed**, **extra-condensed**, **condensed**, **semi-condensed**, **normal**, **semi-expanded**, **expanded**, **extra-expanded** y **ultra-expanded**.
- **font-style** define el estilo de la fuente. Admite tres valores: **normal** (valor por defecto), **italic** (cursiva) y **oblique** (oblicua, que no es soportada por la mayoría de los navegadores).
- **font-variant** admite dos valores: **normal** (valor por defecto) y **small-caps** (versales. Letras mayúsculas de un tamaño más pequeño que el normal).
- **font-weight** admite dos valores: **normal** (valor por defecto) y **bold** (negrita). Otra opción es asignarle un valor numérico: 100, 200, 300, 400 (**normal**), 500, 600, 700 (**bold**), 800 ó 900.

Tablas

Las tablas mantienen sus propias alternativas en cuanto a las propiedades, entre las que podemos citar las siguientes:

- **border-collapse** permite que los bordes de las celdas contiguas puedan mostrarse como uno solo (**collapse**) o como dos bordes contiguos (**separate**).
- **border-spacing** determina el espacio entre celdas contiguas en la tabla. Si pasamos una sola unidad de medida, todos los lados de cada celda la usarán para su separación. Si pasamos dos unidades de medida, la primera se usará para la separación horizontal y la segunda para la separación vertical.
- **empty-cells** dibuja los bordes y fondos de las celdas vacías. Puede tomar el valor **show** (para mostrarlos) o **hide** (para ocultarlos). No funciona con todos los navegadores.
- **table-layout**. Si toma el valor **fixed**, la tabla tomará una dimensión fija sin importar cuán extenso sea el contenido de sus celdas (si éste excede el espacio, el contenido se truncará). Para que sí tenga en cuenta la longitud se utiliza el valor **auto**.

Fondos

Las propiedades más importantes en cuanto al tipo de fondo y su posicionamiento son las que mencionamos a continuación:

- **background-attachment** permite especificar si una imagen de fondo permanecerá fija (**fixed**) o acompañará al contenido si deslizamos la página con la barra de scroll (**scroll**).
- **background-color** nos da la posibilidad de especificar el color de fondo de un elemento tomando valores mediante cualquiera de las opciones vistas. Además, admite el valor **transparent** (transparente).
- **background-image** posibilita asignar una imagen como fondo de un elemento. Los valores que admite son del tipo:

```
url(archivoImagen);
```

III CARACTERÍSTICAS DE COLORES

CSS2 nos permite especificar colores de manera que se integren con el entorno gráfico del usuario. Las reglas de estilo permiten ventajas como crear páginas que coinciden con un determinado aspecto y gusto del usuario, y crear páginas que pueden ser más accesibles en tanto las preferencias del usuario que pueden estar relacionadas con una discapacidad.

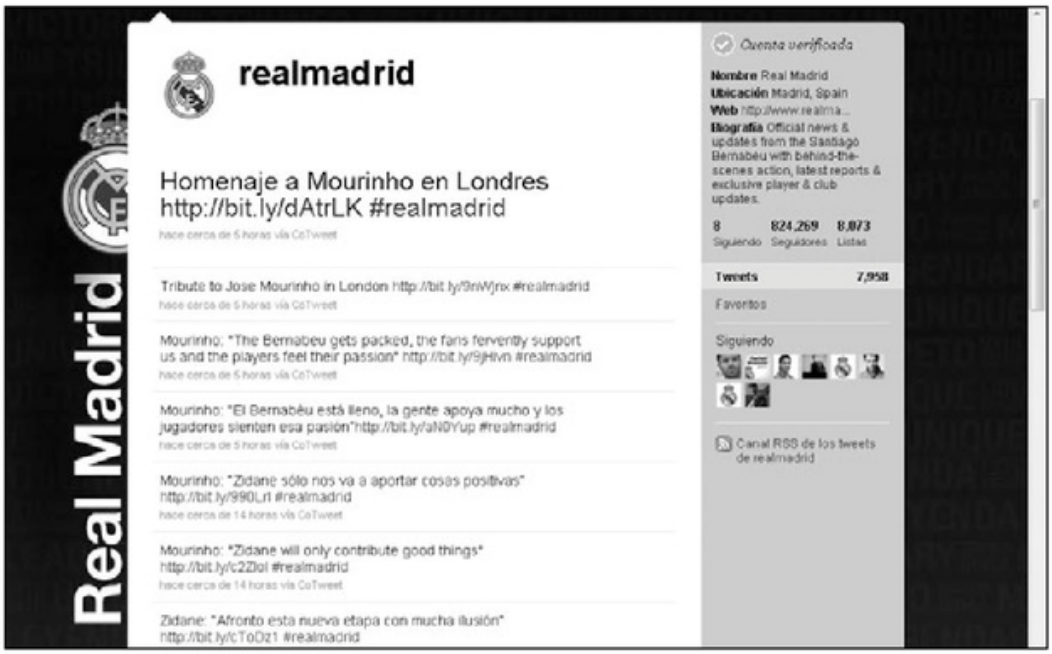


Figura 15. *Las imágenes de fondo son algo habitual en los sitios web y con CSS es posible determinar sus características.*

- **background-position** controla la ubicación (alineación) inicial de la imagen de fondo. La alineación es horizontal (**left**, **center**, **right**) y vertical (**top**, **center**, **bottom**). También pueden usarse porcentajes y longitudes para asignar la posición de la imagen de fondo. Los porcentajes son relativos al tamaño del elemento. Se pueden dar dos argumentos: primero la ubicación horizontal y luego la vertical. Por ejemplo:

```
background-position: left center;
```

```
background-position: 4cm 5cm;
```

```
background-position: 35% 60%;
```

- **background-repeat**. Si la imagen ubicada como fondo de un elemento no alcanza a ocuparlo completamente, mediante esta propiedad podemos definir que se repita hasta cubrir todo el espacio (**repeat**), que solo se repita horizontalmente (**repeat-x**), que solo se repita verticalmente (**repeat-y**) o que no se repita en absoluto (**no-repeat**).

Bordes

En cuanto a la definición de los bordes de los elementos, con su especificación CSS nos provee las siguientes propiedades:

- **border-bottom** trabaja sobre el borde inferior y es similar a **border-style**.
- **border-bottom-color** permite definir el color del borde inferior.
- **border-bottom-style** representa el estilo del borde inferior. Ver **border-style** para más información.
- **border-bottom-width** posee propiedades similares a **border-width** y se encarga de rrepresenta el ancho del borde inferior.
- **border-color** permite especificar el color del borde.
- **border-left** es similar a **border-style** y trabaja sobre el borde izquierdo.
- **border-left-color** define el color del borde izquierdo.
- **border-left-style** nos permite representar el estilo del borde izquierdo. Podemos ver la propiedad **border-style** para más información.
- **border-left-width** es similar a **border-width** y representa el ancho del borde izquierdo.
- **border-right** es similar a **border-style** y trabaja sobre el borde derecho.
- **border-right-color** representa el color del borde derecho.
- **border-right-style** es una propiedad que define el estilo del borde derecho. Para más información podemos ver **border-style**.
- **border-right-width** es similar a **border-width** y representa el ancho del borde derecho.
- **border-style** define el tipo de borde. Los valores posibles son:

solid (sólido), **dashed** (lineado), **dotted** (punteado), **double** (dos líneas), **inset** (efecto 3D), **groove** (efecto 3D), **ridged** (efecto 3D) y **none** (sin borde, valor por defecto).

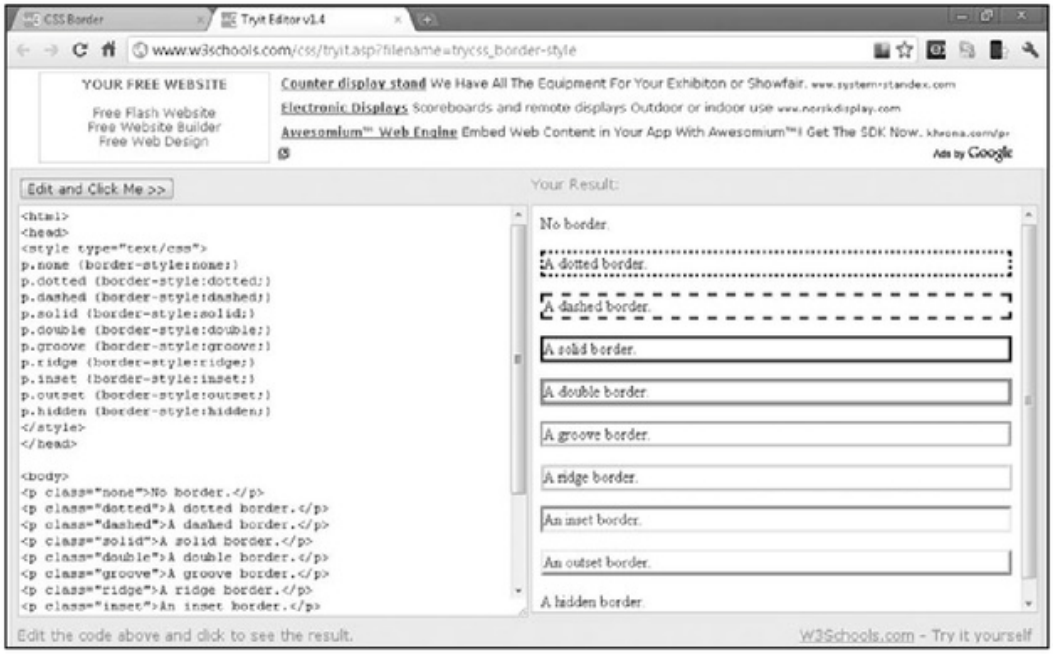


Figura 16. *Los distintos tipos de bordes permiten controlar de manera directa las características del diseño de cada elemento con CSS.*

- **border-top** es similar a **border-style** y trabaja sobre el borde superior.
- **border-top-color** representa el color del borde superior.

- **border-top-style** define el estilo del borde superior. Ver **border-style** para más información.
- **border-top-width** es similar a **border-width** y representa el ancho del borde superior.
- **border-width** define el ancho del borde. Admite como valores **thin** (delgado), **medium** (medio), **thick** (mayor) o cualquier valor absoluto.

Dimensiones

Dentro de las propiedades disponibles para modificar las dimensiones de un elemento a partir de estilos CSS podemos citar:

- **height** representa la altura del elemento.
- **line-height** es la distancia entre líneas. Admite valores representados con cualquier unidad de medida. No admite valores negativos.
- **max-height** define la altura máxima de un elemento.
- **max-width** define el ancho máximo de un elemento.
- **min-height** define la altura mínima de un elemento.
- **min-width** define el ancho mínimo de un elemento.
- **width** representa el ancho del elemento.

Listas

En lo referido a las listas, CSS provee las siguientes opciones:

- **list-style-image** establece una imagen para utilizar como viñeta. La sintaxis es:

```
list-style-image: url(archivoImagen)
```

- **list-style-position** define si la segunda línea de un ítem de una lista comenzará a la altura de la viñeta (**inside**) o a la altura del primer carácter del texto (**outside**).
- **list-style-type** define el tipo de viñeta utilizada en una lista de ítems. Los valores más comúnmente utilizados son:
  - **disc** (valor por defecto).
  - **circle** (círculo).
  - **square** (cuadrado).
  - **decimal** (1, 2, 3, etcétera).
  - **decimal-leading-zero** (01, 02, 03, etcétera).
  - **lower-roman** (números romanos en minúscula)
  - **upper-roman** (números romanos en mayúscula)
  - **lower-alpha** (a, b, c, etcétera).

- **upper-alpha** (A, B, C, etcétera).
- **lower-latin** (latín en minúscula).
- **upper-latin** (latín en mayúscula).
- **lower-greek** (griego en minúscula).
- **upper-greek** (griego en mayúscula).
- **none** (ninguno).

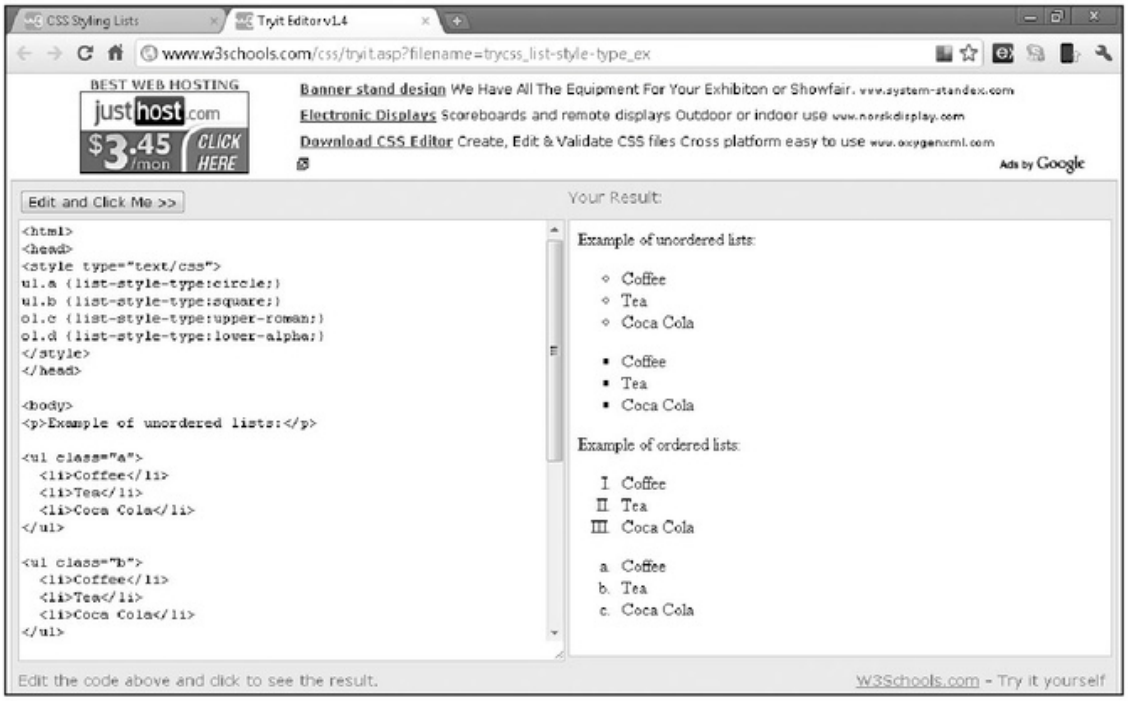


***Figura 17.** Las listas pueden personalizarse a partir de las opciones provistas por CSS mediante constantes predefinidas.*

Márgenes

Las opciones más importantes en lo que se refiere a la especificación de los márgenes de los elementos en CSS son las siguientes:

- **margin-bottom** es la distancia entre el tope inferior del contenedor y la parte inferior del elemento al que se le aplica esta propiedad.
- **margin-left** es la distancia entre el margen izquierdo del contenedor y la parte izquierda del elemento al que se le aplica esta propiedad.
- **margin-right** es la distancia entre el margen derecho del contenedor y la parte derecha del elemento al que se le aplica esta propiedad.
- **margin-top** es la distancia entre el tope del contenedor y la parte superior del elemento al que se le aplica esta propiedad.

En relación con los márgenes interiores (**padding**) las propiedades disponibles para aplicarle a éste, son las siguientes:

- **padding-bottom** define el espacio inferior entre el contenido y el borde del elemento.
- **padding-left** define el espacio izquierdo entre el contenido y el borde del elemento.
- **padding-right** define el espacio derecho entre el contenido y el borde del elemento.
- **padding-top** define el espacio superior entre el contenido y el borde del elemento.

Posicionamiento

Las propiedades referidas al posicionamiento permiten mantener un control sobre el lugar que cada elemento ocupará dentro de la página, su visibilidad, su flotabilidad, etcétera. Entre las más importantes podemos citar las siguientes:

- **bottom** representa la distancia del elemento con respecto al tope inferior de la página o el contenedor.
- **clear** está relacionado con float ya que le da la posibilidad a un elemento de permitir elementos flotantes a su alrededor o no. Los valores posibles son **none** (permite elementos flotantes a ambos lados), **left** (no permite flotantes a su izquierda), **right** (no permite flotantes a su derecha) y **both** (no permite flotantes).
- **clip** permite definir qué parte del elemento se mostrará. Utiliza la construcción **rect** para calcular esa porción, que recibe cuatro argumentos: arriba, derecha, abajo e izquierda, en este orden.
- **float** permite posicionar un elemento a la izquierda o a la derecha de otros elementos adyacentes. Admite tres valores: **right** (derecha), **left** (izquierda) y **none** (ninguno). Un elemento flotante debe tener definido su ancho (con **width**) para ser considerado como tal.
- **left** se aplica a elementos posicionados absoluta o relativamente y define la distancia con respecto al tope izquierdo de la página o el contenedor.
- **overflow**, si el rectángulo formado por las propiedades **top**, **left**, **height** y **width** no coincide con el tamaño normal del elemento al que se aplican, con **overflow** podemos expandir dicho elemento (**visible**), ajustar su tamaño al rectángulo (**hidden**), colocar barras de desplazamiento en el elemento solo si su contenido excede del tamaño del rectángulo (**auto**) y colocar barras de desplazamiento en el elemento si su contenido no excede del tamaño del rectángulo (**scroll**).

III CARACTERÍSTICAS DE POSICIONAMIENTO

Es muy importante tener en cuenta que el posicionamiento realizado con CSS permite un máximo control sobre el lugar que cada elemento de una página web ocupará dentro de ésta, sus condiciones de visibilidad y flotabilidad, así como manejo de capas. El primer borrador de esto fue presentado por la W3C el 19 de agosto de 1997.

