



UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Córdoba

Diplomatura en Desarrollo WEB
Módulo: Introdúcete al mundo SCRIPT

UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Córdoba

Secretaría de Extensión Universitaria

Área de Educación a Distancia

Coordinador General de Educación a Distancia:

Magíster Leandro D. Torres

Curso:

DIPLOMATURA EN DESARROLLO WEB EV 15030

Módulo:

“Introdúcete al mundo SCRIPT”

Autor:

Ing. Ricardo Martín Espeche

Referencias

Este material se confeccionó en base a las últimas tendencias que hoy se encuentra en Internet. Para eso se buscó el mejor material y se recompiló dicha información para facilitar el aprendizaje del mismo.

El material utilizado para esta primera unidad fue:

- Bootstrap 3, el manual oficial (Mark Otto, Jacob Thornton)
- Manual de JQuery (Miguel Angel Alvarez)
- Fundamentos de JQuery (Rebecca Murphey)

Índice

Capítulo 3	8
Bootstrap.....	8
3.1 Introducción básica	8
3.1.1 Ventajas de Bootstrap	8
3.1.2 Contenidos de Bootstrap	9
3.1.3 La primera plantilla Bootstrap	10
3.1.4 Desactivar el diseño Responsive	11
3.2 Diseño con rejilla	13
3.2.1 Antes de empezar	13
3.2.2 Tipos de rejillas	14
3.2.3 Reseteando columnas.....	20
3.2.4 Desplazando columnas	21
3.2.5 Añadiendo columnas.....	21
3.2.6 Reordenando las columnas.....	22
3.3 Tipografía	23
3.3.1 Titulares	23
3.3.2 Texto	24
3.3.3 Énfasis	25
3.3.4 Clases CSS.....	26
3.3.5 Abreviaturas	27
3.3.6 Direcciones	28
3.3.7 Blockquotes.....	28
3.3.8 Listas.....	30
3.3.9 Código.....	34
3.4 Elementos CSS	35
Tutor Ricardo Martín Espeche	

3.4.1 Tablas	35
3.4.2 Imágenes	39
3.4.3 Utilidades	40
3.5 Formularios.....	46
3.5.1 Formulario básico.....	46
3.5.2 Formulario en línea	47
3.5.3 Formularios horizontales	48
3.5.4 Campos de formulario	49
3.5.5 Estados de formulario.....	53
3.5.6 Redimensionando campos de formulario.....	56
3.5.7 Mensajes de ayuda	57
3.5.8 Botones	57
3.6 Componentes	63
3.6.1. Iconos (<i>glyphicons</i>).....	63
3.6.2 Menús desplegables.....	65
3.6.3 Grupos de botones	67
3.6.4 Botones desplegables	70
3.6.5 Grupos de campos de formulario.....	75
3.6.6 Elementos de navegación	81
3.6.7 Barras de navegación.....	84
3.6.8 Migas de pan.....	90
3.6.9 Paginadores	90
3.6.10 Etiquetas	94
3.6.11 Badges	95
3.6.12 Jumbotron	96
3.6.13 Encabezado de página.....	96

3.6.14 Imágenes en miniatura	97
3.6.15 Mensajes de alerta	99
3.6.16 Barras de progreso.....	101
3.6.17 Objetos multimedia.....	104
3.6.18 Listas de elementos.....	106
3.6.19 Paneles	108
3.6.20 Pozos	113
Capítulo 4	114
JavaScript & JQuery.....	114
4.1 Introducción básica	114
4.1.1 Primeros pasos	114
4.2 Elementos básicos.....	117
4.2.1 Comentarios	117
4.2.2 Literales.....	117
4.2.3 Sentencias y bloques	118
4.3 Tipo de datos	119
4.3.1 Variables	119
4.3.2 Referencias	119
4.3.3 Vectores y matrices	120
4.4 Operadores.....	122
4.4.1 Operadores aritméticos	122
4.4.2 Operadores de comparación	123
4.4.3 Operadores lógicos	124
4.4.4 Operadores de asignación	124
4.4.5 Operadores especiales.....	124
4.5 Estructuras de control	126

4.5.1 Bifurcaciones condicionales	126
4.5.2 Bucles	127
4.5.3 Estructuras de manejo de objetos	128
4.6 Funciones	129
4.6.1 Funciones predefinidas	130
4.7 Objetos	132
4.7.1 Propiedades y métodos	132
4.7.2 Creación mediante constructores	132
4.7.3 Herencia	134
4.7.4 Polimorfismo	134
4.8 Objetos predefinidos	136
4.8.1 Objeto Array	136
4.8.2 Objeto Date	137
4.8.3 Objeto Math	138
4.8.4 Objeto Number	139
4.8.5 Objeto String	139
4.8.6 Objeto RegExp	140
4.8.7 Objeto Boolean	140
4.8.8 Objeto Function	140
4.9 Eventos	140
4.9.1 Lista de eventos	141
4.9.2 Definición mediante código	142
4.10 Modelo de objetos del documento	144
Capítulo 5	145
jQuery	145
5.1 Introducción básica	145

5.1.1 Ventajas de jQuery con respecto a otras alternativas	145
5.2 Conceptos básicos.....	146
5.2.1 Descarga la última versión del framework	146
5.2.2 Crear una página HTML simple.....	146
5.2.3 Ejecutar código cuando la página ha sido cargada.....	147
5.2.4 Selección de elementos	148
5.2.5 Comprobar Selecciones	149
5.2.6 Guardar Selecciones.....	149
5.2.7 Refinamiento y Filtrado de Selecciones.....	150
5.2.8 Selección de Elementos de un Formulario	150
5.2.9 Obtenedores (Getters) & Establecedores (Setters)	151
5.2.10 CSS, Estilos, & Dimensiones.....	152
5.2.11 Utilizar Clases para Aplicar Estilos CSS	152
5.2.12 Dimensiones.....	153
5.2.13 Atributos	153
5.2.14 Recorrer el DOM	153
5.2.15 Mover, Copiar y Remover Elementos	154
5.2.16 Crear Nuevos Elementos	155
5.2.17 Manipulación de Atributos	156
5.3 El núcleo de JQuery.....	158
5.3.1 \$ vs \$()	158
5.3.2 Métodos Utilitarios.....	158
5.3.3 Comprobación de Tipos	160
5.3.4 El Método Data.....	161
5.4 Eventos.....	162
5.4.1 Introducción.....	162

5.4.2 Vincular Eventos a Elementos	162
5.4.4 Vinculación de Múltiples Eventos	163
5.4.5 El Objeto del Evento.....	163
5.4.6 Funciones Auxiliares de Eventos.....	165
5.5 Efectos.....	166
5.5.1 Introducción.....	166
5.5.2 Efectos Incorporados en la Biblioteca.....	166
5.5.3 Cambiar la Duración de los Efectos.....	167
5.5.4 Realizar una Acción Cuando un Efecto fue Ejecutado.....	167
5.5.5 Control de los Efectos	168
5.6 AJAX.....	169
5.6.1 Introducción.....	169
5.6.2 Conceptos Clave	169
5.6.3 Métodos Ajax de jQuery	172

Capítulo 3

Bootstrap

3.1 Introducción básica

Bootstrap es un framework Responsive Design desarrollado por Mark Otto y Jacob Thornton de Twitter y está diseñado para ayudar a diseñadores y desarrolladores en el proceso de creación de sitios web y aplicaciones de una manera rápida y fácil controlando la parte front de los sitios.

Bootstrap es uno de los proyectos de código abierto que ha causado gran impacto en los diseñadores web del mundo en los últimos años y muestra de eso lo podemos ver en GitHub, donde aún sustenta el primer lugar como el proyecto más popular de este repositorio donde los mas grande del mundo del diseño y desarrollo web implementan sus grandes y novedosos proyectos.

Bootstrap 3 se construyó basado y pensado para la web móvil “mobile-first approach” o enfoque móvil de primera, donde se prioriza la web moderna, la web que se puede ver correctamente en cualquier dispositivo, desde pantallas pequeñas con poca resolución hasta dispositivos con resoluciones grandes y pantallas de gran tamaño.B.

3.1.1 Ventajas de Bootstrap

Bueno la mayor ventaja de utilizar Twitter Bootstrap consiste en el conjunto de herramientas y componentes de interfaz que trae este sistema para crear diseños responsivos, flexibles o fijas sin tener que hacer muchos Esfuerzos.

- Porque podemos crear componentes de interfaz avanzados como Scrollspy y Typeaheads sin tener que escribir una sola línea de código JavaScript.
- Porque sin duda ahorraremos mucho tiempo y esfuerzo al momento del diseño web con Bootstrap, ya que podemos utilizar un montón de cosas ya predefinidas en este sistema, y cosas que son útiles en cualquier proyecto web.
- Por sus características responsive – El uso de Bootstrap nos garantiza el poder crear diseños responsivos o sensibles fácilmente. Este framework posee características de respuestas que hacen que sus diseños aparezcan de la forma más apropiada en los diferentes dispositivos y resoluciones de pantallas que están en el mercado.
- Porque con Bootstrap podremos crear diseños coherentes ya que todos los componentes comparten la misma base y los mismos estilos de diseño a través de una librería central, por lo que los diseños son consistentes a lo largo del desarrollo.

- por su fácil uso – Bootstrap es muy fácil de usar, cualquier persona con un conocimiento básico de HTML y CSS se puede iniciar en el desarrollo con Twitter Bootstrap y obtener grandes y buenos resultados.
- Por la compatibilidad con los diferentes navegadores – Twitter Bootstrap se crea en base a los diferentes navegadores modernos como Firefox, Chrome, Safari, Opera y Internet Explorer y con cierto soporte a los no tan modernos (IE 8+).
- Porque es totalmente gratuito – este gran framework es open source así que podemos descargarlo y usarlo libremente.

3.1.2 Contenidos de Bootstrap

Bootstrap se puede descargar de dos maneras, compilado o mediante el código fuente original. Dependiendo de la forma que hayas elegido, verás una estructura de directorios u otra. En esta sección se muestran los detalles de cada una de ellas.

3.1.2.1 Contenidos de la versión compilada de Bootstrap

Después de descomprimir el archivo que te has descargado con la versión compilada de Bootstrap, verás la siguiente estructura de archivos y directorios:

```
bootstrap/  
├─ css/  
│   ├─ bootstrap.css  
│   ├─ bootstrap.min.css  
│   ├─ bootstrap-theme.css  
│   └─ bootstrap-theme.min.css  
├─ js/  
│   ├─ bootstrap.js  
│   └─ bootstrap.min.js  
└─ fonts/  
    ├─ glyphsicons-halflings-regular.eot  
    ├─ glyphsicons-halflings-regular.svg  
    ├─ glyphsicons-halflings-regular.ttf  
    └─ glyphsicons-halflings-regular.woff
```

Estos archivos son la forma más sencilla de utilizar Bootstrap en cualquier proyecto web. Para cada archivo se ofrecen dos variantes: los archivos compilados (cuyo nombre es bootstrap.*) y los archivos compilados + comprimidos (cuyo nombre es bootstrap.min.*). También se incluyen las fuentes de los iconos del proyecto Glyphicons y el tema opcional de Bootstrap.

3.1.2.2 Contenidos de la versión original de Bootstrap

La versión original de Bootstrap incluye, además de las versiones compiladas de los archivos CSS y JavaScript, las versiones originales de esos mismos archivos y toda la documentación. Tras descomprimir el archivo que te has descargado con la versión original de Bootstrap, verás la siguiente estructura de archivos y directorios:

```
bootstrap/  
├─ less/  
├─ js/  
├─ fonts/  
├─ dist/  
│   ├─ css/  
│   ├─ js/  
│   └─ fonts/  
└─ docs/  
    └─ examples/
```

Los directorios less/, js/ y fonts/ contienen el código fuente utilizado para generar los archivos CSS, JavaScript y las fuentes. El directorio dist/ contiene los mismos archivos que se han mostrado en la sección anterior para la versión compilada de Bootstrap. El directorio docs/ incluye el código fuente de la documentación de Bootstrap y un directorio llamado examples/ con varios ejemplos de uso. El resto de archivos están relacionados con otros temas secundarios, como por ejemplo las licencias del código.

3.1.3 La primera plantilla Bootstrap

El siguiente código HTML muestra una plantilla muy sencilla creada con Bootstrap:

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="utf-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <title>Plantilla básica de Bootstrap</title>
```

```
<!-- CSS de Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet" media="screen">

<!-- librerías opcionales que activan el soporte de HTML5 para IE8 -->
<!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
    <script
src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->
</head>
<body>
    <h1>¡Hola mundo!</h1>

    <!-- Librería jQuery requerida por los plugins de JavaScript -->
    <script src="http://code.jquery.com/jquery.js"></script>

    <!-- Todos los plugins JavaScript de Bootstrap (también puedes
    incluir archivos JavaScript individuales de los únicos
    plugins que utilices) -->
    <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

3.1.4 Desactivar el diseño Responsive

¿No quieres que tu sitio o aplicación web modifique su aspecto según el dispositivo utilizado por el usuario? En ese caso, y con un poco de esfuerzo, puedes desactivar las características responsive de Bootstrap para que los usuarios con móvil vean el mismo sitio que los usuarios con un navegador de escritorio.

Aplica los siguientes pasos para desactivar el diseño responsive de Bootstrap y echa un vistazo al ejemplo que se muestra después:

1. Elimina de tus páginas (o simplemente no añadas) la etiqueta `<meta name="viewport" ... >` que se explica en los próximos capítulos.
2. Elimina la propiedad `max-width` de todos los elementos `.container` aplicando el estilo `max-width: none !important;` y estableciendo una anchura normal con `width: 970px;`. Obviamente, debes añadir estos estilos después del CSS por defecto aplicado por Bootstrap.

-
3. Si utilizas menús y barras de navegación, debes eliminar todos los estilos que hacen que se compriman en dispositivos pequeños. Como este cambio es enorme, será mejor que veas los estilos CSS del ejemplo que se muestra más adelante.
 4. Utiliza las clases `.col-xs-*` (`xs` = extra-small) para tus diseños basados en rejillas en vez de las otras clases para dispositivos medianos o grandes. No te preocupes porque estas clases `.col-xs-*` escalan bien para cualquier resolución.

En cualquier caso, si utilizas IE8 debes seguir cargando la librería Respond.js, ya que las media queries siguen siendo necesarias a pesar de los cambios anteriores.

3.2 Diseño con rejilla

Antes de comenzar a diseñar el layout o estructura de contenidos de las páginas, es necesario realizar algunos preparativos importantes.

3.2.1 Antes de empezar

Bootstrap utiliza algunos elementos HTML y algunas propiedades CSS que requieren el uso del *doctype* de HTML5. No olvides incluir este *doctype* en todas tus páginas con el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
  ...
</html>
```

3.2.1.1 El móvil es lo más importante

Bootstrap 2 incluía algunas utilidades para hacer que las páginas se adaptaran a los dispositivos móviles. Bootstrap 3 se ha creado desde cero pensando en los móviles. Así que en vez de incluir algunos estilos opcionales para móviles, todo eso ya está incluido en el propio Bootstrap. Por eso nos gusta decir que **para Bootstrap 3, los dispositivos móviles son lo más importante**.

Para que las páginas se muestren correctamente y el zoom funcione bien en los dispositivos móviles, es importante que añadas la siguiente etiqueta dentro de la cabecera `<head>` de las páginas:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Si quieres deshabilitar el zoom en tus páginas, añade la propiedad `user-scalable=no` a la etiqueta anterior:

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no">
```

Al añadir la propiedad `user-scalable=no`, los usuarios ya no podrán hacer zoom en la página y solamente podrán hacer *scroll* en sus contenidos. El resultado es que el comportamiento de la página se parece más al de una aplicación móvil nativa. En cualquier caso, limitar las libertades de los usuarios puede ser contraproducente y por tanto, no te recomendamos que utilices esta opción en todos tus sitios.

3.2.1.2 Imágenes responsive

Bootstrap 3 ya no adapta el tamaño de las imágenes automáticamente como sucedía en Bootstrap 2. Para mantener el mismo comportamiento de antes, debes añadir la clase `.img-responsive` a cada imagen que quieras que se comporte de manera *responsive*. Esta clase incluye las propiedades `max-width: 100%;` y `height: auto;` para que la imagen escale en función del tamaño del elemento en el que se encuentra.

```

```

3.2.1.3 Centrando los contenidos de la página

Si quieres centrar una página respecto a la ventana del navegador, encierra sus contenidos dentro de un elemento y aplícale la clase `.container`:

```
<div class="container">  
  ...  
</div>
```

El ancho del contenedor varía en cada punto de ruptura del diseño para adaptarse a la rejilla. Los contenedores no se pueden anidar debido a la propiedad `padding` y a su anchura fija.

3.2.2 Tipos de rejillas

Bootstrap incluye una rejilla o retícula fluida pensada para móviles y que cumple con el diseño web *responsive*. Esta retícula crece hasta 12 columnas a medida que crece el tamaño de la pantalla del dispositivo. Bootstrap incluye clases CSS para utilizar la rejilla directamente en tus diseños.

3.2.2.1 Introducción

El diseño de páginas basado en rejilla se realiza mediante filas y columnas donde se colocan los contenidos. Así funciona la rejilla de Bootstrap:

- Las filas siempre se definen dentro de un contenedor de tipo `.container` (anchura fija) o de tipo `.container-fluid` (anchura variable). De esta forma las filas se alinean bien y muestran el `padding` correcto.
- Las filas se utilizan para agrupar horizontalmente a varias columnas.
- El contenido siempre se coloca dentro de las columnas, ya que las filas sólo deberían contener como *hijos* elementos de tipo columna.

- Bootstrap define muchas clases CSS (como por ejemplo `.row` y `.col-xs-4`) para crear rejillas rápidamente. También existen *mixins* de Less para crear diseños más semánticos.
- La separación entre columnas se realiza aplicando `padding`. Para contrarrestar sus efectos en la primera y última columnas, las filas (elementos `.row`) aplican márgenes negativos.
- Las columnas de la rejilla definen su anchura especificando cuántas de las 12 columnas de la fila ocupan. Si por ejemplo quieres dividir una fila en tres columnas iguales, utilizarías la clase `.col-xs-4` (el 4 indica que cada columna ocupa 4 de las 12 columnas en las que se divide cada fila).

NOTA Si quieres crear un diseño totalmente fluido que ocupe toda la anchura del navegador, deberías encerrar las rejillas dentro de un elemento al que apliques los estilos `padding: 0 15px;`. De esta forma se pueden neutralizar los márgenes `margin: 0 -15px;` que se aplican a los elementos `.row`.

3.2.2.2 Media queries

Bootstrap utiliza las siguientes *media queries* para establecer los diferentes puntos de ruptura en los que la rejilla se transforma para adaptarse a cada dispositivo.

```
/* Dispositivos muy pequeños (teléfonos de hasta 768px de anchura) */
/* No se define ninguna media query porque este es el estilo por
   defecto utilizado por Bootstrap 3 */

/* Dispositivos pequeños (tablets, anchura mayor o igual a 768px) */
@media (min-width: @screen-sm-min) { ... }

/* Dispositivos medianos (ordenadores, anchura mayor o igual a 992px) */
@media (min-width: @screen-md-min) { ... }

/* Dispositivos grandes (ordenadores, anchura mayor o igual a 1200px) */
@media (min-width: @screen-lg-min) { ... }
```

En ocasiones, también se utilizan las siguientes *media queries* que definen la propiedad `max-width` y permiten restringir los dispositivos a los que se aplican los estilos CSS:

```
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }
@media (min-width: @screen-lg-min) { ... }
```

3.2.2.3 Características de cada rejilla

La siguiente tabla muestra las características de la rejilla de Bootstrap en los diferentes tipos de dispositivos.

	Dispositivos muy pequeños Teléfonos (<768px)	Dispositivos pequeños Tablets (≥768px)	Dispositivos medianos Ordenadores (≥992px)	Dispositivos grandes Ordenadores (≥1200px)
Comportamiento	Las columnas se muestran siempre horizontalmente.	Si se estrecha el navegador, las columnas se muestran verticalmente. A medida que aumenta su anchura, la rejilla muestra su aspecto horizontal normal.		
Anchura máxima del contenedor	Ninguna (auto)	728px	940px	1170px
Prefijo de las clases CSS	.col-xs-	.col-sm-	.col-md-	.col-lg-
Número de columnas	12			
Anchura máxima de columna	auto	60px	78px	95px
Separación	30px (15px a cada lado de la columna)			

entre columnas		
¿Permite anidación?	Si	
¿Permite desplazar columnas?	No	Si
¿Permite reordenación de columnas?	No	Si

3.2.2.4 Ejemplo de rejilla creada con Bootstrap

El siguiente ejemplo muestra cómo crear una rejilla con las clases `.col-md-*`. En los dispositivos móviles (*extra pequeño* o *pequeño*) esta rejilla se muestra verticalmente, pero en un ordenador (*medio* o *grande*) se ve horizontalmente.

```
<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
```

```
</div>

<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>

<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>

<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

3.2.2.5 Ejemplo de contenedor de ancho variable

Si quieres transformar una rejilla de anchura fija en una rejilla de anchura variable que ocupa toda la anchura del navegador, reemplaza la clase CSS `.container` por `.container-fluid` en el elemento que encierra a todos los demás elementos de la rejilla:

```
<div class="container-fluid">
  <div class="row">
    ...
  </div>
</div>
```

3.2.2.6 Ejemplo de rejilla para móviles y ordenadores

Si no quieres que las columnas de la rejilla se muestren verticalmente en los dispositivos pequeños, utiliza a la vez las clases `.col-xs-*` y `.col-md-*`, tal y como muestra el siguiente ejemplo.

```
<!-- En los móviles las columnas se muestran verticalmente porque
una de ellas ocupa toda la anchura del dispositivo y la otra
columna ocupa la mitad -->

<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

```
<!-- En un móvil las columnas ocupan la mitad del dispositivo y en un
ordenador ocupan la tercera parte de la anchura disponible -->
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Las columnas ocupan siempre la mitad de la pantalla, tanto en un
móvil como en un ordenador de escritorio -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

3.2.2.7 Ejemplo de rejilla para móviles, tablets y ordenadores

A partir del ejemplo anterior, puedes hacer que el *layout* sea todavía más dinámico añadiendo las clases `.col-sm-*` pensadas para tablets:

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-sm-6 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<div class="row">
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>

  <!-- Código opcional para limpiar las columnas XS en caso de que el
  contenido de todas las columnas no coincida en altura -->
  <div class="clearfix visible-xs"></div>
  <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
</div>
```

3.2.3 Reseteando columnas

Como las rejillas de Bootstrap tienen cuatro puntos de ruptura en los que las columnas se reordenan, es casi seguro que te vas a encontrar con problemas cuando las columnas tengan diferente altura. Para solucionarlo, utiliza la clase `clearfix` combinándola con alguna de las clases auxiliares tipo `visible-xs`:

```
<div class="row">
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>

  <!-- La clase 'clearfix' sólo se aplica cuando el dispositivo sea
        muy pequeño, tal y como indica la clase 'visible-xs' -->
  <div class="clearfix visible-xs"></div>

  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
  <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
</div>
```

También es posible que en ocasiones necesites resetear los desplazamientos de las columnas. Las clases que resetean estos valores sólo están disponibles para los dispositivos medianos y grandes, que los desplazamientos sólo funcionan en esos dispositivos.

```
<div class="row">
  <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
  <div class="col-sm-5 col-sm-offset-2 col-md-6 col-md-offset-0">.col-sm-5 .col-sm-
offset-2 .col-md-6 .col-md-offset-0</div>
</div>

<div class="row">
  <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-6</div>
  <div class="col-sm-6 col-md-5 col-md-offset-2 col-lg-6 col-lg-offset-0">.col-sm-6
.col-md-5 .col-md-offset-2 .col-lg-6 .col-lg-offset-0</div>
</div>
```

3.2.4 Desplazando columnas

Añade la clase `.col-md-offset-*` para desplazar cualquier columna hacia su derecha. Estas clases aumentan el tamaño del margen izquierdo de la columna en una cantidad equivalente a esas `*` columnas. La clase `.col-md-offset-4` por ejemplo desplaza la columna una anchura equivalente a 4 columnas.

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 col-md-offset-4">.col-md-4 .col-md-offset-4</div>
</div>
```

```
<div class="row">
  <div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>
  <div class="col-md-3 col-md-offset-3">.col-md-3 .col-md-offset-3</div>
</div>
```

```
<div class="row">
  <div class="col-md-6 col-md-offset-3">.col-md-6 .col-md-offset-3</div>
</div>
```

3.2.5 Añadiendo columnas

Bootstrap 3 también permite anidar columnas dentro de otras columnas. Para ello, dentro de una columna con la clase `col-md-*` crea un nuevo elemento con la clase `.row` y añade una o más columnas con la clase `.col-md-*`. Las columnas anidadas siempre tienen que sumar 12 columnas de anchura, tal y como muestra el siguiente ejemplo.

```
<div class="row">
  <div class="col-md-9">
    Level 1: .col-md-9
    <div class="row">
      <div class="col-md-6">
        Level 2: .col-md-6
      </div>
      <div class="col-md-6">
        Level 2: .col-md-6
      </div>
    </div>
  </div>
```

```
</div>  
</div>
```

3.2.6 Reordenando las columnas

Bootstrap 3 introduce la posibilidad de reordenar las columnas para cambiar su posición, lo que es muy importante para los diseños web *responsive*. Añade las clases `.col-md-push-*` y `.col-md-pull-*` para reordenar las columnas.

