Tema 4: Bootstrap

Semana 13: Introducción a Bootstrap

BLOQUE 3: Bootstrap

Módulo: Desarrollo de Interfaces Web
Ciclo: Desarrollo de Aplicaciones Web
Curso: 2023 – 2024
Profesora: Rosa Medina

Índice

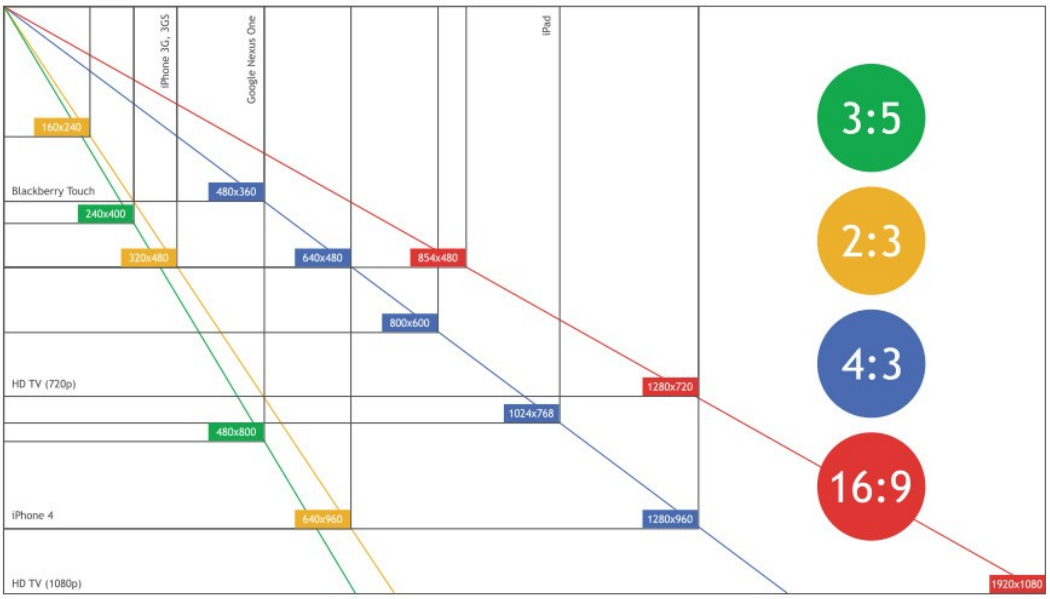
1. Introducción....3
1.1. Frameworks responsive....4
2. Bootstrap....4
2.1. Funcionamiento del diseño adaptable....4
2.2. Herramientas del navegador para el desarrollador....5
2.3. Descarga e instalación....6
2.3.1. Mediante npm....6
2.3.2. Bootstrap & Vite....6
2.4. Contenido del directorio Bootstrap....7
2.5. Página básica....8
3. Sistema de rejilla....9
3.1. Elemento contenedor....9
3.2. Funcionamiento del sistema de rejilla....9
3.2.1. Ejemplo de rejilla....10
3.2.2. Ejemplo de apilado (stacked) en horizontal....11
3.2.3. Ejemplo mix and match....11
3.2.4. Ejemplo row columns....12
3.2.5. Ejemplo de nesting....12
3.3. Cómo ocultar celdas u otro elemento....12
3.4. Márgenes o espaciado entre columnas (margin utilities)....13
3.5. Ordenación de columnas (reordening)....13
3.6. Cómo “romper” una columna (salto)....14
3.7. Cómo mover columnas (offseting)....14
4. Media queries (media object?)....15
4.1. min-width....15
4.2. max-width....16
4.3. single breakpoint....17
4.4. between breakpoint....17
5. Ejemplo de uso....17

1. Introducción

El diseño web responsive (RWD 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 visitarlas. Hoy día las páginas web se ven en multitud de dispositivos como tablets, móviles, portátiles, PC, etc. Además, aún dentro de cada tipo, cada dispositivo tiene sus características concretas: tamaño de pantalla, resolución, potencia de CPU, sistema operativo o capacidad de memoria entre otras. Esta tecnología pretende que con un único diseño web, todo se vea correctamente en cualquier dispositivo.