- **position** permite posicionar elementos en una página. Sus valores posibles son **static** (valor por defecto, los elementos se dibujan uno a continuación de otro), **absolute** (se ubican dentro de la página de forma independiente, incluso se pueden superponer con otros elementos), **relative** (se ubican en determinados lugares dependiendo de la posición del elemento anterior), **fixed** (similar a **absolute**, solo que los elementos no se mueven cuando el documento es desplazado) e **inherit** (hereda el valor **position** de su contenedor).
- **right** representa la distancia del elemento con respecto al tope derecho de la página o el contenedor.
- **top** se aplica a elementos posicionados absoluta o relativamente y define la distancia con respecto al tope superior de la página o el contenedor.
- **vertical-align** sirve tanto para textos como para imágenes. Los valores que admite (salvo **sub** y **super**) toman como referencia el contenedor del texto o la imagen referida (por ejemplo, una tabla o un párrafo). Los valores posibles son: **top** (toma la parte superior del elemento y la alinea con el tope del contenedor), **bottom** (toma la parte inferior del elemento y la alinea con el pie del contenedor), **text-top** (toma la parte superior del elemento y la alinea con el texto superior del contenedor), **text-bottom** (toma la parte inferior del elemento y la alinea con el texto inferior del contenedor), **baseline** (valor por defecto. Coloca en línea el elemento con el contenedor), **middle** (alinea el punto medio del elemento con el punto medio del contenedor), **sub** (subíndice) y **super** (superíndice). Muchos navegadores no soportan la mayoría de los valores posibles de esta propiedad.
- **z-index**. Mediante la utilización de la propiedad **position** existe la posibilidad de que dos elementos o más se superpongan entre sí y formen capas. Con **z-index** podremos definir el orden de apilamiento de estas capas. Esta propiedad admite solo valores enteros positivos y los elementos con mayor valor de **z-index** se ubicarán por encima de los demás.

Visibilidad

Entre las propiedades relativas a la visibilidad de los elementos y sus posibles valores podemos citar las siguientes:

Si nuestro elemento no está contenido dentro de ningún otro, la posición de las coordenadas se mide respecto del inicio de la etiqueta **BODY**. Si nuestro elemento está contenido dentro de otro elemento, y este contenedor está posicionado absoluta o relativamente, las coordenadas del elemento absoluto se calculan desde el origen de su elemento contenedor.

- **display** se refiere a la forma de mostrar (o no mostrar) un texto y admite los valores **block** (muestra un salto de línea antes y después del elemento), **inline** (sin saltos), **list-item** (igual que **block** solo que muestra el contenido como lista), **marker** (identifica un bloque que contiene caracteres generados desde una hoja de estilo. Se utiliza solo con los pseudo elementos **:before** y **:after**), **table** (muestra como tabla), **inline-table** (tablas en línea), **table-row-group** (grupos de celdas en una fila), **table-header-group** (grupos de celdas de cabecera), **table-footer-group** (grupos de celdas de pie), **table-row** (fila de una tabla), **table-column-group** (grupos de celdas en una columna), **table-column** (columna de tabla), **table-cell** (celda de una tabla), **table-caption** (título de una tabla) y **none** (sin visualización).
- **visibility** por su parte, define si el elemento al cual se le aplica esta propiedad deberá estar a la vista (**visible**) u oculto (**hidden**).

***Figura 18**. Para que las páginas de nuestro sitio no obstruyan la carga del documento siempre es recomendable incluir las referencias a las hojas dentro de la etiqueta head.*

Anexo C

Referencia XML

XML es un metalenguaje de propósito general utilizado por un gran número de sistemas. Esto lo hace importante incluso dentro del desarrollo de aplicaciones web. En este capítulo haremos un recorrido por sus características y las premisas necesarias para generar nuestros propios documentos.

Orígenes y principios generales	**342**
Sintaxis	**343**
Reglas de marcado	344
Comentarios	349
Prólogo de un documento	350
DTD	352
Tipos de documento	356
Creación y visualización de documentos	359
Casos de uso	363
Relación con las bases de datos	365
Espacios de nombres	365
Inclusión de hojas de estilo	367
Transformaciones	367

ORÍGENES Y PRINCIPIOS GENERALES

XML (*eXtensive Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje mantenido por la **W3C** (*World Wide Web Consortium*, **www.w3.org**) con el objeto de superar las limitaciones de otro lenguaje de marcado, el **HTML**. Deriva de **SGML** (*Standard Generalized Markup Language*), que es similar en cuanto a prestaciones y principios. No está orientado a almacenar ningún tipo de dato en particular, sus fines son generales y puede ser utilizado por cualquier organización.

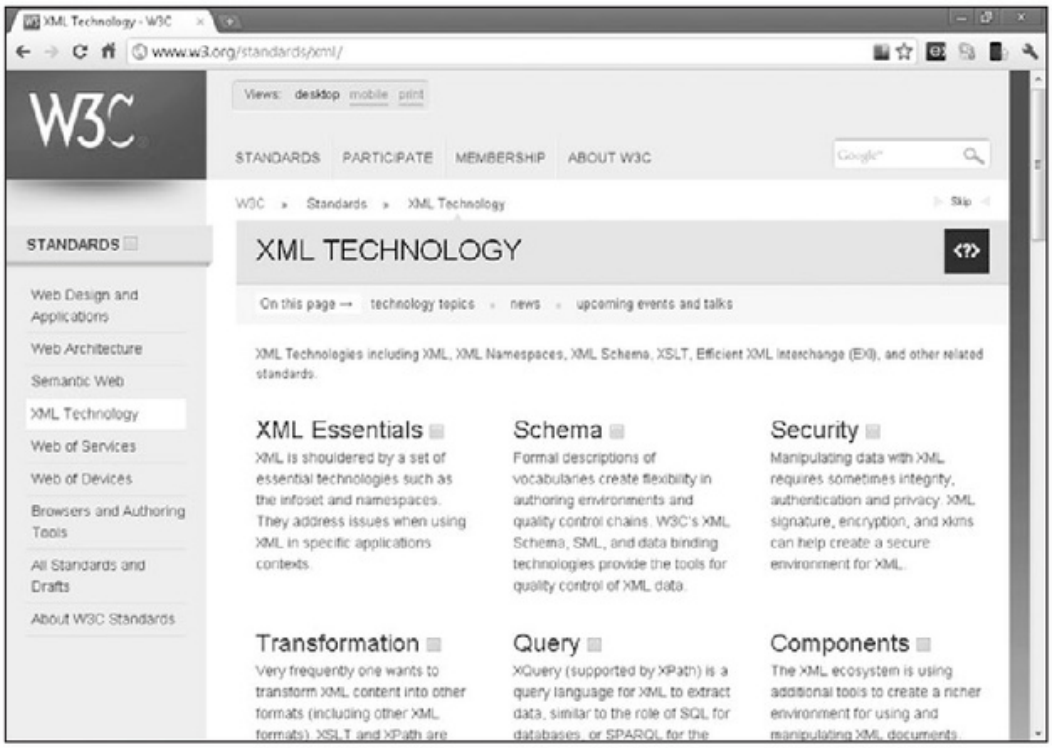


***Figura 1.** XML es uno de los múltiples estándares mantenidos por la organización **W3C** (**World Wide Web Consortium**).*

Tanto HTML como XML son lenguajes de marcado, y la diferencia principal está dada en los objetivos con los que fueron creados. Mientras que HTML es un lenguaje de formato de texto, XML se considera como una manera de estructurar la información, lo que deja la parte de presentación a tecnologías externas como CSS. Otras características de HTML son las siguientes:

- las etiquetas no se encargan de definir la estructura ni tampoco el significado de los datos contenidos, sino solo su formato.
- las etiquetas son limitadas en número. Esto reduce las posibilidades de representación de información sin repetir etiquetas, por lo que es difícil encontrar una información en particular dentro de un documento.
- si bien la información está estructurada en marcas o elementos, no existen reglas estrictas de cómo indicar estas en su creación.

XML y HTML son lenguajes casi opuestos: XML nació y se desarrolló para suplir las necesidades a las que HTML no podía dar respuesta.

SINTAXIS

XML es un metalenguaje, es decir un conjunto de reglas, que permiten crear lenguajes o especificaciones, por ejemplo XHTML. Estas reglas tienen características bien marcadas, entre las que podemos destacar la simpleza y la precisión. Cuando un documento cumple con todas las reglas preestablecidas suele decirse que se trata de un documento **bien formado**.

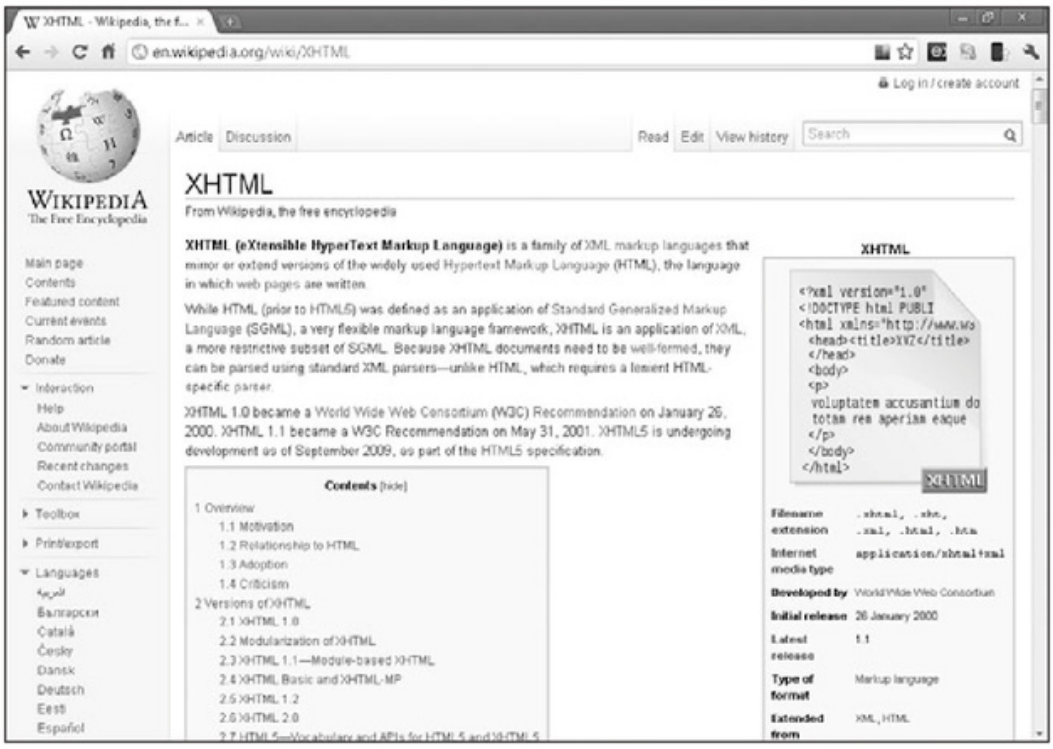


Figura 2. *XHTML es una especificación XML y respeta sus reglas.*

Cuando generamos documentos XML, creamos instancias del metalenguaje, es decir documentos basados en el estándar XML. Al no haber elementos y atributos preestablecidos en este, XML provee la ventaja de permitirle al autor crear sus propias estructuras de información, obteniendo así un nivel de exactitud avanzado que da la posibilidad de representar información de manera inequívoca.
Otro beneficio que deriva de la utilización de XML es la posibilidad de acceder directamente a la información contenida ya que las búsquedas sobre los documentos serán más específicas y consistentes. XML estructura la información de tal manera que, si respetamos determinadas reglas, una aplicación podría llegar a identificar y recuperar datos de manera inequívoca.

Reglas de marcado

Un documento XML está formado básicamente por elementos y atributos. Cada marca o elemento está delimitado por los signos menor que (<) y mayor que (>). Existen las marcas de inicio (**<nombreEtiqueta>**) y de fin (**</nombreEtiqueta>**). Cada elemento está sujeto a reglas específicas, entre las que se encuentran:

- la línea que contenga la versión de XML, en caso de ser incluida, deberá figurar en primer lugar en nuestros documentos XML, ni siquiera podemos dejar líneas en blanco ni espacios antes de esta declaración:

```
<?xml version="1.0"?>
<a>
<b> texto </b>
<b> texto </b>
<b> texto </b>
</a>
```

- la definición de las etiquetas de apertura y cierre, por ejemplo:

```
<nombreEtiqueta> texto </nombreEtiqueta>
```

***Figura 3.** Un documento XML mal formado producirá errores en las aplicaciones que pretendan procesarlo.*

A diferencia de lo que ocurre en los lenguajes como HTML, donde en algunos casos no se considera obligatorio definir la etiqueta de cierre:

```
<p> texto
<p> texto
<p> texto
```

- los nombres de los elementos no pueden contener espacios:

```
<nombre etiqueta> texto </nombre etiqueta>
```

En su lugar podemos utilizar guiones bajos, por ejemplo:

```
<nombre_etiqueta> texto </nombre_etiqueta>
```

- se debe respetar el orden en el que fueron abiertos los elementos y se deben cerrar en el orden inverso. Se pueden anidar elementos, pero deben cerrarse en orden:

```
<a><b><c> texto </c></b></a>
```

- los elementos pueden constar de una sola etiqueta. Se denominan **vacíos** y la misma etiqueta hará de apertura y cierre:

```
<nombreEtiqueta /> texto
```

En XHTML contamos con varios ejemplo de elementos vacíos, entre ellos:

```
<img src="archivo.png" />
```

```
<br />
```

```
<hr />
```

Esta regla no se contradice con la primera, ya que la única etiqueta se considera al mismo tiempo de inicio y fin. Todos los elementos deben tener obligatoriamente una etiqueta de fin, incluso aquellos que consten de una sola etiqueta.

- cada elemento o etiqueta de apertura puede contar con los llamados atributos, o también modificadores, que suelen representar características o cualidades relacionadas directamente con el elemento al que pertenecen. Puede haber más de un atributo por elemento, puede haber solo uno o puede no haber ninguno. El valor del atributo debe encerrarse entre comillas simples o dobles:

```
<nombreEtiqueta> texto </nombreEtiqueta>
```

```
<nombreEtiqueta atr1="xxx"> texto </nombreEtiqueta>
```

```
<nombreEtiqueta atr1="xxx" atr2="yyy"> texto </nombreEtiqueta>
```

Incluso los elemento vacíos pueden tener sus propios atributos:

```
<nombreEtiqueta atr1="xxx" />
```

```
<img src="archivo.png" />
```

Los atributos deben contener valores, aunque sean valores vacíos o nulos:

```
<nombreEtiqueta atr1=""> texto </nombreEtiqueta>
```

En HTML existe la siguiente sintaxis, no válida en XML:

```
<option selected> valor </option>
```

Como XHTML deriva de XML, la sintaxis correcta es ahora:

```
<option selected="selected"> valor </option>
```

Podemos utilizar comillas simples o dobles siempre y cuando el valor del atributo no entre en conflicto y ponga en duda la delimitación del atributo. Si el valor del atributo contiene algún tipo de comillas puede usarse el otro tipo para rodear el valor.

```
<nombreEtiqueta atr1=”xx’x” />
```

```
<nombreEtiqueta atr1=’xx”x’ />
```

En caso de no saber concretamente qué valores tomarán los atributos, que es en general lo que ocurre, podemos reemplazar las comillas simples y dobles en el valor por sus entidades correspondientes, como vimos algunos capítulos atrás. Lo mismo ocurre con otros delimitadores o símbolos que tienen un significado especial:

CARÁCTER	SÍMBOLO	DESCRIPCIÓN
'	'	Comilla simple
"	"	Comilla doble
>	>	Mayor que
<	<	Menor que
&	&	Ampersand

Tabla 1. *Entidades equivalentes en XML.*

```
<nombreEtiqueta atr1=”xx&apos;x” />
```

```
<nombreEtiqueta atr1=’xx&quot;x’ />
```

Al momento de recorrer un documento XML, la aplicación encargada de interpretarlo (**parseador**) leerá una entidad y reemplazarla por el carácter original.

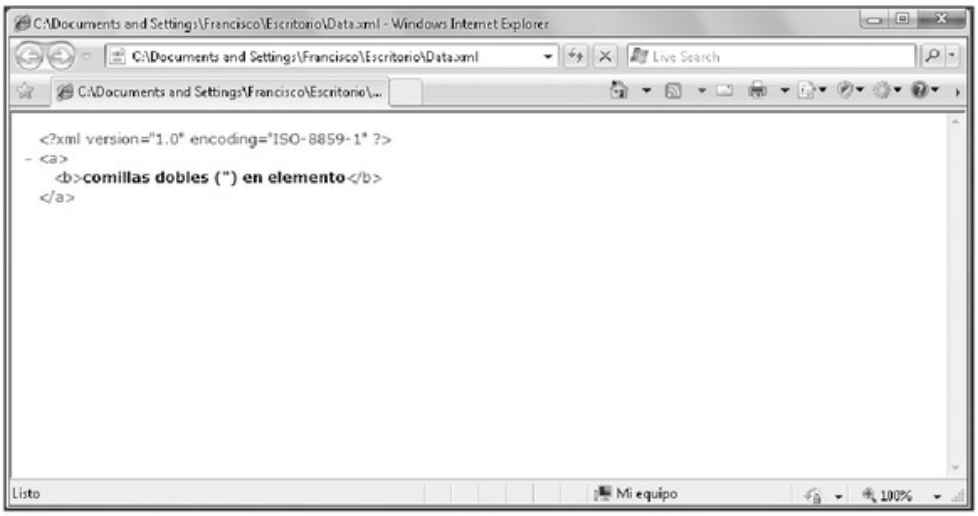


Figura 4. *Los intérpretes leen entidades e intercambian a éstas por el o los caracteres originales.*

La generación de un documento XML queda a cargo de su autor, quien generalmente no tiene restricciones a la hora de insertar elementos y atributos. Como regla general, cuando un dato depende de un elemento o puede tomar valores predefinidos, debe ser un atributo. Otro consejo de diseño puede ser que los valores de los atributos no deben ser demasiado extensos.

```
<tipo_de_documento=”dni” />
<tipo_de_documento=”ci” />
<tipo_de_documento=”lc” />
```

- a diferencia de lo que sucede en HTML, los elementos son sensibles a mayúsculas: se escriben en letra minúscula por convención, pero esto no es obligatorio. Debemos escribir de igual manera los tags de apertura y cierre correspondientes a un mismo elemento:

```
<nombreEtiqueta> incorrecto </nombreetiqueta>
```

```
<nombreEtiqueta> correcto </nombreEtiqueta>
```

- un documento deberá contar con un elemento raíz, es decir aquel que encierra a todos los demás.

```
<nombreEtiqueta>
<elemento> texto </elemento>
<elemento> texto </elemento>
<elemento> texto </elemento>
<elemento> texto </elemento>
</nombreEtiqueta>
```

III COMBINAR XML

XML fue encontrado como una alternativa excelente en la interacción de bases de datos y aplicaciones o portales desarrolladas en Flash con Actionscript. PHP también es otro lenguaje que maneja XML como puente intermedio entre Flash y este lenguaje, lo que permite el desarrollo simple de sitios y portales con e-commerce entre otros.