```
<div class="row">  
  <div class="col-md-9 col-md-push-3">.col-md-9 .col-md-push-3</div>  
  <div class="col-md-3 col-md-pull-9">.col-md-3 .col-md-pull-9</div>  
</div>
```

3.3 Tipografía

Los estilos relacionados con la tipografía y el texto de los contenidos son esenciales en cualquier *framework* CSS. Por esa razón, Bootstrap 3 incluye decenas de estilos para los principales elementos utilizados en los sitios y aplicaciones web.

3.3.1 Titulares

Bootstrap 3 define estilos por defecto para todos los niveles de titulares de las páginas, desde `<h1>` hasta `<h6>`. Ejemplo:

```
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<h4>h4. Bootstrap heading</h4>
<h5>h5. Bootstrap heading</h5>
<h6>h6. Bootstrap heading</h6>
```

Bootstrap también define estilos especiales para los elementos `<small>` incluidos dentro de un titular y utilizados habitualmente para mostrar información secundaria. Ejemplo:

```
<h1>h1. Bootstrap heading <small>Secondary text</small></h1>
<h2>h2. Bootstrap heading <small>Secondary text</small></h2>
<h3>h3. Bootstrap heading <small>Secondary text</small></h3>
<h4>h4. Bootstrap heading <small>Secondary text</small></h4>
<h5>h5. Bootstrap heading <small>Secondary text</small></h5>
<h6>h6. Bootstrap heading <small>Secondary text</small></h6>
```

Así se ve este ejemplo en tu navegador:

h1. Bootstrap heading Secondary text

h2. Bootstrap heading Secondary text

h3. Bootstrap heading Secondary text

h4. Bootstrap heading Secondary text

h5. Bootstrap heading Secondary text

h6. Bootstrap heading Secondary text

3.3.2 Texto

El tamaño de letra (**font-size**) por defecto de Bootstrap 3 es **14px** y el interlineado (**line-height**) es **1.428**. Estos valores se aplican tanto al **<body>** como a todos los párrafos. Estos últimos también incluyen un margen inferior cuyo valor es la mitad que su interlineado (**10px** por defecto).

Ejemplo:

```
<p>Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque  
penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor  
id nibh ultricies vehicula.</p>
```

```
<p>Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus  
mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non  
commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec  
ullamcorper nulla non metus auctor fringilla.</p>
```

```
<p>Maecenas sed diam eget risus varius blandit sit amet non magna. Donec id elit non  
mi porta gravida at eget metus. Duis mollis, est non commodo luctus, nisi erat  
porttitor ligula, eget lacinia odio sem nec elit.</p>
```

Así se ve este ejemplo en tu navegador:

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ullamcorper nulla non metus auctor fringilla. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.

Maecenas sed diam eget risus varius blandit sit amet non magna. Donec id elit non mi porta gravida at eget metus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit.

3.3.2.1 Texto destacado

Para destacar un párrafo sobre los demás, añade la clase `.lead`. Ejemplo:

```
<p class="lead">Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.</p>
```

Así se ve este ejemplo en tu navegador:

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.

3.3.3 Énfasis

Bootstrap 3 permite añadir énfasis a determinadas partes del texto mediante las etiquetas HTML habituales.

3.3.3.1 Texto poco importante

Para marcar una parte del texto o todo un bloque de texto como poco importante, utiliza la etiqueta `<small>`. Bootstrap muestra ese contenido con un tamaño de letra igual al 85% del tamaño de letra de su elemento padre. En el caso de los titulares (`<h1>`, ..., `<h6>`) el tamaño de los elementos `<small>` se ajusta de otra forma para que tengan un mejor aspecto.

Ejemplo:

```
<p><small>Este contenido de texto se verá más pequeño y puede ser utilizado por ejemplo para "la letra pequeña" de la página.</small></p>
```

Así se ve este ejemplo en tu navegador:

Este contenido de texto se verá más pequeño y puede ser utilizado por ejemplo para "la letra pequeña" de la página.

3.3.3.2 Texto en negrita

Utiliza la etiqueta `<bold>` para añadir énfasis a un texto mostrándolo en negrita.

Ejemplo:

```
<p>El siguiente trozo de texto <strong>se muestra en negrita</strong>.</p>
```

Así se ve este ejemplo en tu navegador:

El siguiente trozo de texto **se muestra en negrita.**

3.3.3.3 Texto en cursiva

Utiliza la etiqueta `` para añadir énfasis a un texto mostrándolo en cursiva. Este énfasis es de menor importancia que el definido por la etiqueta ``.

Ejemplo:

```
<p>El siguiente trozo de texto <em>se muestra en cursiva</em>.</p>
```

Así se ve este ejemplo en tu navegador:

El siguiente trozo de texto *se muestra en cursiva.*

3.3.3.4 Etiquetas alternativas

Cuando creas páginas HTML5, también puedes emplear las etiquetas `` y `<i>`. La etiqueta `` se emplea para destacar palabras o frases sin darles énfasis o importancia. La etiqueta `<i>` se emplea sobre todo para marcar términos técnicos, etc.

3.3.4 Clases CSS

3.3.4.1 Clases CSS para alinear texto

Bootstrap 3 define varias clases CSS para alinear de diferentes formas el contenido de texto de los elementos.

Ejemplo:

```
<p class="text-left">Texto alineado a la izquierda.</p>
<p class="text-center">Texto centrado.</p>
<p class="text-right">Texto alineado a la derecha.</p>
```

3.3.4.2 Clases CSS para indicar el tipo de contenido

Aunque no es una práctica recomendada desde el punto de vista de la accesibilidad, Bootstrap 3 también define varias clases CSS para mostrar el tipo de contenido a través del color del texto.

Ejemplo:

```
<p class="text-muted">Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.</p>
<p class="text-primary">Nullam id dolor id nibh ultricies vehicula ut id elit.</p>
<p class="text-success">Duis mollis, est non commodo luctus, nisi erat porttitor ligula.</p>
```

```
<p class="text-info">Maecenas sed diam eget risus varius blandit sit amet non magna.</p>
<p class="text-warning">Etiam porta sem malesuada magna mollis euismod.</p>
<p class="text-danger">Donec ullamcorper nulla non metus auctor fringilla.</p>
```

Así se ve este ejemplo en tu navegador:

Fusce dapibus, tellus ac cursus commodo, tortor mauris nibh.

Nullam id dolor id nibh ultricies vehicula ut id elit.

Duis mollis, est non commodo luctus, nisi erat porttitor ligula.

Maecenas sed diam eget risus varius blandit sit amet non magna.

Etiam porta sem malesuada magna mollis euismod.

Donec ullamcorper nulla non metus auctor fringilla.

3.3.5 Abreviaturas

Si empleas la etiqueta `<abbr>` de HTML para marcar las abreviaturas y los acrónimos, los usuarios podrán ver su contenido completo al pasar el ratón por encima de ellas. Además, si la abreviatura define el atributo `title`, Bootstrap añade un fino borde inferior punteado para indicar al usuario que puede pasar el ratón por encima (también cambia el tipo de puntero que se muestra al pasar el ratón por encima).

3.3.5.1 Abreviaturas básicas

Incluye la definición completa del elemento `<abbr>` dentro de su atributo `title`. Ejemplo:

```
<p>Este texto contiene abreviaturas como <abbr title="señor">Sr.</abbr> y <abbr title="señora">Sra.</abbr></p>
```

3.3.5.2 Iniciales

Añade la clase `.initialism` (*iniciales* en inglés) a un elemento `<abbr>` para reducir ligeramente su tamaño de letra y así hacer que el texto se lea mejor. Ejemplo:

```
<p>Esta página está escrita con <abbr title="HyperText Markup Language" class="initialism">HTML</abbr></p>
```

3.3.6 Direcciones

Utiliza la etiqueta `<address>` para mostrar la información de contacto de tu sitio o página. No olvides añadir `
` al final de cada línea para mantener el texto de la dirección bien formateado. Ejemplo:

```
<address>
  <strong>Empresa S.A.</strong><br>
  Avenida Principal 123<br>
  Ciudad, Provincia 00000<br>
  <abbr title="Phone">Tel:</abbr> 9XX 123 456
</address>

<address>
  <strong>Nombre Apellido</strong><br>
  <a href="mailto:#">nombre.apellido@ejemplo.com</a>
</address>
```

3.3.7 Blockquotes


Estos elementos se emplean para incluir en tus páginas *trozos* de contenidos de otras fuentes.

3.3.7.1 Blockquote por defecto

Utiliza el elemento `<blockquote>` para encerrar el contenido que se va a incluir en la página. También se recomienda utilizar el elemento `<p>` para encerrar el texto dentro del `<blockquote>`. Ejemplo:

```
<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.</p>
</blockquote>
```

Así se ve este ejemplo en tu navegador:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

3.3.7.2 Opciones para los elementos blockquote

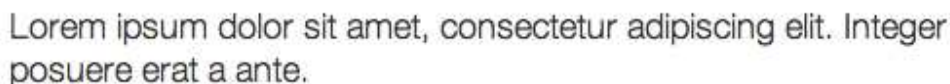
Además del estilo por defecto del elemento `<blockquote>`, Bootstrap 3 define algunos estilos adicionales para los otros elementos que se pueden añadir a los elementos `<blockquote>`.

Mostrando la fuente

Utiliza el elemento `<small>` para mostrar la fuente original del contenido y encierra el nombre concreto de la fuente (una persona, una publicación, un sitio web, etc.) con el elemento `<cite>`. Ejemplo:

```
<blockquote>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a
ante.</p>
  <small>Frase célebre de <cite title="Nombre Apellidos">Nombre
Apellidos</cite></small>
</blockquote>
```

Así se ve este ejemplo en tu navegador:



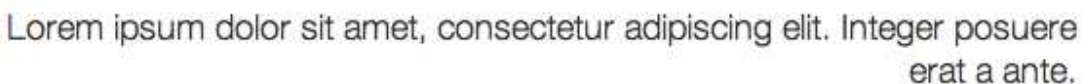
— Frase célebre de Nombre Apellidos

Modificando la alineación

Algunos autores prefieren modificar la alineación de las citas creadas con `<blockquote>` para que destaquen más sobre el resto del texto. Para ello, utiliza la clase `.pull-right`. Ejemplo:

```
<blockquote class="pull-right">
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a
ante.</p>
  <small>Frase célebre de <cite title="Nombre Apellidos">Nombre
Apellidos</cite></small>
</blockquote>
```

Así se ve este ejemplo en tu navegador:



Frase célebre de Nombre Apellidos —

3.3.8 Listas

3.3.8.1 Listas no ordenadas

Utilízalas para agrupar elementos para los que su orden no importa. Bootstrap 3 también incluye los estilos adecuados para las listas anidadas. Ejemplo:

```
<ul>
  <li>Lorem ipsum dolor sit amet</li>
  <li>Consectetur adipiscing elit</li>
  <li>Integer molestie lorem at massa</li>
  <li>Facilisis in pretium nisl aliquet</li>
  <li>Nulla volutpat aliquam velit
    <ul>
      <li>Phasellus iaculis neque</li>
      <li>Purus sodales ultricies</li>
      <li>Vestibulum laoreet porttitor sem</li>
      <li>Ac tristique libero volutpat at</li>
    </ul>
  </li>
  <li>Faucibus porta lacus fringilla vel</li>
  <li>Aenean sit amet erat nunc</li>
  <li>Eget porttitor lorem</li>
</ul>
```

Así se ve este ejemplo en tu navegador:

- Lorem ipsum dolor sit amet
- Consectetur adipiscing elit
- Integer molestie lorem at massa
- Facilisis in pretium nisl aliquet
- Nulla volutpat aliquam velit
 - Phasellus iaculis neque
 - Purus sodales ultricies
 - Vestibulum laoreet porttitor sem
 - Ac tristique libero volutpat at
- Faucibus porta lacus fringilla vel
- Aenean sit amet erat nunc
- Eget porttitor lorem

3.3.8.2 Listas ordenadas

En este caso, el orden de los elementos sí es importante y por eso se muestran numerados. Ejemplo:

```
<ol>
  <li>Lorem ipsum dolor sit amet</li>
  <li>Consectetur adipiscing elit</li>
  <li>Integer molestie lorem at massa</li>
  <li>Facilisis in pretium nisl aliquet</li>
  <li>Nulla volutpat aliquam velit</li>
  <li>Faucibus porta lacus fringilla vel</li>
  <li>Aenean sit amet erat nunc</li>
  <li>Eget porttitor lorem</li>
</ol>
```

Así se ve este ejemplo en tu navegador:

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa
4. Facilisis in pretium nisl aliquet
5. Nulla volutpat aliquam velit
6. Faucibus porta lacus fringilla vel
7. Aenean sit amet erat nunc
8. Eget porttitor lorem

3.3.8.3 Listas sin estilo

Como resulta muy habitual mostrar las listas sin viñetas y sin margen izquierdo, Bootstrap 3 incluye una clase CSS llamada `.list-unstyled` para aplicar esos estilos.

La única pega de esta clase es que sólo se aplica a los elementos de una lista y no a los elementos de sus listas anidadas. Así que si quieres aplicar los estilos a todos los elementos, añade la clase `.list-unstyled` también a las listas anidadas. Ejemplo:

```
<ul class="list-unstyled">
  <li>Lorem ipsum dolor sit amet</li>
  <li>Consectetur adipiscing elit</li>
  <li>Integer molestie lorem at massa</li>
  <li>Facilisis in pretium nisl aliquet</li>
  <li>Nulla volutpat aliquam velit
    <ul>
      <li>Phasellus iaculis neque</li>
      <li>Purus sodales ultricies</li>
      <li>Vestibulum laoreet porttitor sem</li>
```



```
<li>Ac tristique libero volutpat at</li>
</ul>
</li>
<li>Faucibus porta lacus fringilla vel</li>
<li>Aenean sit amet erat nunc</li>
<li>Eget porttitor lorem</li>
</ul>
```

Así se ve este ejemplo en tu navegador:

```

Lorem ipsum dolor sit amet
Consectetur adipiscing elit
Integer molestie lorem at massa
Facilisis in pretium nisl aliquet
Nulla volutpat aliquam velit
  ◦ Phasellus iaculis neque
  ◦ Purus sodales ultricies
  ◦ Vestibulum laoreet porttitor sem
  ◦ Ac tristique libero volutpat at
Faucibus porta lacus fringilla vel
Aenean sit amet erat nunc
Eget porttitor lorem
```

3.3.8.4 Listas en línea

También resulta habitual mostrar los elementos de una lista horizontalmente, como por ejemplo en el menú principal de navegación. Para ello, Bootstrap 3 define una clase CSS llamada `.inline-block`. Ejemplo:

```
<ul class="list-inline">
  <li>Lorem ipsum</li>
  <li>Phasellus iaculis</li>
  <li>Nulla volutpat</li>
</ul>
```

Así se ve este ejemplo en tu navegador:

```

  Lorem ipsum  Phasellus iaculis  Nulla volutpat
```

3.3.8.5 Listas de definición

No se utilizan mucho, pero Bootstrap 3 también incluye estilos por defecto para las listas de definición, formadas por pares término + definición. Ejemplo:

```
<dl>
```

```
<dt>Description lists</dt>
<dd>A description list is perfect for defining terms.</dd>
<dt>Euismod</dt>
<dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.</dd>
<dd>Donec id elit non mi porta gravida at eget metus.</dd>
<dt>Malesuada porta</dt>
<dd>Etiam porta sem malesuada magna mollis euismod.</dd>
</dl>
```

Así se ve este ejemplo en tu navegador:

Description lists
A description list is perfect for defining terms.
Euismod
Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.
Donec id elit non mi porta gravida at eget metus.
Malesuada porta
Etiam porta sem malesuada magna mollis euismod.

Listas de definición horizontales

Si lo prefieres, también es posible mostrar la lista de definición horizontalmente añadiendo la clase `.dl-horizontal`. Ejemplo:

```
<dl class="dl-horizontal">
  <dt>Description lists</dt>
  <dd>A description list is perfect for defining terms.</dd>
  <dt>Euismod</dt>
  <dd>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.</dd>
  <dd>Donec id elit non mi porta gravida at eget metus.</dd>
  <dt>Malesuada porta</dt>
  <dd>Etiam porta sem malesuada magna mollis euismod.</dd>
  <dt>Felis euismod semper eget lacinia</dt>
  <dd>Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</dd>
</dl>
```

Así se ve este ejemplo en tu navegador:

Description lists	A description list is perfect for defining terms.
Euismod	Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit. Donec id elit non mi porta gravida at eget metus.
Malesuada porta	Etiam porta sem malesuada magna mollis euismod.
Felis euismod semper...	Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

Cuando una lista de definición se muestra horizontalmente, se cortan automáticamente con **text-overflow** los términos que son demasiado largos. En los dispositivos pequeños, los términos no se cortan, pero se muestran verticalmente uno encima de otro.

3.3.9 Código

3.3.9.1 Código en línea

Utiliza el elemento `<code>` para encerrar los fragmentos de código fuente que se muestran en línea con el texto. Ejemplo:

```
<p>El elemento <code>&lt;section&gt;</code> es uno de los nuevos elementos de HTML5.</p>
```

Así se ve este ejemplo en tu navegador:

El elemento `<section>` es uno de los nuevos elementos de HTML5.

3.3.9.2 Bloques de código

Cuando quieras mostrar bloques de código, utiliza la etiqueta `<pre>` y asegúrate de reemplazar los caracteres `<` por `<` y `>` por `>` para evitar problemas al mostrar el código. Ejemplo:

```
<pre>&lt;p&gt;Así se escribe un párrafo de texto&lt;/p&gt;</pre>
```

Así se ve este ejemplo en tu navegador:

```
<p>Así se escribe un párrafo de texto</p>
```

Cuando

muestres bloques de código muy largos, quizás prefieras utilizar la clase `.pre-scrollable` para limitar su altura a un máximo de **350px** y añadir una barra de *scroll* al código.

3.4 Elementos CSS

En este capítulo se explican los estilos que define Bootstrap 3 para las tablas y las imágenes, incluyendo todas sus variantes. También se explican las classes CSS genéricas y utilidades que simplifican el diseño de los sitios web y que se pueden aplicar a cualquier elemento.

3.4.1 Tablas

3.4.1.1 Tablas básicas

Añade la clase `.table` a cualquier elemento `<table>` para aplicar los estilos básicos de Bootstrap 3 para tablas. El resultado es una tabla con un *padding* muy sutil y con líneas de separación solamente en las filas.

Puede parecer absurdo tener que añadir la clase `.table` para que se apliquen los estilos a las tablas, pero ten en cuenta que el elemento `<table>` se utiliza para muchas otras cosas que no son necesariamente tablas, como por ejemplo calendarios y selectores de fechas.

Ejemplo:

```
<table class="table">
  ...
</table>
```

Así se ve este ejemplo en tu navegador:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

3.4.1.2 Tablas *cebreadas*

Las tablas *cebreadas* son aquellas cuyas filas alternan su color de fondo para mejorar la legibilidad de los contenidos. Su aspecto recuerda a la piel de una cebra y de ahí su nombre. En inglés se denominan "*striped tables*" y por eso en Bootstrap 3 estas tablas se crean añadiendo la clase `.table-striped`. Ejemplo:

```
<table class="table table-striped">
  ...
</table>
```

Así se ve este ejemplo en tu navegador:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

ADVERTENCIA Como las tablas cebreadas utilizan el selector `:nth-child` de CSS, no funcionan en Internet Explorer 8.

3.4.1.3 Tablas con bordes

Si prefieres utilizar el estilo tradicional de las tablas con los cuatro bordes en todas las celdas y en la propia tabla, añade la clase `.table-bordered`. Ejemplo:

```
<table class="table table-bordered">  
  ...  
</table>
```

Así se ve este ejemplo en tu navegador:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

3.4.1.4 Tablas dinámicas

Para que los contenidos de la tabla sean todavía más fáciles de entender, añade la clase `.table-hover` para modificar ligeramente el aspecto de las filas cuando el usuario pasa el ratón por encima de ellas (sólo funciona para las filas dentro de `<tbody>`).

Ejemplo:

```
<table class="table table-hover">  
  ...  
</table>
```

Así se ve este ejemplo en tu navegador (pasa el ratón por encima de las filas para ver el efecto):

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

3.4.1.5 Tablas condensadas

Cuando una tabla es muy grande o cuando tienes muchas tablas en una misma página, puede ser interesante mostrar sus contenidos de forma más compacta. Añade la clase `.table-condensed` a tus tablas y el *padding* se reducirá a la mitad. Ejemplo:

```
<table class="table table-condensed">
  ...
</table>
```

Así se ve este ejemplo en tu navegador:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

3.4.1.6 Tablas semánticas

Las clases CSS semánticas explicadas en el capítulo anterior para el texto también se pueden aplicar a las filas y a las celdas de una tabla:

`.active`, aplica el color del estado `hover` a la fila o celda para que parezca que está activa.

`.success`, indica que el resultado de alguna operación ha sido un éxito.

`.warning`, avisa al usuario que se ha producido alguna circunstancia que puede requerir su atención.

`.danger`, indica que una acción es negativa o potencialmente peligrosa.

Ejemplo:

```
<table>
<tbody>
  <!-- Aplicadas en las filas -->
  <tr class="active">...</tr>
  <tr class="success">...</tr>
  <tr class="warning">...</tr>
```

```
<tr class="danger">...</tr>

<!-- Aplicadas en las celdas (<td> o <th>) -->
<tr>
  <td class="active">...</td>
  <td class="success">...</td>
  <td class="warning">...</td>
  <td class="danger">...</td>
</tr>
</tbody>
</table>
```

Así se ve este ejemplo en tu navegador:

#	Cabecera de columna	Cabecera de columna	Cabecera de columna
1	Contenido	Contenido	Contenido
2	Contenido	Contenido	Contenido
3	Contenido	Contenido	Contenido
4	Contenido	Contenido	Contenido
5	Contenido	Contenido	Contenido
6	Contenido	Contenido	Contenido
7	Contenido	Contenido	Contenido

3.4.1.7 Tablas responsive

La solución que propone Bootstrap 3 para crear tablas *responsive* que se vean bien en dispositivos pequeños consiste en añadir un *scroll* horizontal a las tablas que sean demasiado anchas. Para ello, encierra cualquier tabla con la clase `.table` dentro de un elemento con la clase `.table-responsive`. Cuando las tablas *responsive* se muestran en dispositivos con una anchura superior a 768px, se ven igual que cualquier otra tabla normal.

Ejemplo:

```
<div class="table-responsive">
  <table class="table">
    ...
  </table>
</div>
```

Así se ve este ejemplo en tu navegador (el ejemplo se muestra con poca anchura para forzar a que la tabla sea *responsive*):

#	Cabecera de tabla	Cabecera de tabla	C
1	Celda de tabla	Celda de tabla	C
2	Celda de tabla	Celda de tabla	C
3	Celda de tabla	Celda de tabla	C

3.4.2 Imágenes

Bootstrap 3 define varias clases CSS para decorar las imágenes de tus sitios web:

`.img-rounded`, añade unas pequeñas esquinas redondeadas en todos los lados de la imagen aplicando el estilo `border-radius: 6px`.

`.img-thumbnail`, muestra la imagen con un relleno blanco y un borde fino simulando el aspecto de las fotografías de las antiguas cámaras instantáneas. Añade además una breve animación para hacer que la imagen aparezca al cargar la página.

`.img-circle`, convierte la imagen en un círculo aplicando el estilo `border-radius: 50%`

ADVERTENCIA El navegador Internet Explorer 8 no soporta el uso de esquinas redondeadas, por lo que los estilos `.img-rounded` y `.img-circle` no tienen ningún efecto sobre las imágenes.

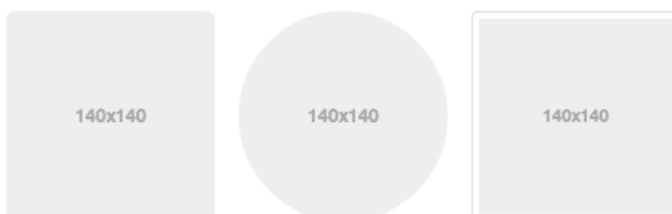
Ejemplo:

```



```

Así se ve este ejemplo en tu navegador:



3.4.3 Utilidades

3.4.3.1 Icono para cerrar

Bootstrap 3 define la clase `.close` para mostrar la entidad HTML `×` como si fuera la típica `X` asociada con el cierre de una ventana o aplicación. Utilízalo para mostrar el icono de cerrar en las ventanas modales o en las alertas sin tener que utilizar una imagen.

Ejemplo:

```
<button type="button" class="close" aria-hidden="true">&times;</button>
```



Así se ve este ejemplo en tu navegador (es una `X` gris claro a la derecha del todo):

3.4.3.2 Elementos flotantes

Flotar un elemento a la derecha o a la izquierda es muy habitual en la mayoría de diseños web. Por eso Bootstrap 3 define dos clases CSS genéricas llamadas `.pull-left` y `.pull-right` que puedes aplicar sobre cualquier elemento:

Ejemplo:

```
<div class="pull-left">...</div>
<div class="pull-right">...</div>
```

Este es el código CSS aplicado a cada clase (la palabra reservada `!important` se utiliza para evitar posibles problemas con la especificidad de los selectores):

```
.pull-left {
  float: left !important;
}
.pull-right {
  float: right !important;
}
```

Bootstrap 3 también define *mixins* para que puedas utilizar estos estilos en tus archivos LESS:

```
.elemento {
  .pull-left();
}
.otro-elemento {
  .pull-right();
}
```

No utilices estas clases para alinear los componentes de las barras de navegación `.navbar`. Utiliza en su lugar las dos clases equivalentes `.navbar-left` y `.navbar-right`.

3.4.3.3 Elementos centrados

Aplica la clase especial `center-block` para centrar horizontalmente cualquier elemento (el elemento centrado se convierte en un elemento de bloque):

Ejemplo:

```
<div class="center-block">...</div>
```

Este es el código CSS aplicado a esta clase:

```
.center-block {  
    display: block;  
    margin-left: auto;  
    margin-right: auto;  
}
```

Bootstrap 3 también define un *mixin* para que puedas utilizar estos estilos en tus archivos LESS:

```
.elemento {  
    .center-block();  
}
```

3.4.3.4 Limpiando floats

Cuando un diseño utiliza muchos elementos flotantes, es común tener que *limpiar* un elemento para que no le afecten otros elementos flotantes. Bootstrap 3 utiliza para ello el famoso [hack clearfix](#) creado originalmente por Nicolas Gallagher.

Ejemplo:

```
<div class="clearfix">...</div>
```

Este es el código CSS aplicado a esta clase:

```
.clearfix:before,  
.clearfix:after {  
    display: table;  
    content: " "  
}  
.clearfix:after {  
    clear: both;
```

```
}
```

Bootstrap 3 también define un *mixin* para que puedas utilizar estos estilos en tus archivos LESS:

```
.elemento {  
  .clearfix();  
}
```

3.4.3.5 Ocultando y mostrando elementos

Otras de las utilidades incluidas por Bootstrap 3 son las clases `.show` y `.hide`, que muestran y ocultan cualquier elemento.

Ejemplo:

```
<div class="show">...</div>  
<div class="hide">...</div>
```

Este es el código CSS aplicado a estas clases (de nuevo se utiliza `!important` para evitar problemas con los selectores):

```
.show {  
  display: block !important;  
}  
.hide {  
  display: none !important;  
}
```

Bootstrap 3 también define dos *mixins* para que puedas utilizar estos estilos en tus archivos LESS:

```
.elemento {  
  .show();  
}  
.otro-elemento {  
  .hide();  
}
```

3.4.3.6 Contenidos ocultos

Cuando se diseña un sitio web accesible, es común añadir ayudas en forma de texto que no se ve por pantalla, pero sí que se lee en los lectores que utilizan para navegar las personas discapacitadas.

Bootstrap 3 define la clase `.sr-only` para marcar un contenido como oculto y que sólo esté disponible para los lectores (*"screen readers"* en inglés, de ahí el nombre de la clase CSS). Ejemplo:

```
<a class="sr-only" href="#contenido">Ir al contenido</a>
```

Este es el código CSS aplicado a esta clase:

```
.sr-only {  
  border: 0;  
  clip: rect(0 0 0 0);  
  height: 1px;  
  width: 1px;  
  margin: -1px;  
  overflow: hidden;  
  padding: 0;  
  position: absolute;  
}
```

Bootstrap 3 también define un *mixin* para que puedas utilizar estos estilos en tus archivos LESS:

```
.saltar-navegacion {  
  .sr-only();  
}
```

3.4.3.7 Reemplazando texto por imágenes

Una de las técnicas más habituales para mostrar el logotipo de los sitios web consiste en ocultar el texto de un elemento `<h1>` para que se vea la imagen de fondo que contiene el logotipo. Esta técnica es tan habitual que Bootstrap 3 define la clase `.text-hide` para que puedas aplicarla a cualquier elemento. Ejemplo:

```
<h1 class="text-hide">Nombre de la empresa</h1>
```

Este es el código CSS aplicado a esta clase:

```
.text-hide {  
  background-color: transparent;  
  border: 0;  
  color: transparent;  
  font: 0/0 a;  
  text-shadow: none;  
}
```

Bootstrap 3 también define un *mixin* para que puedas utilizar estos estilos en tus archivos LESS:

```
.logotipo {
```

```
.text-hide();  
}
```

3.4.3.8 Utilidades *responsive*

Las utilidades *responsive* permite diseñar más rápidamente sitios web móviles, ya que muestran u ocultan elementos en función del tipo de dispositivo que utiliza el usuario para navegar. También se incluyen clases para mostrar/ocultar elementos al imprimir la página.

Estas clases deben utilizarse con moderación y siempre para mejorar el aspecto de los contenidos en cada tipo de dispositivo. Además, sólo funcionan para los elementos de bloque y las tablas, por lo que no podrás aplicarlos a los elementos en línea.

Utiliza alguna de estas clases o combina varias entre sí para definir cómo se ven tus contenidos en cada tipo de dispositivo (teléfono = menos de 768px; tablet = más de 768px; ordenador = más de 992px; ordenador grande = más de 1200px):

Clase	Teléfonos	Tablets	Ordenador	Ordenador grande
<code>.visible-xs</code>	Visible	Oculto	Oculto	Oculto
<code>.visible-sm</code>	Oculto	Visible	Oculto	Oculto
<code>.visible-md</code>	Oculto	Oculto	Visible	Oculto
<code>.visible-lg</code>	Oculto	Oculto	Oculto	Visible
<code>.hidden-xs</code>	Oculto	Visible	Visible	Visible
<code>.hidden-sm</code>	Visible	Oculto	Visible	Visible
<code>.hidden-md</code>	Visible	Visible	Oculto	Visible
<code>.hidden-lg</code>	Visible	Visible	Visible	Oculto

Igualmente, puedes utilizar estas clases para definir qué contenidos se muestran al imprimir la página:

Clase	Navegador	Impresora
<code>.visible-print</code>	Oculto	Visible
<code>.hidden-print</code>	Visible	Oculto

3.5 Formularios

Los formularios son uno de los elementos más importantes de los sitios y aplicaciones web. Por eso Bootstrap 3 permite diseñar formularios con aspectos muy variados y define decenas de estilos para todos los campos de formulario.

3.5.1 Formulario básico

Bootstrap 3 aplica por defecto algunos estilos a todos los componentes de los formularios. Si además añades la clase `.form-control` a los elementos `<input>`, `<textarea>` y `<select>`, su anchura se establece a `width: 100%`. Para optimizar el espaciado, utiliza la clase `.form-group` para encerrar cada campo de formulario con su `<label>`.

Ejemplo:

```
<form role="form">
  <div class="form-group">
    <label for="ejemplo_email_1">Email</label>
    <input type="email" class="form-control" id="ejemplo_email_1"
      placeholder="Introduce tu email">
  </div>
  <div class="form-group">
    <label for="ejemplo_password_1">Contraseña</label>
    <input type="password" class="form-control" id="ejemplo_password_1"
      placeholder="Contraseña">
  </div>
  <div class="form-group">
    <label for="ejemplo_archivo_1">Adjuntar un archivo</label>
    <input type="file" id="ejemplo_archivo_1">
    <p class="help-block">Ejemplo de texto de ayuda.</p>
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Activa esta casilla
    </label>
  </div>
</form>
```

```
<button type="submit" class="btn btn-default">Enviar</button>
</form>
```

Así se ve este ejemplo en tu navegador:



3.5.2 Formulario en línea

Para que el formulario ocupe el menor espacio posible, añade la clase `.form-inline` para que las etiquetas `<label>` se muestren a la izquierda de cada campo del formulario. Ejemplo:

```
<form class="form-inline" role="form">
  <div class="form-group">
    <label class="sr-only" for="ejemplo_email_2">Email</label>
    <input type="email" class="form-control" id="ejemplo_email_2"
      placeholder="Introduce tu email">
  </div>
  <div class="form-group">
    <label class="sr-only" for="ejemplo_password_2">Contraseña</label>
    <input type="password" class="form-control" id="ejemplo_password_2"
      placeholder="Contraseña">
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> No cerrar sesión
    </label>
  </div>
  <button type="submit" class="btn btn-default">Entrar</button>
```


`</form>`

Así se ve este ejemplo en tu navegador:



Como los elementos `<input>`, `<select>` y `<textarea>` tienen por defecto una anchura del 100% en Bootstrap 3, para utilizar los formularios en línea tienes que establecer a mano la anchura de cada campo de formulario.

TRUCO Los lectores utilizados por las personas discapacitadas tienen problemas con los formularios que no incluyen un `<label>` por cada campo de formulario. Si quieres ocultar estos elementos para los formularios en línea, utiliza la clase `.sr-only` explicada en los capítulos anteriores.

3.5.3 Formularios horizontales

Bootstrap 3 también permite alinear los elementos `<label>` y los campos de formulario mediante las clases CSS utilizadas para definir las rejillas de los *layouts*. Para ello, añade la clase `.form-horizontal` al formulario. Además, como esta clase modifica la clase `.form-group` para que se comporte como la fila de una rejilla, no es necesario que añadas en el formulario elementos con la clase `.row`.

```
<form class="form-horizontal" role="form">
  <div class="form-group">
    <label for="ejemplo_email_3" class="col-lg-2 control-label">Email</label>
    <div class="col-lg-10">
      <input type="email" class="form-control" id="ejemplo_email_3"
        placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <label for="ejemplo_password_3" class="col-lg-2 control-label">Contraseña</label>
    <div class="col-lg-10">
      <input type="password" class="form-control" id="ejemplo_password_3">
    </div>
  </div>
</form>
```

```
        placeholder="Contraseña">
    </div>
</div>
<div class="form-group">
    <div class="col-lg-offset-2 col-lg-10">
        <div class="checkbox">
            <label>
                <input type="checkbox"> No cerrar sesión
            </label>
        </div>
    </div>
</div>
<div class="form-group">
    <div class="col-lg-offset-2 col-lg-10">
        <button type="submit" class="btn btn-default">Entrar</button>
    </div>
</div>
</form>
```

Así se ve este ejemplo en tu navegador:



3.5.4 Campos de formulario

Bootstrap 3 define estilos adecuados para todos y cada uno de los campos de formulario existentes.

3.5.4.1 Inputs


Los campos de tipo `<input>` son los más numerosos, ya que con HTML5 la lista se ha ampliado a `text`, `password`, `datetime`, `datetime-local`, `date`, `month`, `time`, `week`, `number`, `email`, `url`, `search`, `tel`, y `color`.

ADVERTENCIA Bootstrap 3 solamente aplica los estilos a los campos `<input>` que definen explícitamente su tipo mediante el atributo `type`.

Ejemplo:

```
<input type="text" class="form-control" placeholder="Campo de texto">
```

Así se ve este ejemplo en tu navegador:



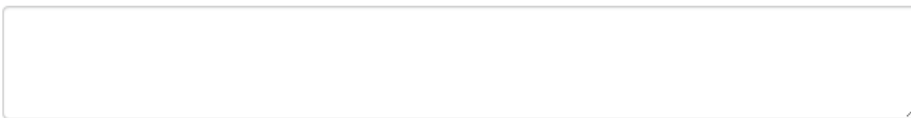
3.5.4.2 Textarea

Utiliza este control para escribir textos largos, modificando el valor del atributo `rows` para ajustarlo al número de líneas que prefieras.

Ejemplo:

```
<textarea class="form-control" rows="3"></textarea>
```

Así se ve este ejemplo en tu navegador:



3.5.4.3 Checkboxes y radio buttons

Los *checkboxes* permiten elegir una o más opciones dentro de una lista, mientras que los *radio buttons* permiten elegir una sola opción entre varias.

Estilo por defecto

Ejemplo:

```
<div class="checkbox">
  <label>
    <input type="checkbox" value="">
    Esta es una opción muy interesante que tienes que pinchar
  </label>
</div>

<div class="radio">
```

```
<label>
  <input type="radio" name="opciones" id="opciones_1" value="opcion_1" checked>
  Esta es una opción muy interesante que tienes que pinchar
</label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="opciones" id="opciones_2" value="opcion_2">
    Esta otra opción también es muy interesante y al pincharla, deseleccionas la
    opción anterior
  </label>
</div>
```

Así se ve este ejemplo en tu navegador:

☐ Esta es una opción muy interesante que tienes que pinchar

☒ Esta es una opción muy interesante que tienes que pinchar

☐ Esta otra opción también es muy interesante y al pincharla, deseleccionas la opción anterior

Checkboxes y radio buttons en línea

Si prefieres mostrar los *checkboxes* y los *radio buttons* en línea para que ocupen menos espacio, utiliza las clases CSS `.checkbox-inline` o `.radio-inline`. Ejemplo:

```
<label class="checkbox-inline">
  <input type="checkbox" id="checkboxEnLinea1" value="opcion_1"> 1
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="checkboxEnLinea2" value="opcion_2"> 2
</label>
<label class="checkbox-inline">
  <input type="checkbox" id="checkboxEnLinea3" value="opcion_3"> 3
</label>
```

Así se ve este ejemplo en tu navegador:

☐ 1 ☐ 2 ☐ 3

3.5.4.4 Listas desplegables

Para mostrar una lista desplegada, añade el atributo `multiple`. Ejemplo:

```
<select class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

```
<select multiple class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

Así se ve este ejemplo en tu navegador:



3.5.4.5 Campos de formulario estáticos

En ocasiones puede ser necesario mostrar texto al lado de un elemento `<label>` en un formulario horizontal. Para ello, añade el texto mediante un `<p>` con la clase `.form-control-static`. Esta técnica es útil por ejemplo para mostrar el valor de los campos de formulario no editables. Ejemplo:

```
<form class="form-horizontal" role="form">
  <div class="form-group">
    <label class="col-lg-2 control-label">Email</label>
    <div class="col-lg-10">
      <p class="form-control-static">email@ejemplo.com</p>
```

```
</div>
</div>
<div class="form-group">
  <label for="inputPassword" class="col-lg-2 control-label">Contraseña</label>
  <div class="col-lg-10">
    <input type="password" class="form-control" id="inputPassword"
placeholder="Contraseña">
  </div>
</div>
</form>
```

Así se ve este ejemplo en tu navegador:

Email

email@ejemplo.com

Contraseña

Contraseña

3.5.5 Estados de formulario

Modificar el estado de los controles del formulario o de sus elementos `<label>` es una de las mejores formas de proporcionar información adicional a los usuarios.

3.5.5.1 Campos seleccionados

Bootstrap 3 aplica una sombra a los campos seleccionados mediante la propiedad `box-shadow` de CSS aplicada a la pseudo-clase `:focus` del elemento. Ejemplo:

```
<input class="form-control" id="inputSeleccionado" type="text" value="Este campo está
seleccionado...">
```

Así se ve este ejemplo en tu navegador:

Este campo está seleccionado...

3.5.5.2 Campos deshabilitados

Añadiendo el atributo `disabled` a cualquier campo de texto evitas que el usuario pueda introducir información y Bootstrap 3 lo muestra con un aspecto muy diferente. Ejemplo:

```
<input class="form-control" id="campoDeshabilitado" type="text" placeholder="Este campo está deshabilitado..." disabled>
```

Así se ve este ejemplo en tu navegador:



3.5.5.3 Deshabilitando grupos de campos de formulario

Además de deshabilitar campos individuales, también es posible añadir el atributo `disabled` a un elemento `<fieldset>` para deshabilitar cualquier campo de formulario que se encuentre en su interior.

NOTA Esta clase sólo afecta al aspecto de los enlaces ``, pero no a su funcionalidad. Para deshabilitar realmente los enlaces, tendrás que utilizar código JavaScript.

ADVERTENCIA Internet Explorer 9 y sus versiones anteriores no soportan el uso del atributo `disabled` en el elemento `<fieldset>`, por lo que tendrás que utilizar algo de código JavaScript para deshabilitar los campos en ese navegador.

Ejemplo:

```
<form role="form">
  <fieldset disabled>
    <div class="form-group">
      <label for="disabledTextInput">Campo deshabilitado</label>
      <input type="text" id="campoDeshabilitado" class="form-control"
        placeholder="Campo deshabilitado">
    </div>
    <div class="form-group">
      <label for="listaDeshabilitada">Lista deshabilitada</label>
      <select id="listaDeshabilitada" class="form-control">
        <option>Lista deshabilitada</option>
      </select>
    </div>
    <div class="checkbox">
      <label>
        <input type="checkbox"> No puedes pinchar esta opción
      </label>
    </div>
  </fieldset>
```

```
<button type="submit" class="btn btn-primary">Enviar</button>
</fieldset>
</form>
```

Así se ve este ejemplo en tu navegador:

Campo deshabilitado

Lista deshabilitada

☐ No puedes pinchar esta opción

Enviar

3.5.5.4 Estados de validación

Bootstrap 3 define varios estilos para indicar el estado de la validación de cada campo del formulario: `.has-warning` para las advertencias, `.has-error` para los errores y `.has-success` para cuando el valor es correcto. Lo mejor es que estas clases se pueden aplicar a cualquier elemento que contenga una de las tres siguientes clases: `.control-label`, `.form-control` y `.help-block`.

Ejemplo:

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess">Campo con un valor válido</label>
  <input type="text" class="form-control" id="campoValido">
</div>
<div class="form-group has-warning">
  <label class="control-label" for="inputWarning">Campo con una advertencia</label>
  <input type="text" class="form-control" id="campoAdvertencia">
</div>
<div class="form-group has-error">
  <label class="control-label" for="inputError">Campo con un error</label>
  <input type="text" class="form-control" id="campoError">
</div>
```

Así se ve este ejemplo en tu navegador:

Campo con un valor válido

Campo con una advertencia

Campo con un error

3.5.6 Redimensionando campos de formulario

Modifica la altura por defecto de los campos con la clase `.input-lg` y modifica sus anchuras con las habituales clases `.col-lg-*` utilizadas para las rejillas de los *layouts*.

3.5.6.1 Cambiando la altura

Aumenta o disminuye la altura de los campos de formulario para que coincidan con la altura de los botones y así queden mejor. Ejemplo:

```
<input class="form-control input-lg" type="text" placeholder=".input-lg">  
<input class="form-control" type="text" placeholder="Campo de texto">  
<input class="form-control input-sm" type="text" placeholder=".input-sm">
```

```
<select class="form-control input-lg">...</select>  
<select class="form-control">...</select>  
<select class="form-control input-sm">...</select>
```

Así se ve este ejemplo en tu navegador:



3.5.6.2 Cambiando el ancho

La forma más sencilla de controlar la anchura de los campos de formulario consiste en encerrarlos en otros elementos y aplicar las clases utilizadas para definir las rejillas de los *layouts*. Ejemplo:

```
<div class="row">
  <div class="col-xs-3">
    <input type="text" class="form-control" placeholder=".col-xs-3">
  </div>
  <div class="col-xs-4">
    <input type="text" class="form-control" placeholder=".col-xs-4">
  </div>
  <div class="col-xs-5">
    <input type="text" class="form-control" placeholder=".col-xs-5">
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:

3.5.7 Mensajes de ayuda

Utiliza la clase `help-block` para mostrar los mensajes de ayuda de los campos del formulario. Ejemplo:

```
<span class="help-block">Un texto de ayuda que ocupa dos líneas porque es muy largo,
pero aún así se ve bien gracias a los estilos de Bootstrap.</span>
```

Así se ve este ejemplo en tu navegador:

Un texto de ayuda que ocupa dos líneas porque es muy largo, pero aún así se ve bien gracias a los estilos de Bootstrap.

3.5.8 Botones

Crea diferentes tipos de botones con ayuda de cualquiera de las clases CSS definidas por Bootstrap 3. Ejemplo:

```
<!-- Botón normal -->
<button type="button" class="btn btn-default">Normal</button>

<!-- Muestra el botón de forma destacada para descubrir fácilmente
      el botón principal dentro de un grupo de botones -->
<button type="button" class="btn btn-primary">Destacado</button>

<!-- Indica una acción exitosa o positiva -->
<button type="button" class="btn btn-success">Éxito</button>

<!-- Botón pensado para los mensajes con alertas informativas -->
<button type="button" class="btn btn-info">Información</button>

<!-- Indica que hay que tener cuidado con la acción asociada al botón -->
<button type="button" class="btn btn-warning">Advertencia</button>

<!-- Indica una acción negativa o potencialmente peligrosa -->
<button type="button" class="btn btn-danger">Peligro</button>

<!-- Resta importancia al botón haciéndolo parecer un simple enlace,
      aunque conserva tu comportamiento original de botón -->
<button type="button" class="btn btn-link">Enlace</button>
```

Así se ve este ejemplo en tu navegador:



3.5.8.1 Botones de diferente tamaño

Cuando necesites crear botones más grandes o más pequeños que el tamaño estándar, utiliza las clases `.btn-lg` (grande), `.btn-sm` (pequeño) y `.btn-xs` (extra pequeño). Ejemplo:

```
<p>
  <button type="button" class="btn btn-primary btn-lg">Botón grande</button>
  <button type="button" class="btn btn-default btn-lg">Botón grande</button>
</p>
<p>
```

```
<button type="button" class="btn btn-primary">Botón normal</button>
<button type="button" class="btn btn-default">Botón normal</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">Botón pequeño</button>
  <button type="button" class="btn btn-default btn-sm">Botón pequeño</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs">Botón extra pequeño</button>
  <button type="button" class="btn btn-default btn-xs">Botón extra pequeño</button>
</p>
```

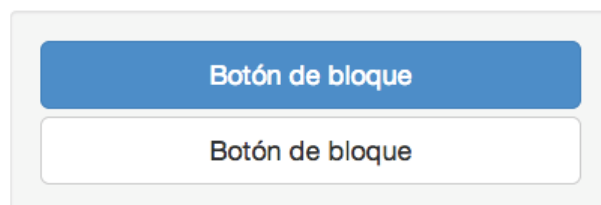
Así se ve este ejemplo en tu navegador:



También puedes convertir el botón en un elemento de bloque para hacer que ocupe toda la anchura del elemento en el que se encuentre. Para ello, añade la clase `.btn-block`. Ejemplo:

```
<button type="button" class="btn btn-primary btn-lg btn-block">Botón de
bloque</button>
<button type="button" class="btn btn-default btn-lg btn-block">Botón de
bloque</button>
```

Así se ve este ejemplo en tu navegador:



3.5.8.2 Botones activados

En ocasiones puede ser útil mostrar un botón con el aspecto de haber sido presionado, es decir, con un color de borde y un color de fondo un poco más oscuro y una sombra que imite la presión del botón. Los elementos `<button>` consiguen este efecto mediante la pseudo-clase `:active` y los elementos `<a>` mediante la clase `.active`.

Activando los elementos `<button>`

Como se utiliza la pseudo-clase `:active`, no es necesario que hagas nada. No obstante, si quieres forzar a que el botón muestre el aspecto presionado, añade la clase `.active`. Ejemplo:

```
<button type="button" class="btn btn-primary btn-lg active">Botón principal</button>
<button type="button" class="btn btn-default btn-lg active">Botón</button>
```

Así se ve este ejemplo en tu navegador:



Activando los elementos `<a>`

Añade la clase `.active` a cualquier elemento `<a>` para hacer que parezca presionado. Ejemplo:

```
<a href="#" class="btn btn-primary btn-lg active" role="button">Enlace principal</a>
<a href="#" class="btn btn-default btn-lg active" role="button">Enlace</a>
```

Así se ve este ejemplo en tu navegador:



3.5.8.3 Botones desactivados

Cuando quieras mostrar al usuario un botón con aspecto desactivado para indicarle que no puede pulsarlo, utiliza los estilos de Bootstrap que reducen su opacidad un 50%.

Desactivando los elementos `<button>`

Añade el atributo `disabled` para dar un aspecto desactivado a los elementos `<button>`. Ejemplo:

```
<button type="button" class="btn btn-lg btn-primary disabled="disabled">Botón
principal</button>
<button type="button" class="btn btn-default btn-lg disabled="disabled">Botón</button>
```

Así se ve este ejemplo en tu navegador:

Botón principal

Botón

ADVERTENCIA Cuando se añade el atributo `disabled` a un elemento `<button>`, el navegador Internet Explorer 9 y sus versiones anteriores muestran el texto de botón en color gris y con una sombra bastante fea que no se puede cambiar.

Desactivando los elementos `<a>`

Añade la clase `.disabled` para dar un aspecto desactivado a los elementos `<a>`. Ejemplo:

```
<a href="#" class="btn btn-primary btn-lg disabled" role="button">Enlace principal</a>
<a href="#" class="btn btn-default btn-lg disabled" role="button">Enlace</a>
```

Así se ve este ejemplo en tu navegador:

Enlace principal

Enlace

NOTA La clase `.disabled` solamente modifica el aspecto de los elementos `<a>`, pero no su funcionalidad. Para deshabilitar realmente los enlaces, tendrás que utilizar código JavaScript.

3.5.8.4 Etiquetas para botones

Gracias a los estilos de Bootstrap 3, puedes utilizar cualquiera de las siguientes etiquetas para mostrar botones: `<a>`, `<button>` e `<input>`. Ejemplo:

```
<a class="btn btn-default" href="#" role="button">Enlace</a>
<button class="btn btn-default" type="submit">Botón</button>
<input class="btn btn-default" type="button" value="Campo input">
<input class="btn btn-default" type="submit" value="Enviar">
```

Así se ve este ejemplo en tu navegador:

Enlace

Botón

Campo input

Enviar

ADVERTENCIA Al margen de que puedas utilizar varios elementos para crear botones, la buena práctica recomendada consiste en utilizar el elemento `<button>` siempre que sea posible, ya que es el que ofrece un aspecto más homogéneo en los diferentes navegadores.

Así, por ejemplo, Firefox sufre un error que impide establecer la propiedad `line-height` en los botones creados con elementos `<input>`, por lo que su aspecto no coincide con el del resto de botones.

3.6 Componentes

Bootstrap 3 incluye numerosos componentes CSS listos para utilizar y que cubren las necesidades más habituales de los diseños actuales para la web.

3.6.1. Iconos (*glyphicons*)

Bootstrap incluye 180 iconos creados mediante una fuente especial llamada *Glyphicon Halflings*. Aunque esta fuente normalmente no es gratuita, su creador permite utilizar estos iconos gratuitamente dentro de Bootstrap 3.

3.6.1.1 Listado completo de los iconos disponibles

A continuación, se muestran todos los iconos de Bootstrap 3 y las clases CSS necesarias para utilizar cada uno: <http://marcoceppi.github.io/bootstrap-glyphicons/>

3.6.1.2 Cómo utilizar los iconos

Por motivos de rendimiento, todos los iconos requieren de una clase CSS común para todos y de una clase CSS específica para cada uno. Para añadir un icono en cualquier elemento de la página, utiliza el siguiente código como ejemplo. Y no olvides añadir un espacio entre el icono y el texto para que se vea mejor:

```
<span class="glyphicon glyphicon-search"></span>
```

3.6.1.3 Ejemplos

Como los iconos son vectoriales, quedan bien en cualquier elemento y a cualquier tamaño. Utilízalos en botones, barras de navegación o incluso en campos de formulario. Ejemplo:

```
<div class="btn-toolbar" role="toolbar">
  <div class="btn-group">
    <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-align-left"></span>
    </button>

    <button type="button" class="btn btn-default">
      <span class="glyphicon glyphicon-align-center"></span>
    </button>
  </div>
</div>
```



```
<button type="button" class="btn btn-default">
  <span class="glyphicon glyphicon-align-right"></span>
</button>

<button type="button" class="btn btn-default">
  <span class="glyphicon glyphicon-align-justify"></span>
</button>
</div>
</div>

<div class="btn-toolbar" role="toolbar">
  <button type="button" class="btn btn-default btn-lg">
    <span class="glyphicon glyphicon-star"></span> Star
  </button>

  <button type="button" class="btn btn-default">
    <span class="glyphicon glyphicon-star"></span> Star
  </button>

  <button type="button" class="btn btn-default btn-sm">
    <span class="glyphicon glyphicon-star"></span> Star
  </button>

  <button type="button" class="btn btn-default btn-xs">
    <span class="glyphicon glyphicon-star"></span> Star
  </button>
</div>
```

Así se ve este ejemplo en tu navegador:



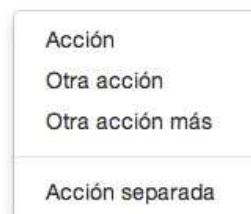
3.6.2 Menús desplegables

Este componente permite mostrar una lista de enlaces como si fuera un menú desplegable con la lista de acciones que el usuario puede realizar. Para que funcionen sus características interactivas, es necesario utilizar también el plugin `dropdown` de JavaScript. Ejemplo:

```
<div class="dropdown">
  <button class="btn dropdown-toggle sr-only" type="button"
    id="dropdownMenu1" data-toggle="dropdown">
    Menú desplegable
    <span class="caret"></span>
  </button>

  <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">
    <li role="presentation">
      <a role="menuitem" tabindex="-1" href="#">Acción</a>
    </li>
    <li role="presentation">
      <a role="menuitem" tabindex="-1" href="#">Otra acción</a>
    </li>
    <li role="presentation">
      <a role="menuitem" tabindex="-1" href="#">Otra acción más</a>
    </li>
    <li role="presentation" class="divider"></li>
    <li role="presentation">
      <a role="menuitem" tabindex="-1" href="#">Acción separada</a>
    </li>
  </ul>
</div>
```

Así se ve este ejemplo en tu navegador:



Como se muestra en el ejemplo anterior, es necesario utilizar la clase `.dropdown` para encerrar tanto el menú como el elemento que lo activa (en este caso, el botón cuyo `id` es `#dropdownMenu1`). También es posible utilizar tu propia clase CSS, siempre que ese elemento aplique el estilo `position: relative;`.

3.6.2.1 Opciones de alineación

Añade la clase `.pull-right` a la lista `.dropdown-menu` que define el menú para alinear sus contenidos a la derecha.

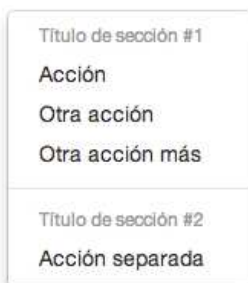
```
<ul class="dropdown-menu pull-right" role="menu" aria-labelledby="dLabel">
  ...
</ul>
```

3.6.2.2 Títulos de sección

Los menús desplegables también pueden contener elementos con la clase `.dropdown-header` para definir el título de un grupo de enlaces. Ejemplo:

```
<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu2">
  <li role="presentation" class="dropdown-header">Título de sección #1</li>
  ...
  <li role="presentation" class="divider"></li>
  <li role="presentation" class="dropdown-header">Título de sección #2</li>
  ...
</ul>
```

Así se ve este ejemplo en tu navegador:



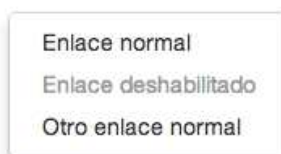
3.6.2.3 Deshabilitando elementos del menú

Si se añade la clase `.disabled` al elemento `` de un enlace del menú, este se muestra deshabilitado, por lo que el usuario puede ver que esa acción existe, pero que no es posible utilizarla en ese momento. Ejemplo:

```
<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu3">
  <li role="presentation">
    <a role="menuitem" tabindex="-1" href="#">Enlace normal</a>
  </li>
```

```
<li role="presentation" class="disabled">
  <a role="menuitem" tabindex="-1" href="#">Enlace deshabilitado</a>
</li>
<li role="presentation">
  <a role="menuitem" tabindex="-1" href="#">Otro enlace normal</a>
</li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.3 Grupos de botones

Bootstrap 3 te permite agrupar varios botones relacionados entre sí para mostrarlos en una única línea. Opcionalmente puedes utilizar el plugin de JavaScript para hacer que los botones se comporten como *radiobuttons* o como *checkboxes*.