Variabilidad en las resoluciones de pantalla

Antiguamente el desarrollo web se basaba en la resolución estándar de 1024 x 768, pero hoy existe una amplia variedad de resoluciones, no solo en ordenadores de sobremesa sino también para tablets y móviles.



Cuando vamos a crear una web, debemos tener en cuenta que los usuarios finales no van a tener toda la misma resolución de pantalla.

A la hora de crear una web, debemos tener en cuenta a qué usuarios va destinados para poder analizar su usabilidad, ya que ya no podemos centrarnos en desarrollar una web pensando que los usuarios van a tener probablemente una única resolución de pantalla.

Es fundamental tener en cuenta que en el diseño responsive **deberemos mostrar en primer lugar los contenidos más importantes e imprescindibles**, y luego el resto del contenido (u ocultar aquello que no queramos mostrar en determinadas resoluciones).

1.1. Frameworks responsive

Hoy en día existen en el mercado una amplia variedad de frameworks responsive, algunos de los más utilizados son:

- **Bootstrap**: es uno de los más populares. Fue desarrollado por el equipo de Twitter. Bootstrap ha sido creado para ofrecer la mejor experiencia de usuario tanto a usuarios de PC como a móviles y tablets. Utiliza el grid responsive de 12 columnas y trae integrado decenas de complementos, plugins de JavaScript, tipografía, controladores de formularios, etc. Además, utiliza el preprocesador de CSS less.
- **Foundation**: Junto con Bootstrap es uno de los frameworks más avanzados que existen en la actualidad. Ha sido desarrollado con Sass.
- **Skeleton**: 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**: Al igual que los demás nos ofrece un montón de utilidades para construir nuestra web responsive de forma rápida y sencilla, con la ventaja de ser uno de los que menos ocupan.

En esta unidad, nos centraremos en Bootstrap por ser uno de los más completos, utilizados y que mejor funciona. ¡Veamos cómo funciona esta librería!

2. Bootstrap

Bootstrap es un conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Bootstrap es el segundo proyecto más destacado en GitHub y es usado por la NASA y la MSNBC entre otras organizaciones.

2.1. Funcionamiento del diseño adaptable

Se basa en adaptar el diseño web dependiendo de la resolución de la pantalla del cliente. De modo que podremos adaptar nuestra web independientemente del dispositivo que esté usando sin necesidad de tener varios archivos CSS.

Antiguamente se hacían dos diseños, uno para móviles y otro para web, sin embargo, responsive trata de estructurar o adaptar el contenido que ya tienes en el diseño original a otros formatos diferentes.

A partir de la versión de CSS 2.1 las hojas de estilo incluyeron los media type, lo cual nos ha facilitado, por ejemplo, proveer un estilo distinto para imprimir.

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

A partir de CSS3, el W3C creó las media queries. Una media query nos permite inspeccionar las características del dispositivo que está renderizando nuestra web. Para utilizarlas podemos incorporar una query al atributo media de un `link` a una hoja de estilos:

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

La query contiene dos componentes:

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

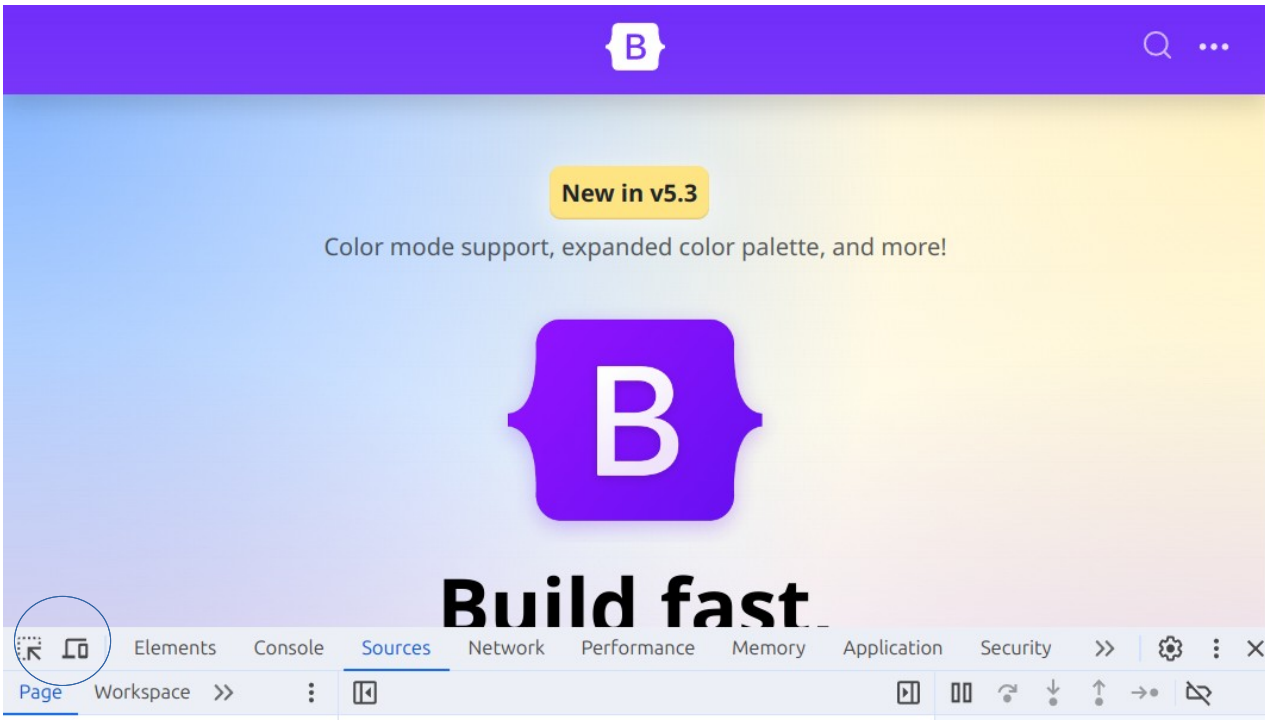
También es posible utilizarlas directamente en nuestro código CSS como parte de una regla:

```
@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
```

Para más información sobre las media queries podéis consultarlo aquí.

2.2. Herramientas del navegador para el desarrollador

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



Además, podemos instalar extensiones del navegador como es Firebug. Con estas herramientas podemos probar nuestra página en local e inspeccionar el código, o modificar los estilos en tiempo real.

2.3. Descarga e instalación

2.3.1.Mediante npm

Para descargar Bootstrap (v5.3) y poder utilizarlo sin necesidad de tener conexión a internet tenemos varias opciones:

1. Descargar los archivos compilados y listos para usar: Compiled CSS and JS
2. Descargar los archivos con la posibilidad de modificarlos con nuestros archivos de Sass: Source files

Una vez descargado el archivo, lo descomprimimos y lo guardamos en el directorio que queramos y/o nos sea fácil acceder a él cada vez que queramos usarlo.

Otra opción para descargar Bootstrap **en el proyecto** que queramos es usar npm: **`npm i bootstrap@5.3.2`** (Si queremos usar otros gestores consultad aquí)

La otra opción que tenemos es en lugar de descargar en nuestro equipo/proyecto el código de Bootstrap es enlazarlo directamente a nuestro código HTML (CDN via jsDelivr)

Enlace al CSS y JS de nuestro bootstrap descargado:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
```

Los componentes que requieren explícitamente JS and Popper son los siguientes: Alerts, Buttons/checkbox/radio con funcionalidades, Carousel, para ver más consultar aquí.

2.3.2.Bootstrap & Vite

Ya vimos cómo instalar npm en el tema anterior, además vimos cómo crear un proyecto Vite. Ahora en este tema vamos a ver cómo podemos crear un proyecto Vite con Bootstrap.

En primer lugar crearemos un proyecto de Vite como vimos en las unidades anteriores: `npm create vite`

Una vez que hemos creado el proyecto (`npm i`), pasamos a instalar bootstrap, en el caso de también necesitar Popper, aprovechamos y hacemos la instalación de las dos cosas:

```
npm i bootstrap @popperjs/core
```

Una vez instalado, en nuestro **index.html** pasaremos a enlazar los estilos de **boostrap** en el **head**:

```
<link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.min.css">
```

Veamos un ejemplo completo, donde muestre un mensaje y el botón con las clases de boostrap para ver que efectivamente funciona todo.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container py-4 px-3 mx-auto">
        <h1>Hello, Bootstrap and Vite!</h1>
        <button class="btn btn-primary">Primary button</button>
    </div>
</body>
</html>
```

Si compilamos(`npm run dev`), el resultado que obtenemos es:

2.4. Contenido del directorio Bootstrap

Muchos de vosotros os preguntaréis qué contiene el directorio que nos hemos descargado. Una vez que hemos descargado y descomprimido el archivo tenemos dentro algo parecido a:

```
bootstrap/
├── dist/
│   ├── css/
│   └── js/
├── site/
│   └──content/
│      └── docs/
│          └── 5.3/
│              └── examples/
├── js/
└── scss/
```

En los directorios **scss/** y **js/** está el código de nuestro CSS y JavaScript. El directorio **dist/** incluye todo lo que aparece en la sección precompilada. La carpeta **site/content/docs/** tiene el código fuente de la documentación y directorio **examples/** ejemplos de cómo usar Bootstrap. Más allá de estos, cualquier otro archivo incluido proporciona soporte para paquetes, información de licencias y desarrollo.

Si queremos ver en qué navegadores/plataformas está soportado Bootstrap, podemos consultarlo aquí.

2.5. Página básica

Bootstrap utiliza elementos HTML y propiedades CSS que requieren el uso del doctype de HTML5 para que funcionen, por lo tanto, es importante añadirlo en TODAS nuestras páginas:

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Bootstrap demo</title>
    </head>
    <body>
        <h1>Hello, world!</h1>
    </body>
</html>
```

Además, para asegurarnos que todo se está visualizando de forma correcta y que podemos utilizar el zoom en los dispositivos móviles, debemos añadir la etiqueta meta dentro de la cabecera <head> como aparece en el head del ejemplo anterior:

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

Ejemplo de una plantilla básica para trabajar con Bootstrap:

```
<!doctype html>
<html lang="es">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Bootstrap demo</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/
Dwwykc2MPK8M2HN" crossorigin="anonymous">
    </head>
    <body>
        <h1>Hello, world!</h1>
```