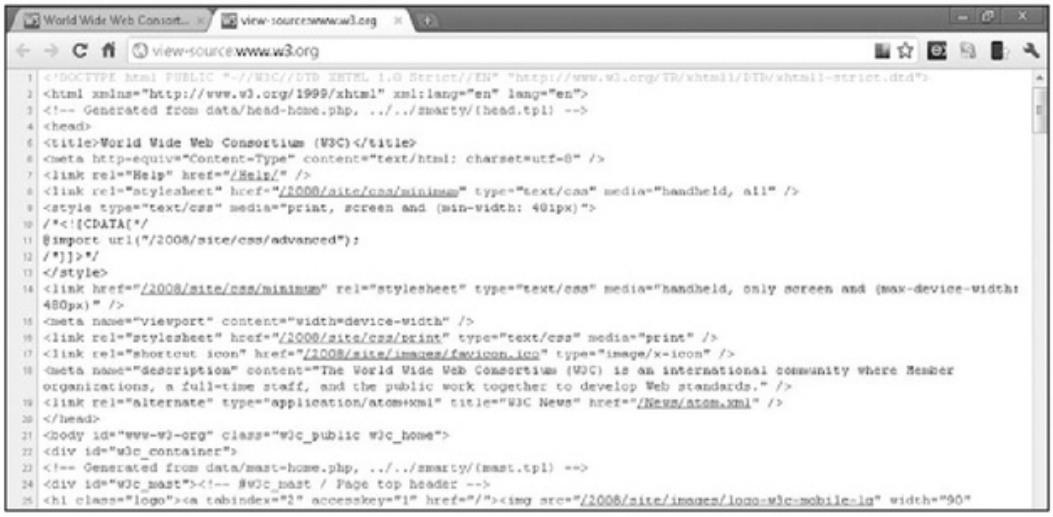


Figura 5. *En* ***XHTML*** *también hay un elemento que contiene a todos los demás, y ese elemento es justamente* ***HTML****.*

- es importante tener en cuenta que en relación con lo anterior, un documento XML debe contener por lo menos un elemento.

Comentarios

Debemos saber que dentro de un documento XML podemos insertar comentarios que serán omitidos por el parseador al momento de interpretar el contenido existente. La sintaxis de un comentario es la siguiente:

```
<!– este es un comentario –>
```

Los comentarios pueden incluirse en cualquier parte de un documento, siempre que estén debajo de la declaración de inicio de XML y no dentro de un elemento.

```
<?xml version=”1.0”?>
<!– ejemplo de comentario –>
<a>
<b> texto </b>
<b> texto </b> <!– ejemplo de comentario –>
<b> texto </b>
</a>
<!– ejemplo de comentario –>
```

El contenido de un comentario puede contener cualquier conjunto de caracteres a excepción de guiones dobles (—). Los comentarios pueden incluir múltiples líneas:

```
<?xml version="1.0"?>
<!-

ejemplo
de
comentario
multilinea

->
<a>
<b> texto </b>
<b> texto </b>
<b> texto </b>
</a>
```

Como vemos, es idéntica a la utilizada en HTML y XHTML.

Prólogo de un documento

Además de contar con elementos y atributos, los documentos contienen información acerca de la versión de XML utilizada, el juego de caracteres y la indicación acerca de si se utilizarán documentos externos o no. Estos datos se ubican en una parte denominada prólogo, que luego les servirá a las aplicaciones encargadas de leer los documentos para identificarlos correctamente como XML. En caso de incluirse (si bien el prólogo para un documento está recomendado en la especificación oficial, no es obligatorio), el prólogo debe ser lo primero que veamos en un documento XML, por encima de los elementos y atributos definidos. Un prólogo tiene la siguiente forma:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

Los tres atributos disponibles son **version** (obligatorio), **encoding** (opcional) y **standalone** (opcional). Cada uno de los valores de estas tres propiedades puede encerrarse entre comillas simples o dobles.

Versión

Como toda tecnología, XML sufre modificaciones y cada determinado tiempo surgen nuevas versiones. Las aplicaciones que hacen uso de esta clase de archivos podrán requerir ciertas funcionalidades de una versión u otra, por lo que es importante definir cuál es la de nuestro documento.

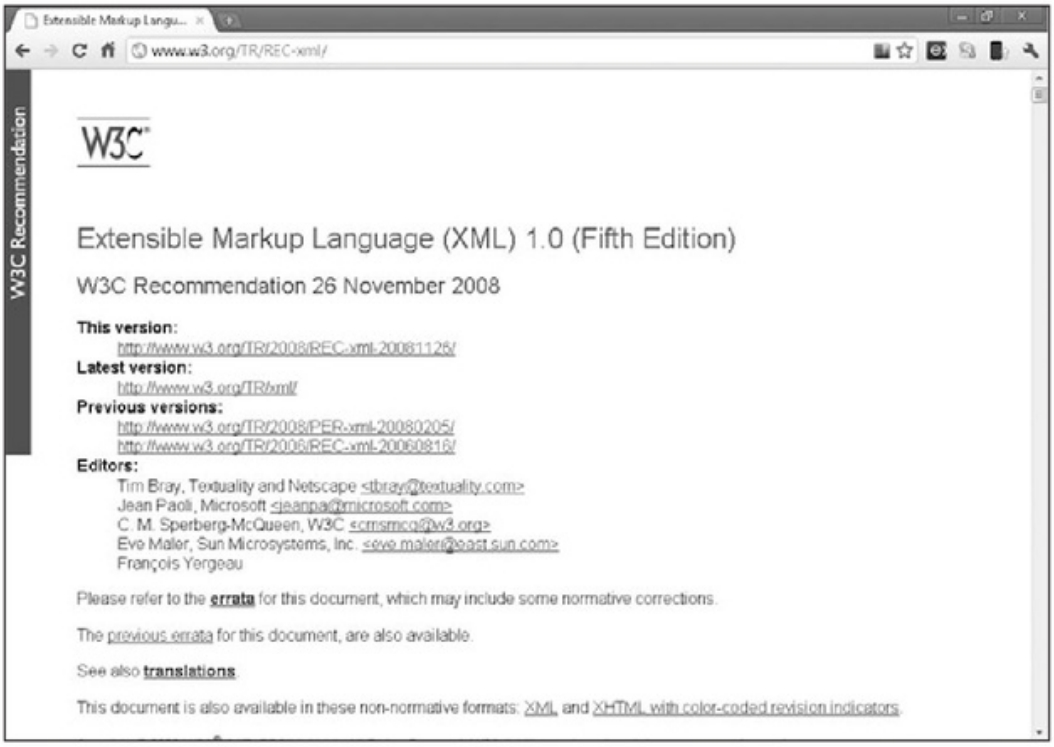


Figura 6. *XML evoluciona y las nuevas versiones ya están en camino.*

A medida que surjan nuevas versiones, deberemos tener en cuenta cuáles son las mejoras introducidas y cuál es la compatibilidad con la utilizada por nosotros hasta ese momento en nuestros documentos. En caso de que actualicemos a una versión distinta de la que utilizamos eventualmente deberemos revisar nuestros documentos y modificarlos según la última especificación.

Encoding

El juego de caracteres permite establecer las alternativas con las que contaremos a la hora de escribir nuestros documentos: cada juego tiene un conjunto de caracteres disponibles y deberemos seleccionar el que se adapte a nuestras necesidades.

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Si no se especifica, se asume que los caracteres pertenecen a **UTF-8**, un estándar **Unicode** que soporta cada carácter e ideograma de cualquier lenguaje del mundo.

Standalone

Un documento XML puede hacer uso de otros documentos externos para tomar forma. Esta propiedad puede tomar uno de dos valores: **yes** (indica que no se utilizarán documentos externos) y **no** (indica que sí se utilizarán documentos externos). Si se omite, asume que existe la posibilidad de que utilicen documentos externos, sin importar si luego se utilizan o no.

DTD

Ligar un documento XML a un DTD (*Document Type Definition*, Definición del Tipo de Documento) hará que podamos definir la estructura de nuestros documentos de forma estricta y detallada. Podríamos definir qué elementos pueden estar dentro de otros, en qué orden deben figurar, qué tipo de datos podrán almacenar, qué atributos contendrán, etcétera. Aquí almacenaremos reglas que nos marcarán una línea en nuestros desarrollos para validar la consistencia de los datos.

***Figura 7.** Los DTD permiten establecer reglas en cuanto a la ubicación y el contenido del contenido XML.*

Un documento XML que respeta las reglas impuestas en un DTD asociado se considera como **válido**. La inclusión de DTD en un documento XML es opcional, es decir que podemos desarrollar bien formados sin utilizar validaciones obligatorias. En un DTD se especifican los elementos, los atributos y las entidades que un documento XML puede contener. En el siguiente ejemplo el XML debe contar con los elementos **a**, **b**, **c** y **d**, y además **a** debe contener a **b**, **c** y **d**, en ese orden. El elemento **a** no puede contener texto (solo elementos) en tanto que **b**, **c** y **d** solo pueden contener texto (declarado con la palabra reservada **PCDATA**) y no subelementos:

```
<!ELEMENT a (b, c, d)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
<!ELEMENT d (#PCDATA)>
```

El siguiente documento es válido en relación con el DTD anterior:

```
<?xml version="1.0"?>
<!DOCTYPE a SYSTEM "documento.dtd">
<a>
  <b> texto </b>
  <c> texto </c>
  <d> texto </d>
</a>
```

Por supuesto que éste es un ejemplo simple: el nivel de personalización es muy profundo y cubre todas las necesidades posibles.

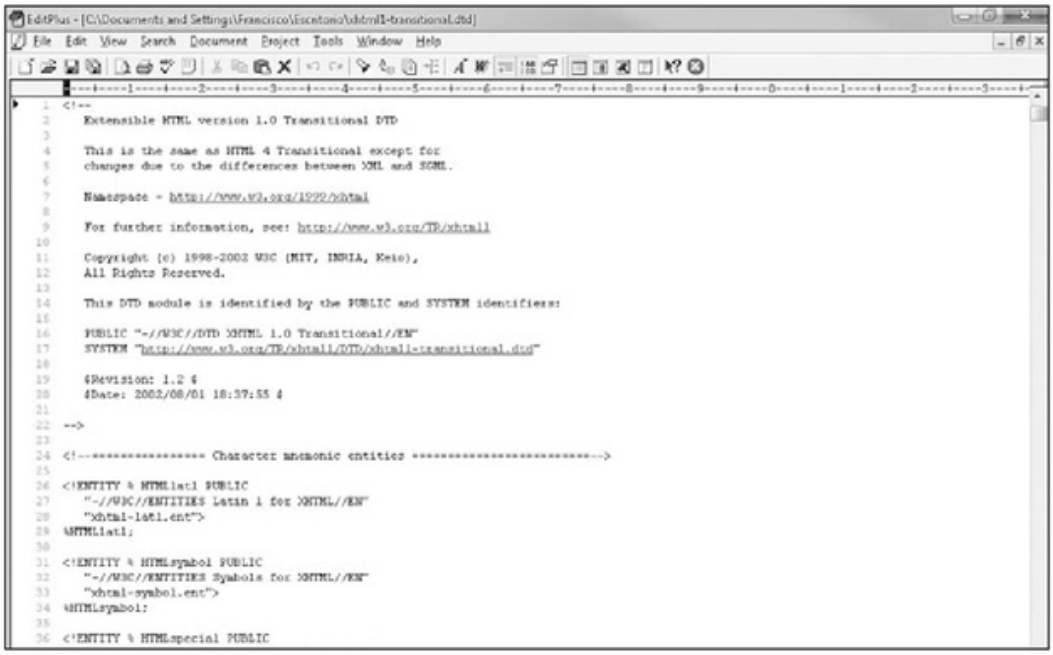


***Figura 8. XHTML** mantiene sus propios documentos DTD asociados, solo que los navegadores web todavía no exigen la validez de los documentos.*

Para validar un documento XML contra un DTD necesitamos un parseador o analizador sintáctico. Existen varias formas de validar un documento: mediante editores XML o con servicios web específicos. Muchos lenguajes de programación (**Java**, **Python**, **C**, **Perl**, etcétera) incorporan librerías y funciones para validación XML. Con el lenguaje JavaScript podemos utilizar el objeto **`XMLDOM`** (Internet Explorer versión 5.0 o superiores) y validar nuestros documentos:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
```

```
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
 </head>
 <body>

<script type="text/javascript">
//<![CDATA[

var archivoXML = "nombreArchivo.xml";

var objXML = new ActiveXObject("Microsoft.XMLDOM")
objXML.async="false"
objXML.validateOnParse="true"
objXML.load(archivoXML)

if (objXML.parseError.errorCode == 0)     {
     document.write("<br>" + archivoXML + ":<br>")
     document.write("<br>No se encontraron errores.")
     document.write(" Documento XML valido.")
} else                                                                {
     document.write("<br>" + archivoXML + ":<br>")
     document.write("<br>Código del error: ")
     document.write(objXML.parseError.errorCode)
     document.write("<br>Descripción del error: ")
     document.write(objXML.parseError.reason)
     document.write("<br>Línea: ")
     document.write(objXML.parseError.line)
}
```

III PARA TENER EN CUENTA

Es importante tener en cuenta que siempre debemos recordar que el documento de definición de tipos **DTD** tiene que aparecer antes del primer elemento que se encuentra en el documento. Debemos tener en cuenta que el nombre que sigue a **DOCTYPE** en el documento de definición de tipos debe ser el mismo que el nombre del elemento raíz.