Opciones especiales para los tooltips y popovers dentro de los botones

Cuando utilices *tooltips* o *popovers* en algún elemento dentro de un `.btn-group`, tendrás que especificar la opción `container: 'body'` para evitar efectos indeseados (como por ejemplo que el elemento se haga más ancho o que pierda sus bordes redondeados al ejecutarse el *tooltip* o el *popover*).

3.6.3.1 Ejemplo sencillo

Para crear un grupo de botones, encierra varios botones con la clase `.btn` dentro de un elemento con la clase `.btn-group`. Ejemplo:

```
<div class="btn-group">
  <button type="button" class="btn btn-default">Izquierdo</button>
  <button type="button" class="btn btn-default">Central</button>
  <button type="button" class="btn btn-default">Derecho</button>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.3.2 Botones en la barra de herramientas

Combinando varios elementos `.btn-group` puedes crear una *toolbar* o barra de herramientas basada en botones. Ejemplo:

```
<div class="btn-toolbar" role="toolbar">
  <div class="btn-group">
    <button type="button" class="btn btn-default">1</button>
    <button type="button" class="btn btn-default">2</button>
    <button type="button" class="btn btn-default">3</button>
    <button type="button" class="btn btn-default">4</button>
  </div>

  <div class="btn-group">
    <button type="button" class="btn btn-default">5</button>
    <button type="button" class="btn btn-default">6</button>
    <button type="button" class="btn btn-default">7</button>
  </div>

  <div class="btn-group">
    <button type="button" class="btn">8</button>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.3.3 Cambiando el tamaño de los botones

Para cambiar el tamaño de los botones de un grupo, en vez de establecer el tamaño individualmente a cada botón, puedes aplicar las clases `.btn-group-*` directamente al elemento que agrupa a todos ellos. Ejemplo:

```
<div class="btn-group btn-group-lg">...</div>
<div class="btn-group">...</div>
```

```
<div class="btn-group btn-group-sm">...</div>
<div class="btn-group btn-group-xs">...</div>
```

Así se ve este ejemplo en tu navegador:



3.6.3.4 Anidando grupos de botones

Bootstrap 3 también permite mostrar menús desplegables dentro de los grupos de botones. Para ello, incluye un elemento con la clase `.btn-group` dentro de otro elemento con la clase `.btn-group`. Ejemplo:

```
<div class="btn-group">
  <button type="button" class="btn btn-default">1</button>
  <button type="button" class="btn btn-default">2</button>

  <div class="btn-group">
    <button type="button" class="btn btn-default dropdown-toggle"
      data-toggle="dropdown">
      Botón Desplegable
    <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li><a href="#">Enlace #1</a></li>
      <li><a href="#">Enlace #2</a></li>
    </ul>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.3.5 Grupos de botones verticales

Aplica la clase `.btn-group-vertical` sobre un grupo de botones para mostrarlos verticalmente en vez de con su estilo horizontal habitual. Ejemplo:

```
<div class="btn-group-vertical">
  ...
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.3.6 Botones justificados

Los grupos de botones también se pueden justificar para que ocupen toda la anchura del elemento en el que se encuentran, haciendo que cada botón ocupe la misma anchura. Para ello, añade la clase `.btn-group-justified` al grupo de botones. La única limitación es que este comportamiento solo funciona para los elementos `<a>`, ya que los elementos `<button>` ignoran estos estilos. Ejemplo:

```
<div class="btn-group btn-group-justified">
  ...
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.4 Botones desplegables

Los botones desplegables se crean añadiendo un botón dentro de cualquier elemento con la clase `.btn-group`.

NOTA Para que funcionen bien estos botones desplegados es necesario utilizar el plugin **dropdown** de JavaScript.

3.6.4.1 Botones desplegados simples

Aplica los cambios que se muestran a continuación para convertir un botón normal en un botón que despliega un completo menú de opciones:

```
<div class="btn-group">
  <button type="button" class="btn btn-default dropdown-toggle"
    data-toggle="dropdown">
    Título del botón <span class="caret"></span>
  </button>

  <ul class="dropdown-menu" role="menu">
    <li><a href="#">Acción #1</a></li>
    <li><a href="#">Acción #2</a></li>
    <li><a href="#">Acción #3</a></li>
    <li class="divider"></li>
    <li><a href="#">Acción #4</a></li>
  </ul>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.4.2 Botones desplegados compuestos

Los botones compuestos se crean realmente con dos botones diferentes: el primero contiene la acción principal y el segundo simplemente muestra una flecha descendente y al pulsarla, se despliega para mostrar el menú de opciones. Ejemplo:

```
<div class="btn-group">
  <button type="button" class="btn btn-danger">Título del botón</button>
```



```
<button type="button" class="btn btn-danger dropdown-toggle"
        data-toggle="dropdown">
  <span class="caret"></span>
  <span class="sr-only">Desplegar menú</span>
</button>

<ul class="dropdown-menu" role="menu">
  <li><a href="#">Acción #1</a></li>
  <li><a href="#">Acción #2</a></li>
  <li><a href="#">Acción #3</a></li>
  <li class="divider"></li>
  <li><a href="#">Acción #4</a></li>
</ul>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.4.3 Cambiando el tamaño de los botones

Los botones desplegables funcionan bien con cualquier tamaño de botón. Ejemplo

```
<div class="btn-group">
  <button class="btn btn-default btn-lg dropdown-toggle"
        type="button" data-toggle="dropdown">
    Botón grande <span class="caret"></span>
  </button>

  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

```
<div class="btn-group">
  <button class="btn btn-default btn-sm dropdown-toggle"
    type="button" data-toggle="dropdown">
    Botón pequeño <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

```
<div class="btn-group">
  <button class="btn btn-default btn-xs dropdown-toggle"
    type="button" data-toggle="dropdown">
    Botón extra pequeño <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    ...
  </ul>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.4.4 Botones desplegados invertidos

Por defecto, los botones desplegados en Bootstrap 3 se despliegan hacia abajo ("*dropdown*" en inglés). Si quieres desplegar el menú hacia arriba ("*dropup*" en inglés), añade la clase `.dropup` al elemento contenedor del botón. Ejemplo:

```
<div class="btn-group dropdown">
  <button type="button" class="btn btn-default">
    Botón dropup</button>

  <button type="button" class="btn btn-default dropdown-toggle"
    data-toggle="dropdown">
    <span class="caret"></span>
    <span class="sr-only">Desplegar menú</span>
  </button>

  <ul class="dropdown-menu">
    ...
  </ul>

  <button type="button" class="btn btn-default">
    Botón dropup a la derecha</button>
    ...

  <ul class="dropdown-menu pull-right">
    ...
  </ul>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.5 Grupos de campos de formulario

Los campos de formulario de tipo texto se pueden modificar para que muestren un texto o un botón delante, detrás o a ambos lados. Para ello, se agrupa el campo dentro de un elemento con la clase `.input-group` y se marca el elemento que va delante o detrás con la clase `.input-group-addon`.

ADVERTENCIA No debes aplicar esta técnica a los elementos `<select>` porque los navegadores basados en el motor WebKit no soportan este tipo de estilos.

Opciones especiales para los tooltips y popovers

Cuando utilices *tooltips* o *popovers* en algún elemento dentro de un `.input-group`, tendrás que especificar la opción `container: 'body'` para evitar efectos indeseados (como por ejemplo que el elemento se haga más ancho o que pierda sus bordes redondeados al ejecutarse el *tooltip* o el *popover*).

3.6.5.1 Ejemplo básico

```
<div class="input-group">
  <span class="input-group-addon">@</span>
  <input type="text" class="form-control" placeholder="Nombre de usuario">
</div>
```

```
<div class="input-group">
  <input type="text" class="form-control">
  <span class="input-group-addon">.00</span>
</div>
```

```
<div class="input-group">
  <span class="input-group-addon">$</span>
  <input type="text" class="form-control">
  <span class="input-group-addon">.00</span>
</div>
```

Así se ve este ejemplo en tu navegador:

3.6.5.2 Cambiando el tamaño de los campos

Los grupos de campos se pueden redimensionar fácilmente cambiando el tamaño del campo de formulario principal. De esta forma, al aplicar la clase CSS correspondiente al elemento `.input-group`, el resto de elementos del grupo se redimensionan automáticamente. Ejemplo:

```
<div class="input-group input-group-lg">
  <span class="input-group-addon">@</span>
  <input type="text" class="form-control" placeholder="Nombre de usuario">
</div>
```

```
<div class="input-group">
  <span class="input-group-addon">@</span>
  <input type="text" class="form-control" placeholder="Nombre de usuario">
</div>
```

```
<div class="input-group input-group-sm">
  <span class="input-group-addon">@</span>
  <input type="text" class="form-control" placeholder="Nombre de usuario">
</div>
```

Así se ve este ejemplo en tu navegador:

3.6.5.3 Adjuntando checkboxes y radiobuttons

Además de texto, a los campos de formulario también se le pueden adjuntar (por delante, por detrás o a ambos lados) otros campos de formulario de tipo *radiobutton* o *checkbox*. Ejemplo:

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="checkbox">
      </span>
      <input type="text" class="form-control">
    </div>
  </div>

  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-addon">
        <input type="radio">
      </span>
      <input type="text" class="form-control">
    </div>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.5.4 Adjuntando botones

Debido a los estilos que aplican por defecto los navegadores, adjuntar un botón a un campo de formulario es un poco diferente a los casos anteriores. En lugar de aplicar la clase `.input-group-addon`, para adjuntar los botones se utiliza la clase `.input-group-btn`. Ejemplo:

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-btn">
        <button class="btn btn-default" type="button">Buscar</button>
      </span>
    </div>
  </div>
</div>
```

```
<input type="text" class="form-control">
</div>
</div>

<div class="col-lg-6">
  <div class="input-group">
    <input type="text" class="form-control">
    <span class="input-group-btn">
      <button class="btn btn-default" type="button">Buscar</button>
    </span>
  </div>
</div>
</div>
```

Así se ve este ejemplo en tu navegador:

Buscar	
	Buscar

3.6.5.5 Adjuntando botones desplegables

Ejemplo:

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <div class="input-group-btn">
        <button type="button" class="btn btn-default dropdown-toggle"
          data-toggle="dropdown">
          Acción <span class="caret"></span>
        </button>

        <ul class="dropdown-menu" role="menu">
          <li><a href="#">Acción #1</a></li>
          <li><a href="#">Acción #2</a></li>
          <li><a href="#">Acción #3</a></li>
          <li class="divider"></li>
```

```
<li><a href="#">Acción #4</a></li>
</ul>
</div>

<input type="text" class="form-control">
</div>
</div>

<div class="col-lg-6">
  <div class="input-group">
    <input type="text" class="form-control">

    <div class="input-group-btn">
      <button type="button" class="btn btn-default dropdown-toggle"
        data-toggle="dropdown">
        Acción <span class="caret"></span>
      </button>

      <ul class="dropdown-menu pull-right" role="menu">
        <li><a href="#">Acción #1</a></li>
        <li><a href="#">Acción #2</a></li>
        <li><a href="#">Acción #3</a></li>
        <li class="divider"></li>
        <li><a href="#">Acción #4</a></li>
      </ul>
    </div>
  </div>
</div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.5.6 Adjuntando botones compuestos

Ejemplo:

```
<div class="input-group">
  <div class="input-group-btn">
    <!-- Botón y menú desplegable -->
  </div>
  <input type="text" class="form-control">
</div>
```

```
<div class="input-group">
  <input type="text" class="form-control">
  <div class="input-group-btn">
    <!-- Botón y menú desplegable -->
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.6 Elementos de navegación

Bootstrap 3 define varios elementos de navegación que comparten el mismo código HTML y la misma clase base `.nav`. Para seleccionar el estilo del elemento, se modifica su clase CSS específica.

3.6.6.1 Navegación con pestañas

Aplica la clase `.nav` para crear un elemento de navegación y después aplica la clase `.nav-tabs` para mostrar sus enlaces en forma de pestaña. Ejemplo:

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#">Inicio</a></li>
  <li><a href="#">Perfil</a></li>
  <li><a href="#">Mensajes</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.6.2. Navegación con botones

Aplicando la clase `.nav-pills` al mismo código HTML del ejemplo anterior, los enlaces se muestran como botones ("*pills*" en inglés) en vez de como pestañas. Ejemplo:

```
<ul class="nav nav-pills">
  <li class="active"><a href="#">Inicio</a></li>
  <li><a href="#">Perfil</a></li>
  <li><a href="#">Mensajes</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



La navegación con botones también se puede mostrar verticalmente añadiendo la clase `.nav-stacked`. Ejemplo:

```
<ul class="nav nav-pills nav-stacked">
  ...
```

``

Así se ve este ejemplo en tu navegador:



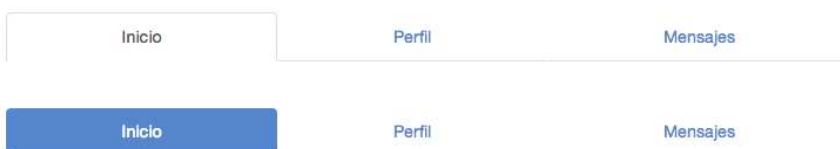
3.6.6.3 Navegación justificada

Bootstrap 3 también permite crear elementos de navegación que ocupan toda la anchura del elemento en el que se encuentran y en los que todos sus enlaces tienen la misma anchura. Para ello, añade la clase `.nav-justified` al elemento de navegación. Ten en cuenta que este comportamiento sólo es posible para los dispositivos con una anchura superior a 768px. En el resto de dispositivos, el elemento de navegación se muestra verticalmente. Ejemplo:

```
<ul class="nav nav-tabs nav-justified">
  <li class="active"><a href="#">Inicio</a></li>
  <li><a href="#">Perfil</a></li>
  <li><a href="#">Mensajes</a></li>
</ul>
```

```
<ul class="nav nav-pills nav-justified">
  <li class="active"><a href="#">Inicio</a></li>
  <li><a href="#">Perfil</a></li>
  <li><a href="#">Mensajes</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.6.4 Deshabilitando enlaces

Aplica la clase `.disabled` a cualquier enlace del elemento de navegación para mostrarlo de color gris y sin que cambie de color al pasar el ratón por encima suyo.

ADVERTENCIA Al aplicar la clase `.disabled` solamente se modifica el aspecto de los enlaces, pero no su funcionamiento. Así que si el usuario pincha el enlace deshabilitado, se ejecutará la acción asociada.

```
<ul class="nav nav-pills">
  <li><a href="#">Enlace activo</a></li>
  <li><a href="#">Enlace activo</a></li>
  <li class="disabled"><a href="#">Enlace deshabilitado</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:

Enlace activo Enlace activo Enlace deshabilitado

3.6.6.5 Navegación con menús desplegables

Usando el plugin `dropdown` de JavaScript y un poco de código HTML, es posible añadir menús desplegables tanto a las pestañas como a los botones.

Pestañas con menús desplegables

```
<ul class="nav nav-tabs">
  ...
  <li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#">
      Menú desplegable <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
      ...
    </ul>
  </li>
  ...
</ul>
```

Así se ve este ejemplo en tu navegador:



Botones con menús desplegables

```
<ul class="nav nav-pills">
  ...
  <li class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#">
      Menú desplegable <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
      ...
    </ul>
  </li>
  ...
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.7 Barras de navegación

Las barras de navegación son componentes adaptados al diseño web *responsive* y que se utilizan como elemento principal de navegación tanto en las aplicaciones como en los sitios web.

En los dispositivos móviles se muestran inicialmente minimizadas y al pulsar sobre ellas, se despliegan todas sus opciones. Si la anchura del dispositivo aumenta hasta un nivel suficiente, las barras de navegación muestran todos sus contenidos horizontalmente.

Dependiendo del contenido de tu barra de navegación, puede ser necesario modificar el punto a partir del cual se muestra horizontal en vez de minimizada. Para ello, modifica el valor de la variable `@grid-float-breakpoint` en tu archivo LESS o añade tu propia *media query* en el archivo CSS.

3.6.7.1 Barra de navegación básica

Ejemplo:

```
<nav class="navbar navbar-default" role="navigation">
  <!-- El logotipo y el icono que despliega el menú se agrupan
        para mostrarlos mejor en los dispositivos móviles -->
  <div class="navbar-header">
    <button type="button" class="navbar-toggle" data-toggle="collapse"
            data-target=".navbar-ex1-collapse">
      <span class="sr-only">Desplegar navegación</span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </button>
    <a class="navbar-brand" href="#">Logotipo</a>
  </div>

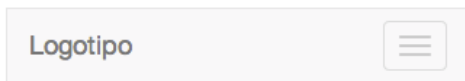
  <!-- Agrupar los enlaces de navegación, los formularios y cualquier
        otro elemento que se pueda ocultar al minimizar la barra -->
  <div class="collapse navbar-collapse navbar-ex1-collapse">
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Enlace #1</a></li>
      <li><a href="#">Enlace #2</a></li>
      <li class="dropdown">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown">
          Menú #1 <b class="caret"></b>
        </a>
        <ul class="dropdown-menu">
```

```
<li><a href="#">Acción #1</a></li>
<li><a href="#">Acción #2</a></li>
<li><a href="#">Acción #3</a></li>
<li class="divider"></li>
<li><a href="#">Acción #4</a></li>
<li class="divider"></li>
<li><a href="#">Acción #5</a></li>
</ul>
</li>
</ul>

<form class="navbar-form navbar-left" role="search">
  <div class="form-group">
    <input type="text" class="form-control" placeholder="Buscar">
  </div>
  <button type="submit" class="btn btn-default">Enviar</button>
</form>

<ul class="nav navbar-nav navbar-right">
  <li><a href="#">Enlace #3</a></li>
  <li class="dropdown">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
      Menú #2 <b class="caret"></b>
    </a>
    <ul class="dropdown-menu">
      <li><a href="#">Acción #1</a></li>
      <li><a href="#">Acción #2</a></li>
      <li><a href="#">Acción #3</a></li>
      <li class="divider"></li>
      <li><a href="#">Acción #4</a></li>
    </ul>
  </li>
</ul>
</div>
</nav>
```

Así se ve este ejemplo en tu navegador cuando el dispositivo es pequeño:



Así se ve este ejemplo en tu navegador cuando el dispositivo es grande:



ADVERTENCIA La barra de navegación *responsive* requiere el uso del plugin `collapse` de JavaScript incluido en Bootstrap 3.

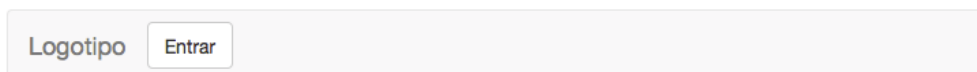
TRUCO Para mejorar la accesibilidad de tus sitios y aplicaciones, no olvides añadir el atributo `role="navigation"` a todas las barras de navegación.

3.6.7.2 Barras de navegación con botones

Cuando un botón no se encuentra dentro de un formulario, debes añadirle la clase `.navbar-btn` para que se muestre alineado verticalmente. Ejemplo:

```
<button type="button" class="btn btn-default navbar-btn">Entrar</button>
```

Así se ve este ejemplo en tu navegador:

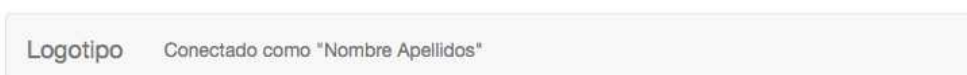


3.6.7.3 Barras de navegación con texto

De la misma forma, si quieres mostrar texto dentro de una barra de navegación, enciérralo en un elemento con la clase `.navbar-text` para que su color y tamaño sea el adecuado. Ejemplo:

```
<p class="navbar-text">Conectado como "Nombre Apellidos"</p>
```

Así se ve este ejemplo en tu navegador:

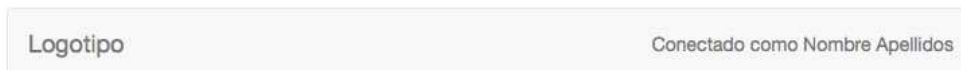


3.6.7.4 Barras de navegación con enlaces

Las barras de navegación de Bootstrap 3 también pueden contener enlaces que no formen parte de las opciones de navegación. Para mostrarlos con los estilos adecuados, aplica la clase `.navbar-link` a todos esos enlaces. Ejemplo:

```
<p class="navbar-text pull-right">  
  Conectado como <a href="#" class="navbar-link">Nombre Apellidos</a>  
</p>
```

Así se ve este ejemplo en tu navegador:



3.6.7.5 Alineando los elementos de la barra de navegación

Para alinear los enlaces, formularios, botones o texto de la barra de navegación, puedes utilizar las clases `.navbar-left` y `.navbar-right`, que añaden a ese elemento un `float` en la dirección correspondiente. Así, para alinear por ejemplo varios enlaces de navegación, puedes encerrarlos dentro de un elemento `` y aplicar sobre este último una de las dos clases CSS mencionadas anteriormente.

En realidad, estas clases se basan en las clases genéricas `.pull-left` y `.pull-right` explicadas en los capítulos anteriores. La diferencia es que han sido adaptadas a los diferentes *media queries* para que los elementos de la barra de navegación se vean lo mejor posible en todo tipo de dispositivos.

3.6.7.6 Barra de navegación fija en la parte superior de la página

Aplica la clase `.navbar-fixed-top` a la barra de navegación para fijarla en la parte superior de la página.

ADVERTENCIA Debido a los estilos aplicados, la barra de navegación fija puede tapar los contenidos que se encuentran en la parte superior de la página. Para evitarlo, añade un `padding` en la parte superior del elemento `<body>`. Como por defecto la barra de navegación tiene una altura de `50px`, este es el estilo recomendado por defecto:

```
body { padding-top: 70px; }
```

No olvides incluir este estilo **después** de todos los estilos de Bootstrap 3.

Ejemplo:

```
<nav class="navbar navbar-default navbar-fixed-top" role="navigation">  
  ...
```

```
</nav>
```

3.6.7.7 Barra de navegación fija en la parte inferior de la página

Aplica la clase `.navbar-fixed-bottom` a la barra de navegación para fijarla en la parte inferior de la página.

ADVERTENCIA Debido a los estilos aplicados, la barra de navegación fija puede tapar los contenidos que se encuentran en la parte inferior de la página. Para evitarlo, añade un `padding` en la parte inferior del elemento `<body>`. Como por defecto la barra de navegación tiene una altura de `50px`, este es el estilo recomendado por defecto:

```
body { padding-bottom: 70px; }
```

No olvides incluir este estilo **después** de todos los estilos de Bootstrap 3.

Ejemplo:

```
<nav class="navbar navbar-default navbar-fixed-bottom" role="navigation">
  ...
</nav>
```

3.6.7.8 Barra de navegación estática en la parte superior de la página

Aplica la clase `.navbar-static-top` para crear una barra de navegación que ocupa toda la anchura del elemento en el que se encuentra, pero que a diferencia de las clases `.navbar-fixed-*`, sí que desaparece al hacer *scroll* en la página.

Ejemplo:

```
<nav class="navbar navbar-default navbar-static-top" role="navigation">
  ...
</nav>
```

3.6.7.9 Barra de navegación invertida

El estilo por defecto de las barras de navegación no queda muy bien en las páginas con fondo oscuro. Por eso Bootstrap 3 define un estilo alternativo muy oscuro que se activa aplicando la clase `.navbar-inverse` a la barra de navegación. Ejemplo:

```
<nav class="navbar navbar-inverse" role="navigation">
  ...
</nav>
```

Así se ve este ejemplo en tu navegador:

Logotipo

Inicio

Enlace #1

Enlace #2

3.6.8 Migas de pan

Las migas de pan (en inglés, "*breadcrumbs*") indican la posición de la página actual dentro de la jerarquía de navegación del sitio. La separación entre los enlaces de las migas de pan se añaden automáticamente mediante el pseudo-selector `:before` y la propiedad `content` de CSS. Ejemplo:

```
<ol class="breadcrumb">
  <li class="active">Inicio</li>
</ol>
```

```
<ol class="breadcrumb">
  <li><a href="#">Inicio</a></li>
  <li class="active">Libros</li>
</ol>
```

```
<ol class="breadcrumb">
  <li><a href="#">Inicio</a></li>
  <li><a href="#">Libros</a></li>
  <li class="active">Bootstrap 3</li>
</ol>
```

Así se ve este ejemplo en tu navegador:

Inicio

Inicio / Libros

Inicio / Libros / Bootstrap 3

3.6.9 Paginadores

Bootstrap 3 incluye dos componentes para paginar los contenidos de tu sitio o aplicación web. El primero es un completo paginador que muestra enlaces para todas las páginas de resultados y el segundo es un paginador simple con los enlaces *Anterior* y *Siguiente*.

3.6.9.1 Paginador por defecto

El paginador por defecto de Bootstrap 3 está basado en el del sitio Rdio y queda bien tanto para las aplicaciones como para los sitios web. Como el paginador es bastante grande, el usuario nunca lo pierde de vista. Además su diseño escala bien a un gran número de páginas y sus enlaces son fáciles de pinchar. Ejemplo:

```
<ul class="pagination">
  <li><a href="#">&laquo;</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">5</a></li>
  <li><a href="#">&raquo;</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:

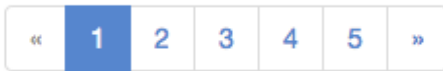


3.6.9.2 Enlaces activos y enlaces deshabilitados

El aspecto de los enlaces del paginador se puede modificar para adaptarlo a tus necesidades. Aplica la clase `.disabled` a los enlaces que no se puedan pinchar y aplica la clase `.active` al enlace que corresponde a la página en la que se encuentra el usuario. Ejemplo:

```
<ul class="pagination">
  <li class="disabled">
    <a href="#">&laquo;</a>
  </li>
  <li class="active">
    <a href="#">1 <span class="sr-only">(página actual)</span></a>
  </li>
  ...
</ul>
```

Así se ve este ejemplo en tu navegador:



Cuando un enlace no sea pinchable, puedes reemplazar su etiqueta `<a>` por la etiqueta ``. De esta forma, además de mostrarse con el estilo correcto, no se podrán pinchar de ninguna manera. Ejemplo:

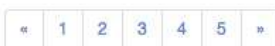
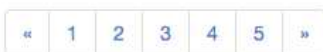
```
<ul class="pagination">
  <li class="disabled">
    <span>&laquo;</span>
  </li>
  <li class="active">
    <span>1 <span class="sr-only">(página actual)</span></span>
  </li>
  ...
</ul>
```

3.6.9.3 Cambiando el tamaño del paginador

Gracias a las clases CSS especiales `.pagination-lg` y `.pagination-sm` puedes hacer el paginador más grande o más pequeño respectivamente. Ejemplo:

```
<ul class="pagination pagination-lg">...</ul>
<ul class="pagination">...</ul>
<ul class="pagination pagination-sm">...</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.9.4 Paginador simple

El paginador simple de Bootstrap 3 es ideal para sitios web sencillos como blogs o revistas. Este paginador solamente muestra dos enlaces para ir a la página *Anterior* o *Siguiente* de los resultados. Ejemplo:

```
<ul class="pager">
  <li><a href="#">Anterior</a></li>
  <li><a href="#">Siguiente</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.9.5 Alineando los enlaces del paginador simple

Si lo prefieres, también puedes alinear los enlaces a la izquierda y a la derecha con las clases CSS especiales `.previous` y `.next`. Ejemplo:

```
<ul class="pager">
  <li class="previous"><a href="#">&larr; Anterior</a></li>
  <li class="next"><a href="#">Siguiente &rarr;</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.9.6 Deshabilitando los enlaces del paginador simple

Los enlaces del paginador simple también pueden utilizar la clase `.disabled` para mostrar un aspecto deshabilitado. Ejemplo:

```
<ul class="pager">
  <li class="previous disabled"><a href="#">&larr; Anterior</a></li>
  <li class="next"><a href="#">Siguiente &rarr;</a></li>
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.10 Etiquetas

Ejemplo:

```
<h3>
  Lorem ipsum dolor sit amet
  <span class="label label-default">Nuevo</span>
</h3>
```

Así se ve este ejemplo en tu navegador:

Lorem ipsum dolor sit amet **Nuevo**

Lorem ipsum dolor sit amet **Nuevo**

Lorem ipsum dolor sit amet **Nuevo**

Lorem ipsum dolor sit amet **Nuevo**

Lorem ipsum dolor sit amet **Nuevo**

Lorem ipsum dolor sit amet **Nuevo**

3.6.10.1 Tipos de etiquetas

Añade cualquiera de las siguientes clases CSS específicas para modificar el tipo de etiqueta que se muestra. Ejemplo:

```
<span class="label label-default">Por defecto</span>
<span class="label label-primary">Principal</span>
<span class="label label-success">Éxito</span>
<span class="label label-info">Información</span>
<span class="label label-warning">Aviso</span>
<span class="label label-danger">Peligro</span>
```

Así se ve este ejemplo en tu navegador:

Por defecto **Principal** **Éxito** **Información** **Aviso** **Peligro**

3.6.11 Badges

Los *badges* se emplean para destacar elementos nuevos o que no han sido leídos. Añade *badges* a los enlaces, los elementos de navegación de Bootstrap y muchos otros tipos de elementos mediante la etiqueta ``. Ejemplo:

```
<a href="#">Bandeja de entrada <span class="badge">42</span></a>
```

Así se ve este ejemplo en tu navegador:



Lo mejor de los *badges* es que se ocultan automáticamente cuando no hay elementos nuevos o sin leer, gracias al selector `:empty` de CSS. Este comportamiento no está disponible en Internet Explorer 8 porque ese navegador no soporta el selector `:empty`.

