Bootstrap 3

Introducción al diseño *Responsive*!

Antonio Javier Gallego Sánchez

Tabla de contenido

Contenidos

El diseño de webs tipo "responsive" permite crear webs adaptables a diferentes tamaños de pantalla, desde webs para pantallas tamaño escritorio, pasando por tablets, hasta webs para móviles. Este tipo de diseño se basa en crear un único código y utilizar reglas y estilos CSS para adaptar los contenidos a los diferentes tamaños de pantalla.

Los contenidos principales del libro son:

Introducción al diseño "responsive"

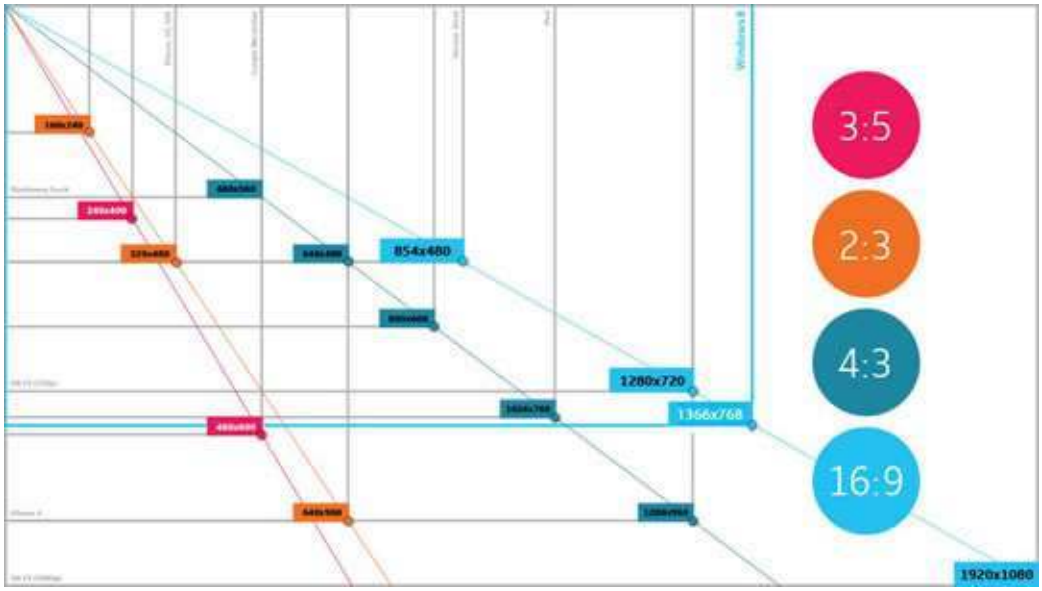
El diseño web *responsive*, adaptable o adaptativo, conocido por las siglas *RWD* (del inglés, *Responsive Web Design*) es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla. Hoy día las páginas web se visualizan en multitud de tipos de dispositivos como tabletas, *smartphones*, libros electrónicos, portátiles, PCs, etc. Esta tecnología pretende que con un solo diseño web tengamos una visualización adecuada en cualquier dispositivo.

El diseño *responsive* se basa en proporcionar a todos los usuarios de una web los mismos contenidos y una experiencia de usuario lo más similar posible, frente a otras aproximaciones al desarrollo web móvil como la creación de *apps*, el cambio de dominio o webs servidas dinámicamente en función del dispositivo.

Aunque todas tienen pros y contras, la web *responsive* es considerada por muchos expertos como la mejor práctica posible, al unificar la web, reducir tiempos de desarrollo y ofrecer grandes ventajas para SEO móvil.

Variabilidad en las resoluciones de pantalla

Durante muchos años el desarrollo web se ha basado en la resolución estándar de 1024×768 (hace apenas 3 años aproximadamente el 40% de los usuarios tenía esta resolución). Pero en la actualidad existe una amplia variedad de resoluciones, no solo en pantallas de ordenadores de escritorio sino también para *tablets* y dispositivos móviles.



Es muy importante conocer todas estas estadísticas así como cuales son las dimensiones de pantalla de los usuarios, a qué público vamos dirigidos, etc. y así poder tenerlo en cuenta en la usabilidad de nuestra web. Ya no es posible centrar el desarrollo pensando que los usuarios van a tener (en un alto porcentaje) una única resolución de pantalla.

Desde hace ya unos años en el desarrollo web se ha sustituido en cierta medida el problema de la compatibilidad de navegadores (gracias a que poco a poco todas las compañías se están ciñendo a los estándares con HTML5/CSS3 y otras se basan directamente en web-kit) por el problema de las resoluciones de los dispositivos.

En la siguiente tabla se pueden ver las últimas estadísticas (2014) de las resoluciones de pantalla más utilizadas:

Resolución	% utilización
> 1920x1080	34%
1920x1080	13%
1366x768	31%
1280x1024	8%
1280x800	7%
1024x768	6%
800×600	0.5%
< 800×600	0.5%

En la actualidad ya no es 1024x768 la resolución más utilizada, sino que es 1366×768 y resoluciones superiores a 1920x1080.

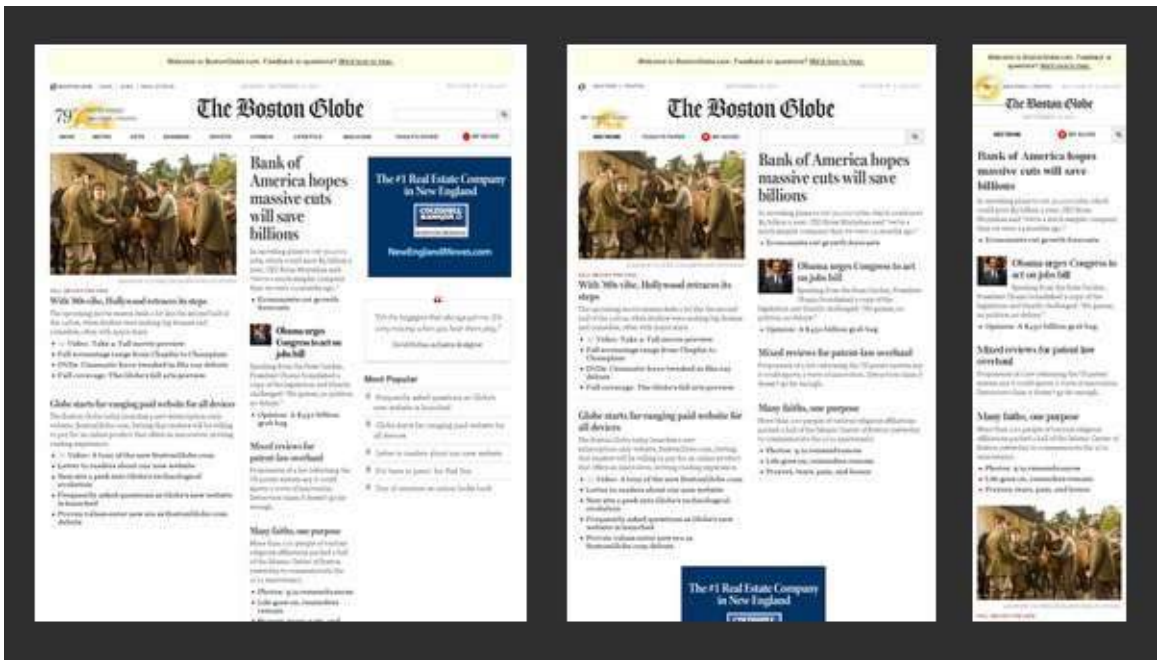
Es fundamental tener en cuenta que en el diseño *responsive*, al variar tanto las posibles resoluciones en las que se verá nuestra web deberemos mostrar en primer lugar los contenidos más importantes e imprescindibles.

Ejemplos de sitios web creados con tecnología *Responsive*

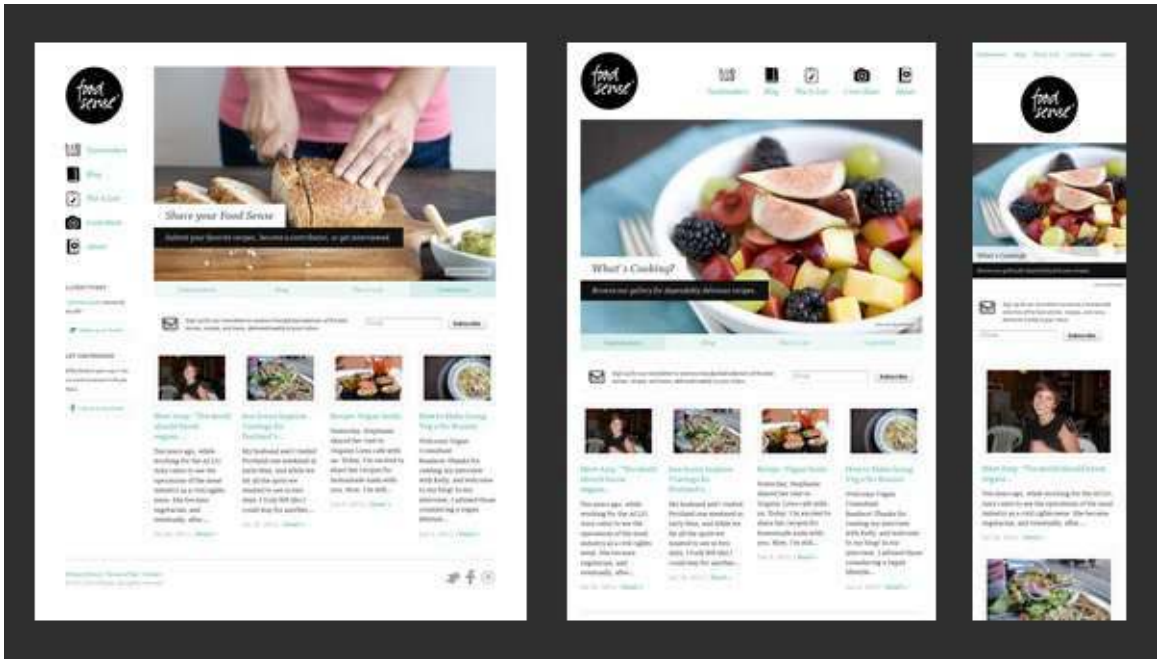
En un artículo llamado: "*Responsive Web Design: 50 Examples and Best Practices*" muestra excelentes ejemplos de la aplicación de esta tecnología. Algunos de estos ejemplos son:

dConstruct 2011

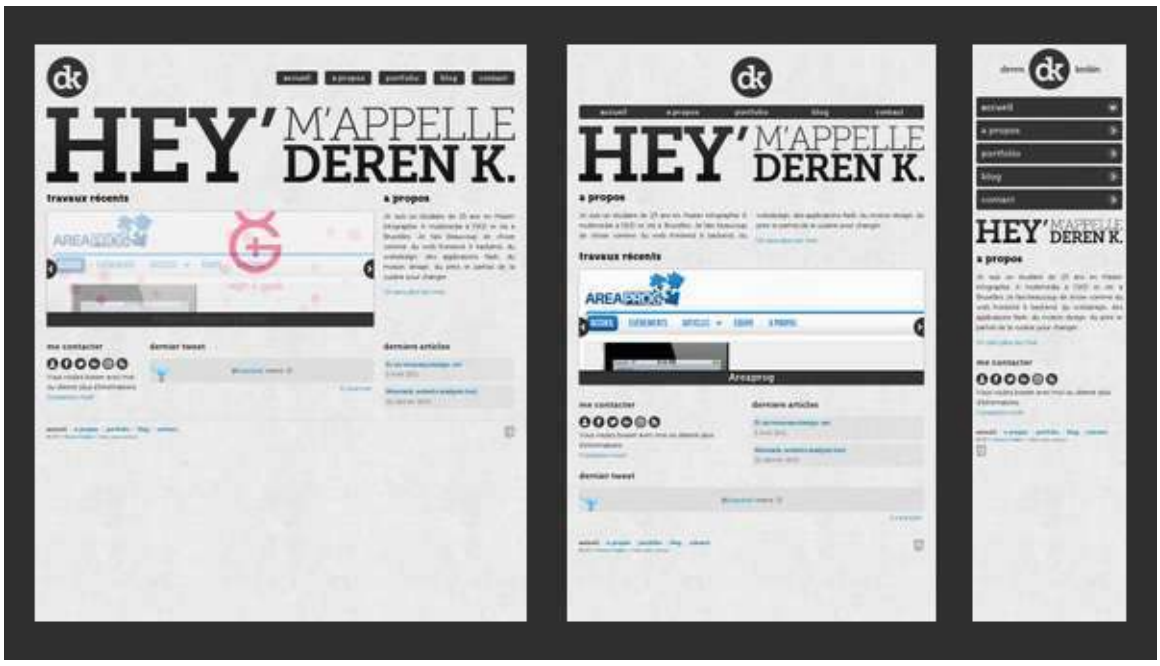
Boston Globe



Food Sense



Deren keskin



Frameworks responsive

Como se suele decir, en vez de reinventar la rueda y programar nosotros todo el diseño *responsive*, podemos aprovechar algunos de los *frameworks* que existen en el mercado para este propósito. Nos ahorrarán muchísimo tiempo, partiremos de código ampliamente probado, y de unos diseños base de todos los elementos web bastante más bonitos que la que tendrían de forma nativa.

Actualmente existen en el mercado una amplia variedad de este tipo *frameworks responsive*, algunos de los más utilizados son:

- **Bootstrap** (http://getbootstrap.com/): Este framework es uno de los más populares del mercado, habiendo sido desarrollado por el equipo de Twitter. Bootstrap ha sido creado pensando en ofrecer la mejor experiencia de usuario tanto a usuarios de PC (IE7 incluido!), como a smartphones y tabletas. Utiliza un grid responsive de 12 columnas y trae integrado decenas de complementos, plugins de JavaScript, tipografía, controladores de formularios y mucho más. Además utiliza el preprocesador de CSS LESS.
- **Foundation** (http://foundation.zurb.com/): Junto con Bootstrap es uno de los *frameworks* más avanzados que existen en la actualidad. Ha sido desarrollado con SASS, un potente preprocesador de CSS que hace de Foundation un *framework* fácilmente personalizable. Además saca partido de las nuevas tecnologías y funciona con IE8+.
- **Skeleton** (http://getskeleton.com/): Skeleton es un *boilerplate* que ofrece un grid responsive basado en una resolución de 960px que se ajusta al tamaño de los dispositivos móviles. Tiene poco peso e incluye una colección de archivos CSS y JS para facilitarnos el diseño de nuestra web.
- **HTML5 Boilerplate** (http://html5boilerplate.com/): Al igual que los demás nos ofrece un set de utilidades para construir nuestra web responsive de forma rápida y sencilla, con la ventaja de ser uno de los que menos ocupan.

En este curso nos vamos a centrar en **Bootstrap** por ser uno de los *frameworks* más completos, más utilizados y que mejor funcionan. En las siguientes secciones estudiaremos en detalle el funcionamiento de esta librería.

Bootstrap

Como ya hemos comentado antes, Bootstrap es uno de los *frameworks* más populares y utilizados del mercado para la creación de páginas *responsive*, habiendo sido desarrollado por el equipo de Twitter.

Entre los navegadores soportados se encuentran Chrome, Firefox, Opera, Safari e Internet Explorer a partir de la versión 8 (aunque en la versión 7 también funciona correctamente).

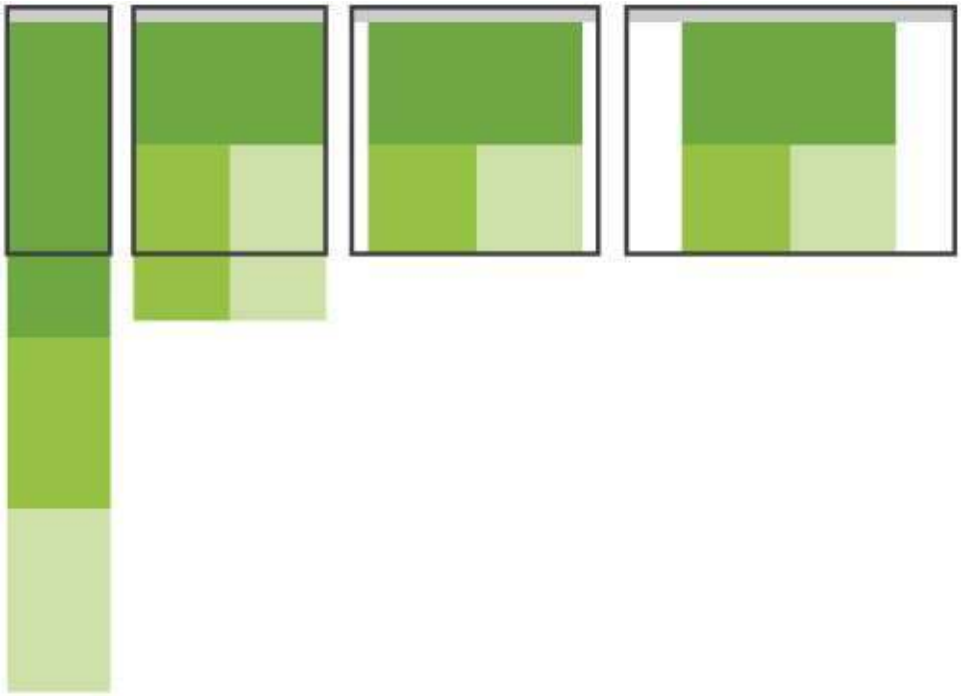
Está preparado para funcionar tanto en navegadores de PCs y portátiles con cualquier tamaño de pantalla así como para *tablets* y *smartphones* de tamaños mucho más reducidos.

Para conseguir que una misma web se pueda visualizar correctamente en todos esos tamaños de pantalla ha diseñado un avanzado sistema de rejilla dividido en columnas para el posicionamiento de los elementos de nuestra web. Además incorpora otras muchas utilidades y complementos (formularios, botones, barras de navegación, etc.) para simplificar el desarrollo de una web *responsive*.

Funcionamiento del diseño adaptable

El diseño *responsive* se basa en adaptar dinámicamente el diseño web en función de la resolución de la pantalla del visitante. De esta forma adaptamos nuestras webs a dispositivos móviles sin necesidad de tener dos sitios separados y al mismo tiempo también podemos adaptar la web a resoluciones grandes para mejorar la experiencia de usuario.

Antiguamente se pensaba en hacer 2 diseños, uno para móviles y otro para web, sin embargo, el diseño *responsive* trata de estructurar o adaptar el contenido que ya tienes en el diseño original a otros formatos diferentes: móviles, *tablets* y versión de escritorio, como bien muestra esta imagen:



La solución técnica que se le ha dado en el desarrollo web al problema de esta diversidad de resoluciones web se llama *Responsive Web Design* y nos permite hacer interfaces adaptadas al entorno del usuario mediante estructuras, bloques, columnas e imágenes fluidas gracias a *media-queries* de CSS.

A partir de CSS 2.1 las hojas de estilo han incluido los *media types*, lo cual nos ha facilitado, por ejemplo, proveer un estilo distinto para el diseño de impresión:

```
<link rel="stylesheet" type="text/css" href="core.css" media="screen" />
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

A partir de CSS 3 el W3C creó las *media queries*. Una media query nos permite apuntar no sólo a ciertas clases de dispositivos, sino realmente inspeccionar las características físicas del dispositivo que está renderizando nuestro trabajo. Para utilizarlas podemos incorporar una *query* al atributo media de un *link* a una hoja de estilos:

```
<link rel="stylesheet" type="text/css" href="shetland.css"
        media="screen and (max-device-width: 480px)" />
```

La *query* contiene dos componentes:

- Un media type (*screen*, *print* o *all*).
- La consulta entre paréntesis, conteniendo una característica a inspeccionar (max-device-width o min-device-width) seguida por el valor al que apuntamos (480px).

También es posible utilizarlas directamente en el CSS como parte de una regla `@media`:

```
@media screen and (max-device-width: 480px) {
    .column {
        float: none;
    }
}
```

Por ejemplo, si quisiéramos crear un estilo de bloques *fluidos* que para pantallas grandes se muestre uno a continuación del otro y para pantallas pantallas cambie a mostrarse de forma apilada, uno encima de otro, podríamos hacer algo como:

```
@media all and (max-width: 800px) {
    .bloque{
        display: block !important;
        /* Cuando el ancho sea inferior a 800px el elemento será un bloque */
        width: auto !important;
    }
}
.bloque {
    display: inline-block;    /* Para que se muestren los bloques en línea */
    height:300px;
    width: 300px;
    border:1px solid #333;
    background: #999;
    margin:20px;
}
```

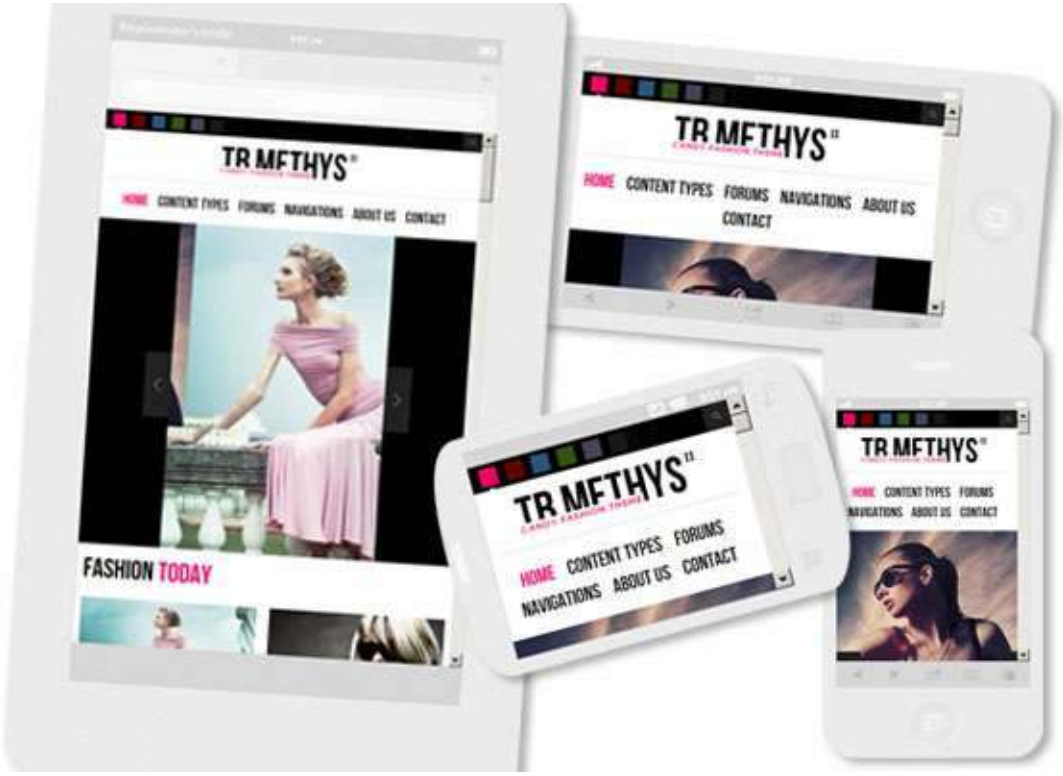
Para más información podéis consultar: http://www.w3.org/TR/css3-mediaqueries/

Probando el responsive

Para probar nuestros diseños *responsive* tenemos varias opciones, una de ellas es usar algunas de las webs que existen para tal fin. Como por ejemplo:

Responsinator (http://www.responsinator.com)

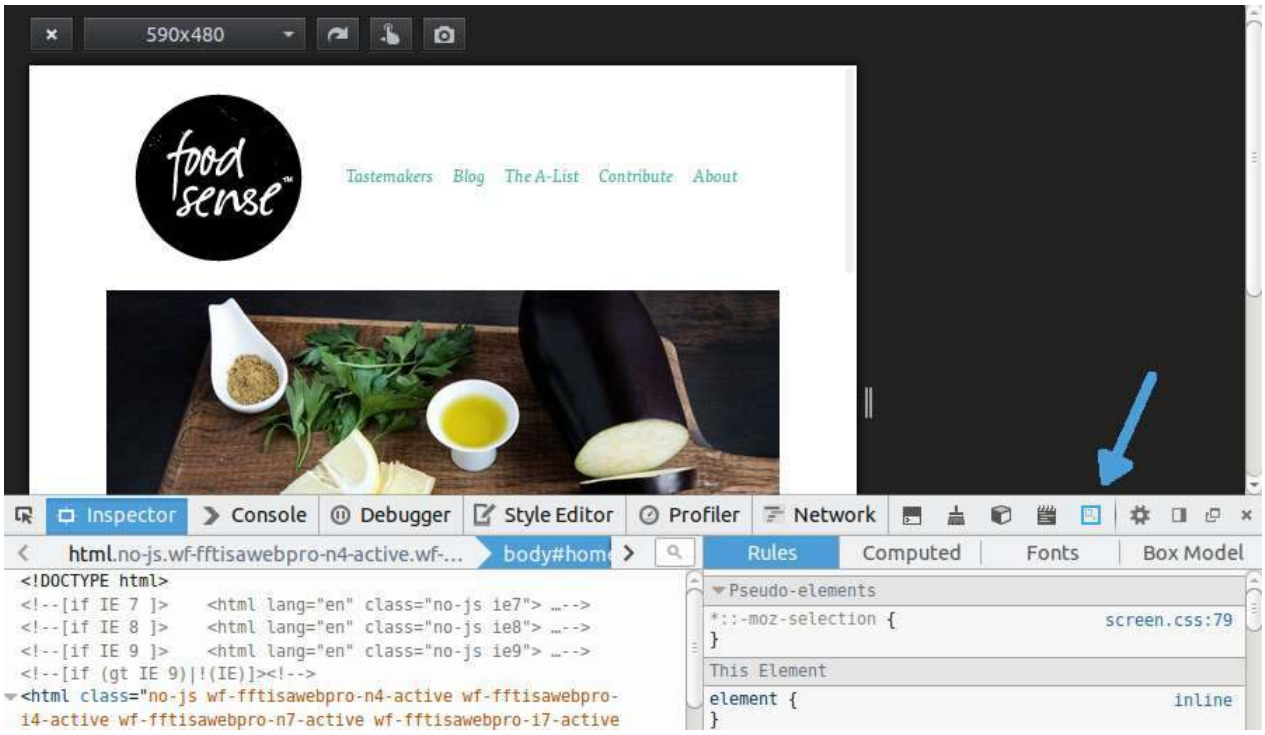
Esta herramienta está disponible solamente de forma *online*, pero nos permite ver de un solo vistazo como se mostraría nuestra web con el tamaño de los *smarthones* y *tablets* más populares, como por ejemplo las diferentes versiones de iPhone, iPad, Kindle y algunas versiones de teléfonos Android.



El problema de estas herramientas es que tenemos que acceder a una versión publicada de nuestra web (no permiten localhost) y son un poco más lentas para realizar pruebas continuas, por esta razón es mucho más recomendable utilizar alguno de los kits de herramientas para el desarrollador web que existen para los diferentes navegadores.

Herramientas del navegador para el desarrollador

Tanto en Firefox como Chrome viene instalado por defecto una serie de herramientas de ayuda para el desarrollador que nos permiten, entre otras cosas, ver la consola de mensajes, inspeccionar el código o ver la secuencia de llamadas al servidor.



Además de estas también existen otras herramientas más avanzadas que podemos instalar como una extensión de nuestro navegador, como por ejemplo Firebug.

La ventaja de estas herramientas frente a las anteriores es que son muchos más rápidas, nos permiten probar nuestra página en local y además podemos inspeccionar el código y modificar los estilos en tiempo real. Usando el inspector de estas herramientas nos podemos ahorrar mucho tiempo a la hora de realizar pruebas sobre la propia página cargada, ya que de otra forma tendríamos que modificar el código directamente, recargar la página y volver a probarlo.

Página básica

Bootstrap utiliza ciertos elementos HTML y propiedades CSS que requieren el uso del *doctype* de HTML 5 para que funcionen, por lo que es importante añadirlo a todas nuestras páginas:

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

Además para asegurar que se muestra correctamente en dispositivos móviles y que permite la utilización del zoom al arrastrar tenemos que añadir la siguiente etiqueta `meta` dentro de la cabecera `<head>` :

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

A continuación se incluye una plantilla HTML base para cualquier proyecto con Bootstrap, a partir de la cual se tendrán que ir añadiendo el resto de elementos:

```
<!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>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- librerías opcionales que activan el soporte de HTML5 para IE8 -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>¡Hola Mundo!</h1>

    <!-- Librería jQuery requerida por los plugins de JavaScript -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.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>
```

Es posible deshabilitar el zoom para dispositivos móviles añadiendo `user-scalable=no` a la etiqueta *meta* del *viewport* (como se puede ver en el ejemplo inferior). De esta forma los usuarios únicamente podrán usar el scroll de la aplicación, haciendo tu web más similar a una aplicación nativa. Sin embargo, hay que usar esta característica con cuidado ya que no es recomendable para todos los sitios.

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

Sistema de rejilla

El sistema de rejilla de Bootstrap se basa en la creación o disposición del contenido de nuestra web dentro de rejillas flexibles, las cuales se escalarán al tamaño y posición adecuada de forma automática dependiendo del tamaño de la pantalla en la que se rendericen.

Elemento contenedor

El sistema de rejilla tiene que ser utilizado dentro de uno de los dos elementos contenedores que provee Bootstrap: `container` ó `container-fluid`. Es importante tener en cuenta que estos elementos se utilizan como raíz de la rejilla y no se podrán anidar unos dentro de otros.

Si lo que queremos es que el contenido de nuestra web aparezca centrado y con un ancho fijo entonces podemos utilizar la etiqueta `.container`, de la forma:

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

Por el contrario, si queremos que el contenido de nuestra web pueda ocupar todo el ancho disponible (hay que tener en mente todos los tamaños de pantalla, incluso las muy grandes), podemos usar la etiqueta `.container-fluid`:

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

Funcionamiento del sistema de rejilla

El sistema de rejilla está pensado para ayudarnos en la disposición de los contenidos de nuestra web y su adaptación a los diferentes tamaños de pantalla de forma automática. Para ello tenemos que poner el contenido dentro de celdas o columnas que irán dentro de filas. Cada fila se puede dividir hasta en 12 columnas, pero seremos nosotros los que definiremos el número de columnas deseado para cada tamaño de pantalla.

A continuación se detalla el funcionamiento de este sistema:

- Las columnas irán agrupadas dentro de filas (`.row`).
- Las filas (`.row`) se deben colocar dentro de una etiqueta contenedora: `.container` (para ancho fijo) o `.container-fluid` (para poder ocupar todo el ancho), esto permitirá alinear las celdas y asignarles el espaciado correcto.
- El contenido se debe disponer dentro de columnas o celdas, las cuales deben de ser el único hijo posible de las filas (`.row`), las cuales, a su vez, serán el único hijo posible del contenedor (`.container` o `.container-fluid`).
- Al seguir este orden el sistema de rejilla funcionará correctamente, creando el espaciado interior y los márgenes apropiados dependiendo de las dimensiones de la pantalla.
- Cada fila se puede dividir hasta un máximo de 12 columnas, pero somos nosotros los que tendremos que definir el número de columnas en el que queremos dividir cada fila y su ancho para cada tamaño de pantalla. Por ejemplo: 3 columnas de igual ancho.
- Si el tamaño total de las columnas de una fila excede de 12 el tamaño sobrante se colocará en la siguiente fila.
- El tamaño de las columnas se especificará con clases css que Bootstrap define para cada tamaño de pantalla, por ejemplo `.col-md-XX` , donde `XX` es el tamaño de la columna, que podrá tomar valores entre 1 y 12.

En la siguiente tabla se muestra un resumen del sistema de rejilla de Bootstrap, su comportamiento según el tamaño del dispositivo y las clases CSS que nos permiten controlarlo:

Pantalla	Prefijo de la clase	Ancho del contenedor
Tamaño extra pequeño Teléfonos (<768px)	.col-xs-	Ninguno (automático)
Tamaño pequeño *Tablets* (≥768px)	.col-sm-	750px
Tamaño medio Escritorios (≥992px)	.col-md-	970px
Tamaño grande Escritorios (≥1200px)	.col-lg-	1170px

Es importante destacar al definir estas clases no solo se aplican para ese tamaño de pantalla sino para los superiores también. Por ejemplo al indicar el tamaño de las columnas con las clases para *tablets* (*.col-sm-*), también se aplicará para los tamaños de pantalla medianos y grandes (si no hubieran otras clases para estos tamaños que los sobreescribieran).

Ejemplos

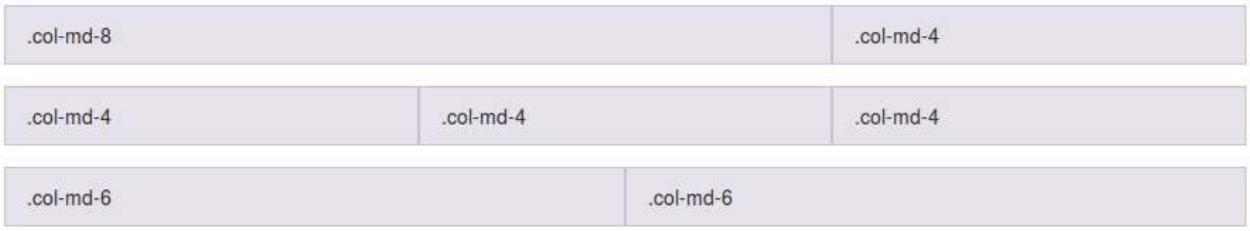
A continuación se incluyen algunos ejemplos de uso del sistema de rejilla que nos ayudarán a comprender mejor su funcionamiento.

Selección de tamaño de las columnas solo para pantallas de escritorio

En el siguiente ejemplo se han creado 3 filas, la primera dividida 2 columnas de tamaño desigual, la segunda en 3 columnas de igual tamaño y la tercera en 2 columnas también de igual tamaño.