```
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
    </body>
</html>
```

3. Sistema de rejilla

Bootstrap se basa en un sistema de rejillas flexibles, las cuales se escalan y posicionan de diferente forma dependiendo de la pantalla en la que se está visualizando la web.

3.1. Elemento contenedor

El sistema de rejilla tiene que ser utilizado dentro de uno de los dos elementos contenedores que tiene Bootstrap: `.container`, `.container-fluid` ó `.container-{breakpoint}`. Los contenedores se utilizan para “contener”, rellenar y a veces centrar el contenido que hay en ellos.

De los tres tipos de contenedores que hay:

- `.container`, establece un max-width en cada breakpoint responsive
- `.container-fluid,` establece un width: 100% en todos los breakpoints
- `.container-{breakpoint},` establece un width: 100% hasta el breakpoint especificado

Si lo que queremos es que el contenido de nuestra web aparezca centrado y con un ancho fijo, entonces utilizamos la clase .container. En cambio, si queremos que ocupe todo el ancho disponible (width: 100%), usamos la clase .container-fluid. (Ver ejemplo **rejillas1.html** y **rejillas2.html**). Para más información aquí.

3.2. Funcionamiento del sistema de rejilla

El Sistema de rejilla está pensado para ayudarnos en la disposición del contenido de la web y su adaptación a las diferentes pantallas. Para ello, el contenido debe estar dentro de celdas o columnas que irán dentro de filas. Cada fila se puede dividir hasta en 12 columnas, y nosotros le iremos definiendo las columnas para cada tamaño de pantalla.

El funcionamiento de las rejillas es:

- El **grid** de **Boostrap** tiene **seis breakpoints responsive**. Cada breakpoint tiene un `min-width`, de tal forma que afecta a ese breakpoint a todos los que hay por encima, es decir, `.col-sm-4` se aplicará tanto a `sm`, `md`, `lg`, `xl` y `xxl`. Esto significa que podemos controlar el comportamiento del tamaño de cada columna con estos breakpoints.
- Los **contenedores** centran y alinean horizontalmente el contenido. Para ello usamos `.container` para un ancho responsive, `.container-fluid` para un `width:100%` en todos los viewports o un contenedor responsive (ejemplo `.container-md`) para combinar entre fluid y px de ancho.

- Las **filas** (`.row`) son **envoltorios** para las **columnas**. Cada columna tiene un `padding` horizontal (**gutter**) parar controlar el espacio que hay entre ellas. Este `padding` se contrarresta en las filas con el fin de asegurar que el contenido de las columnas está alineado en el izquierdo. También las filas admiten clases para modificar uniformemente el tamaño de la columna y clases gutter para cambiar su espaciado.
- Las **columnas son flexibles**. Hay 12 columnas de plantilla disponibles por fila, lo que nos permite crear diferentes combinaciones. Las clases de columna indican el número de colmnas de plantilla que abarcan (por ejemplo, col-4 significa que ocupa 4 columnas). El tamaño (width) se establece en porcentajes, de forma que siempre tendremos el mismo tamaño de forma relativo.
- **Gutters** también son **responsive** y se pueden **personalizar**. Las clases gutter están disponibles en todos los breakpoints con los mismos tamaños que margin y padding. Podemos cambiar los gutter horizontales con las clases `gx-*`, los gutter verticales con `gy-*` o todos los gutter (horizontales y verticales) con `gx-*`. Si usamos `g-0` estamos eliminando los gutter.
- Las **variables de Sass, maps, y mixin** añaden funcionalidades a la cuadrícula de Boostrap. Si no queremos usar las clases de cuadrícula que hay predefinida en bootstrap, podemos usar el fuente para crear el nuestro. Además, hay algunas propiedades personalizadas de CSS para dar una mayor flexibilidad.

En la siguiente tabla se recoge un resumen del funcionamiento de la rejilla de Bootstrap dependiendo del tamaño de la pantalla y las clases CSS:

Pantalla (tamaño)	Prefijo de la clase	Ancho del contenedor
Extra pequeño **(<576px)**	.col-	Ninguno (automático)
Pequeño **(⩾576px)**	.col-sm-	540px
Mediano **(⩾768px)**	.col-md-	720px
Largo **(⩾992px)**	.col-lg-	960px
Extra largo **(⩾1200px)**	.col-xl-	1140px
Extra extra largo **(⩾1400px)**	.col-xxl-	1320px

Es importante tener en cuenta que, al definir estas clases, no sólo se aplican para ese tamaño de pantalla, sino que para las superiores también. Por ejemplo, si no aplicamos clase para tamaños de pantalla medianos y grandes pero sí para pequeños (`.col-sm-`), se usará esta, si los defines se sobrescriben y pasan a utilizar los definidos para dichos tamaños.

3.2.1. Ejemplo de rejilla

Dos filas donde cada columna tendrá el mismo ancho adaptándose a cualquier resolución. (rejillas3.html)

```
<div class="container text-center">
    <div class="row">
        <div class="col">1 of 2</div>
        <div class="col">2 of 2</div>
    </div>
```

```
    <div class="row">
        <div class="col">1 of 3</div>
        <div class="col">2 of 3</div>
        <div class="col">3 of 3</div>
    </div>
</div>
```

1 of 2	2 of 2
1 of 3	2 of 3	3 of 3

3.2.2.Ejemplo de apilado (stacked) en horizontal

Ejemplo donde los elementos están en horizontal hasta que la resolución de pantalla llega a sm y pasan a estar apilados en vertical (rejillas4.html)