Por otra parte, los *badges* disponen por defecto de los estilos adecuados para verse bien cuando se incluyen dentro del enlace seleccionado de un elemento de navegación: Ejemplo:

```
<ul class="nav nav-pills nav-stacked">
  <li class="active">
    <a href="#">
      <span class="badge pull-right">42</span>
      Inicio
    </a>
  </li>
  ...
</ul>
```

Así se ve este ejemplo en tu navegador:



3.6.12 Jumbotron

Se trata de un componente para mostrar contenidos de forma muy destacada. Si no lo encierras dentro de un elemento `.container`, ocupa toda la anchura del dispositivo. Si lo encierras dentro de un `.container`, solamente ocupará la anchura del contenedor y mostrará esquinas redondeadas. Ejemplo:

```
<div class="jumbotron">
  <div class="container">
    <h1>Lorem ipsum</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
      tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
      quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
      consequat.</p>
    <p><a class="btn btn-primary btn-lg" role="button">Leer más</a></p>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.13 Encabezado de página

Resulta muy común que las páginas de sitios web como blogs, periódicos y revistas muestren de forma muy destacada el titular de la página con alguna otra información relacionada. Por ello Bootstrap 3 define un componente llamado "encabezado de página" que puedes utilizar para encerrar este tipo de elementos.

Normalmente, el contenido de este elemento, al que se aplica la clase `.page-header`, está formado por un elemento `<h1>` y un elemento `<small>` asociado para la información secundaria, aunque puede contener cualquier otro tipo de elemento. Ejemplo:

```
<div class="page-header">
```

```
<h1>Encabezado de página <small>con un texto secundario</small></h1>
</div>
<p>...</p>
<p>...</p>
```

Así se ve este ejemplo en tu navegador:

Encabezado de página con un texto secundario

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.6.14 Imágenes en miniatura

Gracias a la rejilla de Bootstrap 3 explicada en los capítulos anteriores y gracias al componente de las imágenes en miniatura (en inglés, *"thumbnails"*), resulta muy sencillo crear galerías de imágenes y vídeos.

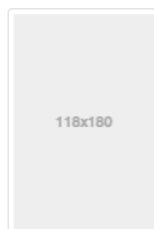
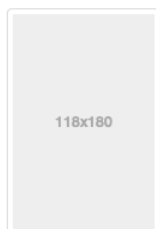
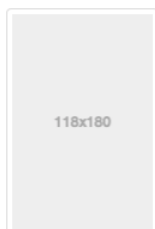
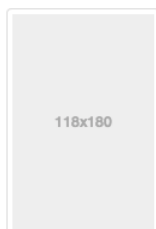
3.6.14.1 Ejemplo sencillo

El componente de imágenes en miniatura de Bootstrap 3 está diseñado para mostrar lo más fácilmente posible varias imágenes que enlazan con su versión en alta resolución. Ejemplo:

```
<div class="row">
  <div class="col-sm-6 col-md-3">
    <a href="#" class="thumbnail">
      
    </a>
  </div>
  ...
</div>
```

Así se ve este ejemplo en tu navegador:

Tutor Ricardo M

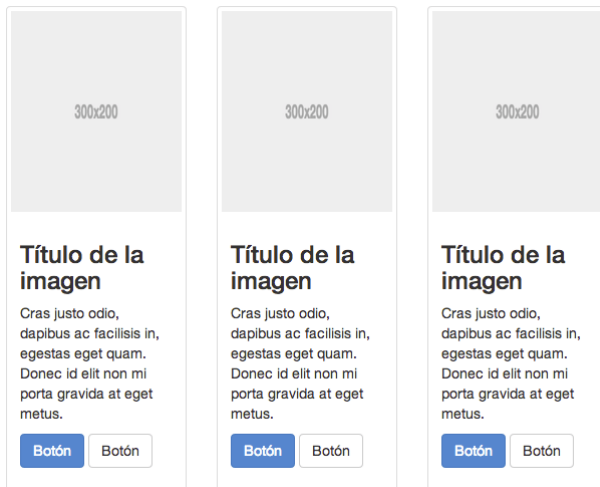


3.6.14.2 Añadiendo contenido a las imágenes en miniatura

Añadiendo un poco de código HTML es posible añadir títulos, párrafos, botones o cualquier otro elemento a cada imagen en miniatura. Ejemplo:

```
<div class="row">
  <div class="col-sm-6 col-md-4">
    <div class="thumbnail">
      
      <div class="caption">
        <h3>Titulo de la imagen</h3>
        <p>...</p>
        <p>
          <a href="#" class="btn btn-primary" role="button">Botón</a>
          <a href="#" class="btn btn-default" role="button">Botón</a>
        </p>
      </div>
    </div>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.15 Mensajes de alerta

Este tipo de mensajes se utilizan normalmente para proporcionar al usuario información contextual sobre el resultado de sus acciones.

3.6.15.1 Ejemplo sencillo

Para mostrar un mensaje de alerta, encierra dentro de un elemento con la clase `.alert` tanto el texto del mensaje como el botón opcional para ocultar la alerta. Además de esta clase base, también debes aplicar cualquiera de las otras cuatro clases `.alert-*` para indicar explícitamente el tipo de mensaje (advertencia, error, éxito, información).

NOTA Los mensajes de alerta no definen un estilo por defecto. Por eso es obligatorio indicar siempre tanto la clase base `.alert` como uno de sus modificadores (`.alert-success`, etc.) El motivo es que en la práctica no tiene sentido mostrar un mensaje de alerta *neutro* (por ejemplo, de color gris claro), ya que las alertas siempre son de algún tipo (advertencia, error, éxito, información).

Ejemplo:

```
<div class="alert alert-success">...</div>
<div class="alert alert-info">...</div>
<div class="alert alert-warning">...</div>
<div class="alert alert-danger">...</div>
```

Así se ve este ejemplo en tu navegador:

¡Bien hecho! Has leído correctamente este mensaje tan importante.

¡Atento! Este mensaje requiere de tu atención, aunque no es tan importante.

¡Cuidado! Es muy importante que leas este mensaje de alerta.

¡Error! Haz algunos cambios antes de volver a enviar el formulario.

3.6.15.2 Cerrando los mensajes de alerta

Opcionalmente los mensajes de alerta pueden mostrar un botón de cierre para ocultar la alerta. Para que funcione este comportamiento, es necesario incluir el plugin `alerts` de JavaScript. Ejemplo:

```
<div class="alert alert-warning alert-dismissible">
  <button type="button" class="close" data-dismiss="alert">&times;</button>
  <strong>¡Cuidado!</strong> Es muy importante que leas este mensaje de alerta.
</div>
```

Así se ve este ejemplo en tu navegador (pulsas sobre la **X** de la derecha para cerrar el mensaje):

¡Cuidado! Es muy importante que leas este mensaje de alerta.

No olvides añadir el atributo `data-dismiss="alert"` al botón que cierra la alerta para que funcione bien en todos los dispositivos.

3.6.15.3 Añadiendo enlaces a las alertas

Si el mensaje de la alerta contiene enlaces, resulta conveniente aplicarles la clase `.alert-link` para que su aspecto se adapte al del resto del mensaje. Ejemplo:

```
<div class="alert alert-success">
  <a href="#" class="alert-link">...</a>
</div>
```

```
<div class="alert alert-info">
  <a href="#" class="alert-link">...</a>
</div>
```

```
<div class="alert alert-warning">
  <a href="#" class="alert-link">...</a>
</div>

<div class="alert alert-danger">
  <a href="#" class="alert-link">...</a>
</div>
```

Así se ve este ejemplo en tu navegador:

¡Bien hecho! Has leído correctamente **este mensaje tan importante.**

¡Atento! Este mensaje **requiere de tu atención**, aunque no es tan importante.

¡Cuidado! Es muy importante que leas **este mensaje de alerta.**

¡Error! Haz **algunos cambios** antes de volver a enviar el formulario.

3.6.16 Barras de progreso

Las barras de progreso permiten mostrar de forma continua el estado de ejecución de una tarea.

3.6.16.1 Ejemplo sencillo

El siguiente ejemplo muestra el código HTML recomendado para crear la barra de progreso, que incluye los atributos necesarios para mejorar su accesibilidad:

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-valuenow="60"
    aria-valuemin="0" aria-valuemax="100" style="width: 60%;">
    <span class="sr-only">60% completado</span>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:

3.6.16.2 Barras de progreso contextuales

Como sucede con muchos otros componentes, Bootstrap 3 permite crear barras de progreso contextuales, que modifican su aspecto para adecuarlo al concepto de éxito, información, aviso o error.

Ejemplo:

```
<div class="progress">
  <div class="progress-bar progress-bar-success" role="progressbar"
    aria-valuenow="40" aria-valuemin="0" aria-valuemax="100"
    style="width: 40%">
    <span class="sr-only">40% completado (éxito)</span>
  </div>
</div>
```

```
<div class="progress">
  <div class="progress-bar progress-bar-info" role="progressbar"
    aria-valuenow="20" aria-valuemin="0" aria-valuemax="100"
    style="width: 20%">
    <span class="sr-only">20% completado</span>
  </div>
</div>
```

```
<div class="progress">
  <div class="progress-bar progress-bar-warning" role="progressbar"
    aria-valuenow="60" aria-valuemin="0" aria-valuemax="100"
    style="width: 60%">
    <span class="sr-only">60% completado (aviso)</span>
  </div>
</div>
```

```
<div class="progress">
  <div class="progress-bar progress-bar-danger" role="progressbar"
    aria-valuenow="80" aria-valuemin="0" aria-valuemax="100"
    style="width: 80%">
```

```
<span class="sr-only">80% completado (peligro / error)</span>
</div>
</div>
```

Así se ve este ejemplo en tu navegador:

3.6.16.3 Barras de progreso cebreadas

Las barras *cebreadas* (del inglés "*striped*") muestran un patrón de franjas oblicuas que modifican el aspecto básico de las barras de progreso por defecto. Como las franjas se crean con gradientes de CSS, este tipo de barras de progreso no funcionan en Internet Explorer 8. Ejemplo:

```
<div class="progress progress-striped">
  <div class="progress-bar progress-bar-success" role="progressbar"
    aria-valuenow="40" aria-valuemin="0" aria-valuemax="100"
    style="width: 40%">
    <span class="sr-only">40% completado (éxito)</span>
  </div>
</div>
```

3.6.16.4 Barras de progreso animadas

Las barras de progreso animadas van un paso más allá de las anteriores barras de progreso cebreadas, ya que añaden una animación a las franjas. Este tipo de barras de progreso son ideales para las aplicaciones web, ya que el usuario tiene en todo momento la sensación de que su tarea está siendo atendida. Ninguna de estas barras de progreso animadas funciona en Internet Explorer. Ejemplo:

```
<div class="progress progress-striped active">
  <div class="progress-bar" role="progressbar"
    aria-valuenow="45" aria-valuemin="0" aria-valuemax="100"
    style="width: 45%">
    <span class="sr-only">45% completado</span>
  </div>
</div>
```

3.6.16.5 Barras de progreso segmentadas

Bootstrap 3 también permite crear barras de progreso compuestas por múltiples segmentos. Para ello, crea una barra de progreso por cada segmento y después enciérralos todos dentro de un elemento con la clase `.progress`. Ejemplo:

```
<div class="progress">
  <div class="progress-bar progress-bar-success" style="width: 35%">
    <span class="sr-only">35% completado (éxito)</span>
  </div>
  <div class="progress-bar progress-bar-warning" style="width: 20%">
    <span class="sr-only">20% completado (aviso)</span>
  </div>
  <div class="progress-bar progress-bar-danger" style="width: 10%">
    <span class="sr-only">10% completado (peligro)</span>
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.17 Objetos multimedia

Bootstrap 3 define varios estilos genéricos para crear diferentes tipos de componentes formados por imágenes y texto, como por ejemplo los comentarios de un blog, los tweets, etc.

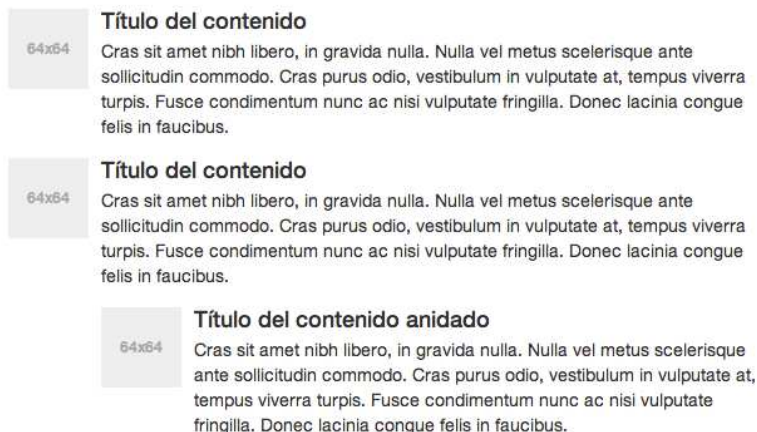
3.6.17.1 Ejemplo sencillo

El estilo por defecto para estos objetos alinea el elemento multimedia (imagen, vídeo , audio) a la derecha o a la izquierda del texto mediante la propiedad `float`. Ejemplo:

```
<div class="media">
  <a class="pull-left" href="#">
    
  </a>
  <div class="media-body">
    <h4 class="media-heading">Título del contenido</h4>
    ...
  </div>
```

</div>

Así se ve este ejemplo en tu navegador:



3.6.17.2 Listados de objetos multimedia

Aplicando un poco de código HTML resulta posible anidar los objetos multimedia dentro de una lista, lo que es muy útil por ejemplo para los comentarios de un blog y para los listados de artículos o noticias.

Ejemplo:

```
<ul class="media-list">
  <li class="media">
    <a class="pull-left" href="#">
      
    </a>
    <div class="media-body">
      <h4 class="media-heading">Título del contenido</h4>
      ...
    </div>
  </li>
</ul>
```

Así se ve este ejemplo en tu navegador:

64x64

Título del contenido
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

64x64

Título del contenido anidado
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

64x64

Título del contenido anidado
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

64x64

Título del contenido anidado
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

Título del contenido
Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin commodo. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis.

64x64

3.6.18 Listas de elementos

Las listas de elementos de Bootstrap 3 son componentes bastante flexibles y poderosos, ya que permiten mostrar listados que están formados por elementos complejos.

3.6.18.1 Ejemplo sencillo

La lista más sencilla está formada por un listado no ordenado de elementos con las clases CSS adecuadas. A partir de este listado básico, puedes añadir otros estilos y elementos, tal y como se explica en las siguientes secciones. Ejemplo:

```
<ul class="list-group">
  <li class="list-group-item">Cras justo odio</li>
  <li class="list-group-item">Dapibus ac facilisis in</li>
  <li class="list-group-item">Morbi leo risus</li>
  <li class="list-group-item">Porta ac consectetur ac</li>
  <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

Así se ve este ejemplo en tu navegador:

Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

3.6.18.2 Listas de elementos con badges

Al añadir un componente de tipo *badge* a cualquier elemento de una lista de objetos, el *badge* se alineará a la derecha automáticamente. Ejemplo:

```
<ul class="list-group">
  <li class="list-group-item">
    <span class="badge">14</span>
    Cras justo odio
  </li>
</ul>
```

Así se ve este ejemplo en tu navegador:

Cras justo odio	14
Dapibus ac facilisis in	2
Morbi leo risus	1

3.6.18.3 Listas de elementos con enlaces

Para crear una lista de elementos pinchables, reemplaza los elementos `` por elementos `<a>` y enciérralos dentro de un elemento `<div>` en vez de un elemento ``. Ejemplo:

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    Cras justo odio
  </a>
  <a href="#" class="list-group-item">Dapibus ac facilisis in</a>
  <a href="#" class="list-group-item">Morbi leo risus</a>
  <a href="#" class="list-group-item">Porta ac consectetur ac</a>
```

```
<a href="#" class="list-group-item">Vestibulum at eros</a>
</div>
```

Así se ve este ejemplo en tu navegador:

Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

3.6.18.4 Listas de elementos complejos

El componente de las listas de elementos es tan flexible que puedes incluir prácticamente cualquier código HTML, incluso cuando todos los elementos de la lista son enlaces. Ejemplo:

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    <h4 class="list-group-item-heading">Título del elemento de la lista</h4>
    <p class="list-group-item-text">...</p>
  </a>
</div>
```

Así se ve este ejemplo en tu navegador:

Título del elemento de la lista
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.
Título del elemento de la lista
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.
Título del elemento de la lista
Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit.

3.6.19 Paneles

En ocasiones es necesario encerrar cierto contenido dentro de una caja. En estos casos deberías considerar el uso del componente para paneles.

3.6.19.1 Ejemplo sencillo

Por defecto, al aplicar la clase `.panel` a un elemento solamente se añade un borde muy sencillo y algo de relleno con la propiedad `padding`. Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-body">
    Ejemplo de panel muy sencillo
  </div>
</div>
```

3.6.19.2 Panel con título

Cuando lo necesites, añade un título al panel aplicando al texto del título la clase `.panel-heading`. Opcionalmente puedes utilizar cualquiera de las etiquetas de título (`<h1>`, ..., `<h6>`) para aplicar también esos estilos. Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-heading">Título del panel con estilo normal</div>
  <div class="panel-body">
    Contenido del panel
  </div>
</div>
```

```
<div class="panel panel-default">
  <div class="panel-heading">
    <h3 class="panel-title">Título del panel con estilo h3</h3>
  </div>
  <div class="panel-body">
    Contenido del panel
  </div>
</div>
```

Así se ve este ejemplo en tu navegador:



3.6.19.3 Panel con pie

Los paneles también pueden contener un pie o zona inferior donde incluir por ejemplo los botones de acción o cualquier otro texto secundario. Estas zonas inferiores no heredan los colores ni los estilos cuando se utilizan los paneles contextuales que se explican en las siguientes secciones. Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-body">
    Contenido del panel
  </div>
  <div class="panel-footer">Pie del panel</div>
</div>
```

3.6.19.4 Paneles contextuales

Al igual que sucede con la mayoría de componentes, los paneles pueden adaptar su estilo al tipo de información que muestran mediante las variantes contextuales definidas por Bootstrap 3. Ejemplo:

```
<div class="panel panel-primary">...</div>
<div class="panel panel-success">...</div>
<div class="panel panel-info">...</div>
<div class="panel panel-warning">...</div>
<div class="panel panel-danger">...</div>
```

Así se ve este ejemplo en tu navegador:



3.6.19.5 Paneles con tablas

Si incluyes dentro de un panel cualquier tabla sin borde y con la clase `.table`, sus contenidos se adaptarán perfectamente al panel. Si además existe un elemento con la clase `.panel-body`, se añade un borde en la parte superior de la tabla para mejorar su separación. Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-heading">Título del panel</div>
  <div class="panel-body">
    <p>...</p>
  </div>

  <table class="table">
    ...
  </table>
</div>
```

Así se ve este ejemplo en tu navegador:

Título del panel			
Aquí va el contenido por defecto del panel. Nulla vitae elit libero, a pharetra augue. Aenean lacinia bibendum nulla sed consectetur. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nullam id dolor id nibh ultricies vehicula ut id elit.			
#	Nombre	Apellido	Nombre de usuario
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

3.6.19.6 Paneles con listas de elementos

Los paneles también están preparados para mostrar correctamente cualquier lista de elementos. Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-heading">Título del panel</div>
  <div class="panel-body">
    <p>...</p>
  </div>

  <ul class="list-group">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Morbi leo risus</li>
    <li class="list-group-item">Porta ac consectetur ac</li>
    <li class="list-group-item">Vestibulum at eros</li>
  </ul>
</div>
```

Así se ve este ejemplo en tu navegador:

Título del panel
Aquí va el contenido por defecto del panel. Nulla vitae elit libero, a pharetra augue. Aenean lacinia bibendum nulla sed consectetur. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nullam id dolor id nibh ultricies vehicula ut id elit.
Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

3.6.20 Pozos

3.6.20.1 Ejemplo sencillo

El estilo por defecto de este componente simplemente aplica un efecto de tipo *hundido* (en inglés, *"inset"*) al elemento. Ejemplo:

```
<div class="well">...</div>
```

Así se ve este ejemplo en tu navegador:

¡Este texto está dentro de un "pozo" y por eso parece que está hundido!

3.6.20.2 Estilos alternativos

Los pozos definen dos clases modificadoras para añadir esquinas redondeadas y para controlar el espacio de relleno. Ejemplo:

```
<div class="well well-lg">...</div>
```

Capítulo 4

JavaScript & JQuery

4.1 Introducción básica

JavaScript, al igual que Java o VRML, es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Al ser la más sencilla, es por el momento la más extendida. Antes que nada conviene aclarar un par de cosas:

- JavaScript no es un lenguaje de programación propiamente dicho. Es un lenguaje *script* u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto. Nunca podrás hacer un programa con JavaScript, tan sólo podrás mejorar tu página Web con algunas cosas sencillas (revisión de formularios, efectos en la barra de estado, etc...) y, ahora, no tan sencillas (animaciones usando HTML dinámico, por ejemplo).
- JavaScript y Java son dos cosas distintas. Principalmente porque Java sí que es un lenguaje de programación completo. Lo único que comparten es la misma sintaxis.

4.1.1 Primeros pasos

Vamos a realizar nuestro primer "programa" en JavaScript. Haremos surgir una ventana que nos muestre el mensaje "hola, mundo". Así podremos ver los elementos principales del lenguaje. El siguiente código es una página Web completa con un botón que, al pulsarlo, muestra el mensaje.

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    function HolaMundo() {
      alert("¡Hola, mundo!");
    }
  </SCRIPT>
</HEAD>
<BODY>
  <FORM>
    <INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
      onClick="HolaMundo()" ">
  </FORM>
</BODY>
```

```
</HTML>
```

Ahora vamos a ver, paso por paso, que significa cada uno de los elementos extraños que tiene la página anterior:

```
<SCRIPT LANGUAGE="JavaScript">
</SCRIPT>
```

Dentro de estas etiquetas será donde se insertará el código JavaScript. Puedes poner cuantos quieras a lo largo del documento y en el lugar que más te guste. Yo he elegido la cabecera para hacer más legible la parte HTML de la página. Si un navegador no acepta JavaScript no leerá el código que se encuentra dentro de estas etiquetas. Así que si programamos algo que sólo funcione con la versión 1.1 pondríamos `LANGUAGE= "JavaScript1.1"` para que los navegadores antiguos obvien el código y no se hagan un lío.

```
function HolaMundo() {
    alert(";Hola, mundo!");
}
```

Esta es nuestra primera función en JavaScript. Aunque JavaScript esté orientado a objetos no es de ningún modo tan estricto como Java, donde nada está fuera de un objeto. En el código de la función vemos una llamada al método `alert` (que pertenece al objeto `window`) que es la que se encarga de mostrar el mensaje en pantalla. Por un fallo del Netscape no se pueden poner las etiquetas HTML de caracteres especiales en una función: no los reconoce. Así que pondremos directamente `";"` arriesgándonos a que salga de otra manera en ordenadores con un juego de caracteres distinto al del nuestro.

```
<FORM>
<INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
onClick="HolaMundo()" ">
</FORM>
```

Dentro del elemento que usamos para mostrar un botón vemos una cosa nueva: `onClick`. Es un controlador de evento. Cuando el usuario pulsa el botón, el evento click se dispara y ejecuta el código que tenga entre comillas el controlador de evento `onClick`, en este caso la llamada a la función `HolaMundo()`, que tendremos que haber definido con anterioridad. Existen muchos más eventos que iremos descubriendo según avancemos en el curso. Más adelante se presentará un resumen de todos ellos.

En realidad, podríamos haber escrito lo siguiente:

```
<FORM>
<INPUT TYPE="button" NAME="Boton" VALUE="Pulsame"
```

```
onClick="alert(' &iexcl;Hola,Mundo! ') ">
```

```
</FORM>
```

y nos habríamos ahorrado el tener que escribir la función y todo lo que le acompaña, además de conseguir que nos reconozca el carácter especial “`¡`”. Sin embargo, me pareció conveniente hacerlo de esa otra manera para mostrar más elementos del lenguaje en el ejemplo.

4.2 Elementos básicos

Lo primero que vamos a ver son los elementos básicos del lenguaje, que nos van a servir para continuar el curso sin encontrar inconvenientes.

4.2.1 Comentarios

Lo primero (por ser lo más fácil) es indicar cómo se ponen los comentarios. Un comentario es una parte de nuestro programa que el compilador ignora y que, por tanto, no realiza ninguna tarea. Se utilizan generalmente para poner en lenguaje humano lo que estamos haciendo en el lenguaje de programación y así hacer que el código sea más comprensible.