```
<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>
```

En la siguiente imagen se puede ver el resultado para una pantalla mediana (de escritorio):



Dado que las columnas se han especificado únicamente mediante las clases `.col-md-*` esto creará estas divisiones solo para las pantallas de escritorio medianas y grandes, pero no para los tamaños de pantalla pequeños (*tablets* y móviles). En estos dos últimos casos las columnas se ampliarán para ocupar todo el ancho y por lo tanto se mostrarán apiladas de la forma:

.col-md-8
.col-md-4

.col-md-4
.col-md-4
.col-md-4

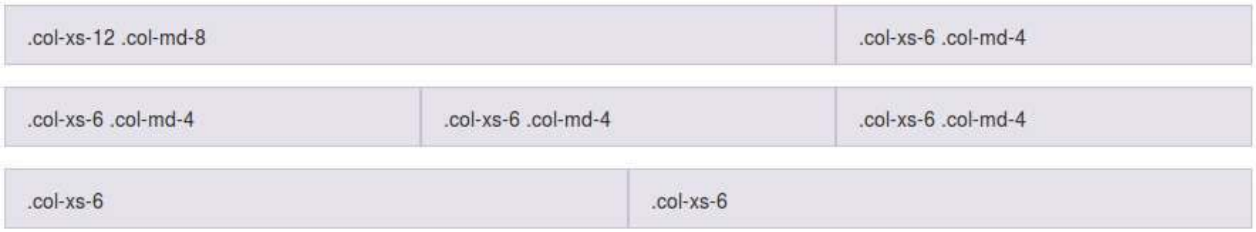
.col-md-6
.col-md-6

Selección del tamaño para móvil y escritorio

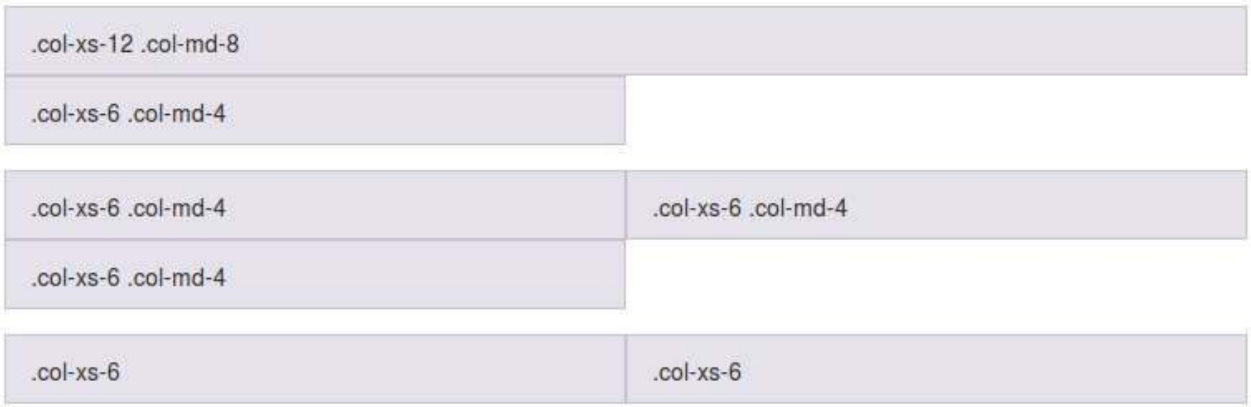
Si no queremos que las columnas se muestren apiladas para tamaños de pantalla pequeños podemos indicar también la disposición para esos casos mediante las clases `.col-xs-*` además de las que ya teníamos `.col-md-*`. Por ejemplo:

```
<!-- En pantallas pequeñas aparecerá una columna que ocupará todo el ancho
y otra que ocupará la mitad de la pantalla -->
<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 pantallas pantallas se indica que ocupe cada columna la mitad
del ancho 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>

<!-- Como no se indica el tamaño para pantallas grandes las columnas
siempre ocuparán el 50% -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

En la siguiente imagen se puede ver como quedaría el código de ejemplo para pantallas medianas (md) y grandes (lg):



En el caso de pantallas pequeñas las columnas se verían de la forma:



Selección del tamaño para móvil, *tablet* y escritorio

Si queremos tener un mayor control podemos especificar también el tamaño de las columnas para las pantallas tipo *tablet* con las clases `.col-sm-*`. Por ejemplo:

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

A continuación se incluye una previsualización de este código de ejemplo para pantallas medianas y grandes:

El mismo código pero en pantallas tipo *tablet* se mostraría como:

.col-xs-12 .col-sm-6 .col-md-8	.col-xs-6 .col-md-4

.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4

Y en el caso de pantallas pequeñas (xs) se vería de la forma:

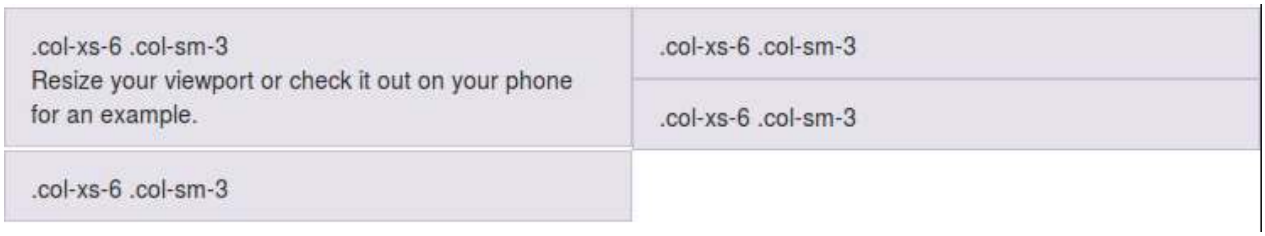
.col-xs-12 .col-sm-6 .col-md-8	
.col-xs-6 .col-md-4	
.col-xs-6 .col-sm-4	.col-xs-6 .col-sm-4
.col-xs-6 .col-sm-4	

Forzar el cambio de fila

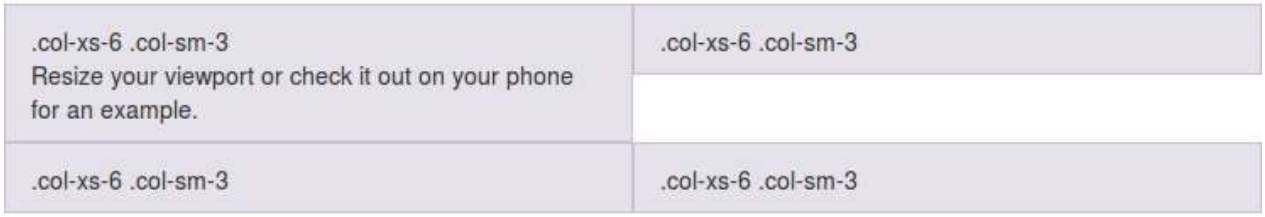
Mediante la clase `.clearfix` podemos forzar el cambio de fila cuando nosotros queremos. Esta clase nos puede ser útil cuando por ejemplo las filas tengan un alto distinto o para forzar el cambio de fila solo para determinados tamaños de pantalla mediante la combinación con otras clases (por ejemplo si añadimos `visible-xs-block` solo se producirá ese cambio de fila para pantallas pequeñas).

```
<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>

  <!-- Add the extra clearfix for only the required viewport -->
  <div class="clearfix visible-xs-block"></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>
```

En la siguiente imagen podemos ver un ejemplo en el que no se ha utilizado la clase `.clearfix` y debido a que las dos primeras columnas tienen un alto distinto la primera columna de la siguiente fila se coloca en una posición incorrecta:



Si añadimos la clase `.clearfix` como en el código de ejemplo podemos solucionar ese problema, quedando:



Anidamiento de columnas

Una característica muy potente del sistema de columnas es que se pueden anidar unas dentro de otras, por ejemplo, dentro de una columna de tamaño 9 podemos crear una nueva fila y subdividirla como queramos (igual que si fuera una fila normal, con hasta 12 columnas). A continuación se incluye un ejemplo:

```
<div class="row">
  <div class="col-sm-9">
    Level 1: .col-sm-9
    <div class="row">
      <div class="col-xs-8 col-sm-6">
        Level 2: .col-xs-8 .col-sm-6
      </div>
      <div class="col-xs-4 col-sm-6">
        Level 2: .col-xs-4 .col-sm-6
      </div>
    </div>
  </div>
</div>
```

Márgenes o espaciado entre columnas

Es posible crear un espaciado entre las columnas o dicho de otra forma, mover o desplazar una columna hacia la derecha, añadiendo un *offset* inicial mediante las clases: `.col-*-offset-*` . Por ejemplo `col-md-offset-4` creará un espacio a la izquierda de la columna de tamaño 4 (como si se creara una columna oculta de tipo `.col-md-4`). En el siguiente código podemos ver un ejemplo más completo:

```
<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>
```

El cual se renderizaría de la forma:

Si en algún caso necesitamos eliminar el *offset* podemos utilizar el tamaño cero (0). Por ejemplo, si especificamos un *offset* de 2 para tamaños pequeños y no queremos que dicho *offset* se aplique para pantallas grandes o medias tendríamos que hacer:

```
  <div class="col-sm-5 col-sm-offset-2 col-md-7 col-md-offset-0"></div>
```

Ordenación de columnas

También podemos modificar el orden de las columnas mediante las clases `.col-*-push-*` y `.col-*-pull-*`. Por ejemplo, con `.col-md-push-3` "empujaríamos" la columna 3 espacios hacia la derecha y con `.col-md-pull-3` la empujaríamos 3 espacios hacia la izquierda. A continuación podemos ver un ejemplo:

```
<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>
```

El cual quedaría de la forma:

.col-md-3 .col-md-pull-9	.col-md-9 .col-md-push-3

Hay que tener cuidado con estas clases si hay un cambio de fila (debido a el número de columnas ocupe más de 12), en estos casos estas clases no funcionarán correctamente.