```
//]]>
</script>

 </body>
</html>
```

La variable **archivoXML** deberá contener el nombre del documento XML a validar.

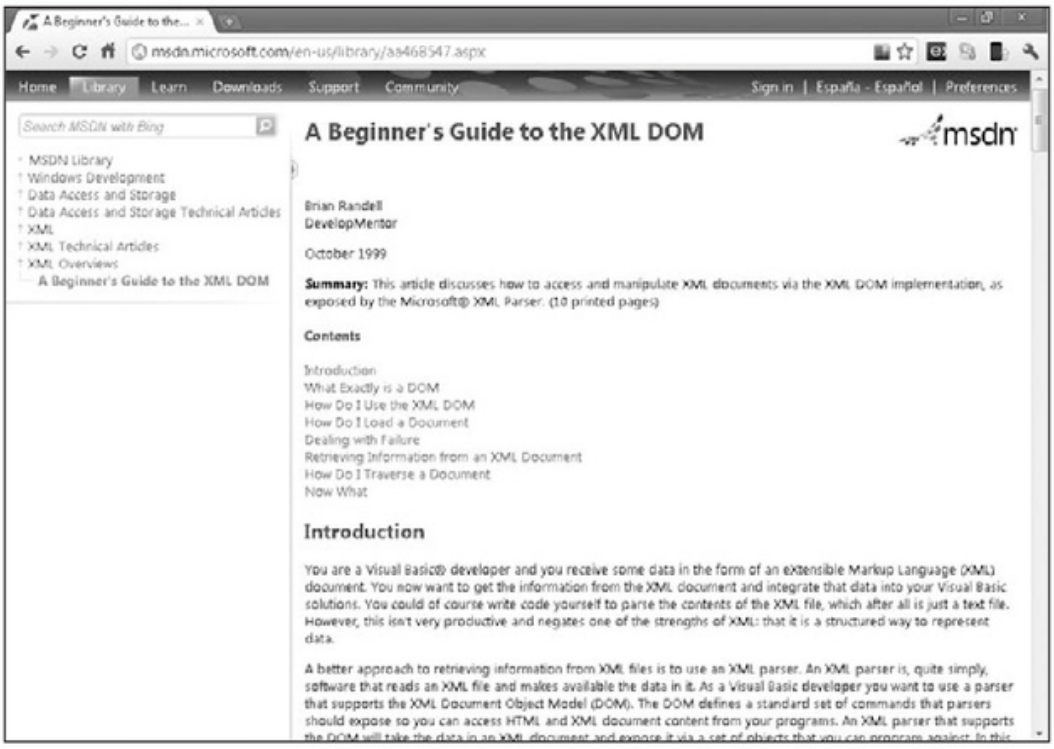


Figura 9. *Internet Explorer dispone de un elemento* ***ActiveX*** *llamado* ***XMLDOM****.*

Los DTD se utilizan cuando los documentos XML son demasiado extensos o complejos y necesitamos mantener un control sobre ellos. Otros casos pueden estar relacionados con las aplicaciones que reciben, y procesan documentos XML y precisan un determinado orden en cuanto a los elementos y atributos contenidos para su correcto funcionamiento.

III USOS DTD

Debemos saber que una de las ideas principales en cuanto a la forma de utilización de los documentos DTD es el intercambio que puede generarse entre los distintos grupos de desarrolladores. Por este motivo, es común que se incluyan externamente, es decir, que el contenido del DTD se almacene en un archivo separado, para que pueda ser accedido sin problemas.

Una alternativa a la utilización de DTD está dada por los **XML Schemas**, que mantienen una sintaxis diferente. Podemos obtener más información en:

- **www.w3.org/TR/xmlschema-0**
- **www.w3.org/TR/xmlschema-1**
- **www.w3.org/TR/xmlschema-2**

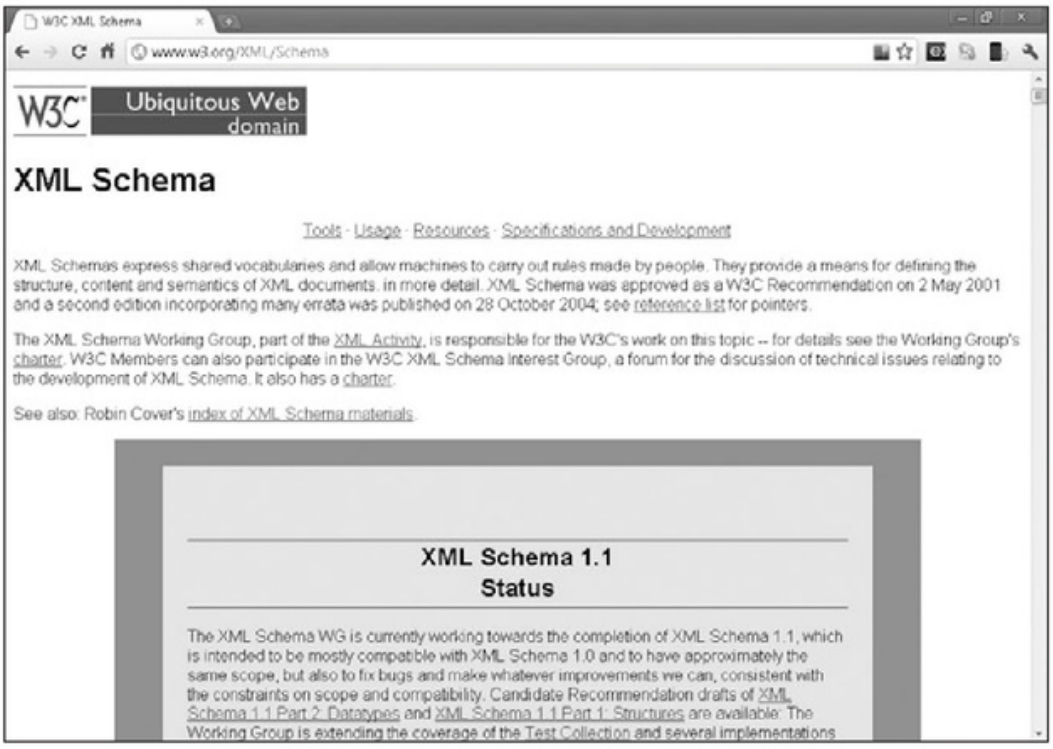


Figura 10. *XML Schemas actualmente es una de las alternativas a los DTD.*

Tipos de documento

Podemos adjuntar información relevante dentro del prólogo de un documento: entre las etiquetas **<!DOCTYPE** y **>** se ubica la declaración del tipo de documento y nos servirá para definir datos como entidades, ubicación del DTD, entre otros.

```
<?xml version="1.0"?>
<!DOCTYPE raiz "documento.dtd"
[
<!ENTITY anio "2008">
]>

<raiz>
...
</raiz>
```

Entidades

Las entidades sirven para declarar una asociación entre una constante y un contenido. Luego podrá ser utilizada a lo largo del documento y el parseador se encarga de reemplazarla por el contenido cada vez que la encuentre. Para hacer uso de una entidad debemos seguir dos pasos básicos: realizar la declaración y llamarla desde la parte principal del documento.

```
<?xml version="1.0"?>
<!DOCTYPE raiz "documento.dtd"
[
<!ENTITY anio "2008">
]>

<raiz>
<fecha>&anio;</fecha>
</raiz>
```

Incluso podemos ubicar el contenido de las entidades en archivos externos:

```
<!ENTITY anio SYSTEM "documentoExterno.xml">
```

Ubicación del DTD

En el caso de que el archivo DTD se encuentre en la maquina local y tengamos esa clase de acceso, podemos referenciarlo de la siguiente manera:

```
<!DOCTYPE raiz SYSTEM "c:\documento.dtd">
```

También existe la posibilidad de vincular DTD ubicados en otro sitio web o servidor, si incluimos la dirección URL completa de éste.
Otra forma de acceder es mediante la utilización del formato **FPI** (*Formal Public Identifier*, Identificador Público Formal). Veamos un ejemplo:

```
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
   "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
```

La sección entre comillas ubicada después de PUBLIC está compuesta por las siguientes partes, separadas por **//**: **reconocido // dueño // descripción // idioma.**

PARTE	DESCRIPCIÓN
reconocido	Puede tomar los valores + (el DTD es reconocido por alguna entidad oficial) y - (el DTD no es reconocido por ninguna entidad oficial).
dueño	Palabra clave que identifica a la institución responsable que se encarga de mantener este documento.
descripción	Indica el tipo de documento (DTD) y su nombre identificador.
idioma	Acrónimo del lenguaje por defecto.

***Tabla 2.** Partes componentes de un **FPI**.*

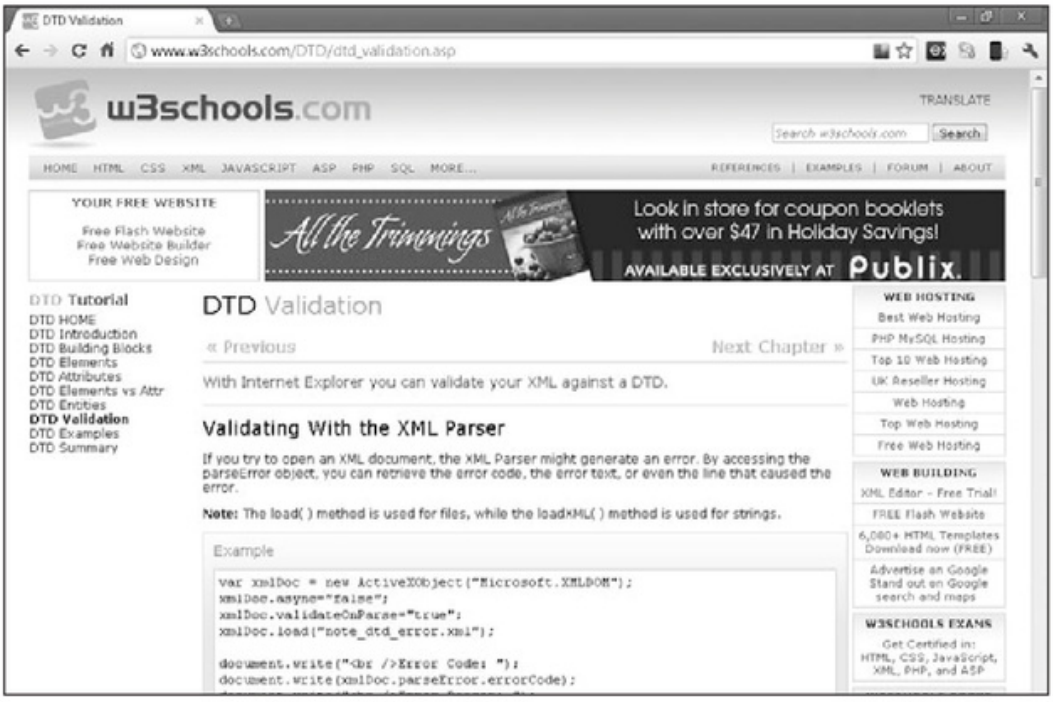


***Figura 11.** La ubicación de un DTD debe especificarse correctamente antes de validar un documento XML.*

Elemento raíz

Antes de la ruta al DTD se incluye el nombre del elemento raíz del XML:

```
<?xml version="1.0"?>
<!DOCTYPE raiz "documento.dtd"
[
<!ENTITY anio "2008">
]>

<raiz>
...
</raiz>
```

El elemento raíz es aquel que encierra o contiene a todos los demás.

Creación y visualización de documentos

Los documentos XML son de código abierto. Es posible acceder a ellos desde cualquier editor de texto como **Emacs**, **WordPad**, **NotePad**, **Vim** o cualquier otro disponible en el mercado. Las herramientas citadas son de propósito general, pero hay otras específicas para XML que proveen opciones avanzadas de validación, por ejemplo. Un editor específico para XML (recordemos que XML es un estándar abierto, por lo que no hay una empresa comercial propietaria y tampoco hay un único entorno de desarrollo) entiende las particularidades del metalenguaje. Esto ayuda al desarrollador en sus tareas de rutina, y además pueden sernos de mucha utilidad cuando los documentos XML se vuelvan tan complejos o extensos que sea imposible editarlos manualmente. Para citar solo algunos ejemplos podemos nombrar:

- **XML Writer** (**http://xmlwriter.net**).
- **XML Spy** (**www.altova.com/products_ide.html**).
- **Stylus Studio** (**www.stylusstudio.com**).

En cualquier caso, si seguimos las reglas preestablecidas y eventualmente adjuntamos hojas de estilo para darle un aspecto amigable a la salida obtenida, como veremos más adelante en este capítulo, podremos generar nuestros propios documentos XML. El proceso es simple: deberemos crear un documento de texto, ingresar el código XML necesario y guardar el archivo con la extensión **XML**. Los documentos XML pueden visualizarse gráficamente mediante estructuras jerárquicas, que es una forma intuitiva de apreciar sus características.

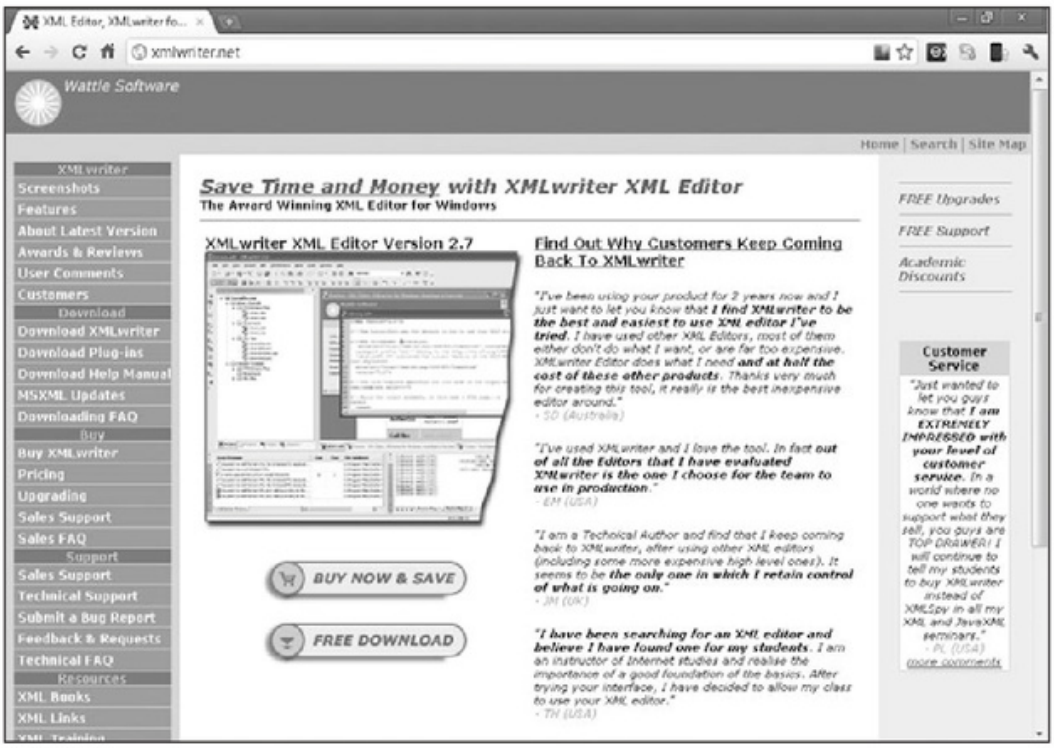


***Figura 12. XML Writer** es una alternativa profesional para el desarrollo con XML.*

Los navegadores web visualizan XML de distintas maneras. Microsoft Internet Explorer (**www.microsoft.com**) diagrama y muestra un documento XML válido mediante una estructura de tipo árbol. En los siguientes casos tomamos como ejemplo un extracto del archivo que genera la aplicación **iTunes** para exportar listas de temas musicales:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
   "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Major Version</key><integer>1</integer>
  <key>Minor Version</key><integer>1</integer>
  <key>Application Version</key><string>7.3.2</string>
  <key>Features</key><integer>1</integer>
  <key>Music Folder</key><string>file://localhost/C:/
     Documents%20and%20Settings/Francisco/Mis%20documentos/
Mi%20m%C3%BAsica/iTunes/iTunes%20Music/</string>
  <key>Library Persistent ID</key><string>1CE33ABF474E4274</string>
  <key>Tracks</key>
  <dict>
    <key>142</key>
    <dict>
      <key>Track ID</key><integer>142</integer>
      <key>Name</key><string>Motorboat</string>
      <key>Artist</key><string>Love is all</string>
      <key>Kind</key><string>Archivo de audio MPEG</string>
      <key>Size</key><integer>3061968</integer>
      <key>Total Time</key><integer>127477</integer>
      <key>Track Number</key><integer>2</integer>
      <key>Track Count</key><integer>3</integer>
      <key>Date Modified</key><date>2006-07-15T22:40:54Z</date>
      <key>Date Added</key><date>2007-12-11T00:09:13Z</date>
      <key>Bit Rate</key><integer>192</integer>
      <key>Sample Rate</key><integer>44100</integer>
      <key>Play Count</key><integer>13</integer>
      <key>Play Date</key><integer>3284577751</integer>
      <key>Play Date UTC</key><date>2008-01-31T00:42:31Z</date>
      <key>Skip Count</key><integer>2</integer>
      <key>Skip Date</key><date>2008-01-24T00:00:56Z</date>
      <key>Persistent ID</key><string>C3FB76498246A9C4</string>
```

```
        <key>Track Type</key><string>File</string>
        <key>Location</key><string>file://localhost/C:/
           Documents%20and%20Settings/Francisco/Escritorio/grabar%203/
           musica/varios/motorboat.mp3</string>
        <key>File Folder Count</key><integer>-1</integer>
        <key>Library Folder Count</key><integer>-1</integer>
      </dict>
    </dict>
</dict>
</plist>
```

Figura 13. *La estructura de árbol exhibida por los navegadores nos permite un acceso intuitivo a las partes de un documento XML.*

Mozilla Firefox (**www.mozilla.org**) nos da información acerca de las hojas de estilo asociadas con el documento y, a continuación, exhibe el árbol correspondiente:

III VALIDEZ DE XML

Que un documento esté bien formado solamente habla de su estructura sintáctica básica, es decir que se componga de elementos, atributos y comentarios como XML establece que se escriban. Cada aplicación de XML necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

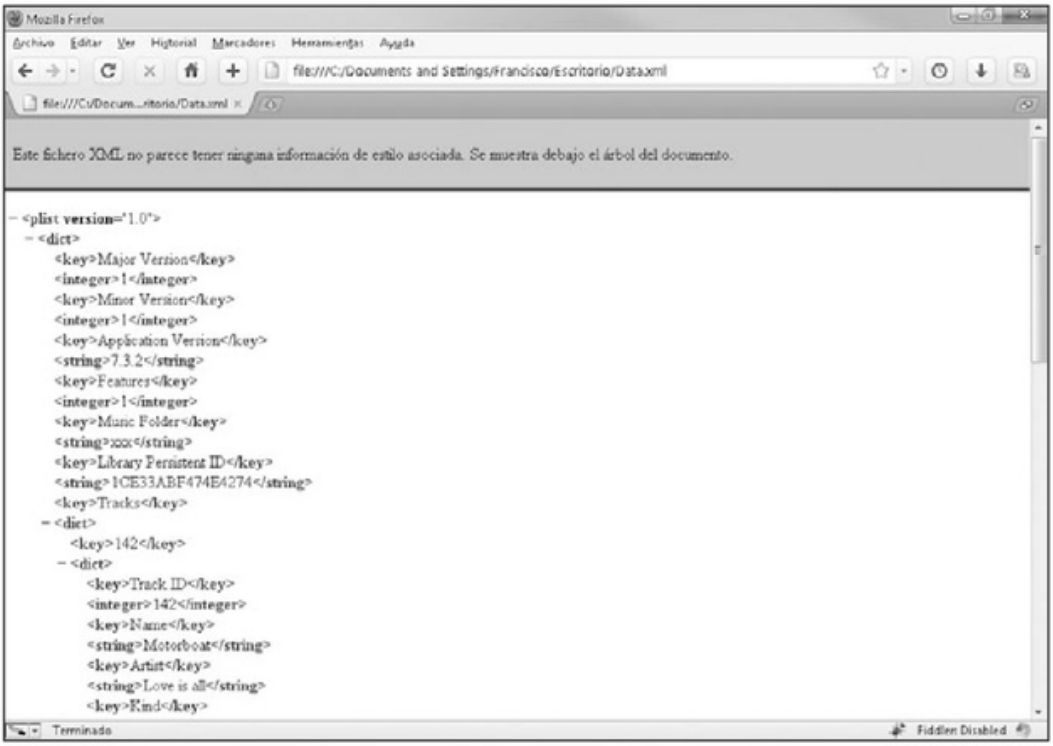


Figura 14. *Mozilla Firefox es uno de los navegadores que más apoyo les brinda a los estándares libres como XML.*

