

CODES

Capacitación Web – HTML – Introducción

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

HTML, siglas de HyperText Markup Language («lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.

El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). Puede incluir o hacer referencia a un tipo de programa llamado script, el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

También consta de varios componentes vitales, entre ellos los elementos y sus atributos, tipos de data y la declaración de tipo de documento.

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de notas de Windows, o cualquier otro editor que admita texto sin formato como Microsoft Wordpad, TextPad, Notepad++, entre otros.

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

<html> define el inicio del documento HTML, le indica al navegador que lo que viene a continuación debe ser interpretado como código HTML.

<script> incrusta un script en una web, o llama a uno mediante src="url del script". Se recomienda incluir el tipo MIME en el atributo type, en el caso de JavaScript text/javascript.

<head> define la cabecera del documento HTML; esta cabecera suele contener información sobre el documento que no se muestra directamente al usuario como, por ejemplo, el título de la ventana del navegador. Dentro de la cabecera **<head>** es posible encontrar:

<title>: define el título de la página. Por lo general, el título aparece en la barra de título encima de la ventana.

<link>: para vincular el sitio a hojas de estilo o iconos. Por ejemplo: <link rel="stylesheet" href="/style.css" type="text/css">.

<style>: para colocar el estilo interno de la página; ya sea usando CSS u otros lenguajes similares. No es necesario colocarlo si se va a vincular a un archivo externo usando la etiqueta **<link>**.

<body>: define el contenido principal o cuerpo del documento. Esta es la parte del documento html que se muestra en el navegador; dentro de esta etiqueta pueden definirse propiedades comunes a toda la página, como color de fondo y márgenes. Dentro del cuerpo <body> es posible encontrar numerosas etiquetas. A continuación se indican algunas a modo de ejemplo:

<h1> a <h6>: encabezados o títulos del documento con diferente relevancia.

<a>: hipervínculo o enlace, dentro o fuera del sitio web. Debe definirse el parámetro de pasada por medio del atributo href.

Por ejemplo:

```
<a href="http://www.example.com" title="Ejemplo" tabindex="1">Ejemplo</a>
```

<div>: división de la página. Se recomienda, junto con css, en vez de <table> cuando se desea alinear contenido.

****: imagen. Requiere del atributo src, que indica la ruta en la que se encuentra la imagen. Por ejemplo: . Es conveniente, por accesibilidad, poner un atributo alt="texto alternativo".

elemento de la lista desordenadas (puntos)

****: etiquetas para listas.

ordenadas (numeradas)

**
**: Instruye al navegador cliente que inserte un salto de línea en un documento HTML. La etiqueta
 tiene el mismo efecto que un retorno de carro en una máquina de escribir. Es una etiqueta especial, pues no precisa de etiqueta de cierre.

<hr>: Dibuja de manera predeterminada una regla horizontal alineada automáticamente, con una apariencia de tercera dimensión. Esta etiqueta especial, por que no necesita de cierre, tiene los siguientes atributos:

- Align establece que la regla se alinee a la izquierda, centro o derecha LEFT,CENTER o RIGHT
- NOSHADE quita el sombreado predeterminado de la regla
- WIDTH permite especificar el ancho de la regla (en pixeles o porcentaje)
- SIZE permite especificar el alto de la regla (en pixeles)

```
<html>
    <head> <title>Curso de HTML</title> </head>
    <body bgcolor="#C0D9D9" text="#000000">
        Bienvenidos al curso de HTML<br>
        <hr align=center width=50%><br>
        Cuando la temperatura es menor a 15&deg;c hace bastante fr&iacute;o.<br>
        <hr align=left width=25% size=5><br>
        Este es un ejemplo de p&aacute;gina WEB :)<br>
    </body>
</html>
```

Ubicación, formato y atributos de texto

Etiqueta **<center> </center>**

Se utiliza para centrar el texto/imagen o datos que se encuentren entre la apertura y el cierre.

Etiqueta ** **

Esta es la etiqueta que nos posibilita un texto en **negrita**.

Etiqueta **<u> </u>**

Etiqueta que posibilita resaltar un texto con subrayado.

Etiqueta **<i> </i>**

Etiqueta que permite resaltar el texto con *inclinación itálica*.

Cabe mencionar que **se pueden combinar entre si todas estas etiquetas**.