Si queremos restaurar la posición podemos utilizar el tamaño cero (0). Por ejemplo, si habíamos desplazado la columna hacia la derecha para tamaños de pantalla pequeños y queremos que no se aplique en tamaños de pantalla medianos y grandes podríamos utilizar la clase `col-md-push-0`.

Utilidades *Responsive*

Bootstrap también incluye una serie de clases para ayudarnos a mostrar u ocultar contenidos según el tamaño del dispositivo. A continuación se incluye una tabla resumen de todas estas clases:

	Extra pequeño Teléfonos (<768px)	**Tamaño pequeño *Tablets* (≥768px)**	**Tamaño medio Escritorios (≥992px)**	**Tamaño grande Escritorios (≥1200px)**
.visible-xs-*	**Visible**	Oculto	Oculto	Oculto
.visible-sm-*	Oculto	**Visible**	Oculto	Oculto
.visible-md-*	Oculto	Oculto	**Visible**	Oculto
.visible-lg-*	Oculto	Oculto	Oculto	**Visible**
.hidden-xs	Oculto	**Visible**	**Visible**	**Visible**
.hidden-sm	**Visible**	Oculto	**Visible**	**Visible**
.hidden-md	**Visible**	**Visible**	Oculto	**Visible**
.hidden-lg	**Visible**	**Visible**	**Visible**	Oculto

En el caso de la clase `.visible-*-*` tenemos la posibilidad de indicar la forma en la que se mostrará el elemento sobre el que se aplique (los posibles valores se corresponden con los que puede adoptar la propiedad `display` de CSS). Por lo que tendríamos:

Group of classes	**CSS display**
.visible-*-block	display: block;
.visible-*-inline	display: inline;
.visible-*-inline-block	display: inline-block;

Por lo que para por ejemplo las pantallas extra pequeñas (xs) podríamos utilizar las clases: `.visible-xs-block`, `.visible-xs-inline`, y `.visible-xs-inline-block`. Siendo `.visible-xs-block` la más común y utilizada, para por ejemplo mostrar una columna solo ante un

determinado tamaño de pantalla.

Notas de Uso

Hay que tener en cuenta que las clases `visible-*-*` solo se mostrarán para el tipo de dispositivo indicado, es decir, si por ejemplo indicamos que un campo solo es visible con la etiqueta `visible-md-block` dicho campo no aparecerá para resoluciones inferiores ni tampoco para pantallas tipo *large*.

Por el contrario, las etiquetas tipo `.hidden-` solo se ocultarán el elemento para el tamaño indicado, quedando visible para el resto de tamaños.

Estas etiquetas son de mucha utilidad para mejorar una web responsive pues nos van a permitir crear un mejor diseño o maquetación. Normalmente nos interesará ocultar determinados contenidos cuando la pantalla sea muy pequeña o mostrar contenidos adicionales para pantallas grandes.

Por ejemplo, imaginaos que en nuestra web tenemos una serie de artículos y que en cada uno de ellos incluimos un título, un subtítulo, una imagen y un texto. Si por ejemplo quisieramos ocultar el subtítulo y la imagen para pantallas extra pequeñas (xs) símplemente tendríamos que añadir la clase " `.hidden-xs` " a estas etiquetas. A continuación se incluye el código de ejemplo:

```
<div class="article">
    <h1>
        Título del artículo
        <small class="hidden-xs">Subtítulo del artículo</small>
    </h1>

    <img src="/article-image.jpg" alt="article image" class="hidden-xs" />

    <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>
</div>
```

Media queries

En la mayoría de los casos gracias a todas las clases que provee Bootstrap nos será suficiente para componer nuestra web. Sin embargo, en algunas situaciones es posible que queramos modificar dicho comportamiento, por ejemplo para aplicar determinados estilos CSS (como colores, alineaciones, etc.) que cambien según el tamaño de pantalla. En estos casos será necesario que creemos nuestra propia *media query* para aplicar los estilos deseados.

Una *media query* se define de la forma:

```
@media (min-width: TAMAÑO-EN-PÍXELES) {
    /* Los estilos aquí contenidos solo se aplicarán a partir
    del tamaño de pantalla indicado */
}
```

En este caso, los estilos que estén dentro de esta *media query* se aplicarán solo a partir del tamaño en píxeles indicado. Además de un tamaño mínimo podemos indicar el tamaño máximo o el rango de tamaño en el que se aplicarán, de la forma:

```
@media (max-width: TAMAÑO-EN-PÍXELES) {
    /* Estos estilos solo se aplicarán hasta el tamaño indicado */
}
@media (min-width: TAMAÑO-EN-PÍXELES) and (max-width: TAMAÑO-EN-PÍXELES) {
    /* Solo se aplicarán entre los tamaños indicados */
}
```

Recordamos que los rangos que define Bootstrap son:

- Pantallas extra pequeñas (móviles) < 768px
- Pantallas pequeñas (*tablets*) ≥ 768px
- Pantallas medianas (escritorio) ≥ 992px
- Pantallas grandes (escritorio) ≥ 1200px

Ejemplos de uso

Si por ejemplo queremos que en las pantallas extra pequeñas (xs) el color de fondo que aplica la clase `.miestilo` sea rojo y para el resto de tamaños sea verde, podríamos hacer:

```
.miestilo {
    background-color: green;
}
@media (max-width: 768px) {
    .miestilo {
        background-color: red;
    }
}
```

O si por ejemplo queremos variar la alineación del texto que se aplica en una clase a partir de las pantallas tipo escritorio:

```
.miestilo {
    text-align: center;
}
@media (min-width: 992px) {
    .miestilo {
        text-align: left;
    }
}
```

Estos sencillos ejemplos nos muestran la idea básica que tenemos que seguir para complementar el código de Bootstrap para hacer que la web se comporte como nosotros queramos. De esta forma podemos llegar a hacer cosas muy avanzadas y personalizar completamente el aspecto de una web según el tamaño del dispositivo.

Otros ejemplos de personalizaciones que podemos hacer usando las *media queries* son:

- Cambiar el tamaño y la posición de una imagen. Por ejemplo hacer que la imagen de cabecera sea muy pequeña para dispositivos móviles y mucho mayor para pantallas de escritorio.
- Cambiar la posición de cualquier elemento. Si por ejemplo tenemos un elemento que no se ve bien con una alineación en pantallas pequeñas podemos colocarlo en otro lugar.
- Cambiar el tamaño de letra, la fuente o su color. Podemos reducir el tamaño de letra de las cabeceras para pantallas xs o aumentarlo para pantallas más grandes.
- Aplicar combinaciones de estilos avanzados. Por ejemplo, podemos crear un estilo ".articulo" que según el tamaño de pantalla reajuste toda la apariencia de un artículo con todos los elementos que contenga.
- Cualquier cosa que se os ocurra!

Componentes *responsive*

Además del sistema de rejilla Bootstrap incluye un completo conjunto de componentes para facilitarnos aún más el diseño de una web *responsive*. Estos componentes aplican estilos a los elementos HTML existentes para crear un diseño más moderno y además adaptable a todos los dispositivos.

Algunos de estos componentes son:

- Barras de navegación
- Botones
- Formularios
- Tablas
- Desplegables
- y muchos más...

A continuación se explica el funcionamiento de los componentes más utilizados.

Botones

Mediante la clase `.btn` podemos aplicar estilo a los elementos tipo `button`. Esta clase la podemos combinar con `.btn-default`, `.btn-primary`, `.btn-success`, `.btn-info`, `.btn-warning`, `.btn-danger`, `.btn-link` para crear botones para diferentes estados o acciones en nuestros formularios:

El código HTML que tendríamos que escribir para conseguir estos botones es el siguiente:

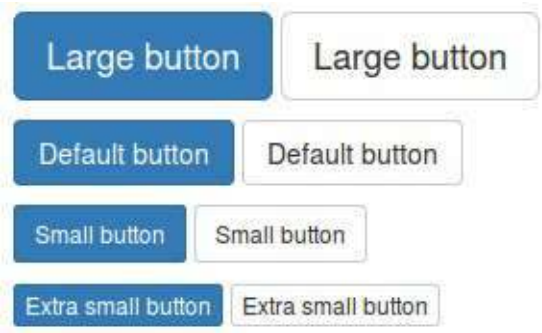
```
<button type="button" class="btn btn-default">Default</button>
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-link">Link</button>
```

Tamaño de los botones

Podemos variar el tamaño de los botones simplemente añadiendo las clases `.btn-lg`, `.btn-sm`, o `.btn-xs`:

```
<p>
  <button type="button" class="btn btn-primary btn-lg">Large button</button>
  <button type="button" class="btn btn-default btn-lg">Large button</button>
</p>
<p>
  <button type="button" class="btn btn-primary">Default button</button>
  <button type="button" class="btn btn-default">Default button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">Small button</button>
  <button type="button" class="btn btn-default btn-sm">Small button</button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-xs">Extra small button</button>
  <button type="button" class="btn btn-default btn-xs">Extra small button</button>
</p>
```

Los botones del código anterior se mostrarían de la forma:



Elementos tipo botón

El estilo tipo botón no solo lo podemos aplicar sobre las etiquetas `button` sino que funcionará de la misma forma con `<a>` y `<input>`:

```
<a class="btn btn-default" href="#" role="button">Link</a>
<button class="btn btn-default" type="submit">Button</button>
<input class="btn btn-default" type="button" value="Input">
<input class="btn btn-default" type="submit" value="Submit">
```

Todos estos elementos se renderizarán de la misma forma:

Desplegables

Bootstrap nos facilita la creación de botones con listas de opciones desplegables mediante la clase `.dropdown`. Este elemento requiere que el *plugin* JavaScript de Bootstrap esté incluido en la plantilla. La estructura básica para crear un elemento de este tipo es la siguiente:

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle" type="button" id="dropdownMenu1"
          data-toggle="dropdown" aria-expanded="true">
    Dropdown
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">
    <li role="presentation"><a role="menuitem" tabindex="-1" href="#">Action</a></li>
    <li role="presentation"><a role="menuitem" tabindex="-1" href="#">Another action</a
></li>
    <li role="presentation"><a role="menuitem" tabindex="-1" href="#">Something else h
ere</a></li>
    <li role="presentation"><a role="menuitem" tabindex="-1" href="#">Separated link</a
></li>
  </ul>
</div>
```

Para alinear un menú a la derecha se puede utilizar la clase `.dropdown-menu-right`, por ejemplo:

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

Encabezados en un desplegable

Para añadir un encabezado y dividir en secciones un desplegable podemos utilizar la clase `.dropdown-header` de la forma:

```
<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu2">
  ...
  <li role="presentation" class="dropdown-header">Dropdown header</li>
  ...
</ul>
```

Separadores en un desplegable

También podemos añadir separadores en un desplegable simplemente utilizando la clase `.divider` de la forma:

```
<ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenuDivider">
  ...
  <li role="presentation" class="divider"></li>
  ...
</ul>
```

Grupos de botones

Podemos crear un grupo de botones en una línea agrupándolos dentro de un elemento contenedor con la etiqueta `.btn-group`.