En JavaScript existen dos tipos de comentarios. El primero nos permite que toda la línea sea un comentario. Para ello se utilizan dos barras inclinadas:

```
var i = 1; // Aqui esta el comentario
```

Sin embargo, también permite un tipo de comentario que puede tener las líneas que quisiéramos. Estos comentario comienzan con `/*` y terminan por `*/`. Por ejemplo:

```
/* Aqui comienza nuestro maravilloso comentario  
   que sigue por aquí  
   e indefinidamente hasta que le indiquemos el final */
```

4.2.2 Literales

Se llama así a los valores que pueden tomar una variable o una constante. Aparte de los distintos tipos de números y valores booleanos:

```
"Soy una cadena"  
3434  
3.43  
true, false
```

Las cadenas de texto se deberán cerrar entre comillas dobles o comillas simples. Es indiferente. Les recomiendo acostumbrarse a las comillas simples si se trata de código Javascript. Los valores numéricos o booleanos no necesitan encerrarse entre comillas.

También podemos especificar vectores:

```
vacaciones = ["Navidad", "Semana Santa", "Verano"];  
alert(vacaciones[0]);
```

Dentro de las cadenas podemos indicar varios caracteres especiales, con significados especiales. Estos son los más usados:

Carácter	Significado
<code>\n</code>	Salto de línea
<code>\t</code>	Tabulador
<code>\'</code>	Comilla simple
<code>\"</code>	Comilla doble
<code>\\</code>	Barra invertida
<code>\999</code>	El número ASCII (según la codificación Latin-1) del carácter en hexadecimal

De este modo, el siguiente literal:

```
"La bebida Fanta (\xA9 Coca-Cola) es \"dulce\"."
```

se corresponde con la cadena:

La bebida Fanta (© Coca-Cola) es "dulce".

Por último, también se pueden especificar objetos como literales, aunque solo funcionan en Netscape 4 y superiores:

```
miNavegador = {nombre: "Netscape", version: 4.5,
                idioma: "Español", plataforma: "PC"};
alert(miNavegador.plataforma);
```

4.2.3 Sentencias y bloques

En Javascript las sentencias se separan con un punto y coma, y se agrupan mediante llaves ({ y }).

```
function HolaMundo()
{
    alert("¡Hola, mundo!");
}
```

Diagram illustrating the structure of a JavaScript function block:

- Llaves de agrupamiento** (Grouping braces) points to the opening curly brace {.
- Comillas de cierre de sentencia** (Closing statement quote) points to the closing curly brace }.

4.3 Tipo de datos

Un tipo de datos es la clase de valores que puede tomar un identificador (es decir, una variable o una constante). Si el tipo de datos es fecha, el identificador que tenga ese tipo sólo podrá almacenar fechas. En Javascript los tipos de datos se asignan dinámicamente según asignamos valores a las distintas variables y son los clásicos: cadenas, varios tipos de enteros y reales, valores booleanos, vectores, matrices, referencias y objetos.

4.3.1 Variables

Las variables son nombres que ponemos a los lugares donde almacenamos la información. En Javascript, deben comenzar por una letra o un subrayado (`_`), pudiendo haber además dígitos entre los demás caracteres. No es necesario declarar una variable, pero cuando se hace es por medio de la palabra reservada `var`. Una variable, cuando no es declarada, tiene siempre ámbito global, mientras que en caso contrario será de ámbito global si está definida fuera de una función, y local si está definida dentro:

```
var x;          // Accesible fuera y dentro de pruebas
y = 2;         // Accesible fuera y dentro de pruebas
function pruebas() {
    var z;      // Accesible sólo dentro de pruebas
    w = 1;      // Accesible fuera y dentro de pruebas
}
```

Se pueden declarar varias variables en una misma sentencia separándolos por comas:

```
var x, y, z;
```

El tipo de datos de la variable será aquel que tenga el valor que asignemos a la misma, a no ser que le asignemos un objeto por medio del operador `new`. Por ejemplo, si escribimos:

```
b = 200;
```

Es de esperar que la variable `b` tenga tipo numérico.

4.3.2 Referencias

La parte sin duda más complicada de comprender y manejar en los lenguajes de programación tradicionales (y especialmente en C y C++) son los punteros. Por eso mismo (entre otras razones) fueron eliminados tanto en Java como en JavaScript. Sin embargo, algunas de sus capacidades han tenido que ser suplantadas con otras estructuras.

Los punteros se pueden usar para apuntar a otras variables, es decir, un puntero puede ser como un nuevo nombre de una variable dada. A esto se le suele llamar referencia. En JavaScript se pueden usar referencias a objetos y a funciones. Su mayor utilidad está en el uso de distinto código para distintos navegadores de forma transparente. Por ejemplo, supongamos que tenemos una función que sólo funciona en Internet Explorer 4, y tenemos una variable llamada IE4 que hemos puesto previamente a `true` (verdadero) sólo si el explorador del usuario es ese.

```
function funcionIE4() {  
    ...  
}  
  
function funcionNormal() {  
    ...  
}  
  
var funcion = (IE4) ? funcionIE4 : funcionNormal;  
    // Si IE4 es verdadero, funcion es una referencia de funcionIE4  
    // Si no, funcion es una referencia de funcionNormal  
funcion();  
    // La llamada que haremos realmente depende de la  
    // línea anterior
```

En este código, cuando llamemos finalmente a `funcion` en realidad llamaremos a la función a la que en la línea anterior hemos decidido que se refiera.

4.3.3 Vectores y matrices

Estos tipos de datos complejos son un conjunto ordenado de elementos, cada uno de los cuales es en sí mismo una variable distinta. En Javascript, los vectores y las matrices son objetos. Como veremos que hacen todos los objetos, se declaran utilizando el operador `new`:

```
miEstupendoVector = new Array(20)
```

El vector tendrá inicialmente 20 elementos (desde el 0 hasta el 19). Si queremos ampliarlo no tenemos más que asignar un valor a un elemento que esté fuera de los límites del vector:

```
miEstupendoVector[25] = "Algo"
```

De hecho, podemos utilizar de índices cualquier expresión que deseemos utilizar. Ni siquiera necesitamos especificar la longitud inicial del vector si no queremos:

```
vectorRaro = new Array();
```

```
vectorRaro["A colocar en los bookmark"] = "HTML en castellano";
```

Hacer una matriz bidimensional es más complicado, ya que tenemos que hacer un bucle que cree un vector nuevo en cada elemento del vector original.

4.4 Operadores

Los operadores nos permiten unir identificadores y literales para formar expresiones. Las expresiones son el resultado de operaciones matemáticas o lógicas. Un literal o una variable son expresiones, pero también lo son esos mismos literales y variables unidos entre sí mediante operadores.

JavaScript dispone de muchos más operadores que la mayoría de los lenguajes, si exceptuamos a sus padres C, C++ y Java. Algunos de ellos no los estudiaremos debido a su escasa utilidad y con algunos otros (especialmente el condicional) deberemos manejarlos con cuidado, ya que puede lograr que nuestro código no lo entendamos ni nosotros.

4.4.1 Operadores aritméticos

JavaScript dispone de los operadores aritméticos clásicos y algún que otro más:

Descripción	Símbolo	Expresión de ejemplo	Resultado del ejemplo
Multiplicación	*	2*4	8
División	/	5/2	2.5
Resto de una división entera	%	5 % 2	1
Suma	+	2+2	4
Resta	-	7-2	5
Incremento	++	++2	3
Decremento	--	--2	1
Menos unario	-	-(2+4)	-6

Los operadores de incremento y decremento merecen una explicación auxiliar. Se pueden colocar tanto antes como después de la expresión que deseemos modificar pero sólo devuelven el valor modificado si están delante.

```
a = 1;
```

```
b = ++a;
```

En este primer caso, **a** valdrá 2 y **b** valdrá 2 también. Sin embargo:

```
a = 1;
```

```
b = a++;
```

Ahora, **a** sigue valiendo 2, pero **b** es ahora 1. Es decir, estos operadores modifican siempre a su operando, pero si se colocan detrás del mismo se ejecutan después de todas las demás operaciones.

4.4.2 Operadores de comparación

Podemos usar los siguientes:

Descripción	Símbolo	Expresión de ejemplo	Resultado del ejemplo
Igualdad	<code>==</code>	<code>2 == '2'</code>	Verdadero
Desigualdad	<code>!=</code>	<code>2 != 2</code>	Falso
Igualdad estricta	<code>===</code>	<code>2 === '2'</code>	Falso
Desigualdad estricta	<code>!==</code>	<code>2 !== 2</code>	Falso
Menor que	<code><</code>	<code>2 < 2</code>	Falso
Mayor que	<code>></code>	<code>3 > 2</code>	Verdadero
Menor o igual que	<code><=</code>	<code>2 <= 2</code>	Verdadero
Mayor o igual que	<code>>=</code>	<code>1 >= 2</code>	Falso

La igualdad y desigualdad estricta son iguales a las normales, pero hacen una comprobación estricta de tipo. Han sido incluidos en el estándar ECMAScript y lo soportan Netscape 4.06 y superiores, como también Explorer 3 y superiores. Hay que indicar que versiones más antiguas de Netscape tratan la igualdad normal como si fuera estricta.

4.4.3 Operadores lógicos

Estos operadores permiten realizar expresiones lógicas complejas:

Descripción	Símbolo	Expresión de ejemplo	Resultado del ejemplo
Negación	!	!(2 = 2)	Falso
Y	&&	(2 = 2) && (2 >= 0)	Verdadero
Ó		(2 = 2) (2 <> 2)	Verdadero

4.4.4 Operadores de asignación

Normalmente los lenguajes tienen un único operador de asignación, que en JavaScript es el símbolo `=`. Pero en este lenguaje, dicho operador se puede combinar con operadores aritméticos y lógicos para dar los siguientes:

Operador	Significado	Operador	Significado
<code>x += y</code>	<code>x = x + y</code>	<code>x -= y</code>	<code>x = x - y</code>
<code>x /= y</code>	<code>x = x / y</code>	<code>x *= y</code>	<code>x = x * y</code>
<code>x % y</code>	<code>x = x % y</code>		

4.4.5 Operadores especiales

Vamos a incluir en este apartado operadores que no hayan sido incluidos en los anteriores. La concatenación de cadenas, por ejemplo, se realiza con el símbolo `+`. El siguiente ejemplo devolverá la variable `a` con el valor "Mi nombre es Ricardo".

```
nombre="Ricardo"
a= "Mi nombre es " + nombre
```

Un operador condicional tiene la siguiente estructura:

```
condicion ? valor1 : valor2
```

Si la condición se cumple devuelve el primer valor y, en caso contrario, el segundo. El siguiente ejemplo asignaría a la variable `a` un 2:

```
a = 2 > 3 ? 1 : 2
```

Como se puede ver no resulta muy legible. A veces resulta engorroso leer e interpretar el código con estas condiciones, más cuando anidamos unas cuantas de ellas.

Para tratar con objetos disponemos de tres operadores:

Descripción	Símbolo	Expresión de ejemplo	Resultado del ejemplo
Crear un objeto	<code>new</code>	<code>a = new Array()</code>	<code>a</code> es ahora un vector
Borrar un objeto	<code>delete</code>	<code>delete a</code>	Elimina el vector anteriormente creado
Referencia al objeto actual	<code>this</code>		

`this` se suele utilizar en el código de los métodos de un objeto para referirse a otros métodos o a propiedades del mismo objeto.

4.5 Estructuras de control

Ningún programa es una secuencia lineal de instrucciones. En todo lenguaje de programación existen estructuras que nos permiten variar el orden de ejecución dependiendo de ciertas condiciones. Estas estructuras se pueden clasificar en dos grandes grupos: bifurcaciones condicionales y bucles.

Aparte de los dos tipos clásicos de estructuras de control, en Javascript disponemos de algunas estructuras adicionales para facilitar el manejo de objetos. Dispone de algunas estructuras más de las que explicaremos en esta página (el soporte de etiquetas), pero debido a que no son más que un recuerdo de lenguajes desfasados y como su utilidad resulta escasa, he preferido no incluirlas.

4.5.1 Bifurcaciones condicionales

Una bifurcación condicional es una estructura que realiza una tarea u otra dependiendo del resultado de evaluar una condición. La primera que vamos a estudiar es la estructura `if...else`. Esta estructura es la más sencilla y antigua de todas:

```
if (bso.compositor == "Manuel")
    alert('Nombre Masculino');
else
    alert('Nombre Femenino');
```

Hay que indicar que el `else` es opcional, no siempre es necesario que realice una acción cuando la condición no se cumpla. Otra particularidad de esta estructura es que cuando tenemos un grupo de sentencias que ejecutar, se las encierra con una llave `{...}`.

```
if (bso.compositor == "Manuel")
{
    a='Hombre';
    alert('Nombre Masculino');
}
else
    alert('Nombre Femenino');
```

La siguiente estructura bifurca según los distintos valores que pueda tomar una variable específica y es una forma simplificada de representar varias condiciones. Es la sentencia `switch`:

```
switch(directorPreferido) {
    case "John Ford":
        alert('Eso es tener buen gusto, sí señor');
        break;
```

```
case "Joel Coen":  
    alert('Parece que te gustan las cosas raras');  
    break;  
default:  
    alert('¿Y ese quien es?');  
}
```

Hay que indicar que no es compatible con estándar ECMA y no es soportado por el Explorer 3.

4.5.2 Bucles

Un bucle es una estructura que permite repetir una tarea un número de veces, determinado por una condición. Para hacer bucles podemos utilizar las estructuras `while` y `do...while`. Estos bucles se ejecutan indefinidamente mientras se cumpla una condición. La diferencia entre ellas es que la primera comprueba dicha condición antes de realizar cada iteración y la segunda lo hace después:

```
var numero=0;  
while (numero==1) {  
    alert('Soy un while');  
}  
do {  
    alert('Soy un do...while');  
} while (numero==1);
```

En este caso solo veríamos aparecer una ventana diciendo que es un `do...while`. ¿Qué por qué? Veamos. El `while` comprueba primero si `numero` es igual a 1 y, como no lo es, no ejecutaría el código que tiene dentro del bucle. En cambio, el `do...while` primero ejecuta el código y luego, viendo que la condición es falsa, saldría. Hay que resaltar que `do...while` no pertenece al estándar y no es soportado por el Explorer 3.

En Javascript, el bucle `for` es singularmente potente. No se reduce a casos numéricos como en muchos otros lenguajes sino que nos da mucha más libertad. Tiene la siguiente estructura:

```
for (inicio; condición; incremento)  
    código
```

El código contenido en el bucle se ejecutará mientras la condición se cumpla. Antes de comenzar la primera iteración del bucle se ejecutará la sentencia `inicio` y en cada iteración lo hará `incremento`. La manera más habitual de usar estas posibilidades es, claro está, la numérica:

```
var numero = 4;  
for (n = 2, factorial = 1; n <= numero; n++)
```

```
factorial *= n;
```

Por último, hay que decir que la ejecución de la sentencia `break` dentro de cualquier parte del bucle provoca la salida inmediata del mismo. Aunque a veces no hay más remedio que utilizarlo, es mejor evitarlo para mejorar la legibilidad y elegancia del código.

4.5.3 Estructuras de manejo de objetos

JavaScript dispone de dos estructuras bien diferenciadas. La primera es el bucle `for...in`, que nos permitirá recorrer todas las propiedades de un objeto. Se usa principalmente con vectores. Por ejemplo:

```
var vector = [1, 2, 2, 5];  
for (i in vector)  
    vector[i] += 2;
```

Este ejemplo sumaría dos a todos los elementos del vector. Sin embargo, conviene tener cuidado ya que, de los navegadores de Microsoft, sólo la versión 5 lo soporta.

La otra estructura es `with`, que nos permitirá una mayor comodidad cuando tengamos que tratar con muchas propiedades de un mismo objeto. En lugar de tener que referirnos a todas ellas con un `objeto.propiedad` podemos hacer:

```
with (objeto) {  
    propiedad1 = ...  
    propiedad2 = ...  
    ...  
}
```

Que resulta más cómodo (tenemos que teclear menos) y legible.

4.6 Funciones

Incluso los programas más sencillos tienen la necesidad de dividirse. Las funciones son los únicos tipos de subprogramas que acepta JavaScript. Tienen la siguiente estructura:

```
function nombre(argumento1, argumento2,..., argumento n) {  
    código de la función  
}
```

Los parámetros se pueden pasar por valor o por referencia. Si no se especifica nada, por defecto es por valor. Esto significa que si cambiamos el valor de un argumento dentro de una función, este cambio no se verá afuera. En otras palabras, si pasamos como parámetro una variable y modificamos el valor de dicho parámetro dentro de la función, no se trasladarán los cambios a la variable original:

```
function sumarUno(num) {  
    num++;  
}  
  
var a = 1;  
sumarUno(a);
```

En este ejemplo, **a** seguirá valiendo 1 después de llamar a la función. Esto tiene una excepción, que son las referencias o parámetros por referencias. Cuando se cambia el valor de una referencia dentro de una función también se cambia afuera.

Para devolver un valor de retorno desde la función se utiliza la palabra reservada **return**:

```
function cuadrado(num) {  
    return num * num;  
}  
  
a = cuadrado(2);
```

En este ejemplo, **a** valdrá 4.

Se pueden definir funciones con un número variable de argumentos. Para poder luego acceder a dichos parámetros dentro de la función se utiliza el vector **arguments**. Este ejemplo sumaría el valor de todos los parámetros:

```
function sumarArgumentos() {  
    resultado = 0;  
    for (i=0; i<arguments.length; i++)  
        resultado += arguments[i];  
    return resultado;  
}
```

```
window.alert (sumarArgumentos(1,2,3,4));
```

EL resultado de dicho código, será la visualización de una ventana que mostrará la suma de todos los parámetros de la función. El resultado será el número 10.

4.6.1 Funciones predefinidas

JavaScript dispone de las siguientes funciones predefinidas:

```
eval(cadena)
```

Ejecuta la expresión o sentencia contenida en la cadena que recibe como parámetro.

```
mensaje = 'Hola';  
eval("alert('\" + mensaje + '\"');");
```

Este ejemplo nos muestra una ventana con un saludo.

```
parseInt(cadena [, base])
```

Convierte en un número entero la cadena que recibe, asumiendo que está en la base indicada. Si este parámetro falta, se asume que está en base 10 (decimal). Si fracasa en la conversión devolverá el valor NaN.

```
parseInt("3453");
```

Devuelve el número 3453.

```
parseFloat(cadena)
```

Convierte en un número real la cadena que recibe, devolviendo NaN si fracasa en el intento.

```
parseFloat("3.12.3");
```

Este ejemplo devuelve NaN ya que la cadena no contiene un número real válido.

```
isNaN(valor)
```

Devuelve true sólo si el argumento es NaN.

```
isFinite(numero)
```

Devuelve true si el número es un número válido y no es infinito.

```
Number(referencia)
```

Convierte a número el objeto que se les pase como argumento.

```
String(referencia)
```

Convierte a una cadena de caracteres (referencia) el objeto que se les pase como argumento.

4.7 Objetos

Un objeto es una entidad que contiene propiedades (variables) como los métodos que manipulan dichas propiedades (funciones). A partir de esta estructura se ha creado un nuevo modelo de programación (la programación orientada a objetos) que atribuye a las mismas propiedades como herencia o polimorfismo. Como veremos, JavaScript simplifica en algo este modelo.

El modelo de la programación orientada a objetos puede contener dos grupos: clases e instancias. Las primeras son entes más abstractos que definen un conjunto determinado de objetos. Las segundas son miembros de una clase, teniendo las mismas propiedades de la clase a la que pertenece. En Javascript esta distinción se difumina. Sólo tenemos objetos.

4.7.1 Propiedades y métodos

Por si acaso ya lo hemos olvidado, recordemos ahora cómo se accede a los métodos y propiedades de un objeto:

```
objeto.propiedad  
objeto.metodo(parametros)
```

4.7.2 Creación mediante constructores

Vamos a aprender a hacer nuestros propios objetos. Ya habíamos visto como hacerlo por medio de literales, pero dado que es una solución bastante propietaria y poco flexible, estudiaremos el método normal de lograrlo mediante constructores.

Un *constructor* es una función que inicializa un objeto. Cuando creamos un objeto nuevo del tipo que sea, lo que hacemos en realidad es llamar al constructor pasándole argumentos. Por ejemplo, si creamos un objeto Array de esta manera:

```
vector = new Array(9);
```

En realidad, estamos llamando a un constructor llamado Array que admite un parámetro. Sería algo así:

```
function Array(longitud) {  
    ...  
}
```

Vamos a crear nuestro primer objeto. Supongamos que queremos codificar en Javascript una aplicación que lleve nuestra biblioteca de libros técnicos de Informática. Para lograrlo, crearemos un objeto Libro que guarde toda la información de cada libro. Este sería el constructor:

```
function Libro(titulo, autor, tema) {  
    this.titulo = titulo;  
    this.autor = autor;  
    this.tema = tema;  
}
```

Como vemos, accederemos a las propiedades y métodos de nuestros objetos por medio de la referencia `this` (objeto con el cuál se está trabajando actualmente). Ahora podemos crear y acceder a nuestros objetos tipo Libro:

```
miLibro = new Libro("JavaScript Bible", "Danny Goodman", "JavaScript");  
alert(miLibro.autor);
```

Sencillo, ¿no? Sin embargo, para disfrutar de toda la funcionalidad de los objetos nos falta algo. Ese algo son los métodos. Vamos a incluir uno que nos saque una ventana con el contenido de las propiedades escrito:

```
function escribirLibro() {  
    alert("El libro " + this.titulo + " de " + this.autor +  
        " trata sobre " + this.tema);  
}
```

Para incluirlo en nuestro objeto añadimos la siguiente línea a nuestra función constructora:

```
this.escribir = escribirLibro;
```

Y podremos acceder al mismo de la manera normal:

```
miLibro.escribir();
```

Nuestro ejemplo nos quedaría de la siguiente manera:

```
function escribirLibro() {  
    alert("El libro " + this.titulo + " de " + this.autor +  
        " trata sobre " + this.tema);  
}  
function Libro(titulo, autor, tema) {  
    this.titulo = titulo;  
    this.autor = autor;  
    this.tema = tema;  
    this.escribir = escribirLibro;  
}
```

Constructor (objeto)

Método que trabaja sobre
las propiedades de nuestro

```
miLibro = new Libro("JavaScript Bible", "Danny Goodman", "JavaScript");  
miLibro.escribir();
```

4.7.3 Herencia

Una de las capacidades más empleadas de la programación orientada a objetos es la herencia. La herencia supone crear objetos nuevos y distintos que, aparte de los suyos propios, disponen de las propiedades y métodos de otro objeto, al que llamaremos padre.

Vamos a ver en nuestro ejemplo cómo se puede aplicar:

En la actualidad, en muchos libros de informática, se incluye un CD-ROM con ejemplos y a veces aplicaciones relacionadas con el tema del libro. Si quisiéramos tener también esa información nos sería difícil, ya que algunos libros tienen CD y otros no. Podríamos crear otro objeto Libro, pero tendríamos que reproducir el código (al menos del constructor) y si cambiáramos algo del mismo en el objeto Libro también tendríamos que hacerlo en el nuevo. Lo mejor será crear un objeto que herede las características de Libro y añada otras nuevas.

```
function LibroConCD (titulo, autor, tema, ejemplos, aplicaciones) {  
    this.base = Libro  
    this.base(titulo, autor, tema);  
    this.tieneEjemplos = ejemplos;  
    this.tieneAplicaciones = aplicaciones;  
}
```

Y podremos acceder a las propiedades de Libro sin mayores problemas:

```
miLibro = new LibroConCD("JavaScript Bible", "Danny Goodman",  
    "JavaScript", true, true);  
miLibro.escribir();
```

4.7.4 Polimorfismo

Existe otra manera de añadir métodos a nuestros objetos sin tener que tocar para nada el código del constructor ni, de hecho, el código del objeto ya escrito. Supongamos que queremos ampliar el objeto anterior añadiendo un método parecido a escribir() pero que nos indique algo sobre los atributos extra que tiene la nueva clase:

```
function escribirCD() {  
    if (this.tieneEjemplos && this.tieneAplicaciones)  
        alert("Este libro, " + this.titulo + ", tiene un CD completo.");  
}
```

```
else
    alert("El CD de este libro es incompleto.");
}
```

```
LibroConCD.prototype.escribirExtra = escribirCD;
```

Ahora sólo tendríamos que llamarlo con el objeto miLibro:

```
miLibro = new LibroConCD("JavaScript Bible", "Danny Goodman",
    "JavaScript", true, true);
miLibro.escribirExtra();
```

Como tenemos siempre acceso al código de nuestros objetos, el polimorfismo se usa, generalmente, como forma de añadir funcionalidad a los objetos predefinidos de Javascript.

4.8 Objetos predefinidos

JavaScript dispone de varios objetos predefinidos para acceder a muchas de las funciones normales de cualquier lenguaje, como puede ser el manejo de vectores o el de fechas. En algunos casos estaremos tratando con objetos aunque no nos demos cuenta, ya que los usos más habituales de los mismos disponen de abreviaturas que esconden el hecho de que sean objetos.

Este apartado no pretende ser una referencia completa, sino un resumen de las propiedades y métodos más usados. Si quieres más información consulta el manual de referencia de Javascript de Netscape.

4.8.1 Objeto Array

Como dijimos antes, este objeto permite crear vectores. Se inicializa de cualquiera de las siguientes maneras:

```
vector = new Array(longitud);  
vector = new Array(elemento1, elemento2, ..., elementoN);
```

En el primer caso crearemos un vector con el número especificado de elementos, mientras que en el segundo tendremos un vector que contiene los elementos indicados y de longitud N. Para acceder al mismo debemos recordar que el primero elemento es el número cero.

El objeto Array tiene, entre otros, los siguientes métodos y propiedades:

`length`

Propiedad que contiene el número de elementos del vector.

`concat(vector2)`

Añade los elementos de vector2 al final de los del vector que invoca el método, devolviendo el resultado. No funciona en Explorer 3 y no forma parte del estándar ECMA.

`sort(funcionComparacion)`

Ordena los elementos del vector alfabéticamente. Si se añade una función de comparación como parámetro los ordenará utilizando ésta. Dicha función (como se muestra en el siguiente ejemplo) debe aceptar dos parámetros y devolver 0 si son iguales, menor que cero si el primer parámetro es menor que el segundo y mayor que cero si es al revés.

```
function compararEnteros(a,b) {  
    return a<b ? -1 : (a==b ? 0 : 1);  
}
```

Usando esta función ordenaría numéricamente (y de menor a mayor) los elementos del vector.

