Bootstrap 3, el manual oficial

Capítulo 1.

Primeros pasos

Capítulo 2.

Diseñando con rejilla

Capítulo 3.

Tipografía

Capítulo 6.

Componentes

Capítulo 7.
Plugins de JavaScript

Capítulo 2. Diseñando con rejilla

2.1. Preparando la página

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

2.1.1. Se requiere el doctype de HTML5

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>
```

2.1.2. 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, maxim
um-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.

2.1.3. 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.

```
<img src="..." class="img-responsive" alt="Imagen responsive">
```

2.1.4. Tipografía y enlaces

Bootstrap establece una serie de estilos por defecto para la tipografía de todos los elementos y para los enlaces de la página. En concreto:

- Se establece a blanco el color de fondo del `body` con la propiedad `background-color: white;`
- Se utiliza el valor de las variables `@font-family-base`, `@font-size-base` y `@line-height-base` definidas por LESS como atributos de las propiedades tipográficas de los elementos.
- Se establece el color de los enlaces al valor de la variable `@link-color` de LESS y sólo se muestran los enlaces subrayados en el estado `:hover`

Esta primera inicialización de estilos se define en el archivo `scaffolding.less`.

2.1.5. Normalización de estilos

Para homogeneizar los estilos iniciales en los diferentes navegadores, Bootstrap utiliza la hoja de estilos Normalize, que es un proyecto creado por Nicolas Gallagher y Jonathan Neal.

2.1.6. 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>
```

La anchura 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.

2.2. Tipos de rejillas

Bootstrap incluye una rejila o retícula fluída 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 y también define *mixins* de LESS para que puedas crear diseños más semánticos.

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`.

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) { ... }
```

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	**Dispositivos pequeños** **Tablets**	**Dispositivos medianos** **Ordenadores**	**Dispositivos grandes** **Ordenadores**

	Teléfonos (<768px)	(≥768px)	(≥992px)	(≥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 entre columnas	`30px` (`15px` a cada lado de la columna)			
¿Permite anidación?	Si			

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

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>
  <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>
```

Ver este ejemplo en una nueva página

2.2.5. Ejemplo de contenedor de anchura 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>
```

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>
```

[Ver este ejemplo en una nueva página]

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>
```

Ver este ejemplo en una nueva página

← Anterior
Capítulo 2. Diseñando con rejilla

Siguiente →
2.3. Reseteando columnas

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>
```


← Anterior

2.2. Tipos de rejillas

Siguiente →

2.4. Desplazando columnas

INDICE DE CONTENIDOS

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>
```

Ver este ejemplo en una nueva página

← **Anterior**

2.3. Reseteando columnas

Siguiente →

2.5. Anidando columnas

INDICE DE CONTENIDOS

2.5. Anidando columnas

Bootsrap 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>
```

Ver este ejemplo en una nueva página

← **Anterior**

2.4. Desplazando columnas

Siguiente →

2.6. Reordenando las columnas

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>
```

Ver este ejemplo en una nueva página

← **Anterior**

2.5. Anidando columnas

Siguiente →

2.7. Variables y *mixins* de LESS

INDICE DE CONTENIDOS

2.7. Variables y mixins de LESS

Además de las clases CSS listas para definir rejillas rápidamente, Bootstrap incluye variables y *mixins* de LESS para generar fácilmente tus propios diseños web semánticos.

2.7.1. Variables

Las variables establecen el número de columnas, su separación y la anchura del navegador a partir de la cual las columnas flotan horizontalmente en vez de mostrarse una encima de otra. Los valores por defecto de estas variables son los que se muestran a continuación:

```
@grid-columns:              12;
@grid-gutter-width:         30px;
```

```
@grid-float-breakpoint: 768px;
```

2.7.2. Mixins

Los *mixins*, junto con las variables anteriores, permiten crear estilos semánticos para los diferentes elementos de la rejilla.

```
// Crea un elemento contenedor de varias columnas
.make-row(@gutter: @grid-gutter-width) {
  // Limpiar las columnas flotadas
  .clearfix();

  @media (min-width: @screen-small) {
    margin-left:  (@gutter / -2);
    margin-right: (@gutter / -2);
  }

  // Aplicar un margen negativo a la fila para alinear el
  // contenido de las columnas
  .row {
    margin-left:  (@gutter / -2);
```

```
    margin-right: (@gutter / -2);
  }
}

// Generar las columnas extra pequeñas
.make-xs-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Evitar que las columnas no se vean cuando están vacías
  min-height: 1px;
  // Utilizar padding para separar las columnas
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calcular la anchura en función del número de columnas
  @media (min-width: @grid-float-breakpoint) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generar las columnas pequeñas
```

```
.make-sm-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Evitar que las columnas no se vean cuando están vacías
  min-height: 1px;
  // Utilizar padding para separar las columnas
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calcular la anchura en función del número de columnas
  @media (min-width: @screen-small) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generate the small column offsets
.make-sm-column-offset(@columns) {
  @media (min-width: @screen-small) {
    margin-left: percentage((@columns / @grid-columns));
  }
}
```

```
.make-sm-column-push(@columns) {
  @media (min-width: @screen-small) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-sm-column-pull(@columns) {
  @media (min-width: @screen-small) {
    right: percentage((@columns / @grid-columns));
  }
}

// Generar las columnas medianas
.make-md-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Evitar que las columnas no se vean cuando están vacías
  min-height: 1px;
  // Utilizar padding para separar las columnas
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calcular la anchura en función del número de columnas
```

```
  @media (min-width: @screen-medium) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generar los desplazamientos de las columnas medianas
.make-md-column-offset(@columns) {
  @media (min-width: @screen-medium) {
    margin-left: percentage((@columns / @grid-columns));
  }
}
.make-md-column-push(@columns) {
  @media (min-width: @screen-medium) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-md-column-pull(@columns) {
  @media (min-width: @screen-medium) {
    right: percentage((@columns / @grid-columns));
  }
```

```
}

// Generar las columnas grandes
.make-lg-column(@columns; @gutter: @grid-gutter-width) {
  position: relative;
  // Evitar que las columnas no se vean cuando están vacías
  min-height: 1px;
  // Utilizar padding para separar las columnas
  padding-left:  (@gutter / 2);
  padding-right: (@gutter / 2);

  // Calcular la anchura en función del número de columnas
  @media (min-width: @screen-large) {
    float: left;
    width: percentage((@columns / @grid-columns));
  }
}

// Generar los desplazamientos de las columnas grandes
.make-lg-column-offset(@columns) {
  @media (min-width: @screen-large) {
```

```
    margin-left: percentage((@columns / @grid-columns));
  }
}
.make-lg-column-push(@columns) {
  @media (min-width: @screen-large) {
    left: percentage((@columns / @grid-columns));
  }
}
.make-lg-column-pull(@columns) {
  @media (min-width: @screen-large) {
    right: percentage((@columns / @grid-columns));
  }
}
```

2.7.3. Ejemplo de uso

Utilizando los *mixins* anteriores y modificando el valor de las variables para ajustarlos a tus necesidades, ya puedes crear diseños web semánticos. Este ejemplo muestra cómo crear un diseño a dos columnas con una separación entre los dos:

```
.wrapper {
```

```
    .make-row();
}
.content-main {
    .make-column(8);
}
.content-secondary {
    .make-column(3);
    .make-column-offset(1);
}
```

```
<div class="wrapper">
    <div class="content-main">...</div>
    <div class="content-secondary">...</div>
</div>
```


← Anterior

2.6. Reordenando las columnas

Siguiente →

Capítulo 3. Tipografía