```
<div class="btn-group" role="group" aria-label="...">
  <button type="button" class="btn btn-default">Left</button>
  <button type="button" class="btn btn-default">Middle</button>
  <button type="button" class="btn btn-default">Right</button>
</div>
```

Mediante la librería JavaScript de Bootstrap podemos añadir comportamientos tipo *checkbox* o *radio button* a un grupo de botones.

Barra de botones

La barra de botones nos permite combinar grupos de botones para crear componentes más avanzados:

```
<div class="btn-toolbar" role="toolbar" aria-label="...">
  <div class="btn-group" role="group" aria-label="...">...</div>
  <div class="btn-group" role="group" aria-label="...">...</div>
  <div class="btn-group" role="group" aria-label="...">...</div>
</div>
```

Tamaños de los grupos de botones

Los grupos de botones nos permiten además indicar el tamaño de los botones para todo el conjunto mediante las etiquetas `.btn-group-*`:

```
<div class="btn-group btn-group-lg" role="group" aria-label="...">...</div>
<div class="btn-group" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-sm" role="group" aria-label="...">...</div>
<div class="btn-group btn-group-xs" role="group" aria-label="...">...</div>
```

Grupo de botones con desplegables

También es posible añadir desplegables a los grupos de botones. Para esto el desplegable tendrá que estar contenido a su vez dentro de otro grupo de botones, de la forma:

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

  <div class="btn-group" role="group">
    <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdow
n" aria-expanded="false">
      Dropdown
      <span class="caret"></span>
    </button>
    <ul class="dropdown-menu" role="menu">
      <li><a href="#">Dropdown link</a></li>
      <li><a href="#">Dropdown link</a></li>
    </ul>
  </div>
</div>
```

Como se puede observar la única diferencia con un desplegable normal es que la etiqueta contenedora en vez de ser de tipo `.dropdown` es un `.btn-group`.

Formularios

Bootstrap aplica estilos a los elementos de tipo formulario para mejorar su apariencia y permitirnos crear diferentes alineaciones. La estructura básica de un formulario es la siguiente:

```
<form role="form">
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
        placeholder="Enter email">
  </div>
</form>
```

Para permitir que Bootstrap ajuste correctamente el espaciado, cada bloque o grupo de un formulario (normalmente formado por una etiqueta `label` y algún elemento de entrada de datos como un *input*, *textarea*, etc.) tendrá que estar agrupado por una caja contenedora con la clase `.form-group` . Además a cada *input* se le tiene que aplicar la clase `.form-control` .

Bootstrap sobrecarga y aplica estilos a los principales elementos de formulario definidos en HTML 5, como son: *text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel,* y *color*.

Formulario *inline*

Mediante la utilización de la clase `.form-inline` sobre la etiqueta `<form>` podemos crear formularios que se dispondrán en una sola línea. A continuación se incluye un ejemplo de este tipo de formularios:

```
<form class="form-inline" role="form">
  <div class="form-group">
    <div class="input-group">
      <label class="sr-only" for="exampleInputEmail2">Email address</label>
      <div class="input-group-addon">@</div>
      <input type="email" class="form-control" id="exampleInputEmail2" placeholder="En
ter email">
    </div>
  </div>
  <div class="form-group">
    <label class="sr-only" for="exampleInputPassword2">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword2" placeholder=
"Password">
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Remember me
    </label>
  </div>
  <button type="submit" class="btn btn-default">Sign in</button>
</form>
```

Nota 1: aunque los campos del formulario no contengan etiquetas (*labels*) es necesario incluirlas por cuestiones de accesibilidad, para dar soporte a los lectores de pantalla. Por este motivo se han incluido en el ejemplo anterior con la clase `.sr-only` (*screen readers only*).

Nota 2: Este estilo no se aplicará en pantallas pequeñas tipo móvil.

Formulario horizontal

Mediante la combinación de la clase `.form-horizontal` con el sistema de rejilla de Bootstrap podemos crear formularios horizontales como en el ejemplo siguiente:

```
<form class="form-horizontal" role="form">
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3" placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3" placeholder="Pas
sword">
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember me
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Sign in</button>
    </div>
  </div>
</form>
```

Es importante que nos fijemos que no se utilizan las clases `.row` para crear filas, ya que son sustituidas por `.form-groups`. Además, podemos aplicar la clase de las columnas para las etiquetas `label` directamente sobre dicho elemento, sin necesidad de crear una caja contenedora.

Estados de validación de un formulario

Bootstrap también incluye clases para aplicar diferentes estados de validación a un formulario. Para utilizarlo simplemente tenemos que añadir las clases: `.has-warning`, `.has-error`, o `.has-success` al elemento contenedor, en este caso a `.form-group`. De esta forma, el color de los elementos del formulario de dicho grupo cambiarán. A continuación podemos ver un ejemplo:

```
<div class="form-group has-success">
  <label class="control-label" for="inputSuccess1">Input with success</label>
  <input type="text" class="form-control" id="inputSuccess1">
</div>
<div class="form-group has-warning">
  <label class="control-label" for="inputWarning1">Input with warning</label>
  <input type="text" class="form-control" id="inputWarning1">
</div>
<div class="form-group has-error">
  <label class="control-label" for="inputError1">Input with error</label>
  <input type="text" class="form-control" id="inputError1">
</div>
```

Agrupar *inputs* con otros elementos

Podemos añadir texto o botones al principio, final o a ambos lados de campo tipo `<input>`. Para esto tenemos que agrupar dicho *input* dentro de un `.input-group` y añadir dentro del grupo el elemento que queremos agrupar con la etiqueta `.input-group-addon`. A continuación se incluye un ejemplo:

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

Navegación

Los elementos de navegación de Bootstrap comparten la etiqueta `.nav` para su marcado en la clase contenedora. Estos elementos necesitan la librería JavaScript para su correcto funcionamiento. Algunos de los elementos de navegación que podemos utilizar son las fichas o pestañas y las "píldoras".

Fichas o pestañas

Mediante la clase `.nav-tabs` podemos crear un grupo de pestañas o fichas, para ello tenemos que seguir la siguiente estructura:

```
<div role="tabpanel">
    <ul class="nav nav-tabs">
        <li role="presentation" class="active">
            <a href="#home" aria-controls="home" role="tab" data-toggle="tab">Home</a>
        </li>
        <li role="presentation">
            <a href="#profile" aria-controls="profile" role="tab" data-toggle="tab">Pr
ofile</a>
        </li>
        <li role="presentation">
            <a href="#messages" aria-controls="messages" role="tab" data-toggle="tab">
Messages</a>
        </li>
    </ul>

    <!-- Tab panes -->
    <div class="tab-content">
        <div role="tabpanel" class="tab-pane active" id="home">...</div>
        <div role="tabpanel" class="tab-pane" id="profile">...</div>
        <div role="tabpanel" class="tab-pane" id="messages">...</div>
    </div>
</div>
```

Píldoras

La clase `.nav-pills` se define de igual forma que la `.nav-tab` pero sus elementos adoptarán una apariencia más similar a botones o "píldoras":

```
<ul class="nav nav-pills">
  <li role="presentation" class="active"><a href="#">Home</a></li>
  <li role="presentation"><a href="#">Profile</a></li>
  <li role="presentation"><a href="#">Messages</a></li>
</ul>
```

En este caso también podemos crear un menú vertical o apilado añadiendo la clase `.nav-stacked` a la etiqueta contenedora:

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

Justificado

También podemos indicar que el ancho de las pestañas o de las píldoras se distribuya equitativamente según el ancho disponible. Para esto simplemente tenemos que aplicar la clase `.nav-justified` a la etiqueta contenedora, de la forma:

```
<ul class="nav nav-tabs nav-justified">
  ...
</ul>
<ul class="nav nav-pills nav-justified">
  ...
</ul>
```

Nota: Este estilo no funcionará para pantallas con un ancho menor a 768px, que son las pantallas definidas como extra pequeñas o de *smartphone*.

Elementos de navegación con desplegables

Podemos añadir desplegables a nuestros elementos de navegación de la forma:

```
<ul class="nav nav-tabs">
  ...
  <li role="presentation" class="dropdown">
    <a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-expa
nded="false">
      Dropdown <span class="caret"></span>
    </a>
    <ul class="dropdown-menu" role="menu">
      ...
    </ul>
  </li>
  ...
</ul>
```

Barra de navegación

Bootstrap nos facilita la creación de la barra principal de navegación de nuestra web mediante la clase `.navbar`. Esta barra se adaptará al tamaño de pantalla, mostrando los elementos colapsados en un botón en pantallas pequeñas y de forma normal para pantallas más grandes.

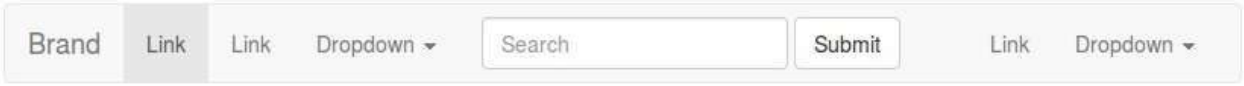
Este elemento requiere que el *plugin* de JavaScript de Bootstrap esté incluido. Además, para cumplir con las reglas de accesibilidad se recomienda añadir `role="navigation"` a nuestras barras de navegación.