Tablas

En HTML también podemos incluir arreglos de tablas. Se deben utilizar varias etiquetas:

Etiqueta **<table></table>**

Señala el **inicio y final de una tabla**. Sus **atributos** son:

- **Align** Establece la **alineación de la tabla o texto** mediante ALIGN=LEFT o ALIGN=RIGHT
- **Bgcolor** Establece el **color de fondo de las celdas** de la tabla
- **Border** Determina el **ancho del borde** en pixeles
- **BorderColor** Asigna un **color al borde**
- **BorderDark** Determina el **color de la parte oscura** de un borde de 3 dimensiones
- **BorderLight** Asigna el **color de la parte clara** de un borde de 3 dimensiones
- **Caption** Especifica el **título para la tabla**
- **Cellpadding** Establece la **cantidad de espacio libre junto al contenido** de una celda
- **Cellspacing** Asigna la **cantidad de espacio entre** las celdas de una tabla
- **Width** Determina el **ancho de la tabla** en pixeles o en un porcentaje

Etiqueta **<tr> </tr>**

Indica al navegador que exhiba el **texto dentro de una fila**; puede también interpretarse como la **etiqueta que define filas**.

Etiqueta **<td> </td>**

La etiqueta de datos de la tabla, es la que identifica a las columnas o celdas específicas de una tabla. Atributos principales:

- **Align** Alineación del texto/objeto de la celda
- **Colspan** especifica el número de celdas que cubre el encabezado
- **Bgcolor** Color de fondo de la celda
- **Background** imagen de fondo de una celda
- **Width** Ancho de la celda/columna con respecto al ancho de la tabla

Solo precisa definir el ancho en la primera celda de la columna.

Recuerde que dentro de una celda, usted puede insertar desde texto o un gráfico hasta una tabla entera.

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

3. Estructura de página HTML

A continuación se muestra una visualización de una estructura de la página HTML

```
<html>
  <body>
    Esta <h1> un título </ h1>
    <p> Este es un párrafo. </ p>
    <p> Este es otro párrafo. </ p>
  </ Body>
</ Html>
```

<! DOCTYPE>: El <! DOCTYPE> ayuda a que el navegador muestre una página web correctamente.

Hay muchos documentos en la web y un navegador sólo puede mostrar una página HTML 100% correctamente si se conoce el tipo de HTML y la versión utilizada.

Atributos HTML

Los atributos proporcionan información adicional acerca de los elementos HTML.

- Se especifican en la etiqueta de inicio
- Vienen en pares de nombre / valor como: nombre = "valor"

Ejemplo: los vínculos HTML se definen con la etiqueta <a>. La dirección del enlace se especifica en el atributo href:

```
<a href="http://www.google.com.ar">Link a Google</a>
```

Atributos más comunes en HTML

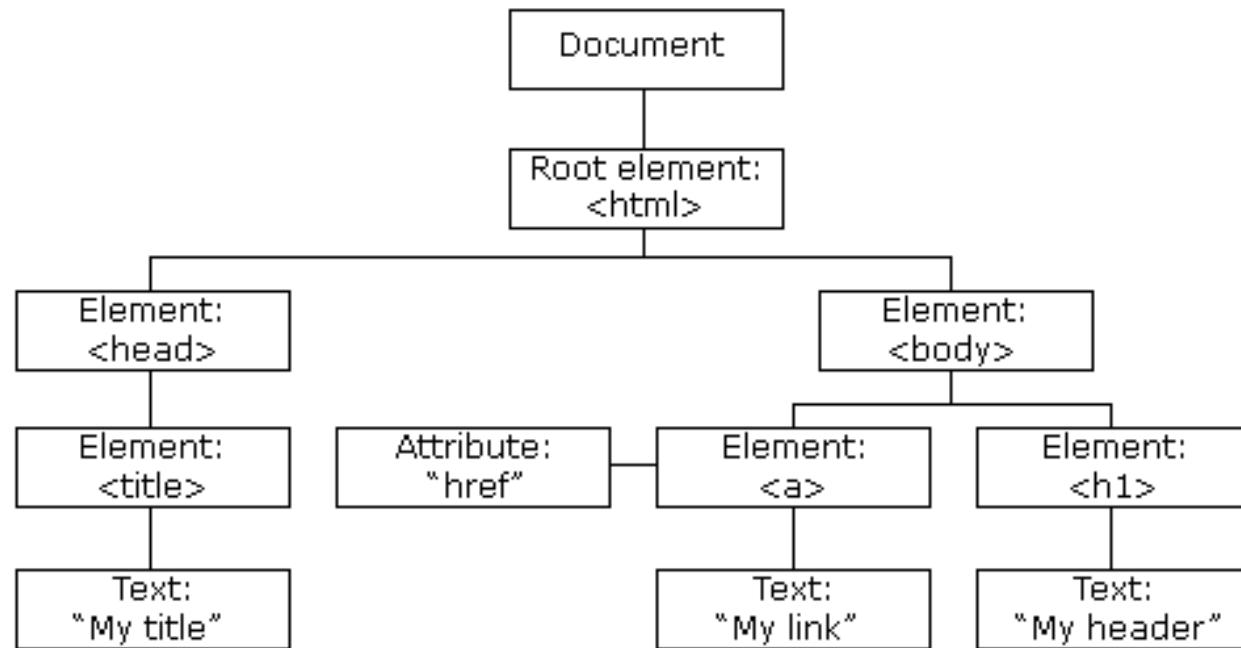
Atributo	Descripción
class	Especifica uno o más nombres de clases para un elemento (las clases deben estar definidas en la hoja de estilos)
id	Especifica un ID único para un elemento
style	Especifica un estilo CSS “inline” para un elemento
title	Especifica información extra sobre un elemento (tool tip)

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

El **Modelo de Objetos del Documento** (DOM) es una **interfaz de programación de aplicaciones** (**API - Application Programming Interface**) para documentos **HTML**. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

Cuando se carga una página, el navegador crea el **Modelo de Objetos del Documento** (DOM), el cual es construido como un árbol de objetos.