Opera (se encuentra en el sitio web **www.opera.com**) por su parte se encarga de generar una salida en texto plano, ya que necesita que el documento tenga una hoja de estilos adjunta como lo que vemos a continuación:

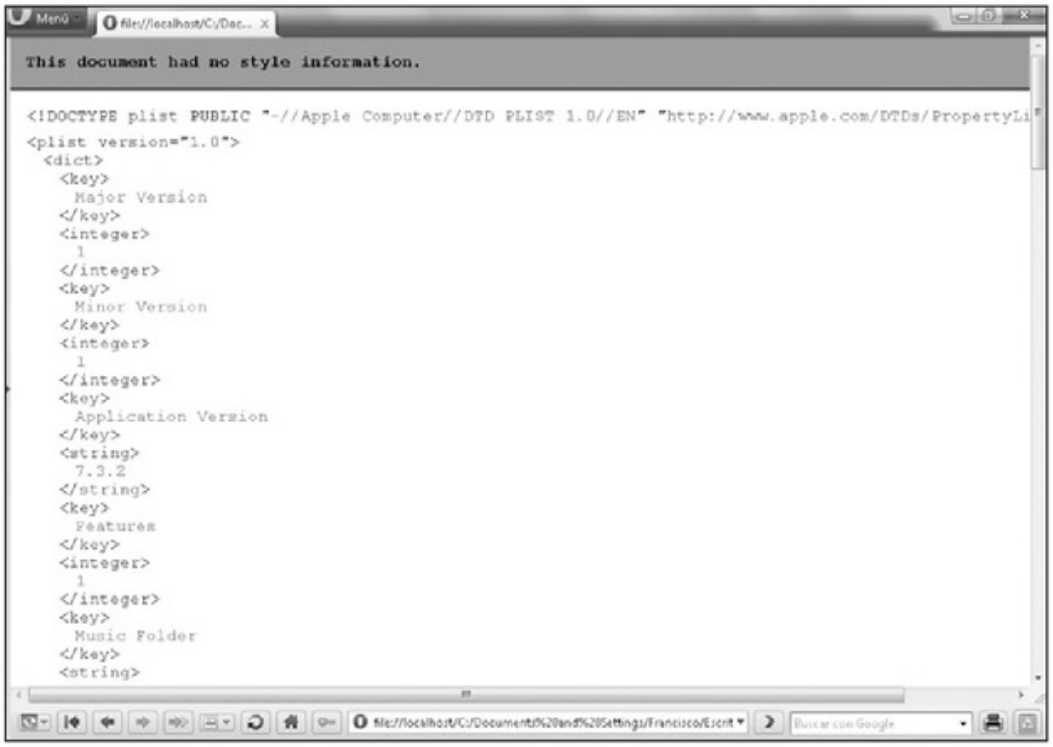


Figura 15. *El navegador Opera se caracteriza por su rapidez y por el soporte brindado a las tecnologías emergentes.*

Algo similar sucede con el navegador web **Safari** (**www.apple.com/safari**):

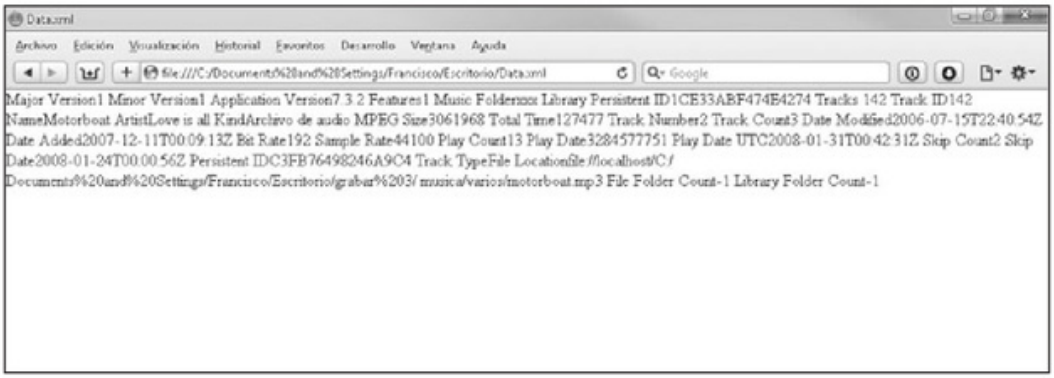


***Figura 16.** Safari ya es un clásico navegador para **Mac OS X** y **Windows**.*

El soporte brindado por los navegadores web a documentos XML asociados con una hoja de estilo **CSS** no es completo. **CSS2**, que incorpora mejoras para trabajar específicamente con XML, es soportado en forma muy completa.
Al contrario de lo que sucede con el lenguaje (X)HTML, donde hay elementos preestablecidos que son directamente comprendidos por el navegador web, aquí están definidos por el autor del documento XML, lo que implica la necesidad de una tecnología externa para definir el diseño buscado.

Casos de uso

Un documento XML puede ser utilizado en muchos contextos diferentes, ya que se trata solo de un formato de transporte e incluso almacenamiento de datos. Algunos usos pueden ser los siguientes:

- Configuración: las aplicaciones pueden almacenar datos particulares de configuración en esta clase de documentos. Así como en un tiempo eran muy comunes los archivos **.INI**, hoy en día podemos comenzar a ver XML en este campo.
- Base de datos de aplicaciones: cuando la utilización de un sistema gestor de bases de datos es innecesaria o técnicamente imposible, XML surge como formato de almacenamiento estándar para distintas clases de aplicaciones. Como ejemplos

III PREMISAS DE XML

Es interesante mencionar que desde el comienzo del desarrollo con XML se tuvieron en cuenta temas como el de la necesidad de mantener de alguna manera la **consistencia** de los datos utilizados, darles un **formato** diferente según la necesidad particulares del receptor, recuperar información específica y también validar los contenidos.

podemos citar a los desarrollos hechos en **Flash** o incluso programas como **Windows Live Messenger**, entre muchos otros en la actualidad.

Figura 17. *Cada vez más aplicaciones utilizan XML para almacenar e incluso exhibir información.*

- Intercambio: al ser un formato estándar, popular y con reglas claramente definidas, dos aplicaciones o más pueden establecer como idioma a XML para generar, enviar, recibir y procesar mensajes. Esto va más allá de las aplicaciones web ya que puede extenderse a cualquier tipo de organización que requiera comunicar interna o externamente sus propios sistemas componentes (aplicaciones desarrolladas en lenguajes, bases de datos, plataformas e incluso protocolos de comunicación diferentes) sin modificar radicalmente su estructura sino agregando la posibilidad de exportar e importar documentos estructurados que contengan los mensajes correspondientes. Esto tiene un beneficio extra que es el de reducir la carga en la red. Cuando se opta por utilizar como medio de transporte a XML lo que se hace es

III FORMATO DTD

Al igual que un archivo XML, un DTD es un documento conformado por texto plano que podrá ser manipulado con la mayoría de los editores disponibles en el mercado, por ejemplo el Bloc de notas incorporado en Microsoft Windows. Esto va en sintonía con XML, que impulsa la facilidad y la transparencia en los distintos proyectos.

evitar el traslado de información anexa que no tenga que ver estrictamente con lo necesario para los servidores y clientes, lo que hacemos es trasportar solo las estructuras y los datos que ellas contienen.

- Servicios web: otra de las posibilidades de alguna manera relacionada con el punto anterior es que los servicios web requieran entradas o devuelvan salidas en formato XML, que es algo muy común hoy en día. Luego, las aplicaciones que se encarguen de leer y manipular los documentos XML podrán delimitar sus regiones para identificar y separar las partes requeridas.

Relación con las bases de datos

Algo que puede llegar a ser útil para comprender qué es XML y cómo funciona es la comparación con el trabajo realizado por una base de datos. Cuando definimos la estructura de las tablas de una base, lo hacemos según nuestros requerimientos particulares y la aplicación que intente acceder a los datos almacenados deberá entre comillas comprender la lógica de la estructuración especificada. En definitiva, una base de datos por sí misma no tiene ningún valor si no hay una aplicación que interprete los datos almacenados y los manipule de alguna manera. Lo que importa tanto en las bases de datos tradicionales como en XML es que la información se almacena de forma organizada para que las aplicaciones accedan y obtengan lo que buscan. En síntesis, XML nos da la libertad de generar estructuras a nuestra medida pero nos ata a respetar algunas reglas de forma precisa.

Espacios de nombres

Al no tener elementos predefinidos y con la posibilidad de usar más de un documento o una especificación XML a la vez, puede suceder que los nombres se repitan, por lo que hay que buscar la manera de diferenciarlos según su origen.
Por ejemplo, un elemento llamado **nombre** tendrá diferente significado según la circunstancia en que se utilice, es decir si aparece en dos estructuras diferentes o más. La solución a este problema potencial es que las construcciones de documentos posean nombres universales, es decir diferenciables unos de otros, cuyo

Hay una gran cantidad de aplicaciones diseñadas como editores XML que se encargan de incorporan la posibilidad de validar los documentos creados. Pero recomendamos utilizar algún editor especializado ya que hay mucha variedad en el mercado y podemos encontrar versiones de prueba de cada uno de ellos para realizar la elección final.

ámbito se extienda más allá del documento que las contiene. Los espacios de nombres (**namespaces**) de XML ofrecen una respuesta a este inconveniente, ya que permiten separar en conjuntos a los elementos según su origen.
Los espacios de nombres primero deberán ser declarados de la forma **xmlns:nombre = "url"**, en donde **xmlns** le avisa al parseador XML que se está declarando un espacio de nombres. **nombre** es el prefijo que luego podremos utilizar para identificar a qué espacio de nombres corresponde cada elemento de nuestro documento y **url** usualmente debe corresponder a la dirección URL de la entidad que mantiene el espacio de nombres. Debido a que en ocasiones los espacios de nombres son creados por usuarios y no tienen una entidad oficial relacionada, la URL solo es necesaria para aportarle información adicional al espacio utilizado.
La declaración de cada espacio de nombres se incluye relacionándolo con un elemento. Para que podamos acceder a realizar la definición del espacio de nombres al cual corresponde un determinado elemento, utilizamos la sintaxis que se muestra en la porción de código que mostramos a continuación:

```
<prefijo:nombreElemento xmlns:prefijo="URL">
```

Luego, todos los descendientes de ese elemento serán parte del mismo espacio de nombres, a menos que se especifique lo contrario:

```
<prefijo:otroElemento> texto </prefijo:otroElemento>
```

Otra opción es utilizar un espacio de nombres por defecto para no tener que especificarlo en cada elemento. Solo puede haber un espacio de nombres por defecto dentro de un documento, los demás deberán ser declarados y utilizados explícitamente:

```
<nombreElemento xmlns="URL">
```

Luego, todos los descendientes de ese elemento serán parte del mismo espacio de nombres, a menos que se especifique lo contrario:

```
<otroElemento> texto </otroElemento>
```

Debemos saber que los espacios de nombres son muy utilizados dentro del ámbito de trabajo con documentos XML, por esta razón es importante que tengamos claro su modo de funcionamiento y la sintaxis asociada.

Inclusión de hojas de estilo

Un archivo XML puede ser vinculado a una hoja de estilos CSS, y para hacerlo contamos con la instrucción **xml-stylesheet**:

```
<?xml-stylesheet href="hoja.css" type="text/css" ?>
```

El atributo denominado **href** define la ubicación de la hoja de estilos, y en **type** se ubica el tipo **MIME** correspondiente, que es **text/css**, para identificar el tipo de medio. Las hojas de estilo que están enlazadas o importadas prevalecen unas sobre otras, en cascada, en el orden en que son llamadas.

La idea detrás de utilizar tecnologías como la popular CSS en conjunto con XML es separar y proceder a delimitar las capas de una aplicación: las reglas de negocio, el diseño y los datos. Esto, que sin dudas es un desafío para cualquier aplicación no solamente del ámbito web, provee una manera sencilla y rápida de mantener el código escrito y hacerlo más claro y comprensible.

Transformaciones

XSLT (*eXtensible Stylesheet Language for Transformations, Lenguaje eXtensible de Hojas de Estilo para Transformaciones*) permite, tomando como entrada a un documento XML, aplicarle una transformación y generar una salida personalizada para restringir la muestra solo a determinados elementos y atributos, o a valores que cumplan con un requisito en particular. Admite la ejecución de bucles, estructuras condicionales, comparaciones lógicas, ordenamiento de datos, utilización de plantillas e inclusión de archivos externos, entre otras opciones.

XSLT selecciona qué elementos y atributos se incluirán en el procesamiento mediante **XPath**, que permiten acceder a partes de un documento mediante jerarquías. Un documento XSLT tiene la extensión **xsl** y puede vincularse a uno o varios archivos XML. La aplicación de mensajería instantánea **Windows Live Messenger** de Microsoft utiliza XML para almacenar diversa información y XSLT para darles un formato de salida. Actualmente, algunos de los navegadores que dan soporte a esta

III COCOON

El framework **Cocoon** (que podemos encontrar en la dirección **http://cocoon.apache.org**) forma parte del proyecto Apache cuyo sitio wen es **www.apache.org**, y nos permite, entre otras cosas, generar transformaciones de documentos XML en el lado servidor en tiempo real, permitiendo solucionar posibles incompatibilidades entre navegadores.

tecnología son Mozilla Firefox, Netscape 8 (soporte parcial) y Microsoft Internet Explorer 6 (la versión 5 da un soporte parcial).

XML y sus tecnologías asociadas permiten estructurar la información y aplicarle transformaciones para su presentación. Hoy en día es normal que la información esté almacenada en una base de datos, se convierta a XML y luego se transforme en otra clase de documento mediante, justamente, el uso de las transformaciones.

Como CSS no puede ser utilizado para modificar la estructura de un documento XML, lo que implica que la salida dependerá de la organización de tales documentos, ni tampoco puede seleccionar qué elementos mostrar y cuáles no, la utilización de XSLT se vuelve importante. Una solución es la de transformar (mediante XSLT) documentos XML a un formato comprendido por la totalidad de los navegadores web (XHTML) y asociar al resultado de esa transformación a una hoja de estilos CSS.

Existen aplicaciones que incorporan un **procesador XSLT**, que toma como entrada un documento XML y una o varias hojas de estilo, realiza sobre sus datos una serie de tareas y devuelve como salida la información formateada según lo indicado para generar transformaciones y enviar la salida a archivos externos, por ejemplo **Xalan**. Esto puede sernos de utilidad para evitar las posibles incompatibilidades en cuanto al soporte nativo por parte de los navegadores. Podemos obtener más información acerca de este lenguaje basado en XML en los siguientes sitios:

- **www.w3.org/Style/XSL/WhatIsXSL.html**
- **www.w3.org/TR/xml-stylesheet**
- **www.w3.org/Style/XSL**

Referencia JSON

Además de XML existen otras alternativas en cuanto al formato de los datos intercambiados entre programas. JSON es una de ellas, y en este apartado incursionamos en sus características, formas de uso y ventajas.

Introducción	370
Implementaciones	371
XML y JSON	372
Sintaxis de JSON	377
JavaScript y JSON	377
Ejemplo: Listado de productos	383
con Prototype y JSON	383
Implementación PHP de JSON	389

SERVICIO DE ATENCIÓN AL LECTOR: usershop@redusers.com

INTRODUCCIÓN

JSON (*JavaScript Object Notation*, Notación en Objetos de JavaScript) es una sintaxis (o notación) para representar información. No es un programa ni un lenguaje, solo es una manera de describir y estructurar datos que eventualmente podrán ser intercambiados entre aplicaciones.

Como se mencionó en el **Capítulo 6**, el modelo Ajax propone una forma de trabajar, una arquitectura en la que intervienen ciertas tecnologías, que cumplen con determinados roles. Si bien se proponen algunas, no se especifican de manera concreta. Es posible intercambiarlas con otras siempre y cuando cumplan con las funciones que el modelo demanda. JSON tendría en el modelo la tarea de representar y estructurar el contenido de los mensajes intercambiados.