A continuación se incluye un ejemplo completo de una barra de navegación:

```
<nav class="navbar navbar-default" role="navigation">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed"
            data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Brand</a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a>
</li>
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
                aria-expanded="false">Dropdown <span class="caret"></span>
          </a>
          <ul class="dropdown-menu" role="menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li class="divider"></li>
            <li><a href="#">Separated link</a></li>
            <li class="divider"></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </li>
```

```
      </ul>
      <form class="navbar-form navbar-left" role="search">
        <div class="form-group">
          <input type="text" class="form-control" placeholder="Search">
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
      </form>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
                aria-expanded="false">Dropdown <span class="caret"></span>
          </a>
          <ul class="dropdown-menu" role="menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li class="divider"></li>
            <li><a href="#">Separated link</a></li>
          </ul>
        </li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
```

La cual se mostraría como en la siguiente figura en pantallas medianas y grandes:



En las pantallas de *smartphone* los elementos de navegación se colapsarían en un botón, de la forma:

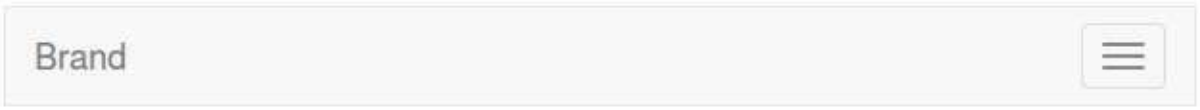
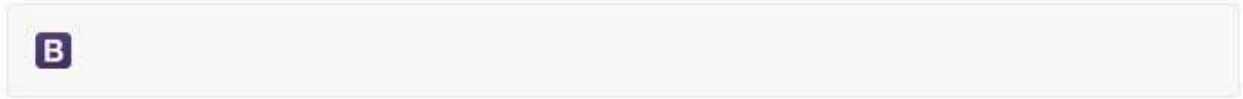


Imagen en la barra de navegación

Para incluir el logotipo de nuestra web en la barra de navegación tenemos que modificar la sección `navbar-header` del ejemplo anterior para incluir la etiqueta `<img>`, de la forma:

```
<nav class="navbar navbar-default" role="navigation">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">
        <img alt="Brand" src="...">
      </a>
    </div>
  </div>
</nav>
```



Nota: Es posible que sea necesario añadir o modificar los estilos para disponer correctamente la imagen en la barra de navegación.

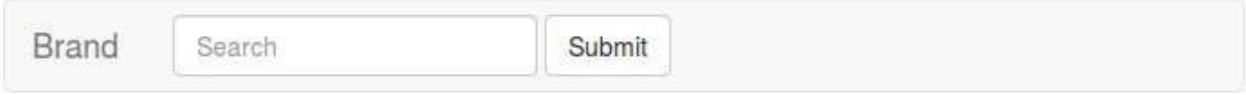
Alineación de los elementos de la barra

Mediante las clases `.navbar-left` y `.navbar-right` podemos indicar la alineación de los elementos en la barra de navegación, ya sean enlaces, botones, texto o formularios.

Barra de navegación con formulario

Podemos añadir un formulario a nuestra barra de navegación añadiendo la clase `.navbar-form` a la etiqueta del formulario, esto hará que se alinee correctamente y se colapse en pantallas pequeñas. De forma opcional podemos utilizar las clases `.navbar-left` o `.navbar-right` para indicar su alineación en la barra.

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



Fijar la barra a la parte superior

Podemos fijar la barra a la parte superior de la pantalla simplemente añadiendo la clase `.navbar-fixed-top` junto a las clases `navbar navbar-default` . Además podemos indicar la disposición de los elementos de la barra añadiendo una clase contenedora de los tipos `.container` o `.container-fluid` para que aparezcan centrados o para que ocupen todo el ancho, respectivamente.

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

Dado que la barra se colará de forma "flotante" sobre el contenido es posible que oculte una parte del mismo. Para solucionar esto es necesario añadir un pequeño espaciado inicial a la etiqueta `<body>` . El alto de la barra es de 50px, por lo que se suele recomendar un espaciado de 70px, de la forma:

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

Barra fija en la parte inferior

También podemos crear una barra de navegación que permanezca fija en la parte inferior de la pantalla. Para esto simplemente tenemos que añadir la clase `.navbar-fixed-bottom` a nuestra barra. Además podemos añadir un contenedor de los tipos `.container` o `.container-fluid` para indicar la disposición de los elementos.

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

En este caso también será necesario modificar el espaciado de la etiqueta `<body>` pero por la parte inferior, para que la barra no oculte los contenidos.

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

Barra de navegación principal

Para crear la barra de navegación principal de nuestro sitio se recomienda añadir la clase `.navbar-static-top` para que ocupe todo el ancho posible. Esto nos permitirá definir si queremos que el contenido aparezca centrado con una clase contenedora tipo `.container` o que por el contrario ocupe todo el ancho posible con `.container-fluid`.

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

Colores invertidos

Podemos invertir los colores de la barra añadiendo la clase `.navbar-inverse`:

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

Tablas

Bootstrap también define una serie de clases para aplicar estilos sobre las tablas de HTML. La más básica es la clase `.table` :

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

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

Colores alternos

Si además aplicamos la clase `.table-striped` a nuestra tabla conseguiremos que las filas presenten colores alternos:

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

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

Tablas con bordes

También podemos dibujar un borde al rededor de la tabla añadiendo la clase `.table-bordered` , de la forma:

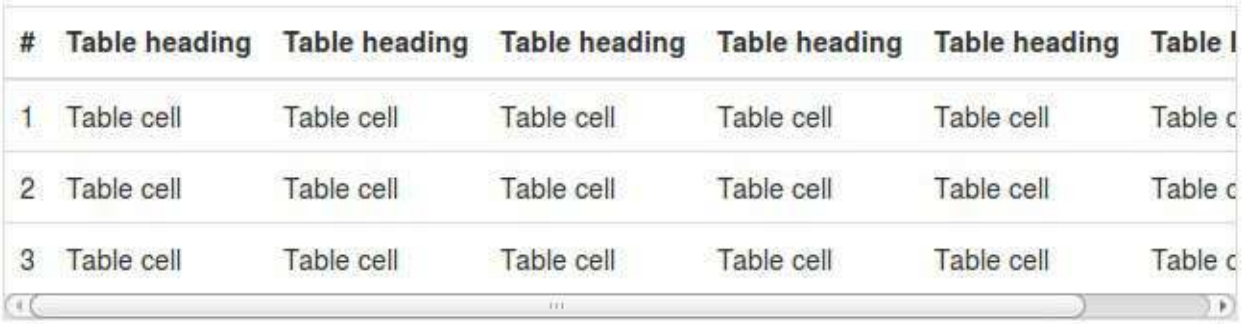
```
<table class="table table-bordered">
  ...
</table>
```

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

Tablas *Responsive*

Bootstrap proporciona una forma de crear tablas *responsive* que se basa en crear un *scroll* horizontal para que se vean correctamente. Para que esto funcione tenemos que crear una caja contenedora a nuestra tabla con la clase `.table-responsive` :

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



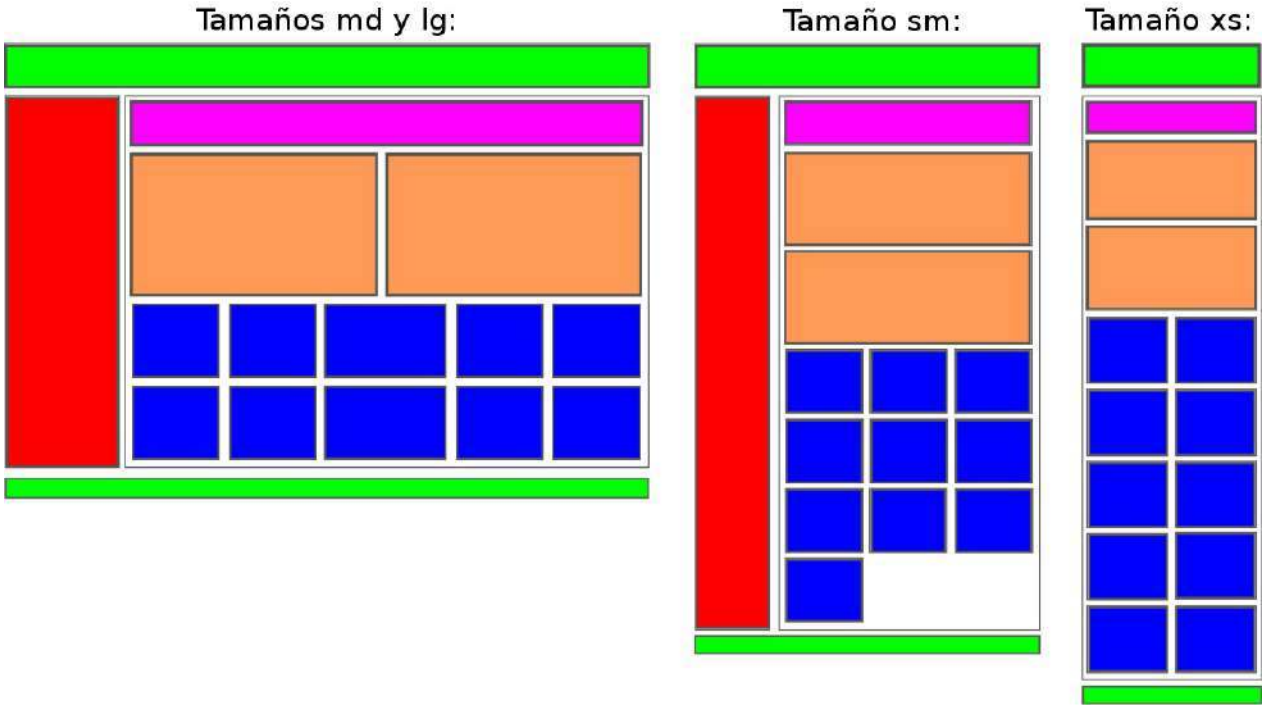
#	Table heading	Table heading	Table heading	Table heading	Table heading	Table h
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table c
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table c
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table c

> *Nota*: este efecto se aplicará únicamente sobre dispositivos pequeños (`<768px`) mientras que en el resto de dispositivos no se notará la diferencia.