Con el DOM, a través de JavaScript podemos manipular el HTML dinámicamente:

- Se puede cambiar todos los elementos HTML de la página
- Se puede cambiar todos los atributos HTML en la página
- Se puede cambiar todos los estilos CSS en la página
- Se puede quitar elementos y atributos HTML existentes
- Se puede añadir nuevos elementos y atributos HTML
- Se puede reaccionar a todos los eventos de HTML existentes en la página
- Se puede crear nuevos eventos HTML en la página

El método **getElementById**

La forma más común para acceder a un elemento HTML es utilizar el id del elemento:

```
var txt=document.getElementById("intro").innerHTML;
```

La propiedad **innerHTML**

La forma más fácil de obtener el contenido de un elemento es mediante el uso de la innerHTML propiedad.

La propiedad innerHTML es útil para conseguir o reemplazar el contenido de los elementos HTML.

4. HTML DOM – métodos de acceso a elementos

Método	Descripción
<code>document.getElementById()</code>	Recupera un elemento por su ID
<code>document.getElementsByTagName()</code>	Recupera elementos por su nombre
<code>document.getElementsByClassName()</code>	Recupera elementos por su nombre de clase
<code>document.forms[]</code>	Recupera elementos como colección de objetos HTML

Para realizar cambios de elementos HTML, se utilizan los siguientes métodos:

Método	Descripción
<code>element.innerHTML=</code>	Cambiar el inner HTML de un elemento
<code>element.attribute=</code>	Cambiar el atributo de un elemento HTML
<code>element.setAttribute(attribute,value)</code>	Cambiar el atributo de un elemento HTML
<code>element.style.property=</code>	Cambiar el estilo de un elemento HTML

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Explicación del ejemplo arriba descrito:

- La declaración **DOCTYPE** define el **tipo de documento**
- El texto entre **<html>** y **</ html>** **describe la página web**
- El texto entre **<body>** y **</ body>** es el **contenido de la página visible**
- El texto entre **<h1>** y **</ h1>** **se muestra como un título**
- El texto entre **<p>** y **</ p>** **se muestra como un párrafo**

5. Ejemplos – Comentarios

Se puede agregar **comentarios** al código fuente HTML utilizando la siguiente sintaxis:

```
<!-- Escribir comentarios aquí -->
```

Los comentarios **no se muestran en el navegador**, pero pueden ayudar a documentar el código HTML.

Con los comentarios se pueden colocar las notificaciones y recordatorios en el código HTML:

```
<!-- Este es un comentario -->
```

```
<p>Este es un párrafo.</p>
```

```
<!-- Recordar agregar más información aquí. -->
```

Hipervínculos HTML (Links)

La etiqueta <a> HTML define un hipervínculo. Sirve para saltar a otro documento. El atributo más importante del elemento <a> es el atributo href, que indica el destino del vínculo.

Sintaxis Enlace HTML

El código HTML de un enlace es simple. Se parece a esto: Link text

Ejemplo:

```
<a href="http://www.w3schools.com/">Visitar W3Schools</a>
```

Nota: El “Texto del enlace” no tiene que ser un texto. Puede ser una imagen o cualquier otro elemento HTML.

Enlaces HTML - el atributo de destino

El atributo target especifica dónde abrir el documento vinculado.

El ejemplo siguiente abrirá el documento vinculado en una nueva ventana o una nueva pestaña:

```
<a href="http://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

Enlaces HTML – El ID del elemento <a>

El atributo id puede ser usado para crear un marcador dentro de un documento HTML.

Nota: Los marcadores no se muestran de ninguna manera especial. Ellos son invisibles para el lector.

Ejemplo:

Un ancla con un id dentro de un documento HTML:

```
<a id="tips">Sección Consejos Útiles</a>
```

El siguiente es un enlace al ancla “Sección Consejos Útiles” dentro del mismo documento:

```
<a href="#tips">Visite la Sección de Consejos Útiles</a>
```

O bien, se puede crear un vínculo a la “Sección Consejos Útiles” de otra página:

```
<a href="http://www.w3schools.com/html_links.htm#tips">Visite la Sección Consejos Útiles</a>
```

Tablas HTML

Las tablas se definen con la etiqueta <table>.

Una tabla se divide en las filas con la etiqueta <tr>. (Tr es sinónimo de fila de la tabla)

Una fila se divide en celdas de datos con la etiqueta <td>. (Td significa datos de la tabla)

Una fila también se puede dividir con la etiqueta <th>. (Th significa encabezado de la tabla)

Los elementos <td> son los contenedores de datos en la tabla. Pueden contener todo tipo de elementos HTML, como texto, imágenes, listas, otras tablas, etc.

El ancho de una tabla puede ser definido mediante CSS.