4.8.2 Objeto Date

Este objeto nos permitirá manejar fechas y horas. Se invoca así:

```
fecha = new Date();  
fecha = new Date(año, mes, día);  
fecha = new Date(año, mes, día, hora, minuto, segundo);
```

Si no utilizamos parámetros, el objeto fecha contendrá la fecha y hora actuales, obtenidas del reloj del sistema. En caso contrario hay que tener en cuenta que los meses comienzan por cero. Así, por ejemplo:

```
navidad99 = new Date(1999, 11, 25)
```

El objeto Date dispone, entre otros, de los siguientes métodos:

```
getTime()  
setTime(milisegundos)
```

Obtienen y ponen, respectivamente, la fecha y la hora tomadas como milisegundos transcurridos desde el 1 de enero de 1970.

```
getFullYear()  
setYear(año)
```

Obtiene y pone, respectivamente, el año de la fecha. Éste se devuelven como números de 4 dígitos excepto en el caso en que estén entre 1900 y 1999, en cuyo caso se devolverán las dos últimas cifras. Hay que tener cuidado, ya que la implementación de éstos métodos puede variar en las últimas versiones de Netscape.

```
getFullYear()  
setFullYear(año)
```

Realizan la misma función que los anteriores, pero sin tantos líos, ya que siempre devuelven números con todos sus dígitos. Funciona en Explorer 4 y Netscape 4.06 y superiores.

```
getMonth()  
setMonth(mes)  
getDate()  
setDate(día)  
getHours()  
setHours(horas)  
getMinutes()  
setMinutes(minutos)  
getSeconds()  
setSeconds(segundos)
```

Obtienen y ponen, respectivamente, el mes, día, hora, minuto y segundo de la fecha.

```
getDay()
```

Devuelve el día de la semana de la fecha en forma de número que va del 0 (domingo) al 6 (sábado).

4.8.3 Objeto Math

Este objeto no está construido para que tengamos nuestras variables Math, sino como un contenedor donde meter diversas constantes (como Math.E y Math.PI) y los siguientes métodos matemáticos:

Método	Descripción	Expresión de ejemplo	Resultado
<code>abs</code>	Valor absoluto	<code>Math.abs(-2)</code>	2
<code>sin, cos, tan</code>	Funciones trigonométricas, reciben el argumento en radianes	<code>Math.cos(Math.PI)</code>	-1
<code>asin, acos, atan</code>	Funciones trigonométricas inversas	<code>Math.asin(1)</code>	1.57
<code>exp, log</code>	Exponenciación y logaritmo, base E	<code>Math.log(Math.E)</code>	1
<code>ceil</code>	Devuelve el entero más pequeño mayor o igual al argumento	<code>Math.ceil(-2.7)</code>	-2
<code>floor</code>	Devuelve el entero más grande menor o igual al argumento	<code>Math.floor(-2.7)</code>	-3
<code>round</code>	Devuelve el entero más cercano o igual al argumento	<code>Math.round(-2.7)</code>	-3
<code>min, max</code>	Devuelve el menor (o mayor) de sus dos argumentos	<code>Math.min(2,4)</code>	2
<code>pow</code>	Exponenciación, siendo el primer argumento la base y el segundo el exponente	<code>Math.pow(2,4)</code>	16
<code>sqrt</code>	Raíz cuadrada	<code>Math.sqrt(4)</code>	2

4.8.4 Objeto Number

Al igual que en el caso anterior, no se pueden crear objetos de tipo Number, sino que debemos referirnos al genérico. Este objeto contiene como propiedades los siguientes valores numéricos

Propiedad	Descripción
<code>NaN</code>	Valor que significa "no es un número"
<code>MAX_VALUE</code> y <code>MIN_VALUE</code>	Máximo y mínimo número representable
<code>NEGATIVE_INFINITY</code> <code>POSITIVE_INFINITY</code>	y Infinito negativo y positivo, se utilizan cuando hay desbordamiento al realizar alguna operación matemática

4.8.5 Objeto String

Este es un objeto que se puede confundir con los datos normales del tipo cadena. Conviene utilizar estos últimos, ya que los objetos String tienen un comportamiento extraño cuando se utilizan como cadenas normales. Además, al crear una cadena estamos creando a la vez un objeto String asociado. Su utilidad está en sus métodos, entre los que cabe destacar:

```
charAt(pos)  
charCodeAt(pos)
```

Devuelven el carácter o el código numérico del carácter que está en la posición indicada de la cadena. El último no funciona en Explorer 3.

```
indexOf(subcadena)
```

Devuelven la posición de la subcadena dentro de la cadena, o -1 en caso de no estar.

```
split(separador)
```

Devuelven un vector con subcadenas obtenidas separando la cadena por el carácter separador. No funciona en Explorer 3.

```
cadena = "Navidad,Semana Santa,Verano";  
vector = cadena.split(",");
```

En el ejemplo, el vector tendrá tres elementos con cada una de las vacaciones de un alumno escolar.

```
concat(cadena2)
```

Devuelve el resultado de concatenar cadena2 al final de la cadena. No funciona en Explorer 3 y no forma parte del estándar ECMA.

```
substr(indice, longitud)  
substring(indice1, indice2)
```

Devuelven una subcadena de la cadena, ya sea tomando un número de caracteres a partir de un índice o pillando todos los caracteres entre dos índices.

```
toLowerCase()  
toUpperCase()
```

Transforman la cadena a minúsculas y mayúsculas, respectivamente.

4.8.6 Objeto RegExp

Este objeto se utiliza para comprobar si las cadenas se ajustan a ciertos patrones formalizados como expresiones regulares. Funcionan en las versiones 4 y superiores de los dos navegadores. Resultan bastante complicadas de manejar, así que les dedicamos un apartado en exclusiva un poco más adelante.

4.8.7 Objeto Boolean

Objeto creado para crear confusiones a los programadores. Se supone que sirve para meter en un objeto los valores lógicos de verdadero y falso pero resulta que no se pueden utilizar de la manera habitual. Por eso no le daremos mucha importancia a este objeto.

4.8.8 Objeto Function

Sirve para crear funciones de manera ilegible y para que las referencias (que son siempre referencias a objetos) puedan usarse con funciones como en el ejemplo que vimos. La manera de crear funciones es la siguiente:

```
funcion = new Function([arg1, arg2, ..., argN], codigo);
```

Por ejemplo:

```
esElMejor = new Function([cadena], "return cadena=='Daniel' ? true : false");
```

Se lo dejo a su criterio según a sus conocimientos hasta el momento adquiridos, para que interprete el código anterior. No es difícil.

4.9 Eventos

Un evento, como su mismo nombre indica, es algo que ocurre en consecuencia a una acción ejecutada. Para que una rutina nuestra se ejecute sólo cuando suceda algo extraño deberemos llamarla desde un controlador de eventos. Estos controladores se asocian a un elemento HTML y se incluyen así:

```
<A HREF="http://home.netscape.com" onMouseOver="MiFuncion()" ">
```

4.9.1 Lista de eventos

Aquí tienes una pequeña guía de eventos definidos en JavaScript.

Evento	Descripción	Elementos que lo admiten
<code>onLoad</code>	Terminar de cargarse una página	<BODY...> <FRAMESET...>
<code>onUnLoad</code>	Salir de una página (descargarla)	<BODY...><FRAMESET...>
<code>onMouseOver</code>	Pasar el ratón por encima	<A HREF..> <AREA...>
<code>onMouseOut</code>	Que el ratón deje de estar encima	<A HREF..> <AREA...>
<code>onSubmit</code>	Enviar un formulario	<FORM...>
<code>onClick</code>	Pulsar un elemento	<INPUT TYPE="button, checkbox, link, radio"...>
<code>onBlur</code>	Perder el cursor	<INPUT TYPE="text"...> <TEXTAREA...>
<code>onChange</code>	Cambiar de contenido o perder el cursor	<INPUT TYPE="text"...> <TEXTAREA...>
<code>onFocus</code>	Conseguir el cursor	<INPUT TYPE="text"...> <TEXTAREA...>

onSelect	Seleccionar texto	<INPUT <TEXTAREA...>	TYPE="text"...>
----------	-------------------	-------------------------	-----------------

Como ejemplo, vamos a hacer que una ventana aparezca automáticamente en cuanto pasemos un cursor por encima de un elemento `<A>` (e impidiendo, de paso, que quien esté viendo la página pueda hacer uso del mismo).

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    <!-- Los comentarios esconden el código a navegadores sin JavaScript
    function Alarma() {
      alert("No me pises.");
      return true;
    }
    // -->
  </SCRIPT>
</HEAD>
<BODY>
<A HREF="eventos.html" onMouseOver="Alarma()" ">
  Pasa por aquí encima
</A>
</BODY>
</HTML>
```

4.9.2 Definición mediante código

Hemos visto como declarar un controlador de eventos desde etiquetas HTML. Sin embargo, y desde las versiones 3 de Netscape y 4 de Explorer, existe otro modo de hacerlo mediante código.

Muchos objetos cuyas etiquetas HTML correspondientes permiten atributos que definen controladores de evento, permiten el acceso a dichos controladores por medio de propiedades con el mismo nombre. Por ejemplo, la página:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    <!-- Los comentarios esconden el código a navegadores sin JavaScript
    function Alarma() {
```

```
    alert("Hola");  
  }  
  // -->  
</SCRIPT>  
</HEAD>  
<BODY onload="Saludo()">  
...  
</BODY>  
</HTML>
```

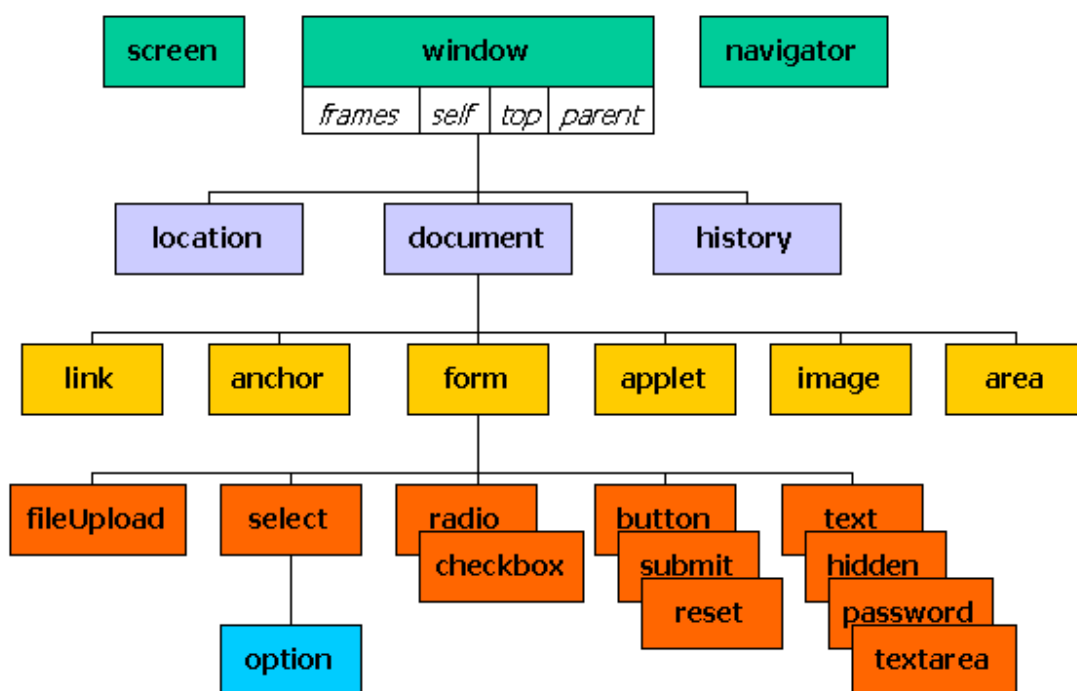
Se puede reescribir como:

```
<HTML>  
<HEAD>  
  <SCRIPT LANGUAGE="JavaScript">  
    <!-- Los comentarios esconden el código a navegadores sin JavaScript  
    function Saludo() {  
      alert("Hola");  
    }  
    window.onload = Saludo;  
    // -->  
  </SCRIPT>  
</HEAD>  
<BODY>  
...  
</BODY>  
</HTML>
```


4.10 Modelo de objetos del documento

El lenguaje JavaScript dispone de una serie de objetos que se refieren a cosas como la ventana actual, el documento sobre el que trabajamos, o el navegador que estamos utilizando. En los próximos apartados vamos a hacer un pequeño repaso de algunos de ellos con los métodos y propiedades más usados.

La jerarquía de dichos objetos toma la siguiente forma:



Para los más iniciados en la programación orientada a objetos, conviene aclarar que, en esta jerarquía, contra lo que pueda parecer, no existe herencia alguna. Los objetos se relacionan por composición, es decir, un objeto Window se compone (entre otras cosas) de un objeto Document, y éste a su vez se compone de diversos objetos Form, Image, etc.

El padre de esta jerarquía es el objeto Window, que representa una ventana de nuestro navegador. Dado que cada marco se considera una ventana distinta, cada uno de ellos dispone de su propio objeto Window. El objeto Document representa el documento HTML y cada uno de los objetos que lo componen se corresponde con diversas etiquetas HTML.

Capítulo 5

JQuery

5.1 Introducción básica

jQuery es un framework Javascript, pero quizás ustedes se preguntarán qué es un framework. Pues es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, framework son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

Por ejemplo, en el caso que nos ocupa, jQuery es un framework para el lenguaje Javascript, luego será un producto que nos simplificará la vida para programar en este lenguaje. Como probablemente sabremos, cuando un desarrollador tiene que utilizar Javascript, generalmente tiene que preocuparse por hacer scripts compatibles con varios navegadores y para ello tiene que incorporar mucho código que lo único que hace es detectar el browser del usuario, para hacer una u otra cosa dependiendo de si es Internet Explorer, Firefox, Opera, etc. jQuery es donde más nos puede ayudar, puesto que implementa una serie de clases (de programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales.

5.1.1 Ventajas de jQuery con respecto a otras alternativas

Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, que básicamente nos sirven para hacer lo mismo. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc.

5.2 Conceptos básicos

En este apartado vamos a explicar cómo comenzar a utilizar jQuery en tus páginas web, paso a paso, para que puedas hacer tu primer script jQuery y así comprender los fundamentos de uso de este framework Javascript.

5.2.1 Descarga la última versión del framework

Accede a la página de [jQuery](http://www.jquery.com) (www.jquery.com) para descargar la última versión del framework. En el momento de escribir esta unidad la versión actual es la 2.1.4, y con la que hemos realizado estos ejemplos. Sin embargo, puede que haya publicado una nueva versión que mejore la actual.

Dan dos posibilidades para descargar, una que le llaman PRODUCTION, que es la adecuada para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida. La otra posibilidad es descargar la versión que llaman DEVELOPMENT, que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos bucear en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework.

5.2.2 Crear una página HTML simple

Ahora, en el mismo directorio donde has colocado el archivo js, coloca un archivo html con el siguiente código.

```
<html>
  <head>
    <script src="jquery-2.1.4.min.js" type="text/javascript"></script>
    <script>

    </script>
  </head>
  <body>
    <a href="http://www.ejemplo.com/">Link</a>
  </body>
</html>
```

Como podrás ver, es una página bien simple, en la que tenemos una llamada a un script externo llamado jquery-2.1.4.min.js. Este es el código de jQuery que hemos descargado de la página del framework. Si has descargado una versión distinta, quizás el archivo se llame de otra manera, así que es posible que tengas que editar ese nombre de archivo para colocar el que tengas en el directorio.

Con ese script ya hemos incluido todas las funciones de jQuery dentro de nuestra página. Sólo tienes que prestar atención a que tanto el archivo .html de esta página, como el archivo .js del framework estén en el mismo directorio. Y, como decía, que el archivo que incluimos con la etiqueta SCRIPT sea el .js que nosotros hemos descargado.

Además, como se puede ver, hemos dejado dentro del HEAD una etiqueta SCRIPT de apertura y cierre que está vacía. Todo el código que vamos a explicar a continuación tendrás que colocarlo en dentro de esa etiqueta.

5.2.3 Ejecutar código cuando la página ha sido cargada

Se trata de detectar el momento en que la página está lista para recibir comandos Javascript que hacen uso del DOM.

Cuando hacemos ciertas acciones complejas con Javascript tenemos que estar seguros que la página haya terminado de cargar y esté lista para recibir comandos Javascript que utilicen la estructura del documento con el objetivo de cambiar cosas, como crear elementos, quitarlos, cambiar sus propiedades, etc. Si no esperamos a que la página esté lista para recibir instrucciones corremos un alto riesgo de obtener errores de Javascript en la ejecución.

Generalmente, cuando se desea ejecutar Javascript después de la carga de la página, si no utilizamos ningún framework, lo más normal será utilizar un código como este:

```
window.onload = function () {  
    alert("cargado...");  
}
```

Pero esta sentencia, que carga una funcionalidad en el evento onload del objeto window, sólo se ejecutará cuando el navegador haya descargado completamente Todos los elementos de la página, lo que incluye imágenes, iframes, banners, etc. lo que puede tardar bastante, dependiendo de los elementos que tenga esa página y su peso.

Pero en realidad no hace falta esperar todo ese tiempo de carga de los elementos de la página para poder ejecutar sentencias Javascript que alteren el DOM de la página. Sólo habría que hacerlo cuando el navegador ha recibido el código HTML completo y lo ha procesado al renderizar la página. Para ello, jQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo. Esto se hace con la siguiente sentencia.

```
$(document).ready(function(){  
    //código a ejecutar cuando el DOM está listo para recibir instrucciones.  
});
```

Existe una forma abreviada para `$(document).ready()` la cual podrá encontrar algunas veces; sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

```
$(function() {  
    console.log('el documento está preparado');  
});
```

Además es posible pasarle a `$(document).ready()` una función nombrada en lugar de una anónima:

```
function readyFn() {  
    // código a ejecutar cuando el documento este listo  
}  
  
$(document).ready(readyFn);
```

Por dar una explicación a este código, digamos que con `$(document)` se obtiene una referencia al documento (la página web) que se está cargando. Luego, con el método `ready()` se define un evento, que se desata al quedar listo el documento para realizar acciones sobre el DOM de la página.

5.2.4 Selección de elementos

El concepto más básico de jQuery es el de “seleccionar algunos elementos y realizar acciones con ellos”. La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados.

A continuación, se muestran algunas técnicas comunes para la selección de elementos:

Selección de elementos en base a su ID

```
$('#myId'); // notar que los IDs deben ser únicos por página
```

Selección de elementos en base al nombre de clase

```
$('.div.myClass'); // si se especifica el tipo de elemento,  
// se mejora el rendimiento de la selección
```

Selección de elementos por su atributo

```
$('input[name=first_name]'); // tenga cuidado, que puede ser muy lento
```

Selección de elementos en forma de selector CSS

```
$('#contents ul.people li');
```

Pseudo-selectores

```
$('#a.external:first'); // selecciona el primer elemento <a>
                        // con la clase 'external'

$('#tr:odd');           // selecciona todos los elementos <tr>
                        // impares de una tabla

$('#myForm :input');    // selecciona todos los elementos del tipo input
                        // dentro del formulario #myForm

$('#div:visible');      // selecciona todos los divs visibles

$('#div:gt(2)');        // selecciona todos los divs excepto los tres primeros

$('#div:animated');     // selecciona todos los divs actualmente animados
```

5.2.5 Comprobar Selecciones

Una vez realizada la selección de los elementos, querrá conocer si dicha selección entregó algún resultado. Para ello, pueda que escriba algo así:

```
if ($('#div.foo')) { ... }
```

Sin embargo, esta forma no funcionará. Cuando se realiza una selección utilizando `$()`, siempre es devuelto un objeto, y si se lo evalúa, éste siempre devolverá `true`. Incluso si la selección no contiene ningún elemento, el código dentro del bloque `if` se ejecutará.

En lugar de utilizar el código mostrado, lo que se debe hacer es preguntar por la cantidad de elementos que posee la selección que se ejecutó. Esto es posible realizarlo utilizando la propiedad JavaScript `length`. Si la respuesta es 0, la condición evaluará falso, caso contrario (más de 0 elementos), la condición será verdadera.

```
if ($('#div.foo').length) { ... }
```

5.2.6 Guardar Selecciones

Cada vez que se hace una selección, una gran cantidad de código es ejecutado. jQuery no guarda el resultado por sí solo, por lo tanto, si va a realizar una selección que luego se hará de nuevo, deberá salvar la selección en una variable.

```
var $divs = $('#div');
```

Una vez que la selección es guardada en la variable, se la puede utilizar en conjunto con los métodos de jQuery y el resultado será igual que utilizando la selección original.

5.2.7 Refinamiento y Filtrado de Selecciones

A veces, puede obtener una selección que contiene más de lo que necesita; en este caso, es necesario refinar dicha selección. jQuery ofrece varios métodos para poder obtener exactamente lo que desea.

```
$('#div.foo').has('p');           // el elemento div.foo contiene elementos <p>
$('#h1').not('.bar');             // el elemento h1 no posee la clase 'bar'
$('ul li').filter('.current');   // un item de una lista desordenada
                                // que posee la clase 'current'
$('ul li').first();              // el primer item de una lista desordenada
$('ul li').eq(5);                // el sexto item de una lista desordenada
```

5.2.8 Selección de Elementos de un Formulario

jQuery ofrece varios pseudo-selectores que ayudan a encontrar elementos dentro de los formularios, éstos son especialmente útiles ya que dependiendo de los estados de cada elemento o su tipo, puede ser difícil distinguirlos utilizando selectores CSS estándar.

`:button`

Selecciona elementos <button> y con el atributo type='button'

`:checkbox`

Selecciona elementos <input> con el atributo type='checkbox'

`:checked`

Selecciona elementos <input> del tipo checkbox seleccionados

`:disabled`

Selecciona elementos del formulario que están deshabilitados

`:enabled`

Selecciona elementos del formulario que están habilitados

`:file`

Selecciona elementos <input> con el atributo type='file'

`:image`

Selecciona elementos <input> con el atributo type='image'

`:input`

Selecciona elementos `<input>`, `<textarea>` y `<select>`

`:password`

Selecciona elementos `<input>` con el atributo `type='password'`

`:radio`

Selecciona elementos `<input>` con el atributo `type='radio'`

`:reset`

Selecciona elementos `<input>` con el atributo `type='reset'`

`:selected`

Selecciona elementos `<options>` que están seleccionados

`:submit`

Selecciona elementos `<input>` con el atributo `type='submit'`

`:text`

Selecciona elementos `<input>` con el atributo `type='text'`

5.2.9 Obtenedores (Getters) & Establecedores (Setters)

jQuery “sobrecarga” sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés setter). En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenido (en inglés getter).

El método `$.fn.html` utilizado como establecedor

```
$('h1').html('hello world');
```

El método `html` utilizado como obtenido

```
$('h1').html();
```

Los métodos establecedores devuelven un objeto jQuery, permitiendo continuar con la llamada de más métodos en la misma selección, mientras que los métodos obtenedores devuelven el valor por el cual se consultó, pero no permiten seguir llamando a más métodos en dicho valor.

5.2.10 CSS, Estilos, & Dimensiones

jQuery incluye una manera útil de obtener y establecer propiedades CSS a los elementos.

Obtener propiedades CSS

```
$('#h1').css('fontSize'); // devuelve una cadena de caracteres como "19px"
$('#h1').css('font-size'); // también funciona
```

Establecer propiedades CSS

```
$('#h1').css('fontSize', '100px'); // establece una propiedad individual CSS
$('#h1').css({
    'fontSize' : '100px',
    'color' : 'red'
}); // establece múltiples propiedades CSS
```

A partir de la versión 1.6 de la biblioteca, utilizando \$.fn.css también es posible establecer valores relativos en las propiedades CSS de un elemento determinado:

```
$('#h1').css({
    'fontSize' : '+=15px', // suma 15px al tamaño original del elemento
    'paddingTop' : '+=20px' // suma 20px al padding superior original del
elemento
});
```

5.2.11 Utilizar Clases para Aplicar Estilos CSS

Para obtener valores de los estilos aplicados a un elemento, el método \$.fn.css es muy útil, sin embargo, su utilización como método establecedor se debe evitar (ya que, para aplicar estilos a un elemento, se puede hacer directamente desde CSS). En su lugar, lo ideal, es escribir reglas CSS que se apliquen a clases que describan los diferentes estados visuales de los elementos y luego cambiar la clase del elemento para aplicar el estilo que se desea mostrar.

```
var $h1 = $('#h1');
$h1.addClass('big');
$h1.removeClass('big');
$h1.toggleClass('big');
if ($h1.hasClass('big')) { ... }
```

Las clases también pueden ser útiles para guardar información del estado de un elemento, por ejemplo, para indicar que un elemento fue seleccionado.

5.2.12 Dimensiones

jQuery ofrece una variedad de métodos para obtener y modificar valores de dimensiones y posición de un elemento.

```
$('h1').width('50px');    // establece el ancho de todos los elementos H1
$('h1').width();          // obtiene el ancho del primer elemento H1

$('h1').height('50px');   // establece el alto de todos los elementos H1
$('h1').height();         // obtiene el alto del primer elemento H1

$('h1').position();       // devuelve un objeto conteniendo
                           // información sobre la posición
                           // del primer elemento relativo al
                           // "offset" (posición) de su elemento padre
```

5.2.13 Atributos

Los atributos de los elementos HTML que conforman una aplicación pueden contener información útil, por eso es importante poder establecer y obtener esa información.

El método `$.fn.attr` actúa tanto como método establecedor como obtenedor. Además, al igual que el método `$.fn.css`, cuando se lo utiliza como método establecedor, puede aceptar un conjunto de palabra clave-valor o un objeto conteniendo más conjuntos.

```
$('a').attr('href', 'allMyHrefsAreTheSameNow.html');
$('a').attr({
    'title' : 'all titles are the same too',
    'href'  : 'somethingNew.html'
});
```

Obtener atributos

```
$('a').attr('href');    // devuelve el atributo href perteneciente
                        // al primer elemento <a> del documento
```

5.2.14 Recorrer el DOM

Una vez obtenida la selección, es posible encontrar otros elementos utilizando a la misma selección.

```
$('h1').next('p');      // seleccionar el inmediato y próximo
                        // elemento <p> con respecto a H1
```

```
$('#div:visible').parent(); // seleccionar el elemento contenedor
                             // a un div visible
$('#input[name=first_name]').closest('form'); // seleccionar el elemento
                                                // <form> más cercano a un input
$('#myList').children(); // seleccionar todos los elementos
                        // hijos de #myList
$('li.selected').siblings(); // seleccionar todos los items
                             // hermanos del elemento <li>
```

También es posible interactuar con la selección utilizando el método `$.fn.each`. Dicho método interactúa con todos los elementos obtenidos en la selección y ejecuta una función por cada uno. La función recibe como argumento el índice del elemento actual y al mismo elemento. De forma predeterminada, dentro de la función, se puede hacer referencia al elemento DOM a través de la declaración `this`.

5.2.15 Mover, Copiar y Remover Elementos

Existen varias maneras para mover elementos a través del DOM; las cuales se pueden separar en dos enfoques:

- querer colocar el/los elementos seleccionados de forma relativa a otro elemento;
- querer colocar un elemento relativo a el/los elementos seleccionados.

Por ejemplo, jQuery provee los métodos `$.fn.insertAfter` y `$.fn.after`. El método `$.fn.insertAfter` coloca a él/los elementos seleccionados después del elemento que se haya pasado como argumento; mientras que el método `$.fn.after` coloca al elemento pasado como argumento después del elemento seleccionado. Otros métodos también siguen este patrón: `$.fn.insertBefore` y `$.fn.before`; `$.fn.appendTo` y `$.fn.append`; y `$.fn.prependTo` y `$.fn.prepend`.