```
<div class="container text-center">
    <div class="row">
        <div class="col-sm-8">col-sm-8</div>
        <div class="col-sm-4">col-sm-4</div>
    </div>
    <div class="row">
        <div class="col-sm">col-sm</div>
        <div class="col-sm">col-sm</div>
        <div class="col-sm">col-sm</div>
    </div>
</div>
```

col-sm-8	col-sm-4
col-sm	col-sm	col-sm

3.2.3.Ejemplo mix and match

Combinación de diferentes formas de agrupar los elementos en función de la resolución(rejillas5.html)

```
<div class="container text-center">
    <!-- Stack the columns on mobile by making one full-width and the other half-width -->
    <div class="row">
        <div class="col-md-8">.col-md-8</div>
        <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    </div>
    <!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
    <div class="row">
        <div class="col-6 col-md-4">.col-6 .col-md-4</div>
        <div class="col-6 col-md-4">.col-6 .col-md-4</div>
        <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    </div>
    <!-- Columns are always 50% wide, on mobile and
desktop -->
    <div class="row">
        <div class="col-6">.col-6</div>
        <div class="col-6">.col-6</div>
    </div>
</div>
```

.col-md-8	
.col-6 .col-md-4	
.col-6 .col-md-4	.col-6 .col-md-4
.col-6 .col-md-4	
.col-6	.col-6

3.2.4.Ejemplo row columns

Si usamos las clases .row-cols-* podemos establecer la cantidad de columnas que podemos tener. (rejillas6.html)

```
<div class="container text-center">
    <div class="row row-cols-2">
        <div class="col">Column</div>
        <div class="col">Column</div>
        <div class="col">Column</div>
        <div class="col">Column</div>
    </div>
</div>
```

Column	Column
Column	Column

3.2.5.Ejemplo de nesting

Podemos anidar contenido, podemos añadir un nuevo .row y más columnas .col- dentro de una columna existente. Las filas que anidamos debemos tener que sumarían como mucho 12 o menos (no es necesario que usemos las 12 columnas disponibles de bootstrap). (rejillas7.html)

```
<div class="container text-center">
    <div class="row">
        <div class="col-sm-3">Level 1: .col-sm-3</div>
        <div class="col-sm-9">
            <div class="row">
                <div class="col-8 col-sm-6">Level 2: .col-8 .col-sm-6</div>
                <div class="col-4 col-sm-6">Level 2: .col-4 .col-sm-6</div>
            </div>
        </div>
    </div>
</div>
```

Level 1: .col-sm-3	Level 2: .col-8 .col-sm-6	Level 2: .col-4 .col-sm-6

3.3. Cómo ocultar celdas u otro elemento

Con el objetivo de facilitar el desarrollo de aplicaciones donde el contenido de la web sea menor en función de si se están visualizando en el móvil o en el ordenador, bootstrap tiene la clase .d-none para ocultarlo para todas las pantallas o .d-{sm,md,lg,xl,xxl}-none si queremos que se oculte en alguna de esas resoluciones en concreto.

Para mostrar los elementos que sea sólo para esa resolución y que se vea con el resto se deberá especificar la clase .d-md-block por ejemplo, donde se ocultarían en todas menos en las medias.

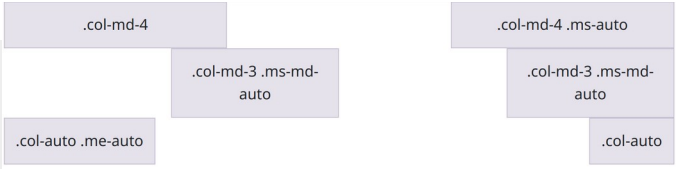
Tamaño de la pantalla	Clase
Oculto en todas	.d-none
Oculto sólo en xs	.d-none .d-sm-block
Oculto sólo en sm	.d-sm-none .d-md-block
Oculto sólo en md	.d-md-none .d-lg-block
Oculto sólo en lg	.d-lg-none .d-xl-block
Oculto sólo en xl	.d-xl-none -d-xxl-block
Oculto sólo en xxl	.d-xxl-none

Visible en todas	.d-block
Visible sólo en xs	.d-block .d-sm-none
Visible sólo en sm	.d-none .d-sm-block .d-md-none
Visible sólo en md	.d-none .d-md-block .d-lg-none
Visible sólo en lg	.d-none .d-lg-block .d-xl-none
Visible sólo en xl	.d-none .d-xl-block .d-xxl-none
Visible sólo en xl	.d-none .d-xxl-block

Más info aquí.

3.4. Márgenes o espaciado entre columnas (margin utilities)

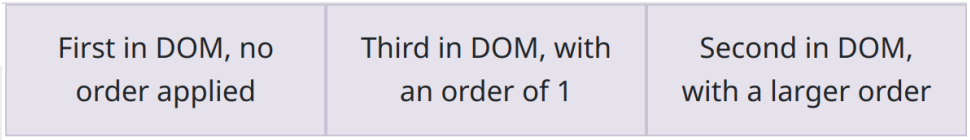
Desde la versión 4 de Bootstrap, ya no se usa el offset que se empleaba en las versiones anteriores, en lugar de eso utilizamos la clase .me-auto por ejemplo para forzar a las columnas a que tengan un espacio a la derecha. Más info aquí. Veamos un ejemplo (rejillas8.html):