```
<table style="width:300px">
<tr>
  <td>Jill</td>
  <td>Smith</td>
  <td>50</td>
</tr>
<tr>
  <td>Eve</td>
  <td>Jackson</td>
  <td>94</td>
</tr>
</table>
```

Los **Formularios** HTML se utilizan para pasar datos a un servidor.

Un formulario HTML puede contener elementos de entrada como **campos de texto, casillas de verificación, botones de radio, presentar los botones** y más. Un formulario puede contener **listas de selección, textarea, fieldset, legend, y demás etiquetas**.

La etiqueta **<form>** se utiliza para crear un formulario HTML:

```
<form>  
  .  
  . input elements  
  .  
</form>
```

El elemento de entrada más importante es el elemento **<input>**, el cual **se utiliza para seleccionar la información del usuario**.

Un elemento **<input>** **puede variar** de muchas maneras, **dependiendo del tipo de atributo**. Un elemento **<input>** puede ser de tipo campo de texto, casilla de verificación, botón de contraseña, radio, botón de enviar, y más.

<input type="text"> define un campo de entrada de una línea donde un usuario puede introducir texto:

```
<form>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>
```

<input type="password"> define un campo de contraseña:

```
<form>
Password: <input type="password" name="pwd">
</form>
```

Nota: Los caracteres en un campo de contraseña se enmascaran (mostrado como asteriscos o círculos).

<input type="radio"> define un botón de radio. Los botones de radio permiten al usuario seleccionar sólo una de un número limitado de opciones:

```
<form>
<input type="radio" name="sex" value="male">Male<br>
<input type="radio" name="sex" value="female">Female
</form>
```

5. Ejemplos – Elementos de entrada más importantes

<input type="checkbox"> define una casilla de verificación. Las casillas de verificación permiten al usuario seleccionar cero o más opciones de un número limitado de opciones.

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>
```

<input type="submit"> define un botón de envío.

Un botón de enviar se utiliza para enviar datos del formulario a un servidor. Los datos se envían a la página especificada en el atributo action del formulario.

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

5. Ejemplos – Elementos de entrada más importantes

<select><option value="1">valor 1</option></select>

El elemento **<select>** se usa para crear **listas de valores desplegables** (comboboxes). El tag **<option>** dentro del elemento **<select>** define las opciones en la lista.

El siguiente ejemplo muestra una lista de marcas de automóviles:

```
<select disabled="disabled">
  <option value="volvo">Volvo</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

Si se especifica el atributo **multiple** a **true**, la lista cambia su forma de visualización y permite la selección múltiple de elementos:

```
<select multiple="true">
  <option value="volvo">Volvo</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

Y si se especifica el atributo **disabled** en “**disabled**”, la lista no está habilitada para su uso:

```
<select disabled="disabled">
  <option value="volvo">Volvo</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

5. Ejemplos – El elemento <script>

La etiqueta <script> se utiliza para definir un script del lado del cliente, tales como JavaScript.

El elemento <script> puede contener declaraciones de scripting, o apuntar a un archivo de script externo a través del atributo src.

Los usos más comunes de JavaScript son la manipulación de imágenes, validación de formularios, y los cambios dinámicos de contenido.

El siguiente script escribe Hello World! a la salida de HTML:

```
<script>
document.write("Hello World!")
</script>
```

5. Ejemplos – El elemento <script>

He aquí algunos ejemplos de lo que puede hacer JavaScript:

- JavaScript puede escribir directamente en la corriente de salida HTML:

```
document.write("<p>This is a paragraph</p>");
```

- JavaScript puede reaccionar a los eventos:

```
<button type="button" onclick="myFunction()">Click Me!</button>
```

- JavaScript puede manipular estilos HTML:

```
document.getElementById("demo").style.color="#ff0000";
```

5. Ejemplos – Varios controles en una página