Ejercicios 1

Ejercicio 1 - Diseño *responsive* (1 punto)

En este ejercicio vamos a practicar con la librería Bootstrap y su sistema de rejilla. Partiremos de la plantilla para una página web básica facilitada en la teoría, le añadiremos un contenedor de tipo `container` e iremos añadiendo filas y columnas intentando imitar el diseño (y colores) del esquema de la siguiente figura:



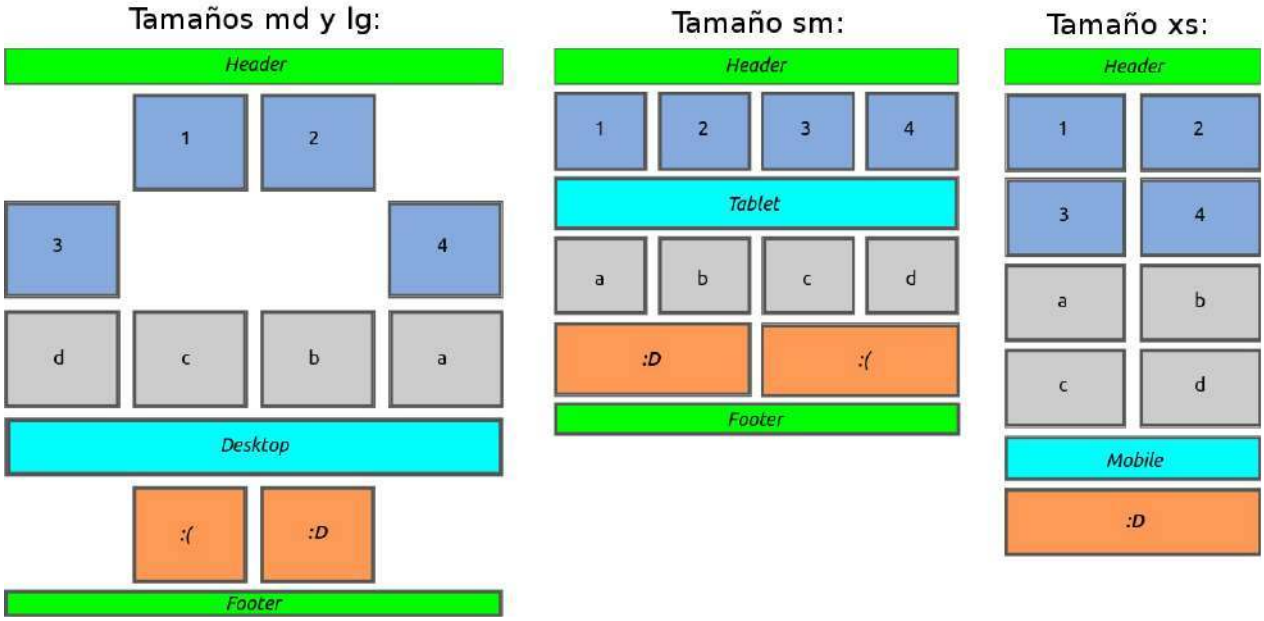
En el esquema de la figura se pueden ver tres disposiciones de la misma web, la de la izquierda se refiere a los tamaños grandes (lg) y medianos (md), la disposición central al tamaño pequeño o de *tablets* (sm) y la de la derecha la correspondiente a móviles (xs).

Tenéis que aplicar las clases de Bootstrap necesarias para que al cambiar el tamaño de la pantalla se cambie la disposición de los bloques como se muestra en el esquema. Tened en cuenta que la columna roja tendrá que desaparecer cuando el tamaño sea extra pequeño (xs).

Ejercicio 2 - *Offset* y ordenación (1 punto)

En este ejercicio vamos a practicar con algunas características más de Bootstrap: la posibilidad de añadir un *offset* (o espacio inicial a las columnas), el cambio de orden de los elementos de una fila y la visibilidad de las columnas según el tamaño del dispositivo.

Para ello nos crearemos una nueva página web partiendo de la plantilla básica, le añadiremos un contendor de tipo `container` e iremos añadiendo filas y columnas intentando imitar el diseño, colores y contenidos del esquema de la siguiente figura:



Tened en cuenta que:

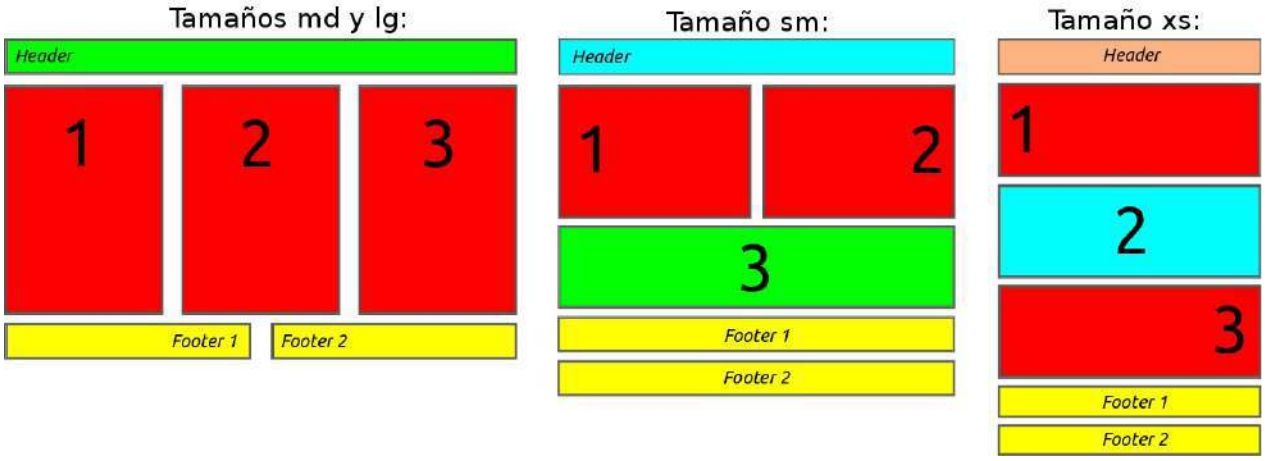
- La segunda fila (que contiene 4 columnas con los números 1, 2, 3 y 4) es solamente una fila a la que se le han añadido *offsets*. Para forzar el cambio de fila se puede añadir un elemento entre la 2ª y la 3ª columna que solo sea visible cuando la pantalla sea mediana o grande (md o lg) y que aplique la clase `clearfix` de Bootstrap.
- El orden de la tercera fila (con las letras a, b, c, d) se ha alterado para las disposiciones de pantalla grandes (md o lg) usando las clases de bootstrap `col-*-push-*` y `col-*-pull-*`.
- En la 5ª fila naranja se ha aplicado un cambio de orden y un offset para las pantallas grandes y medianas (md o lg). Además, cuando la pantalla sea de tipo *xs* se deberá de ocultar una de sus columnas.
- La fila azul claro en la que pone *Desktop* (para pantallas lg o md), *Tablet* (para sm) y *Mobile* (cuando la pantalla es xs) en realidad son 3 filas distintas con clases para que solo se muestren en dichos tamaños de pantalla.
- La última fila se deberá de ocultar solamente cuando la pantalla sea del tipo *xs*.

Ejercicio 3 - Personalizando mediante *media query* (1 punto)

En este ejercicio se pide que creéis una nueva página web usando la librería Bootstrap. El contenido aparecerá centrado en la pantalla y constará de tres filas con el siguiente contenido y disposición cuando la pantalla sea de tamaño medio (md) y grande (lg):

- Una fila en la parte superior con una única columna con fondo verde que ocupará todo el ancho, en dicha columna aparecerá el texto "*Header*" alineado a la izquierda y en grande.
- Una segunda fila con tres columnas en color rojo con el mismo ancho y con los números 1, 2 y 3 (respectivamente) centrados y en letras grandes.
- La tercera y última fila contendrá dos columnas de igual ancho y en color amarillo, la primera columna tendrá el texto "*footer 1*" alineado a la derecha y la segunda el texto "*footer 2*" alineado a la izquierda (ambos usando un tamaño de fuente grande).

En la siguiente imagen se puede ver un esquema de la web a realizar:



Como se puede ver en el esquema de la imagen la disposición de las columnas y la alineación de los textos variará dependiendo del tamaño de la pantalla. Tenéis que reproducir este comportamiento para que la apariencia de la web sea similar al esquema (número de columnas, alineaciones de los textos y colores) cuando el tamaño de la pantalla sea la de un *tablet* (sm) o la de un teléfono (xs).

Tened en cuenta que:

- Siempre que sea posible se utilizarán las clases que provee Bootstrap.
- Cuando no sea posible (por ejemplo para controlar la alineación de los textos y el cambio de color de fondo) tendréis que definir una *media query* que lo haga.

Ejercicios 2

Ejercicio 1 - Crear una Web responsive (3 puntos)

Para poner en práctica los conceptos teóricos vistos sobre diseño *responsive* se propone como ejercicio la creación de un pequeño sitio Web estático que use los estilos y componentes vistos de Bootstrap.

La temática, contenidos y estilos del sitio son libres, pero deberá tener al menos las siguientes características:

- El sitio estará formado por al menos 3 páginas enlazadas entre sí (con contenidos estáticos).
- Ser completamente *responsive*, de forma que se adapte tanto a pantallas extra pequeñas de *smartphone* como a *tablets* y pantallas más grandes de portátiles y de escritorio.
- Tener una barra de navegación principal que se contraiga cuando la pantalla sea pequeña. Esta barra tendrá al menos:
  - dos enlaces,
  - una imagen como logotipo,
  - un buscador (aunque no sea funcional).
- Contener los siguientes elementos (un ejemplo de cada uno en alguna de las páginas del sitio web):
  - botones,
  - un desplegable,
  - una sección con fichas o pestañas,
  - un formulario horizontal,
  - una tabla responsive con bordes y de tipo *striped*.
- El estilo base a utilizar será el que define Bootstrap, si se definen estilos CSS personalizados tendrán que estar en un fichero separado, llamado "custom.css", y que será común para todas las páginas del sitio.

Un posible ejemplo de una web que podéis realizar sería, por ejemplo, una web de recetas. Esta podría tener una página principal con la información más importante, una página con una receta de ejemplo (aquí se podrían utilizar las fichas o pestañas para cambiar entre

elaboración e ingredientes, los cuales podrían estar en una tabla) y otra página para el envío de recetas (con un formulario horizontal, botones para envíar y cancelar, y un desplegable para elegir la categoría).

De forma similar se podría crear la web sobre coches u otro tipo de vehículos, mascotas, bicicletas, etc.

Nota: al ser una web estática tendréis que repetir partes del código en todas las páginas, por ejemplo la barra de menú principal tendrá que ser igual en todas las páginas. Por este motivo se recomienda realizar primero estas partes y una vez probadas copiar y pegar el codigo en el resto de páginas.

Bibliografía

- http://getbootstrap.com/

  Página oficial de Bootstrap desde donde descargar la librería y consultar toda la documentación.

- http://blog.getbootstrap.com/

  El blog oficial de Bootstrap donde se publican las últimas novedades.

- Temas y plantillas gratuitas para Bootstrap:
  - http://startbootstrap.com/
  - http://bootstrapzero.com/
  - http://bootswatch.com/
  - http://www.bootbundle.com/

- http://bootsnipp.com

  Ejemplos y trozos de código útiles para Bootstrap. Aquí podrás encontrar cientos de ejemplos, desde como hacer un formulario de login hasta todo tipo de elementos con animaciones o estilos avanzados.

- http://expo.getbootstrap.com/

  Ejemplos *inspiradores* de uso de Bootstrap.

- http://startbootstrap.com/bootstrap-resources/

  Listado completísimo con todo tipo de recursos disponibles para Bootstrap.