La utilización de uno u otro método dependerá de los elementos que tenga seleccionados y el tipo de referencia que se quiera guardar con respecto al elemento que se está moviendo.

```
// hacer que el primer item de la lista sea el último
var $li = $('#myList li:first').appendTo('#myList');

// otro enfoque para el mismo problema
$('#myList').append($('#myList li:first'));

// debe tener en cuenta que no hay forma de acceder a la
```

```
// lista de items que se ha movido, ya que devuelve  
// la lista en sí
```

5.2.15.1 Clonar Elementos

Cuando se utiliza un método como `$.fn.appendTo`, lo que se está haciendo es mover al elemento; pero a veces en lugar de eso, se necesita mover un duplicado del mismo elemento. En este caso, es posible utilizar el método `$.fn.clone`.

```
// copiar el primer elemento de la lista y moverlo al final de la misma  
$('#myList li:first').clone().appendTo('#myList');
```

5.2.15.2 Remover elementos

Existen dos formas de remover elementos de una página: Utilizando `$.fn.remove` o `$.fn.detach`. Cuando desee remover de forma permanente al elemento, utilice el método `$.fn.remove`. Por otro lado, el método `$.fn.detach` también remueve el elemento, pero mantiene la información y eventos asociados al mismo, siendo útil en el caso que necesite reinsertar el elemento en el documento.

Por otro lado, si se desea mantener al elemento pero se necesita eliminar su contenido, es posible utilizar el método `$.fn.empty`, el cual “vaciará” el contenido HTML del elemento.

5.2.16 Crear Nuevos Elementos

jQuery provee una forma fácil y elegante para crear nuevos elementos a través del mismo método `$()` que se utiliza para realizar selecciones.

Crear nuevos elementos

```
$('<p>Un nuevo párrafo</p>');  
$('<li class="new">nuevo item de la lista</li>');
```

Crear un nuevo elemento con atributos utilizando un objeto

```
$('<a/>', {  
    html : 'Un <strong>nuevo</strong> enlace',  
    'class' : 'new',  
    href : 'foo.html'  
});
```

Cuando se crea un elemento, éste no es añadido inmediatamente a la página, sino que se debe hacerlo en conjunto con un método.

Crear un nuevo elemento en la página

```
var $myNewElement = $('<p>Nuevo elemento</p>');
$myNewElement.appendTo('#content');

$myNewElement.insertAfter('ul:last'); // eliminará al elemento <p>
// existente en #content
$('ul').last().after($myNewElement.clone()); // clonar al elemento <p>
// para tener las dos versiones
```

Crear y añadir al mismo tiempo un elemento a la página

```
$('#ul').append('<li>item de la lista</li>');
```

5.2.17 Manipulación de Atributos

Las capacidades para la manipulación de atributos que ofrece la biblioteca son extensos. La realización de cambios básicos son simples, sin embargo el método \$.fn.attr permite manipulaciones más complejas.

Manipular un simple atributo

```
$('#myDiv a:first').attr('href', 'newDestination.html');
```

Manipular múltiples atributos

```
$('#myDiv a:first').attr({
    href : 'newDestination.html',
    rel : 'super-special'
});
```

Utilizar una función para determinar el valor del nuevo atributo

```
$('#myDiv a:first').attr({
    rel : 'super-special',
    href : function(idx, href) {
        return '/new/' + href;
    }
});

$('#myDiv a:first').attr('href', function(idx, href) {
```

```
return '/new/' + href;  
});
```

5.3 El núcleo de JQuery

5.3.1 \$ vs \$()

Hasta ahora, se ha tratado completamente con métodos que se llaman desde el objeto jQuery. Por ejemplo:

```
$('h1').remove();
```

Dichos métodos son parte del espacio de nombres (en inglés namespace) \$.fn, o del prototipo (en inglés prototype) de jQuery, y son considerados como métodos del objeto jQuery.

Sin embargo, existen métodos que son parte del espacio de nombres de \$ y se consideran como métodos del núcleo de jQuery.

Estas distinciones pueden ser bastantes confusas para usuarios nuevos. Para evitar la confusión, debe recordar estos dos puntos:

- los métodos utilizados en selecciones se encuentran dentro del espacio de nombres \$.fn, y automáticamente reciben y devuelven una selección en sí;
- métodos en el espacio de nombres \$ son generalmente métodos para diferentes utilidades, no trabajan con selecciones, no se les pasa ningún argumento y el valor que devuelven puede variar.

Existen algunos casos en donde métodos del objeto y del núcleo poseen los mismos nombres, como sucede con \$.each y \$.fn.each. En estos casos, debe ser cuidadoso de leer bien la documentación para saber que objeto utilizar correctamente.

5.3.2 Métodos Utilitarios

jQuery ofrece varios métodos utilitarios dentro del espacio de nombres \$. Estos métodos son de gran ayuda para llevar a cabo tareas rutinarias de programación. A continuación, se muestran algunos ejemplos, para una completa documentación sobre ellos, visite:

<http://api.jquery.com/category/utilities/>

\$.trim

Remueve los espacios en blanco del principio y final.

```
$.trim('    varios espacios en blanco    ');  
// devuelve 'varios espacios en blanco'
```

\$.each

Interactúa en vectores y objetos.

```
$.each([ 'foo', 'bar', 'baz' ], function(idx, val) {  
    console.log('elemento ' + idx + 'es ' + val);  
});
```

```
$.each({ foo : 'bar', baz : 'bim' }, function(k, v) {  
    console.log(k + ' : ' + v);  
});
```

\$.inArray

Devuelve el índice de un valor en un vector, o -1 si el valor no se encuentra en el vector.

```
var myArray = [ 1, 2, 3, 5 ];  
  
if ($.inArray(4, myArray) !== -1) {  
    console.log('valor encontrado');  
}
```

\$.extend

Cambia la propiedad del primer objeto utilizando las propiedades de los subsecuentes objetos.

```
var firstObject = { foo : 'bar', a : 'b' };  
var secondObject = { foo : 'baz' };  
  
var newObject = $.extend(firstObject, secondObject);  
console.log(firstObject.foo); // 'baz'  
console.log(newObject.foo);   // 'baz'
```

Si no se desea cambiar las propiedades de ninguno de los objetos que se utilizan en \$.extend, se debe incluir un objeto vacío como primer argumento.

```
var firstObject = { foo : 'bar', a : 'b' };  
var secondObject = { foo : 'baz' };  
  
var newObject = $.extend({}, firstObject, secondObject);  
console.log(firstObject.foo); // 'bar'  
console.log(newObject.foo);   // 'baz'
```

\$.proxy

Devuelve una función que siempre se ejecutará en el alcance (scope) provisto — en otras palabras, establece el significado de this (incluido dentro de la función) como el segundo argumento.

```
var myFunction = function() { console.log(this); };  
var myObject = { foo : 'bar' };
```

```
myFunction(); // devuelve el objeto window
```

```
var myProxyFunction = $.proxy(myFunction, myObject);  
myProxyFunction(); // devuelve el objeto myObject
```

Si se posee un objeto con métodos, es posible pasar dicho objeto y el nombre de un método para devolver una función que siempre se ejecuta en el alcance de dicho objeto.

```
var myObject = {  
    myFn : function() {  
        console.log(this);  
    }  
};
```

```
$('#foo').click(myObject.myFn); // registra el elemento DOM #foo  
$('#foo').click($.proxy(myObject, 'myFn')); // registra myObject
```

5.3.3 Comprobación de Tipos

Como se mencionó en el capítulo “JavaScript”, jQuery ofrece varios métodos útiles para determinar el tipo de un valor específico.

Comprobar el tipo de un determinado valor

```
var myValue = [1, 2, 3];
```

```
// Utilizar el operador typeof de JavaScript para comprobar tipos primitivos  
typeof myValue == 'string'; // falso (false)  
typeof myValue == 'number'; // falso (false)  
typeof myValue == 'undefined'; // falso (false)  
typeof myValue == 'boolean'; // falso (false)
```

```
// Utilizar el operador de igualdad estricta para comprobar valores nulos (null)
```

```
myValue === null; // falso (false)

// Utilizar los métodos jQuery para comprobar tipos no primitivos
jQuery.isFunction(myValue); // falso (false)
jQuery.isPlainObject(myValue); // falso (false)
jQuery.isArray(myValue); // verdadero (true)
jQuery.isNumeric(16); // verdadero (true). No disponible en versiones inferiores a
jQuery 1.7
```

5.3.4 El Método Data

A menudo encontrará que existe información acerca de un elemento que necesita guardar. En JavaScript es posible hacerlo añadiendo propiedades al DOM del elemento, pero esta práctica conlleva enfrentarse a pérdidas de memoria (en inglés *memory leaks*) en algunos navegadores. jQuery ofrece una manera sencilla para poder guardar información relacionada a un elemento, y la misma biblioteca se ocupa de manejar los problemas que pueden surgir por falta de memoria.

Guardar y recuperar información relacionada a un elemento

```
$('#myDiv').data('keyName', { foo : 'bar' });
$('#myDiv').data('keyName'); // { foo : 'bar' }
```

A través del método `$.fn.data` es posible guardar cualquier tipo de información sobre un elemento. Es difícil exagerar la importancia de este concepto cuando se está desarrollando una aplicación compleja.

5.4 Eventos

5.4.1 Introducción

jQuery provee métodos para asociar controladores de eventos (en inglés event handlers) a selectores. Cuando un evento ocurre, la función provista es ejecutada. Dentro de la función, la palabra clave `this` hace referencia al elemento en que el evento ocurre.

Para más detalles sobre los eventos en jQuery, puede consultar:

<http://api.jquery.com/category/events/>

La función del controlador de eventos puede recibir un objeto. Este objeto puede ser utilizado para determinar la naturaleza del evento o, por ejemplo, prevenir el comportamiento predeterminado de éste. Para más detalles sobre el objeto del evento, visite:

<http://api.jquery.com/category/events/event-object/>

5.4.2 Vincular Eventos a Elementos

jQuery ofrece métodos para la mayoría de los eventos — entre ellos `$.fn.click`, `$.fn.focus`, `$.fn.blur`, `$.fn.change`, etc. Estos últimos son formas reducidas del método `$.fn.on` de jQuery (`$.fn.bind` en versiones anteriores a jQuery 1.7). El método `$.fn.on` es útil para vincular (en inglés binding) la misma función de controlador a múltiples eventos, para cuando se desea proveer información al controlador de evento, cuando se está trabajando con eventos personalizados o cuando se desea pasar un objeto a múltiples eventos y controladores.

Vincular un evento utilizando un método reducido

```
$('.p').click(function() {  
    console.log('click');  
});
```

Vincular un evento utilizando el método `$.fn.on`

```
$('.p').on('click', function() {  
    console.log('click');  
});
```

Vincular un evento utilizando el método `$.fn.on` con información asociada

```
$('.input').on(  
    'click blur', // es posible vincular múltiples eventos al elemento  
    { foo : 'bar' }, // se debe pasar la información asociada como argumento
```

```
function(eventObject) {  
    console.log(eventObject.type, eventObject.data);  
    // registra el tipo de evento y la información asociada { foo : 'bar' }  
}  
);
```

5.4.3 Desvincular Eventos

Para desvincular (en ingles unbind) un controlador de evento, puede utilizar el método `$.fn.off` pasándole el tipo de evento a desconectar. Si se pasó como adjunto al evento una función nombrada, es posible aislar la desconexión de dicha función pasándola como segundo argumento.

Desvincular todos los controladores del evento click en una selección

```
$('p').off('click');
```

5.4.4 Vinculación de Múltiples Eventos

Muy a menudo, elementos en una aplicación estarán vinculados a múltiples eventos, cada uno con una función diferente. En estos casos, es posible pasar un objeto dentro de `$.fn.on` con uno o más pares de nombres claves/valores. Cada clave será el nombre del evento mientras que cada valor será la función a ejecutar cuando ocurra el evento.

Vincular múltiples eventos a un elemento

```
$('p').on({  
    'click': function() {  
        console.log('clickeado');  
    },  
    'mouseover': function() {  
        console.log('sobrepasado');  
    }  
});
```

5.4.5 El Objeto del Evento

Como se menciona en la introducción, la función controladora de eventos recibe un objeto del evento, el cual contiene varios métodos y propiedades. El objeto es comúnmente utilizado para prevenir la

acción predeterminada del evento a través del método `preventDefault`. Sin embargo, también contiene varias propiedades y métodos útiles:

`pageX, pageY`

La posición del puntero del ratón en el momento que el evento ocurrió, relativo a las zonas superiores e izquierda de la página.

`type`

El tipo de evento (por ejemplo "click").

`which`

El botón o tecla presionada.

`data`

Alguna información pasada cuando el evento es ejecutado.

`target`

El elemento DOM que inicializó el evento.

`preventDefault()`

Cancela la acción predeterminada del evento (por ejemplo: seguir un enlace).

`stopPropagation()`

Detiene la propagación del evento sobre otros elementos.

Por otro lado, la función controladora también tiene acceso al elemento DOM que inicializó el evento a través de la palabra clave `this`. Para convertir a dicho elemento DOM en un objeto jQuery (y poder utilizar los métodos de la biblioteca) es necesario escribir `$(this)`, como se muestra a continuación:

```
var $this = $(this);
```

Cancelar que al hacer click en un enlace, éste se siga

```
$('#a').click(function(e) {  
    var $this = $(this);  
    if ($this.attr('href').match('evil')) {  
        e.preventDefault();  
        $this.addClass('evil');  
    }  
});
```

```
}  
});
```

5.4.6 Funciones Auxiliares de Eventos

jQuery ofrece dos funciones auxiliares para el trabajo con eventos:

5.4.6.1 \$.fn.hover

El método \$.fn.hover permite pasar una o dos funciones que se ejecutarán cuando los eventos mouseenter y mouseleave ocurran en el elemento seleccionado. Si se pasa una sola función, esta será ejecutada en ambos eventos; en cambio si se pasan dos, la primera será ejecutada cuando ocurra el evento mouseenter, mientras que la segunda será ejecutada cuando ocurra mouseleave.

La función auxiliar hover

```
$('#menu li').hover(function() {  
    $(this).toggleClass('hover');  
});
```

5.4.6.2 \$.fn.toggle

Al igual que el método anterior, \$.fn.toggle recibe dos o más funciones; cada vez que un evento ocurre, la función siguiente en la lista se ejecutará. Generalmente, \$.fn.toggle es utilizada con solo dos funciones. En caso que utiliza más de dos funciones, tenga cuidado, ya que puede ser dificultar la depuración del código.

La función auxiliar toggle

```
$('p.expander').toggle(  
    function() {  
        $(this).prev().addClass('open');  
    },  
    function() {  
        $(this).prev().removeClass('open');  
    }  
);
```

5.5 Efectos

5.5.1 Introducción

Con jQuery, agregar efectos a una página es muy fácil. Estos efectos poseen una configuración predeterminada pero también es posible proveerles parámetros personalizados. Además, es posible crear animaciones particulares estableciendo valores de propiedades CSS.

Para una completa documentación sobre los diferentes tipos de efectos puede visitar la sección effects:

<http://api.jquery.com/category/effects/>

5.5.2 Efectos Incorporados en la Biblioteca

Los efectos más utilizados ya vienen incorporados dentro de la biblioteca en forma de métodos:

`$.fn.show`

Muestra el elemento seleccionado.

`$.fn.hide`

Oculto el elemento seleccionado.

`$.fn.fadeIn`

De forma animada, cambia la opacidad del elemento seleccionado al 100%.

`$.fn.fadeOut`

De forma animada, cambia la opacidad del elemento seleccionado al 0

`$.fn.slideDown`

Muestra el elemento seleccionado con un movimiento de deslizamiento vertical.

`$.fn.slideUp`

Oculto el elemento seleccionado con un movimiento de deslizamiento vertical.

`$.fn.slideToggle`

Muestra o oculta el elemento seleccionado con un movimiento de deslizamiento vertical, dependiendo si actualmente el elemento está visible o no.

Uso básico de un efecto incorporado

```
$( 'h1' ).show();
```

5.5.3 Cambiar la Duración de los Efectos

Con la excepción de \$.fn.show y \$.fn.hide, todos los métodos tienen una duración predeterminada de la animación en 400ms. Este valor es posible cambiarlo.

Configurar la duración de un efecto

```
$('#h1').fadeIn(300);           // desvanecimiento en 300ms
$('#h1').fadeOut('slow');      // utilizar una definición de velocidad interna
```

5.5.4 Realizar una Acción Cuando un Efecto fue Ejecutado

A menudo, querrá ejecutar una acción una vez que la animación haya terminado — ya que si ejecuta la acción antes que la animación haya acabado, puede llegar a alterar la calidad del efecto o afectar a los elementos que forman parte de la misma. [Definición: Las funciones de devolución de llamada (en inglés callback functions) proveen una forma para ejecutar código una vez que un evento haya terminado.] En este caso, el evento que responderá a la función será la conclusión de la animación. Dentro de la función de devolución, la palabra clave this hace referencia al elemento en donde el efecto fue ejecutado y al igual que sucede con los eventos, es posible transformarlo a un objeto jQuery utilizando \$(this).

Ejecutar cierto código cuando una animación haya concluido

```
$('#div.old').fadeOut(300, function() { $(this).remove(); });
```

Note que, si la selección no retorna ningún elemento, la función nunca se ejecutará. Este problema lo puede resolver comprobando si la selección devuelve algún elemento; y en caso que no lo haga, ejecutar la función de devolución inmediatamente.

Ejecutar una función de devolución incluso si no hay elementos para animar

```
var $thing = $('#nonexistent');

var cb = function() {
    console.log('realizado');
};

if ($thing.length) {
    $thing.fadeIn(300, cb);
} else {
    cb();
}
```


5.5.5 Control de los Efectos

jQuery provee varias herramientas para el manejo de animaciones.

\$.fn.stop

Detiene las animaciones que se están ejecutando en el elemento seleccionado.

\$.fn.delay

Espera un tiempo determinado antes de ejecutar la próxima animación.

```
$('h1').show(300).delay(1000).hide(300);
```

jQuery.fx.off

Si el valor es verdadero (true), no existirán transiciones para las animaciones; y a los elementos se le establecerá el estado final de la animación. Este método puede ser especialmente útil cuando se está trabajando con navegadores antiguos.

5.6 AJAX

5.6.1 Introducción

El método XMLHttpRequest (XHR) permite a los navegadores comunicarse con el servidor sin la necesidad de recargar la página. Este método, también conocido como Ajax (Asynchronous JavaScript and XML), permite la creación de aplicaciones ricas en interactividad.

Las peticiones Ajax son ejecutadas por el código JavaScript, el cual envía una petición a una URL y cuando recibe una respuesta, una función de devolución puede ser ejecutada la cual recibe como argumento la respuesta del servidor y realiza algo con ella. Debido a que la respuesta es asíncrona, el resto del código de la aplicación continúa ejecutándose, por lo cual, es imperativo que una función de devolución sea ejecutada para manejar la respuesta.

A través de varios métodos, jQuery provee soporte para Ajax, permitiendo abstraer las diferencias que pueden existir entre navegadores. Los métodos en cuestión son `$.get()`, `$.getScript()`, `$.getJSON()`, `$.post()` y `$.load()`.

A pesar que la definición de Ajax posee la palabra “XML”, la mayoría de las aplicaciones no utilizan dicho formato para el transporte de datos, sino que en su lugar se utiliza HTML plano o información en formato JSON (JavaScript Object Notation).

En general, Ajax no trabaja a través de dominios diferentes. Sin embargo, existen excepciones, como los servicios que proveen información en formato JSONP (JSON with Padding), los cuales permiten una funcionalidad limitada a través de diferentes dominios.

5.6.2 Conceptos Clave

La utilización correcta de los métodos Ajax requiere primero la comprensión de algunos conceptos clave.

5.6.2.1 GET vs. POST

Los dos métodos HTTP más comunes para enviar una petición a un servidor son GET y POST. Es importante entender la utilización de cada uno.

El método GET debe ser utilizado para operaciones no-destructivas — es decir, operaciones en donde se está “obteniendo” datos del servidor, pero no modificando. Por ejemplo, una consulta a un servicio de búsqueda podría ser una petición GET. Por otro lado, las solicitudes GET pueden ser almacenadas en la cache del navegador, pudiendo conducir a un comportamiento impredecible si no se lo espera.

Generalmente, la información enviada al servidor, es enviada en una cadena de datos (en inglés query string).

El método POST debe ser utilizado para operaciones destructivas — es decir, operaciones en donde se está incorporando información al servidor. Por ejemplo, cuando un usuario guarda un artículo en un blog, esta acción debería utilizar POST. Por otro lado, este tipo de método no se guarda en la cache del navegador. Además, una cadena de datos puede ser parte de la URL, pero la información tiende a ser enviada de forma separada.

5.6.2.2 Tipos de Datos

Generalmente, jQuery necesita algunas instrucciones sobre el tipo de información que se espera recibir cuando se realiza una petición Ajax. En algunos casos, el tipo de dato es especificado por el nombre del método, pero en otros casos se lo debe detallar como parte de la configuración del método:

text

Para el transporte de cadenas de caracteres simples.

html

Para el transporte de bloques de código HTML que serán ubicados en la página.

script

Para añadir un nuevo script con código JavaScript a la página.

json

Para transportar información en formato JSON, el cual puede incluir cadenas de caracteres, vectores y objetos.

Es recomendable utilizar los mecanismos que posea el lenguaje del lado de servidor para la generación de información en formato JSON.

jsonp

Para transportar información JSON de un dominio a otro.

xml

Para transportar información en formato XML.

A pesar de los diferentes tipos de datos de que se puede utilizar, es recomendable utilizar el formato JSON, ya que es muy flexible, permitiendo, por ejemplo, enviar al mismo tiempo información plana y HTML.

5.6.2.3 Asincronismo

Debido a que, de forma predeterminada, las llamadas Ajax son asíncronas, la respuesta del servidor no está disponible de forma inmediata. Por ejemplo, el siguiente código no debería funcionar:

```
var response;  
$.get('foo.php', function(r) { response = r; });  
console.log(response); // indefinido (undefined)
```

En su lugar, es necesario especificar una función de devolución de llamada; dicha función se ejecutará cuando la petición se haya realizado de forma correcta ya que es en ese momento cuando la respuesta del servidor esta lista.

```
$.get('foo.php', function(response) { console.log(response); });
```

5.6.2.4 Políticas de Mismo Origen y JSONP

En general, las peticiones Ajax están limitadas a utilizar el mismo protocolo (http o https), el mismo puerto y el mismo dominio de origen. Esta limitación no se aplica a los scripts cargados a través del método Ajax de jQuery.

La otra excepción es cuando se hace una petición que recibirá una respuesta en formato JSONP. En este caso, el proveedor de la respuesta debe responder la petición con un script que puede ser cargado utilizando la etiqueta <script>, evitando así la limitación de realizar peticiones desde el mismo dominio. Dicha respuesta contendrá la información solicitada, contenida en una función

5.6.2.5 Ajax y Firebug

Firebug (o el inspector WebKit que viene incluido en Chrome o Safari) son herramientas imprescindibles para trabajar con peticiones Ajax, ya que es posible observarlas desde la pestaña Consola de Firebug (o yendo a Recursos > Panel XHR desde el inspector de Webkit) y revisar los detalles de dichas peticiones. Si algo está fallando cuando trabaja con Ajax, este es el primer lugar en donde debe dirigirse para saber cuál es el problema.

5.6.3 Métodos Ajax de jQuery

Como se indicó anteriormente, jQuery posee varios métodos para trabajar con Ajax. Sin embargo, todos están basados en el método \$.ajax, por lo tanto, su comprensión es obligatoria. A continuación, se abarcará dicho método y luego se indicará un breve resumen sobre los demás métodos.

Generalmente, es preferible utilizar el método \$.ajax en lugar de los otros, ya que ofrece más características y su configuración es muy comprensible.

5.6.3.1 \$.ajax

El método \$.ajax es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar la petición. Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código.

Puede visitar <http://api.jquery.com/jquery.ajax/> para consultar la documentación sobre las opciones disponibles en el método.

Utilizar el método \$.ajax

```
$.ajax({  
    // la URL para la petición  
    url : 'post.php',  
  
    // la información a enviar  
    // (también es posible utilizar una cadena de datos)  
    data : { id : 123 },  
  
    // especifica si será una petición POST o GET  
    type : 'GET',  
  
    // el tipo de información que se espera de respuesta  
    dataType : 'json',  
  
    // código a ejecutar si la petición es satisfactoria;  
    // la respuesta es pasada como argumento a la función  
    success : function(json) {
```

```
$( '<h1/>' ).text( json.title ).appendTo( 'body' );
$( '<div class="content"/>' )
    .html( json.html ).appendTo( 'body' );
},

// código a ejecutar si la petición falla;
// son pasados como argumentos a la función
// el objeto jqXHR (extensión de XMLHttpRequest), un texto con el estatus
// de la petición y un texto con la descripción del error que haya dado el
servidor
error : function( jqXHR, status, error ) {
    alert( 'Disculpe, existió un problema' );
},

// código a ejecutar sin importar si la petición falló o no
complete : function( jqXHR, status ) {
    alert( 'Petición realizada' );
}
});
```

5.6.3.2 Opciones del método \$.ajax

El método \$.ajax posee muchas opciones de configuración, y es justamente esta característica la que hace que sea un método muy útil. Para una lista completa de las opciones disponibles, puede consultar <http://api.jquery.com/jQuery.ajax/>; a continuación, se muestran las más comunes:

async

Establece si la petición será asíncrona o no. De forma predeterminada el valor es true. Debe tener en cuenta que si la opción se establece en false, la petición bloqueará la ejecución de otros códigos hasta que dicha petición haya finalizado.

cache

Establece si la petición será guardada en la cache del navegador. De forma predeterminada es true para todos los dataType excepto para "script" y "jsonp". Cuando posee el valor false, se agrega una cadena de caracteres anti-cache al final de la URL de la petición.

complete

Establece una función de devolución de llamada que se ejecuta cuando la petición está completa, aunque haya fallado o no. La función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHttpRequest) y un texto especificando el estatus de la misma petición (success, notmodified, error, timeout, abort, o parsererror).

context

Establece el alcance en que la/las funciones de devolución de llamada se ejecutaran (por ejemplo, define el significado de this dentro de las funciones). De manera predeterminada this hace referencia al objeto originalmente pasado al método \$.ajax.

data

Establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo, foo=bar&baz=bim.)

dataType

Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME que posee la respuesta.

error

Establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHttpRequest), un texto especificando el estatus de la misma petición (timeout, error, abort, o parsererror) y un texto con la descripción del error que haya enviado el servidor (por ejemplo, Not Found o Internal Server Error).

jsonp

Establece el nombre de la función de devolución de llamada a enviar cuando se realiza una petición JSONP. De forma predeterminada el nombre es "callback"

success

Establece una función a ejecutar si la petición ha sido satisfactoria. Dicha función recibe como argumentos el objeto jqXHR (en versiones anteriores o iguales a jQuery 1.4, recibe en su lugar el objeto de la petición en crudo XMLHttpRequest), un texto especificando el estatus de la misma petición y la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma.

timeout

Establece un tiempo en milisegundos para considerar a una petición como fallada.

traditional

Si su valor es true, se utiliza el estilo de serialización de datos utilizado antes de jQuery 1.4. Para más detalles puede visitar <http://api.jquery.com/jquery.param/>.

type

De forma predeterminada su valor es "GET". Otros tipos de peticiones también pueden ser utilizadas (como PUT y DELETE), sin embargo, pueden no estar soportados por todos los navegadores.

url

Establece la URL en donde se realiza la petición.

La opción url es obligatoria para el método \$.ajax;