```
<html>
<head>
<title>Testing...</title>
<script language="javascript" type="text/javascript">
function IrAGoogle() { if(confirm("Desea abandonarnos?")) {window.location.href = "http://www.google.com.ar";} }
function LeerValor() { alert(document.getElementById("txtValor1").value); }
function LeerCombo() {
    alert("value: " + document.getElementById("cboValores").value);
    var comboAux = document.getElementById("cboValores");
    alert("text: " + comboAux.options[comboAux.selectedIndex].text);
}
function VerOpcion() {
    var optionAux = document.getElementsByName("opciones");
    if(optionAux[0].checked) {alert("Seleccionó " + optionAux[0].value); }
    else if(optionAux[1].checked) { alert("Seleccionó " + optionAux[1].value); }
    else { alert("Seleccionó " + optionAux[2].value); }
}
</script>
</head>
<body>
<input type="button" value="ir a google" onclick="IrAGoogle();">
<br/>
<input type="text" value="Valor1" id="txtValor1"><input type="button" value="Leer Valor" onclick="LeerValor();">
<br/>
<select id="cboValores">
<option value="1">Valor 1</option>
<option value="2">Valor 2</option>
<option value="3">Valor 3</option>
</select>
<input type="button" value="Leer combo" onclick="LeerCombo();">
<br/>
<input type="radio" name="opciones" value="opcionA" />Opción A
<input type="radio" name="opciones" value="opcionB" />Opción B
<input type="radio" name="opciones" value="opcionNinguna" />Ninguna
<input type="button" value="Ver opción" onclick="VerOpcion();">
</body>
</html>
```

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

6. Acentos y caracteres especiales

Muchas veces necesitamos incluir en nuestros textos, signos que tienen un significado especial en HTML (por ej. "<" - menor que).

Para poder mostrarlos, debemos usar su nombre en código.

- Los nombres de las entidades están compuestos por el signo(&), luego el nombre de la entidad y al final (";" - punto y coma).
- Los números de estos caracteres están compuestos por (&), luego (# - numeral), el número de la entidad y al final (";" - punto y coma).

Por ejemplo para mostrar el signo "<" debemos escribir < o <.

El uso más común de los caracteres especiales es el espacio en blanco. Si en un texto figuran 5 espacios en blanco seguidos, HTML automáticamente borra 4.

Para ingresar espacios en blanco usamos " " y HTML los dejará en su lugar.

Otro uso muy frecuente es el de insertar acentos en el código html por medio de los números de las entidades

6. Acentos y caracteres especiales de uso frecuente

Resultado	Descripción	Nombre	Número
	Espacio en blanco	 	
<	Menor que	<	<
>	Mayor que	>	>
&		&	&
"	Comillas	"	"
i	Apertura signo de exclamación	¡	¡
¿	Apertura signo de interrogación	?	¿
®	Marca registrada	®	®
©	Derecho de autor	©	©
á	a minúscula con acento	á	á
é	e minúscula con acento	é	é
í	i minúscula con acento	í	í
ó	o minúscula con acento	ó	ó
ú	u minúscula con acento	ú	ú
ñ	ñ minúscula	ñ	ñ
ü	u minúscula con diéresis	ü	ü
Á	A mayúscula con acento	Á	Á
É	E mayúscula con acento	É	É
Í	I mayúscula con acento	Í	Í
Ó	O mayúscula con acento	Ó	Ó
Ú	U mayúscula con acento	Ú	Ú
Ñ	Ñ mayúscula	Ñ	Ñ

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

Se pueden llegar a tener 16 millones de colores en una página web.
Existen dos formas para aplicar colores a una página web:

1. Se especifica el color deseado directamente con el nombre del color en inglés: Ej: blue, green, yellow
2. Se especifica el color deseado mediante números hexadecimales mediante la siguiente estructura:

#RRVVAA

El color tiene un signo de numeral # antecediendo a los 6 números.

Existen dos números para cada color principal: rojo, verde y azul.
Cada uno de los números varía hexadecimalmente {0,1,2,...,9,A,B,...F}.

Ejemplos de Colores:

#RRVVAA Color #RRVVAA Color

#FFFFFF Blanco #000000 Negro
#FF0000 Rojo #00FF00 Verde
#0000FF Azul #FF00FF Magente
#00FFFF Cyan #FFFF00 Amarillo
#70DB93 Agua Marino #000080 Azul Marino
#FF7F00 Coral #A62A2A Café
#C0C0C0 Plomo #4F2F4F Violeta

Utilizando estos datos, haremos una página con fondo celeste y letras negras. Usaremos para este efecto los atributos *bgcolor* y *text*.:

Ejemplo 3:

```
<html>
<head> <title>Curso de HTML</title> </head>
<body bgcolor="#C0D9D9" text="#000000">Bienvenidos al curso de HTML</body>
</html>
```

Grabe este archivo seleccionando la opción de *Guardar/Grabar* de su editor de texto, de modo que se mantenga el nombre index.html. Cuando usted vaya a su navegador WWW y seleccione la opción de *Actualizar*, notara el cambio.

ÍNDICE

1. HTML – Introducción
2. Códigos HTML básicos
3. Estructura de página HTML
4. HTML DOM
5. Ejemplos
6. Acentos y caracteres especiales
7. Manejo de Colores
8. Ejercicios

8. Ejercicios

1. Armar su CV en HTML. Utilice tablas para presentar la información.
2. Dibuje la siguiente página:

Nombre

Apellido

Aceptar

El botón ACEPTAR mostrará un ALERT diciendo: "Hola, {nombre} {apellido}!", donde {nombre} es lo ingresado en el campo NOMBRE, y {apellido} lo ingresado en el campo APELLIDO.

3. Dibuje una página que pida ingresar dos números, y que tenga un botón que realice la operación seleccionada con los dos números. Mostrar el resultado con un ALERT.

Número 1

Número 2

Sumar Restar Multiplicar Dividir

Aceptar

FIN
Gracias

CODES

Capacitación Web - Conceptos

4. HOJAS DE ESTILO

Que es CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

```
h1 {color: red;}  
    - h1 es el selector  
    - {color: red;} es la declaración
```

Las tres formas más conocidas de dar estilo a un documento son las siguientes:

Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento <link>, el cual debe ir situado en la sección <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">  
<html>  
  <head>  
    <title>Título</title>  
    <link rel="stylesheet" type="text/css" href="estilo.css" />  
  </head>  
<body>
```

4. HOJAS DE ESTILO

Utilizando el elemento `<style>`, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección `<head>`. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">
<html>
  <head>
    <title>hoja de estilo interna</title>
    <style type="text/css">
      body {
        padding-left: 11em;
        font-family: Georgia, "Times New Roman", serif;
        color: red;
        background-color: #d8da3d;
      }
    </style>
```

Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo `<style>` dentro de `<body>`. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

4. HOJAS DE ESTILO

AÑADIENDO ESTILO CON CSS

Para aplicar estilo a nuestras páginas podemos usar el lenguaje de programación conocido como CSS. CSS es la abreviación de hoja de estilos en cascada. El siguiente código (explicado en los pasos a continuación) fija como rojo el color de fondo de los párrafos de nuestra página y nos permite ver el aspecto de CSS.

```
p { background-color: red;}
```

1. Seleccionamos el elemento al que queremos aplicar estilo. En este ejemplo, el elemento <p>. Observe, que en CSS no se pone <>alrededor del nombre de la etiqueta.
1. Especificamos la propiedad a la que queremos aplicar estilo, en este caso el color de fondo del elemento. En inglés background-color.
1. Entre el nombre de la propiedad y el valor de la misma ponemos dos puntos.
2. Para definir el color de fondo como rojo, fijamos el valor de la propiedad a red.
3. Al final ponemos punto y coma
4. Recuerde colocar todos los elementos de estilo para el elemento <p>entre {}.

4. HOJAS DE ESTILO

Para añadir estilos CSS directamente en HTML, añada las etiquetas de apertura y cierre del elemento <style> con el atributo type de valortext/css en el elemento <head>.

Esto se muestra a continuación:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Rock and Beer Guadalajara</title>
    <style type="text/css">
      p { background-color: red; }
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

A menudo, nos va interesar que todas nuestras páginas usen las mismas reglas de estilo. Los periodicos, por ejemplo, usan a lo largo de todas sus páginas el mismo tamaño de letra para los encabezados, el mismo espacio entre lineas etc. ¿Cómo podemos hacer que todas nuestras páginas tengan las mismas reglas de estilo?. Imaginemos un portal con dos archivos; index.html (la página de inicio) ybebidas.html. Para lograr aplicar a ambas páginas las mismas reglas de estilo, realizaríamos los siguientes pasos:

- Sacar las reglas CSS del archivo index.html y colocarlas en un archivo llamado estilo.css.
- Crear un enlace externo a este archivo desde el archivoindex.html
- Añadir el mismo enlace externo en el archivo bebidas.html

4. HOJAS DE ESTILO

Usamos el elemento link para enlazar con información externa. En este caso enlazamos con un archivo externo de estilos. El código sería algo así:

```
<link type="text/css" rel="stylesheet" href="estilo.css"/>
```

Con el atributo type indicamos que el tipo de esta información es text/css. En otras palabras, una hoja de estilos en cascada, un CSS. El atributo href indica donde se encuentra el archivo de estilos. En este caso un camino relativo, aunque podría ser una URL completa. El atributo rel especifica la relación entre el archivo HTML y lo que estamos enlazando. Estamos enlazando una hoja de estilos, por lo que le damos el valor stylesheet.

CSS permite agrupar declaraciones de estilo comunes para varios elementos en una sola regla. Por ejemplo:

```
h1 { font-family: sans-serif; color: gray; }  
h1 { font-family: sans-serif; color: gray; }
```

Es equivalente a:

```
h1, h2 { font-family: sans-serif; color: gray; }
```

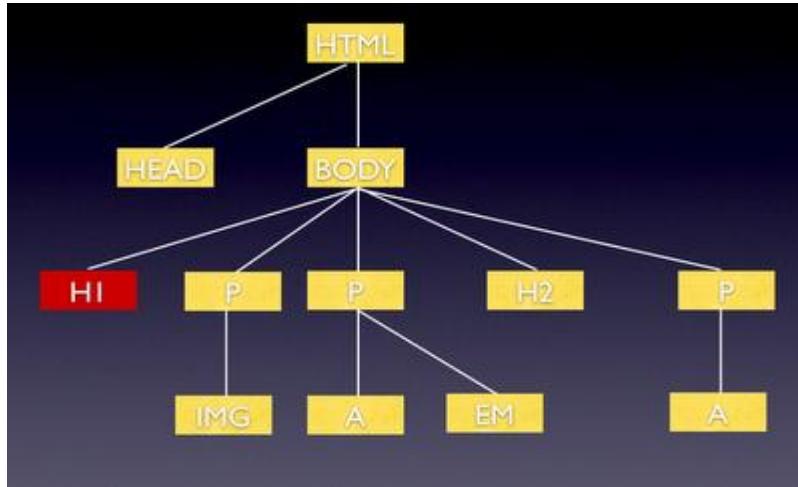
HERENCIA

Sin duda la propiedad más interesante de CSS es la herencia de reglas de estilos entre elementos. Podemos imaginarnos nuestros archivos HTML como un árbol invertido de elementos que se anidan los unos en los otros. Si quisieramos fijar la tipografía de nuestras s de primer nivel a sans-serif escribiríamos una regla como la siguiente:

```
h1 { font-family: sans-serif; }
```

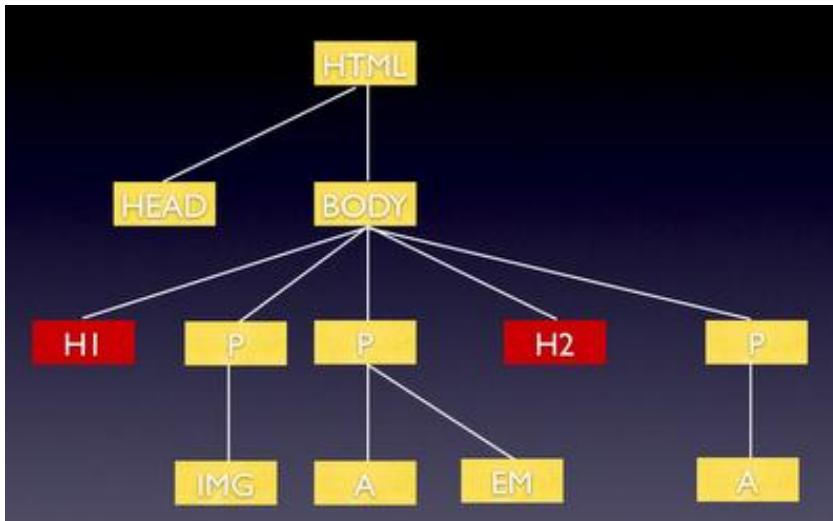
Con CSS, sólo podemos aplicar estilo a los elementos dentro del elemento body. Debido a esto los siguientes gráficos no muestran los elementos dentro de head. La regla anterior seleccionaría todos los elementos <h1> de la página. En este ejemplo sólo hay uno como se muestra en el siguiente gráfico.

4. HOJAS DE ESTILO



Si quisieramos cambiar la tipografía de las s de primer y segundo nivel escribiríamos la siguiente regla.
h1, h2 { font-family: sans-serif;}

Ahora el selector selecciona todos los elementos `<h1>` y `<h2>`



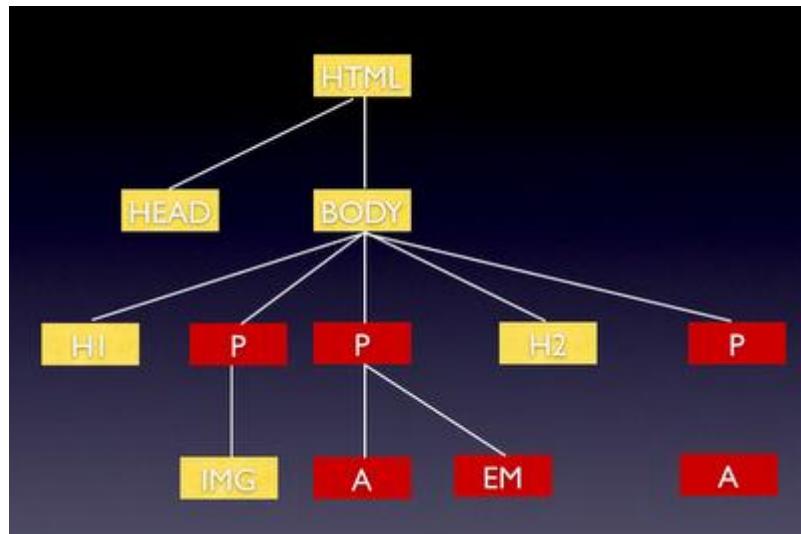
4. HOJAS DE ESTILO

Si quisieramos cambiar la tipografía de los párrafos escribiríamos la siguiente regla.