```
<div class="container text-center">
    <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
    </div>
    <div class="row">
        <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
        <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
    </div>
    <div class="row">
        <div class="col-auto me-auto">.col-auto .me-auto</div>
        <div class="col-auto">.col-auto</div>
    </div>
</div>
```

3.5. Ordenación de columnas (reordering)

Podemos usar la clase .order- para controlar de forma visual el orden de nuestro contenido. Estas clases son responsive, por tanto, podemos establecer el orden teniendo en cuenta que el grid tiene 12 columnas. Si por ejemplo queremos que en una resolución md (o superior) esta columna esté la segunda usaremos la clase .order-md-2, y también está la posibilidad de usar .order-first y .order-last (ver rejillas9.html).



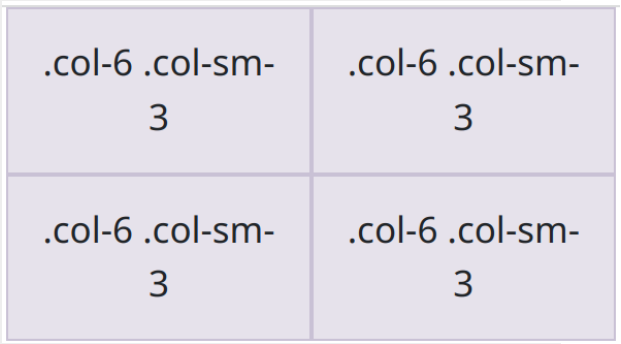

```
<div class="container text-center">
    <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
    </div>
    <div class="row">
        <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
        <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
```

```
    </div>
    <div class="row">
        <div class="col-auto me-auto">.col-auto .me-auto</div>
        <div class="col-auto">.col-auto</div>
    </div>
</div>
```

3.6. Cómo “romper” una columna (salto)

Con la clase .w-100, lo que conseguimos es romper para que la siguiente columna baje a una línea nueva (realmente estás añadiendo un elemento con un width del 100%). Normalmente esto se consigue insertando una nueva .row, pero si por ejemplo queremos que esto ocurra en ciertas resoluciones usaremos este elemento de width: 100% (.w-100). (rejillas10.html)

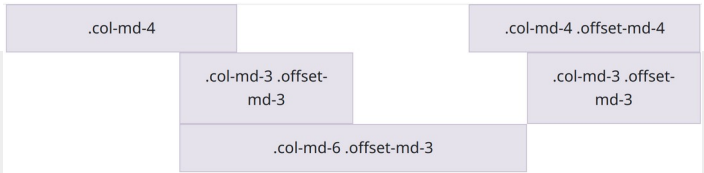
```
<div class="container text-center">
    <div class="row">
        <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
        <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
        <!-- Force next columns to break to new line -->
        <div class="w-100"></div>
        <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
        <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    </div>
</div>
```



3.7. Cómo mover columnas (offseting)

Podemos desplazar las columnas de dos formas en bootstrap: la que veremos aquí es usando las clases propias del grid .offset-, la otra es utilizando las utilidades de los márgenes.

Para mover las columnas a la derecha usaremos .offset-md-*, con esto moveremos la columna * posiciones cuando el tamaño de la pantalla sea media. Veamos un ejemplo (rejillas11.html)