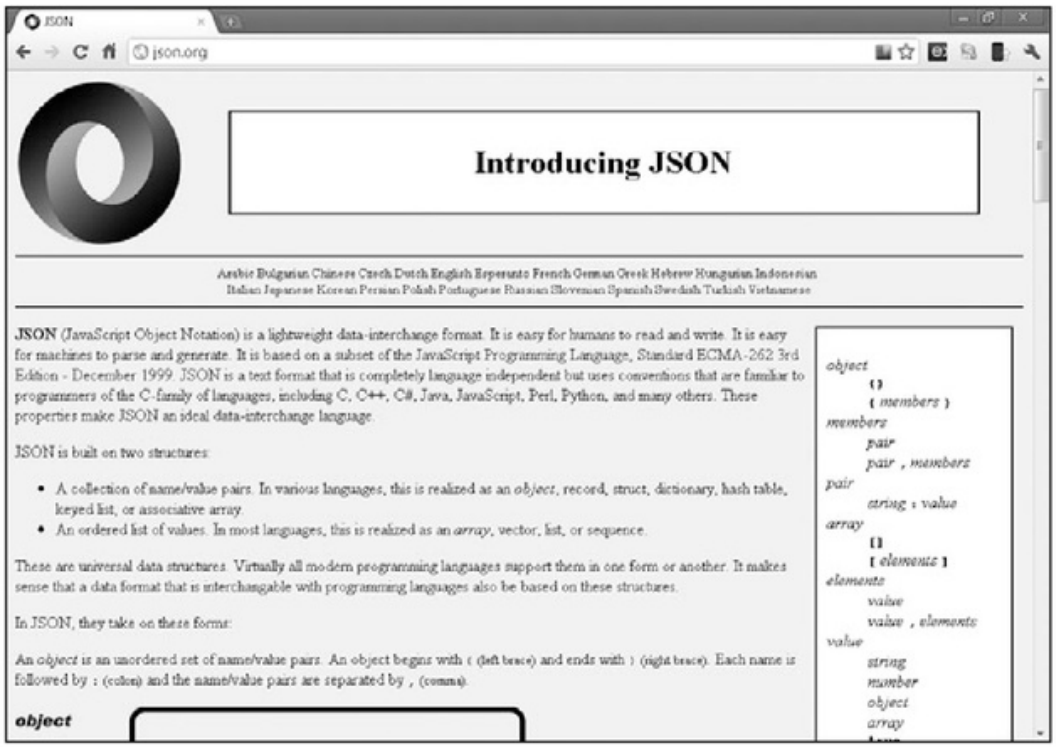


***Figura 1.** Del sitio web oficial podemos descargar implementaciones JSON para múltiples lenguajes de programación.*

Debemos saber que XML cumple originariamente el papel referido al intercambio de datos entre el cliente y el servidor, pero puede reemplazarse por cualquier formato que pueda ser interpretado por los receptores en el marco de la comunicación. Uno de estos formatos posibles es JSON.

La sintaxis utilizada tiene su origen en un subconjunto del lenguaje de programación JavaScript (según el estándar denominado **ECMA-262** 3ra edición, Diciembre 1999). Se caracteriza por ser un formato de transportación comprensible e interpretable por humanos. Su sintaxis se considera completamente independiente de cualquier lenguaje de programación y consiste en:

- pares nombre / valor.
- listas ordenadas de valores (*arrays*).

Es importante tener en cuenta que cada vez son más los servicios web que incorporan a JSON como una alternativa más a la hora de intercambiar información. En este sentido podemos mencionar a Yahoo! ya que se presenta como una de las empresas pioneras en este aspecto y podemos obtener más información al respecto en la dirección web **http://developer.yahoo.com/yui/json**.

IMPLEMENTACIONES

Existen implementaciones de la sintaxis JSON para una gran cantidad de lenguajes de programación, entre las que podemos citar:

LENGUAJE	SITIO WEB
.net	http://jayrock.berlios.de
	http://james.newtonking.com/pages/json-net.aspx
	http://www.ajaxpro.info
C / C++	http://oss.metaparadigm.com/json-c
	http://sourceforge.net/projects/jsoncpp
	http://zoolib.sourceforge.net
Cold Fusion	http://jehiah.cz
	www.bennadel.com/blog/757-Learning-ColdFusion-8-Javascript-Object-Notation-JSON-Part-I-Data-Conversion.htm
Delphi	https://sourceforge.net/projects/is-webstart
Java	www.json.org/java/index.html
	http://json-lib.sourceforge.net
	http://developer.berlios.de/projects/jsontools
	http://blog.stringtree.org/2006/08/12/json
JavaScript	www.json.org/js.html
Perl	http://search.cpan.org/dist/JSON
PHP	http://framework.zend.com/manual/en/zend.json.html
	http://mike.teczno.com/json.html
	www.aurore.net/projects/php-json
	http://pear.php.net/pepr/pepr-proposal-show.php?id=198
Python	http://undefined.org/python/#simple_json
Ruby	http://rubyforge.org/snippet/detail.php?type=snippet&id=29

***Tabla 1**. Algunas de las implementaciones JSON disponibles para los lenguajes de programación más utilizados actualmente.*

Además, JSON es soportado por **ActionScript**, **E**, **ML**, **Objective CAML**, **Rebol** y **Lua**. Para obtener un listado completo podemos recurrir al sitio oficial **www.json.org**.

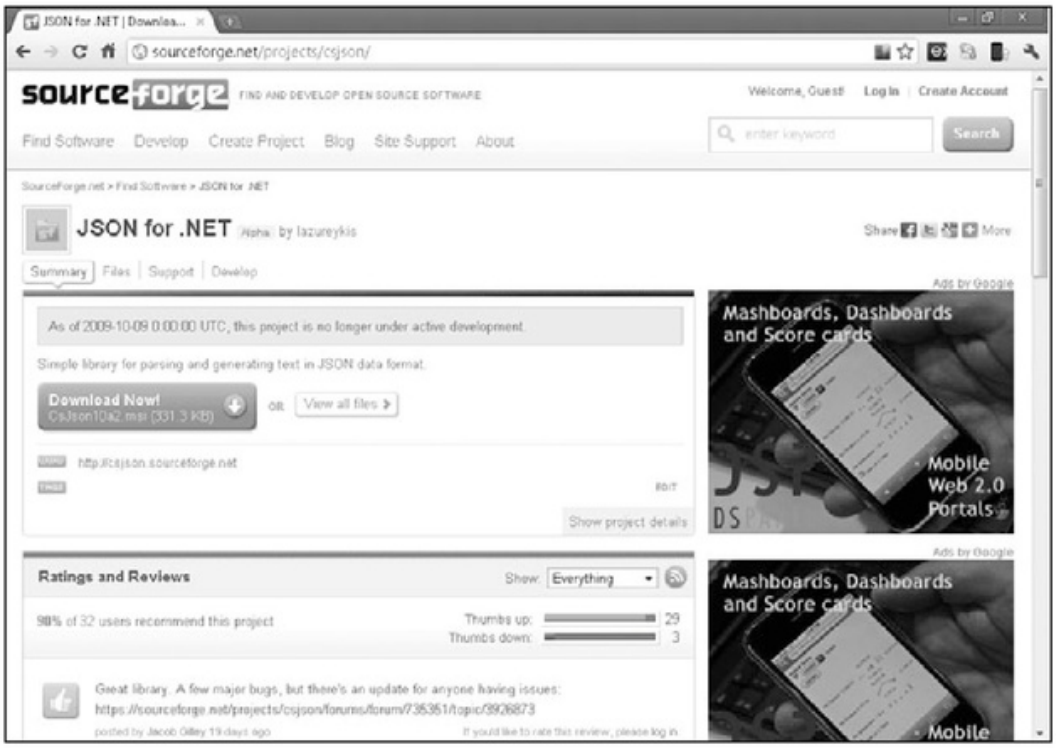


***Figura 2. ASP.Net** admite soporte JSON para implementar en aplicaciones web.*

XML Y JSON

El siguiente fragmento XML:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<mensaje>
   <autor>Ned Flanders</autor>
   <email>ned@test.com</email>
   <asunto>hola JSON</asunto>
   <contenido>este es el contenido</contenido>
   <destino>
       <autor>Homero Simpson</autor>
      <email>homero@test.com</email>
   </destino>
   <destino>
       <autor>Marge Simpson</autor>
      <email>marge@test.com</email>
```

```
    </destino>
</mensaje>
```

Luciría en JSON de la forma:

```
{
    mensaje:{
        autor:'Ned Flanders',
        email:'ned@test.com',
        asunto:'hola JSON',
        contenido:'este es el contenido',
        destino:[
            {
                autor:'Homero Simpson',
                email:'homero@test.com'
            },
            {
                autor:'Marge Simpson',
                email:'marge@test.com'
            }
        ]
    }
}
```

Si solo observamos las dos estructuras podemos decir que:

- ambas mantienen una jerarquía (elementos dentro de elementos).
- ambas están formadas por texto plano.
- ambas son intuitivas y comprensibles a simple vista (los datos que se incluyen se identifican mediante un título o un nombre de elemento).
- JSON es menos redundante, requiere menos texto y el tamaño del paquete intercambiado por las aplicaciones se reduce.
- XML en general solo está compuesto por elementos y atributos, lo que lo hace quizás más simple de comprender para aquellos que no tienen conocimientos de programación o de estructuras de datos medianamente complejas como arrays u objetos.

Al igual que XML, el transportar datos con JSON necesita que las aplicaciones emisoras y receptoras tengan las herramientas necesarias para comprender e interpretar los paquetes recibidos y generar otros nuevos para enviar.

Más información acerca de las diferencias entre JSON y XML en:

- **www.json.org/xml.html**
- **www.softwaresecretweapons.com/jspwiki/Wiki.jsp?page AJAXWithoutXML**

Hay muchas aplicaciones disponibles para transformar estructuras definidas en XML en JSON, y se puede acceder a una de ellas en:

- **http://www.thomasfrank.se/xml_to_json.html**

***Figura 3.** En Internet hay una gran cantidad de artículos acerca de cómo reemplazar a XML en cuanto al intercambio de información entre aplicaciones.*

XML es el método estándar para el intercambio de datos, pero en ocasiones no es la mejor opción. La sintaxis de un documento XML puede ser compleja y acceder a determinada información desde un lenguaje como JavaScript requiere eventualmente

III SIMILITUD

Al igual que ocurre con el metalenguaje XML, **JSON** se ha convertido en una de esas tecnologías que se ve en pequeños, medianos y grandes proyectos, sin importar el nivel de complejidad o la finalidad de la aplicación en la que se implementen las soluciones. De esta forma debemos estar seguros que se trata de una herramienta que puede ayudarnos.

mucha cantidad de código. Si bien existen analizadores XML para la mayoría de las aplicaciones, la codificación o la decodificación de fragmentos XML se vuelve un trabajo extra para el desarrollador. Ambos pueden ser interpretados por múltiples lenguajes de programación y además encajan en el modelo propuesto por Ajax.

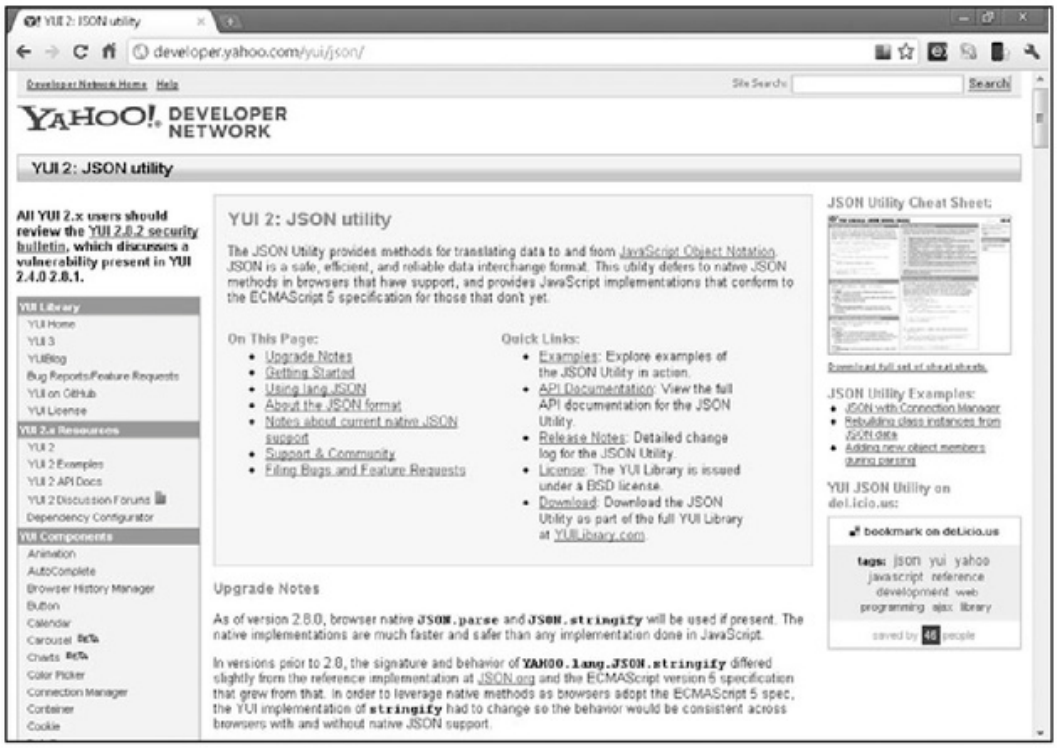


Figura 4. *Cada vez más servicios web utilizan JSON, como es el caso de los ofrecidos por Yahoo!.*

Veamos otro ejemplo comparativo. Versión XML:

```
<a a1="a1" a2="a2">
  <b>
    <c c1="c1" funcion="f_c1()"/>
    <c c2="c2" funcion="f_c2()"/>
  </b>
</a>
```

Versión JSON:

```
{
 "a": {
  "a1": "a1",
  "a2": "a2",
```

```
  "b": {
    "c": [
      {"c1": "c1", "funcion": "f_c1()"},
      {"c2": "c2", "funcion": "f_c2()"}
    ]
  }
 }
}
```

Además de XML y JSON, que son las alternativas más populares, existen otras como **YAML** (*YAML Ain't Markup Language*), que es una versión más completa y compleja que JSON. **Ruby** es uno de los lenguajes de programación que utiliza YAML. Para obtener más información ingresando en **www.yaml.org**.

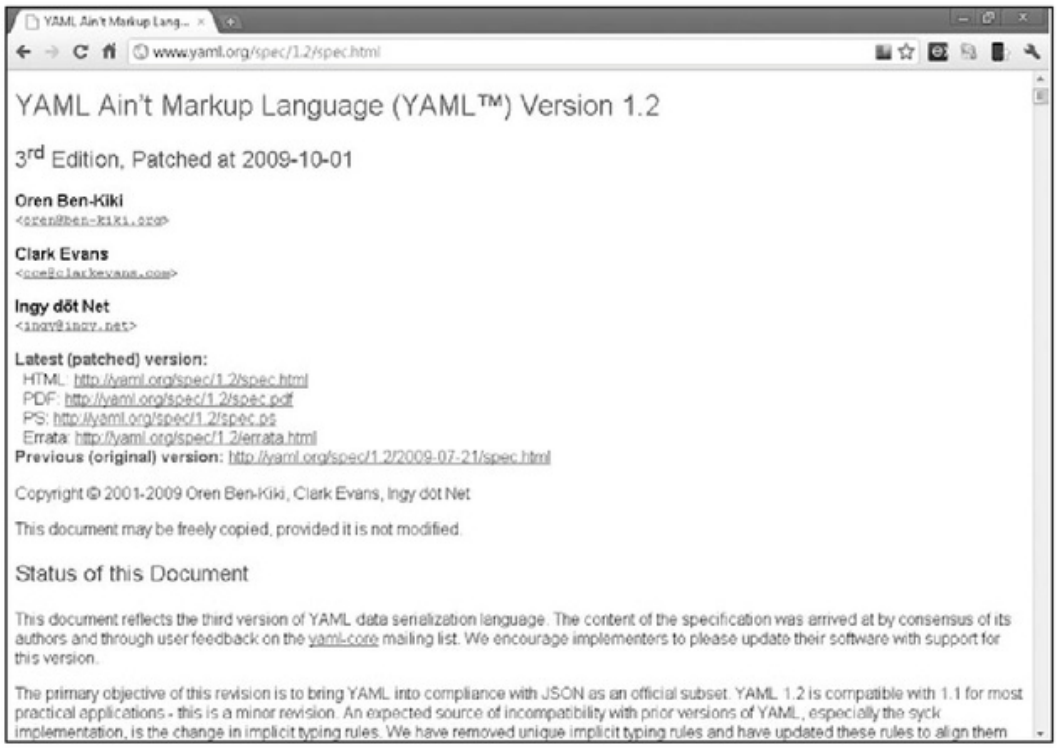


***Figura 5.** En los últimos tiempos, **YAML** se ha convertido en una aternativa a JSON.*

III FIREFOX

Es importante tener en cuenta que el navegador Mozilla Firefox nos brinda una extensión desarrollada para que los usuarios puedan visualizar archivos JSON de una forma agradable a la vista. Podemos descargar e instalar JSONView desde **https://addons.mozilla.org/en-US/firefox/addon/10869**.

SINTAXIS DE JSON

La estructura propuesta por JSON es un objeto que contiene otros objetos, arrays, cadenas de caracteres, números, booleanos (verdadero o falso) o valores nulos. En XML es posible utilizar cualquier nombre para identificar elementos si se siguen ciertas reglas como no incluir espacios o símbolos, mientras que en JSON no es posible utilizar palabras reservadas del lenguaje JavaScript.
Un documento JSON cuenta con algunos símbolos que delimitan partes con significado diferente, como analizamos a continuación:

- las llaves ({}) actúan como contenedores, que definen jerarquías que luego permitirán acceder a partes de la estructura para recuperar un dato en particular.
- los corchetes ([]) definen arrays, que usualmente se utilizan para representar elementos repetitivos. Un array puede contener a su vez otros arrays, objetos, cadenas de caracteres, números, booleanos (verdadero / falso) o valores nulos.
- los dos puntos (**:**) separan los pares nombre / valor.
- las comas (**,**) se encargan de separar los elementos de un array, en caso de haber más de uno, y los pares nombre / valor.

JAVASCRIPT Y JSON

A pesar de que su nombre lo contradiga, JSON no necesita de JavaScript para funcionar: la referencia se debe a que utiliza la notación literal de objetos del lenguaje, algo así como un subconjunto de éste.
Debemos tener en cuenta que según la forma tradicional, los objetos en JavaScript podrían definirse de la formaque vemos a cotinuación:

```
var objeto = new Object();
objeto.nombre = "Peter";
objeto.obtenerNombre = function() {
                           return this.nombre;
                       }
```

Con la forma alternativa mediante la notación literal, o sea JSON:

```
var objeto = {
nombre: "Peter",
```

```
obtenerNombre: function() {
                    return this.nombre;
                  }
}
```

Dada esta característica, una estructura JSON puede ser analizada en JavaScript con la función **eval**, que genera un objeto desde la respuesta obtenida del servidor.
Uno de los mayores beneficios al utilizar JSON se visualiza en el tratamiento de los datos por parte de una aplicación una vez obtenida la respuesta. **JavaScript es el lenguaje por excelencia utilizado en Ajax**, y JSON encaja perfectamente ya que con unos pocos pasos la información queda disponible de una forma intuitiva. Esto evita el proceso de interpretación y conversión requerido al utilizar XML.
En el sitio oficial de JSON (**www.json.org**) se distribuye una clase incluida en el archivo **json.js** que permite convertir desde este formato y hacia éste.
Si bien provee ciertas facilidades de uso, podemos observar que de todas maneras se continúa utilizando la función **eval** para la interpretación de las estructuras:

```
j = eval('(' + this + ')');
```

Dos de los métodos disponibles sobre un objeto de tipo JSON son **toJSONString** y **parseJSON**. Veamos las características de funcionamiento de cada uno:

- **toJSONString** se aplica sobre objetos (incluye como caso particular a los arrays) y devuelve su estructura y valores correspondientes en formato JSON.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> JSON - Ejemplo con objetos </title>
  <script type="text/javascript" language="javascript"
     src="json.js"></script>
 </head>
 <body>
<script type="text/javascript">
//<![CDATA[
```

```
function x() {
  this.a = 10;
  this.b = 20;
  this.c = new y();
  this.d = new Array('j', 's', 'on');
}

function y() {
  this.f = 11;
  this.g = 12;
  this.h = 13;
}

x1 = new x();
x1.b = 'veinte';

alert(x1.toJSONString());

//]]>
</script>

 </body>
</html>
```

La salida sería:

```
{
  "a":10,
  "b":"veinte",
  "c":{
```

III INTERPRETACIÓN

Es importante mencionar que un **parseador** es una pieza de código que intenta **leer** un documento, **recorrerlo**, e **interpretar** su contenido. Debemos tener en cuenta que JavaScript puede cumplir este papel al intentar evaluar el contenido de un documento en formato JSON, destacándose por su facilidad y la velocidad que alcanza.

```
        "f":11,
        "g":12,

        "h":13
        },
   "d":["j","s","on"]
}
```

En el ejemplo anterior hemos trabajado con objetos a través de JSON. En el ejemplo a continuación trabajamos con arrays:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> JSON - Ejemplo con arrays </title>
  <script type="text/javascript" language="javascript"
     src="json.js"></script>
 </head>
 <body>

<script type="text/javascript">
//<![CDATA[

function x() {
  this.w = 10;
  this.z = 20;
}

a = new Array(1, 2, new Array(11, 22));
b = new x();
c = new Array('j', 's', 'o', 'n', a, b);

alert(c.toJSONString());

//]]>
</script>
```

```
 </body>
</html>
```

La salida sería:

```
[
  "j",
  "s",
  "o",
  "n",
  [1,2,[11,22]],
  {
    "w":10,
    "z":20
  }
]
```

- por su parte, el método **parseJSON** realiza la operación inversa, es decir a partir de una estructura en formato JSON devuelve un objeto JavaScript:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> JSON - Ejemplo </title>
  <script type="text/javascript" language="javascript"
     src="json.js"></script>
 </head>
 <body>

<script type="text/javascript">
//<![CDATA[

a = new Array('j', 's', 'o', 'n');
b = a.toJSONString();
c = b.parseJSON();
```

```
alert(c[0]);
alert(c[1]);
alert(c[2]);
alert(c[3]);

//]]>
</script>

 </body>
</html>
```

La salida sería:

```
j
s
o
n
```

Otro ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> JSON - Ejemplo </title>
  <script type="text/javascript" language="javascript"
      src="json.js"></script>
 </head>
 <body>

<script type="text/javascript">
//<![CDATA[

function p() {
     this.z = 10;
}
```

```
a = new p();
b = a.toJSONString();
c = b.parseJSON();

alert(c.z);

//]]>
</script>

 </body>
</html>
```

La salida sería:

```
10
```

Estos métodos vistos hasta ahora pueden sernos de gran utilidad para tratar los datos obtenidos en formato JSON.

EJEMPLO: LISTADO DE PRODUCTOS CON PROTOTYPE Y JSON

En el siguiente ejemplo a través del objeto **XMLHttpRequest** recuperamos una respuesta en formato JSON que contiene un listado de productos. Cada uno de estos productos dispone de un nombre, una marca, un modelo y un precio. A continuación utilizaremos Prototype para realizar las peticiones y recuperar las respuestas.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title> titulo del documento </title>
  <script type="text/javascript" language="javascript"
```

```
    src="prototype.js"></script>
</head>
<body>

<script type="text/javascript">
//<![CDATA[

Event.observe(window, 'load', realizarPeticion, true);

function realizarPeticion() {
 url = 'productos.txt';
 var peticion = new Ajax.Request(
  url,
  {
  onComplete: recibirRespuesta
  }
 );
}

function recibirRespuesta(respuesta) {
 var salida = '';
 var json = eval('(' + respuesta.responseText + ')');
 var cantidad = json.productos.informatica.length;
 for (var c=0; c<cantidad; c++) {
  var item = json.productos.informatica[c];
  producto = item.producto;
  marca = item.marca;
  modelo = item.modelo;
  precio = item.precio;
  salida += producto + ' ' + marca + ' ' + modelo + ' ' + '($' +
     precio + ') <br />';
```

III OPCIONES

Es importante mencionar que se podría decir que JSON es un competidor directo del lenguaje XML puesto que cumple un papel similar dentro del contexto de la implementación de una aplicación. En este sentido, ninguno es marcadamente mejor que el otro, es solo una cuestión de preferencias personales a la hora de trabajar en desarrollo.

```
  }

  $('contenedor').innerHTML = salida;
 }

//]]>
</script>

</head>
<body>
  <div id="contenedor"></div> <br />

</body>
</html>
```

Inicialmente llamamos a la función **realizarPeticion**:

```
Event.observe(window, 'load', realizarPeticion, true);
```

Creamos una instancia del objeto **XMLHttpRequest** con Prototype y realizamos una petición al documento **productos.txt**:

```
function realizarPeticion() {
  url = 'productos.txt';
  var peticion = new Ajax.Request(
   url,
   {
   onComplete: recibirRespuesta
   }
  );
 }
```

La salida del archivo **productos.txt** es un texto plano en formato JSON, pero que bien podría haberse generado de forma dinámica:

```
{
    productos:{
```

```
        informatica:[
            {
                producto:'Netbook',
                marca:'Dell',
                modelo:'Inspiron Mini 1018',
                precio:299.99
            },
            {
                producto:'Laptop',
                marca:'Dell',
                modelo:'Inspiron 15',
                precio:499.99
            },
            {
                producto:'Laptop',
                marca:'Dell',
                modelo:'Inspiron 15R',
                precio:549.99
            }
        ]
    }
}
```

Su equivalente en XML sería:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<productos>
   <informatica>
      <producto>Netbook</producto>
      <marca>Dell</marca>
      <modelo>Inspiron Mini 1018</modelo>
      <precio>299.99</precio>
   </informatica>
   <informatica>
      <producto>Laptop</producto>
      <marca>Dell</marca>
      <modelo>Inspiron 15</modelo>
      <precio>499.99</precio>
   </informatica>
```

```
    <informatica>
       <producto>Laptop</producto>
       <marca>Dell</marca>
       <modelo>Inspiron 15R</modelo>
       <precio>549.99</precio>
    </informatica>
</productos>a>
</productos>
```

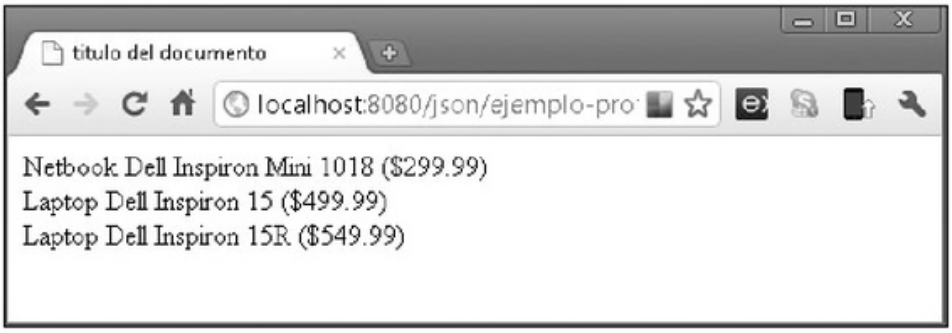


Figura 6. *En este ejemplo tomamos la información de un documento JSON y la mostramos en la página con JavaScript.*

Una vez que haya sido respondida la petición se procede a invocar a la función denominada **recibirRespuesta**, cuya definición es la siguiente:

```
function recibirRespuesta(respuesta) {
  var salida = '';
  var json = eval('(' + respuesta.responseText + ')');
  var cantidad = json.productos.informatica.length;
  for (var c=0; c<cantidad; c++) {
   var item = json.productos.informatica[c];
   producto = item.producto;
   marca = item.marca;
```

III VER ARCHIVOS JSON

Es importante que tengamos en cuenta que así como los documentos XML pueden ser visualizados directamente a través del uso de un navegador web, es importante mencionar que no ocurre lo mismo con los archivos en formato JSON, los cuales necesitan un procesamiento previo por parte de una aplicación que lea su contenido y lo interprete.

```
    modelo = item.modelo;
    precio = item.precio;
    salida += producto + ' ' + marca + ' ' + modelo + ' ' + '($' +
       precio + ') <br />';
  }
```

Con la función **eval** transformamos el texto recuperado en un objeto, desde el cual podremos acceder a las distintas partes disponibles en la respuesta:

```
var json = eval('(' + respuesta.responseText + ')');
```

En el documento JSON el elemento **informatica** está definido como array, por lo que para saber la cantidad de posiciones de las que dispone utilizamos el método **length**:

```
var cantidad = json.productos.informatica.length;
```

Para acceder al primer producto:

```
var item = json.productos.informatica[0];
```

Para acceder al nombre del modelo del primer producto:

```
json.productos.informatica[0].modelo;
```

Como podemos darnos cuenta, todo se almacena en la variable denominada **salida**, que posteriormente se le asigna como contenido al elemento llamado **contenedor**, a continuación vemos el código implicado:

III EVALUAR CÓDIGO

La función denominada **eval** forma parte de la distribución base de JavaScript y permite, justamente, realizar el proceso de **evaluar** como código una cadena de caracteres, pudiendo incorporar cualquier clase de sentencias que luego podrían ser incluidas y procesadas en lo que resta del script. De esta forma puede convertirse en fundamental para nosotros.

```
$('contenedor').innerHTML = salida;
```

```
<div id="contenedor"></div>
```

IMPLEMENTACIÓN PHP DE JSON

Es importante tener en cuenta que como se mencionó al principio del capítulo, tenemos a nuestra disposición diversas implementaciones de JSON, las cuales están desarrolladas para distintos lenguajes de programación. Una de las implementaciones JSON disponibles para PHP es la llamada **PHP-JSON**, cuya clase principal incorpora dos métodos básicos: **encode** y **decode**.

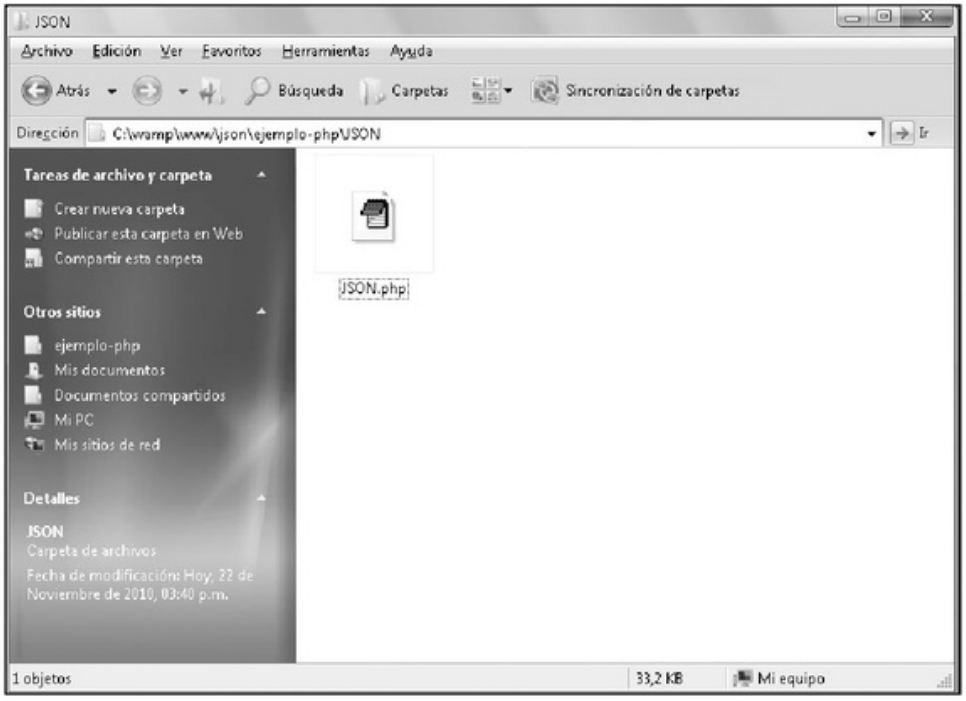


***Figura 7.** Para trabajar **PHP-JSON** deberemos incluir el archivo **JSON.php** en nuestros proyectos.*

Con la sintaxis **encode** se codifican las estructuras generadas en PHP y se obtiene el equivalente en JSON.

- Ejemplo 01. Codificación - Arrays

```
<?php
```

```
include ‘JSON/JSON.php’;
$json = new Services_JSON();
$array = Array(‘cero’, ‘uno’, ‘dos’, ‘tres’);
$json = $json->encode($array);
// [“cero”,”uno”,”dos”,”tres”]

?>
```

- Ejemplo 02. Codificación - Objetos

```
<?php


class ejemplo {
  var $a;
  var $b;
  var $c;

  function ejemplo() {
    $this->a = 10;
    $this->b = 11;
    $this->c = $this->a - $this->b;
  }
}

$x = new ejemplo();
$x->b = $x->a + $x->c;
include 'JSON/JSON.php';
$json = new Services_JSON();
$objeto = $x;
```

EVAL EN PHP

Es importante mencionar que dentro del lenguaje de programación PHP también contamos con una función denominada **eval**, en esta ocasión se encarga de cumplir una tarea similar a la de la función disponible dentro del lenguaje JavaScript, es decir, esta función nos permite **evaluar** el contenido de una cadena de caracteres que indiquemos.

```
$json = $json->encode($objeto);

?>
```

Con **decode** se realiza el proceso inverso: desde una notación JSON se obtiene una estructura comprensible por el lenguaje.

• Ejemplo 03. Decodificación

```
<?php

$var  = "{                                                    ";
$var .= "    xxx:{                                  ";
$var .= "        yyy:{                              ";
$var .= "            zzz:[                          ";
$var .= "                'abc',          ";
$var .= "                'def',          ";
$var .= "                'ghi'           ";
$var .= "            ]                              ";
$var .= "        }                                  ";
$var .= "    }                                                ";
$var .= "}                                                    ";

include 'JSON/JSON.php';
$json = new Services_JSON();
$json = $json->decode($var);

/*

estructura de $json en PHP:

stdClass Object
(
  [xxx] => stdClass Object
    (
     [yyy] => stdClass Object
       (
           [zzz] => Array
           (
```

```
                [0] => abc
                [1] => def
                [2] => ghi
            )
        )
    )
)

*/

echo $json->xxx->yyy->zzz[1]; //salida: def

?>
```

Podemos obtener más información acerca de esta clase en el sitio oficial, el cual se encuentra en la dirección **http://mike. teczno.com/json.html**.

Servicios al lector

En este último apartado encontraremos un índice temático con las palabras más importantes de la obra. Esto nos servirá para ubicar con mayor facilidad el contenido que necesitemos.

ÍNDICE TEMÁTICO

A

Accept-encoding 185
Adobe Dreamweaver 36
Ajax 45, 114, 127
Alt 27, 270
Alternate StyleSheet 65
Apache 182
Applet 59
Array 121, 129
Asincronía 154
ASP.Net 45, 176

B

Background 27, 75, 151, 331
Border 72, 333
Bottom 71, 336
BR 59
Break 293
Breakpoint 225
Button 27

C

C# 176
Case sensitive 29
CDATA 48, 49
CDN 191
Center 40
Charset 35, 88
Class 47, 313
Cliente HTTP 182
Content-encoding 185
Content-type 65, 186, 193
Continue 293
CSS 27, 49, 64

D

Debugging 222
Description 195
DHTML 46, 154, 172
DIAL 59
DirectoryIndex 201
Div 46, 92, 154, 300, 314
Do while 292
Doctype 43
DOM 114, 121, 155, 172, 299
DTD 26, 350

E

Each 118
ECMA 16, 280, 368
Else if 291
Else 291
Encoding 34, 349
Enumerable 122
ErrorDocument 200, 206
Estándares 16, 44
Eval 378

F

Field 126
Flash 132, 134, 269, 276, 362
Font 24, 40, 74, 329
For in 292
For 292
Form 127
Form.element 127
FPI 357
Frame 28, 29, 45, 56, 193
Frameset 29

G

GIF 134
Global 287
Google Web Toolkit 174
Googlebot 273
Gzip 184, 214, 244

H

Head 31, 35, 36, 50, 65, 66, 70, 191
Herencia 75
Hexadecimal 77, 87, 324
Hojas de estilo 36, 49
HR 59
HREF 59, 365
HTACCESS 197
HTML 24, 30
HTTP 213
Httpd.conf 186, 201
Http-equiv 35

I

Id 32, 33, 47, 115, 124, 173, 312
If 289
Image 147
Img 27, 59, 142, 315
Include 59, 119
Inline 69, 317
InnerHTML 47
Input 28
IP 182, 190, 202

J

Java 174
JavaScriptPacker 108
JPEG 133
JS 48, 97, 100, 281
JScript 161, 280
JSMin 103
JSON 154, 160, 174, 368
JSP 45, 111

K

Keys 119
Keywords 195, 271

L

Label 28
Lang 30
Left 71, 336
Link 65
List-style 75, 334

M

Margin 335
MathML 25, 38
Media 65, 70
Meta 36, 57, 59, 191, 268
Minify 97, 106
Mod_deflate 185
Mod_gzip 185
Mod_gzip_item_include 188

N

Name 32
Nodos 299
Noscript 281
Number 113

O

Object 59, 113
ObjectRange 123
OnClick 155, 223, 303
OnLoad 148, 303
OnMouseOver 147, 155, 223, 303
Operadores 286
Optimizar 15

P

Pack 110
Padding 72, 335
PageRank 272
Parser 37, 345, 355, 377
Pautas 18
PERL 45
PHP 45, 101, 176
PNG 136
Position 337
Print 65
Profile 36

Q

Query string 205
Quirk mode 43

R

Redirect 203
Reducción 41
Rel 65
Related 275
Request 169
Return 297
RGB 77, 87, 324
Right 71, 337
Robots 196, 270, 273
RollOver 148
Ruby 172, 241, 374

S

Sajax 177
Scalable Vector Graphic 38, 141
Screen 65
Script 48, 57, 59, 96
Section 58
Select 28
Servidor HTTP 182
SGML 67, 340
Sincronía 154
Sitemap 86, 269, 273, 276
Span 46, 92, 154
Src 59, 142, 147
Srctype 59
SSI 45
Standalone 349
Standard mode 43
Status 169
String 113, 125
Style 24, 57, 59, 66, 69, 317
StyleSheet 65
Switch 293

T

Table 27, 55, 92, 331
Td 27, 92
Templates 45
Textarea 28
Th 92
Title 31, 36, 65, 270, 315
Top 71, 337
Tr 27, 92
Transparencia 135, 136
Type 48, 65, 365

U

Updater 173
URL 182, 205
UTF-8 35, 349

V

Validar 15, 36, 88
Var 283, 297
VBScript 161
Visual Basic.Net 176

W

W3C 16, 24, 37, 58, 64
Web 2.0 156
While 292

X

XForms 59
XFrames 46, 59
XHTML 24, 46, 154
XML Schemas 354
XML 30, 154
XMLHTTP 163
XMLHttpRequest 154, 159, 168, 381
Xmlns 30
XSL 25, 27
XSLT 154, 365

CLAVES PARA COMPRAR
UN LIBRO DE COMPUTACIÓN

1 SOBRE EL AUTOR Y LA EDITORIAL

Revise que haya un cuadro "sobre el autor", en el que se informe sobre su experiencia en el tema. En cuanto a la editorial, es conveniente que sea especializada en computación.

2 PRESTE ATENCIÓN AL DISEÑO

Compruebe que el libro tenga guías visuales, explicaciones paso a paso, recuadros con información adicional y gran cantidad de pantallas. Su lectura será más ágil y atractiva que la de un libro de puro texto.

3 COMPARE PRECIOS

Suele haber grandes diferencias de precio entre libros del mismo tema; si no tiene el valor en tapa, pregunte y compare.

4 ¿TIENE VALORES AGREGADOS?

Desde un sitio exclusivo en la Red hasta un CD-ROM, desde un Servicio de Atención al Lector hasta la posibilidad de leer el sumario en la Web para evaluar con tranquilidad la compra, o la presencia de adecuados índices temáticos, todo suma al valor de un buen libro.

5 VERIFIQUE EL IDIOMA

No sólo el del texto; también revise que las pantallas incluidas en el libro estén en el mismo idioma del programa que usted utiliza.

6 REVISE LA FECHA DE PUBLICACIÓN

Está en letra pequeña en las primeras páginas; si es un libro traducido, la que vale es la fecha de la edición original.

usershop.redusers.com
VISITE NUESTRO SITIO WEB

» Vea información más detallada sobre cada libro de este catálogo.
» Obtenga un capítulo gratuito para evaluar la posible compra de un ejemplar.
» Conozca qué opinaron otros lectores.
» Compre los libros sin moverse de su casa y con importantes descuentos.
» Publique su comentario sobre el libro que leyó.
» Manténgase informado acerca de las últimas novedades y los próximos lanzamientos.

TAMBIÉN PUEDE CONSEGUIR NUESTROS LIBROS EN KIOSCOS O PUESTOS DE PERIÓDICOS, LIBRERÍAS, CADENAS COMERCIALES, SUPERMERCADOS Y CASAS DE COMPUTACIÓN.

LLEGAMOS A TODO EL MUNDO VÍA OCA * Y DHL **

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

usershop.redusers.com

Photoshop

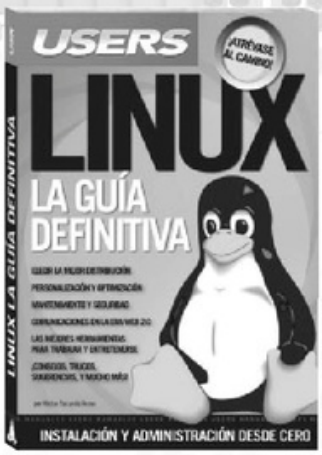
En este libro aprenderemos sobre las más novedosas técnicas de edición de imágenes en Photoshop. El autor nos presenta de manera clara y práctica todos los conceptos necesarios, desde la captura digital hasta las más avanzadas técnicas de retoque.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-05-3

Grabación y producción de música

En este libro repasaremos todos los aspectos del complejo mundo de la producción musical. Desde las cuestiones para tener en cuenta al momento de la composición, hasta la mezcla y el masterizado, así como la distribución final del producto.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-04-6



Linux

Este libro es una completa guía para migrar e iniciarse en el fascinante mundo del software libre. En su interior, el lector conocerá las características de Linux, desde su instalación hasta las opciones de entretenimiento, con todas las ventajas de seguridad que ofrece el sistema.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-8-6

Premiere + After Effects

Esta obra nos presenta un recorrido detallado por las aplicaciones audiovisuales de Adobe: Premiere Pro, After Effects y Soundbooth. Todas las técnicas de los profesionales, desde la captura de video hasta la creación de efectos, explicadas de forma teórica y práctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-9-3

Office 2010

En este libro aprenderemos a utilizar todas las aplicaciones de la suite, en su versión 2010. Además, su autora nos mostrará las novedades más importantes, desde los minigráficos de Excel hasta Office Web Apps, todo presentado en un libro único.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-26013-6-2

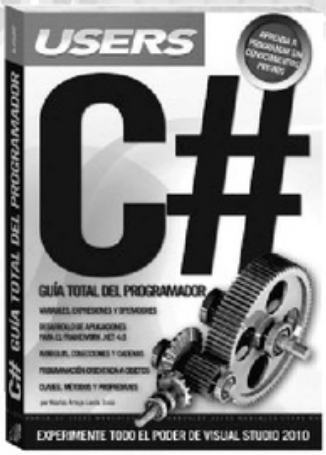
Excel Paso a Paso

En esta obra encontraremos una increíble selección de proyectos pensada para aprender, mediante la práctica, la forma de agilizar todas las tareas diarias. Todas las actividades son desarrolladas en procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-4-8

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



C#

Este libro es un completo curso de programación con C# actualizado a la versión 4.0. Ideal tanto para quienes desean migrar a este potente lenguaje, como para quienes quieran aprender a programar desde cero en Visual Studio 2010.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-26013-5-5

200 Respuestas Seguridad

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos contestar para conseguir un equipo seguro. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-1-7

Funciones en Excel

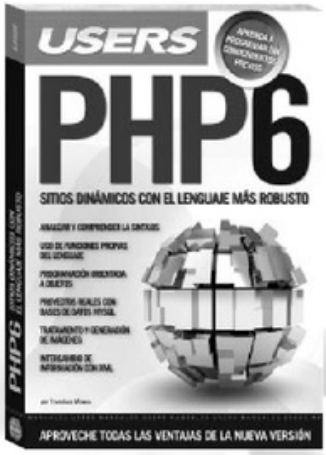
Este libro es una guía práctica de uso y aplicación de todas las funciones de la planilla de cálculo de Microsoft. Desde las funciones de siempre hasta las más complejas, todas presentadas a través de ejemplos prácticos y reales.

→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-26013-0-0

Proyectos con Windows 7

En esta obra aprenderemos cómo aprovechar al máximo todas las ventajas que ofrece la PC. Desde cómo participar en las redes sociales hasta las formas de montar una oficina virtual, todo presentado en 120 proyectos únicos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-036-8



PHP 6

Este libro es un completo curso de programación en PHP en su versión 6.0. Un lenguaje que se destaca tanto por su versatilidad como por el respaldo de una amplia comunidad de desarrolladores, que lo convierten en un punto de partida ideal para quienes comienzan a programar.

→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-663-039-9

200 Respuestas: Blogs

Esta obra es una completa guía que responde a las preguntas más frecuentes de la gente sobre la forma de publicación más poderosa de la Web 2.0. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-663-037-5

USERS PRESENTA...

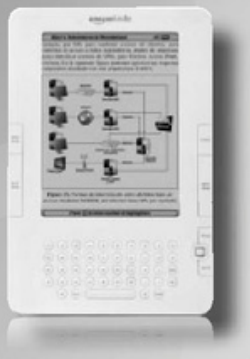
¡EL PRIMER EBOOK USERS!

Sí, ya podés leer Hackers al descubierto en tu PC, notebook, Amazon Kindle, iPad, en el celular...

CONSEGUILO DESDE CUALQUIER PARTE DEL MUNDO

A UN PRECIO INCREÍBLE

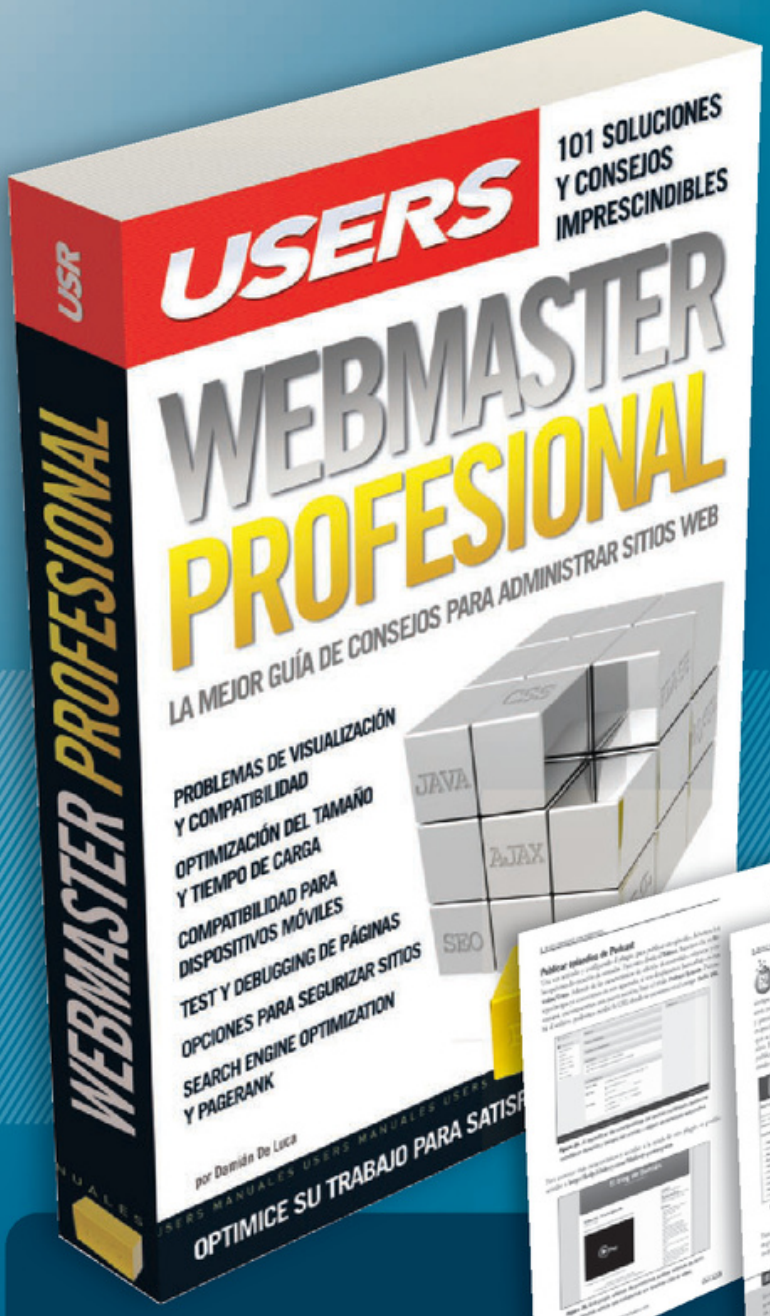
¿QUÉ ESTÁS ESPERANDO?



¡LEELO DONDE QUIERAS!

INGRESA YA A USERSHOP.REDUSERS.COM Y ENTERATE MÁS

LOS MEJORES CONSEJOS DE LOS EXPERTOS PARA ADMINISTRAR SITIOS WEB



Esta obra explica cómo superar los problemas más frecuentes y complejos que enfrenta todo administrador de sitios web. Ideal para quienes necesiten conocer las tendencias actuales y las tecnologías en desarrollo que son materia obligada para dominar la Web 2.0.

» INTERNET / HOME / DESARROLLO
» 336 PÁGINAS
» ISBN 978-987-663-011-5

LLEGAMOS A TODO EL MUNDO VÍA OCA* Y DHL**

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

USERS

CONTENIDO

1 | CLAVES PARA LA OPTIMIZACIÓN DE SITIOS
Conceptos generales / Importancia de los estándares / Reglas de optimización / Herramientas / Conocimientos requeridos

2 | XHTML
Origen e historia / Tipos de documentos / Reglas de marcado / Juegos de caracteres / HTML Tidy / Técnicas de reducción / Inclusión de JavaScript y CSS / Elementos disponibles / Módulos en XHTML / XHTML 2.0

3 | CSS
Inclusión de documentos / Escritura reducida / Agrupación de estilos / Colores RGB / Imágenes en conjunto / Validación de documentos / Compatibilidad entre navegadores / Diseño sin tablas / Opciones adicionales

4 | JAVASCRIPT
Ubicación de los scripts / Unificación de archivos / Reducción de documentos / Prototype

5 | IMÁGENES
Situación y necesidades / Tipos de compresión / Tratamiento de imágenes / Precarga

6 | AJAX
Características y tecnologías / Principios / El objeto XMLHttpRequest / Compatibilidad / Desventajas del modelo tradicional / Implementación con Prototype / Disminución del tiempo de respuesta y uso de cache / AJAX y los lenguajes de programación / JavaScript

7 | HTTP
Contenido comprimido / Reducción de peticiones / Contenido distribuido / Cabeceras / El archivo .HTACCESS / Herramientas de control

8 | HERRAMIENTAS COMPLEMENTARIAS
Introducción / Firebug / Web Developer / Web Inspector / Opera Dragonfly / XRAY / DebugBar / Companion.JS / Internet Explorer Developer Toolbar

9 | POSICIONAMIENTO DE SITIOS EN GOOGLE
Herramientas disponibles / Sitios de calidad / Rastreo / Googlebot / Índices / Sitemaps

APÉNDICE A | REFERENCIA JAVASCRIPT
APÉNDICE B | REFERENCIA CSS 2
APÉNDICE C | REFERENCIA XML
APÉNDICE D | REFERENCIA JSON

NIVEL DE USUARIO

PRINCIPIANTE | INTERMEDIO | AVANZADO | EXPERTO

DESARROLLADOR WEB

XHTML + CSS JAVASCRIPT + AJAX

Esta obra es una guía de referencia útil y versátil, tanto para programadores como para diseñadores o entusiastas con alguna experiencia en el armado de sitios web, que deseen obtener resultados profesionales en sus desarrollos.

En sus capítulos, repasaremos conceptos y técnicas que permitirán mejorar, de manera sustancial, la experiencia final de los usuarios en los tiempos de carga y en el modo en que acceden a la información y navegan el sitio.

Aprenderemos la manera de incrementar la presencia en la Web y de optimizar el posicionamiento en Google, y recibiremos los mejores consejos para aplicar a las principales tecnologías: XHTML, CSS, JavaScript y AJAX.

El autor, Francisco Minera, reconocido desarrollador que cuenta con varios libros sobre el tema en esta editorial, muestra en estas páginas el resultado de años de experiencia en el trabajo con pequeñas y medianas empresas.

Versión completamente mejorada, actualizada y ampliada de la obra *Desarrollo web profesional*

RedUSERS.com

En este sitio encontrará una gran variedad de recursos y software relacionado, que le servirán como complemento al contenido del libro. Además, tendrá la posibilidad de estar en contacto con los editores y de participar del foro de lectores, en donde podrá intercambiar opiniones y experiencias.

Si desea más información sobre el libro, puede comunicarse con nuestro Servicio de Atención al Lector: usershop@redusers.com

WEB DEVELOPER

In this book you will find the most important developer tools and techniques for web site improvements. Every developer's real world needs are fulfilled in this comprehensive XHTML, CSS, JavaScript and AJAX guide.

ISBN 978-987-1773-09-1

MANUALES USERS MANUALES USERS MANUAL

SOLUCIONES PROFESIONALES Y EFICIENTES