```
<div class="container text-center">
    <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
    </div>
    <div class="row">
        <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
        <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
    </div>
    <div class="row">
        <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
    </div>
</div>
```

4. Media queries (media object?)

En la mayoría de los casos con las clases que vienen en Bootstrap nos es suficiente para diseñar nuestra web, sin embargo, en algunas situaciones a veces queremos modificar cierto comportamiento como puede ser para aplicar colores, alineación, etc. que cambie dependiendo del tamaño de la pantalla. Será en estos casos cuando utilicemos los media queries.

Bootstrap usa los siguientes rangos de media query (o breakpoints) en sus archivos de Sass para el layout, grid y componentes.

4.1. min-width

Bootstrap usa las siguientes media queries en sus archivos de sass:

```
// Source mixins

// No media query necessary for xs breakpoint as it's effectively `@media (min-width: 0) { ... }`
@include media-breakpoint-up(sm) { ... }
@include media-breakpoint-up(md) { ... }
@include media-breakpoint-up(lg) { ... }
@include media-breakpoint-up(xl) { ... }
@include media-breakpoint-up(xxl) { ... }

// Usage

// Example: Hide starting at `min-width: 0`, and then show at the `sm` breakpoint
.custom-class {
  display: none;
}
@include media-breakpoint-up(sm) {
  .custom-class {
    display: block;
  }
}
```

Estos mixin se traducen en CSS:

```
// X-Small devices (portrait phones, less than 576px)
// No media query for `xs` since this is the default in Bootstrap

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }
```

```
// X-Large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }

// XX-Large devices (larger desktops, 1400px and up)
@media (min-width: 1400px) { ... }
```

4.2. max-width

Si usamos los media queries en la otra dirección (tamaño máximo):

```
// No media query necessary for xs breakpoint as it's effectively `@media (max-width: 0) { ... }`
@include media-breakpoint-down(sm) { ... }
@include media-breakpoint-down(md) { ... }
@include media-breakpoint-down(lg) { ... }
@include media-breakpoint-down(xl) { ... }
@include media-breakpoint-down(xxl) { ... }

// Example: Style from medium breakpoint and down
@include media-breakpoint-down(md) {
  .custom-class {
    display: block;
  }
}
```

Estos mixin se traducen en CSS:

```
// `xs` returns only a ruleset and no media query
// ... { ... }

// `sm` applies to x-small devices (portrait phones, less than 576px)
@media (max-width: 575.98px) { ... }

// `md` applies to small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }

// `lg` applies to medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }

// `xl` applies to large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }

// `xxl` applies to x-large devices (large desktops, less than 1400px)
@media (max-width: 1399.98px) { ... }
```

4.3. single breakpoint

Cuando queremos un media querie para un cierto tamaño de pantalla (min y max) tenemos:

```
@include media-breakpoint-only(xs) { ... }
@include media-breakpoint-only(sm) { ... }
@include media-breakpoint-only(md) { ... }
@include media-breakpoint-only(lg) { ... }
@include media-breakpoint-only(xl) { ... }
@include media-breakpoint-only(xxl) { ... }
```

Se traduce a

```
@media (min-width: 768px) and (max-width: 991.98px) { ... }
```

4.4. between breakpoint

De formal similar, si queremos abarcar a múltiples anchos:

```
@include media-breakpoint-between(md, xl) { ... }
```

Se traduce a

```
// Example
// Apply styles starting from medium devices and up to extra large devices
@media (min-width: 768px) and (max-width: 1199.98px) { ... }
```

5. Ejemplo de uso

Por ejemplo, si queremos que en pantallas extrapequeñas el fondo sea rojo (`myBackground`) y que para el resto de los tamaños el fondo sea verde haremos:

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