```
p { font-family: sans-serif;}
```

El selector se aplica a todos los elementos <p> del árbol. Más aún, los elementos anidados dentro de los párrafos como son <a>, y <a>heredan también la propiedad font-family de sus elementos padres.

El elemento es un hijo del párrafo, pero no tiene ningún texto por lo que no se ve afectado.



4. HOJAS DE ESTILO

Si quisieramos cambiar la tipografía de los párrafos escribiríamos la siguiente regla.

```
p { font-family: sans-serif;}
```

Usando la herencia de CSS, podemos simplificar las reglas de estilo que hemos usado en los ejemplos anteriores.

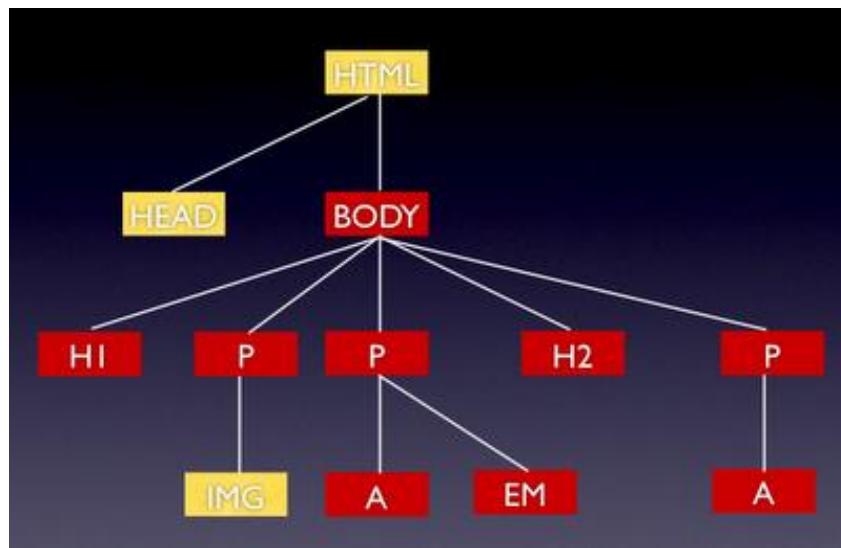
```
h1 { font-family: sans-serif;}
```

```
h2 { font-family: sans-serif;}
```

```
p { font-family: sans-serif;}
```

Podemos lograr el mismo efecto si movemos la propiedad font-family de los párrafos y cabeceras al elemento body. Ahora, todos los elementos anidados bajo body heredan la propiedad font-family. A continuación, se muestra el código y el gráfico que ilustra el cambio.

```
body { font-family: sans-serif;}
```



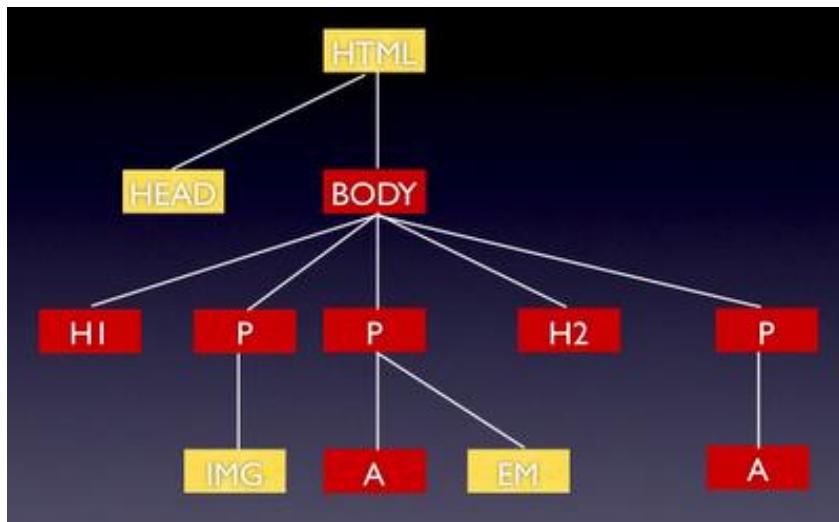
4. HOJAS DE ESTILO

¿CÓMO SOBREESCRIBIR UNA PROPIEDAD HEREDADA?

La propiedad font-family se define en la regla body, de modo que cada elemento dentro de body la hereda. Puedes sobreescribir el valor de font-family que has heredado. Si decides usar, por ejemplo, la tipografía serif para los elementos , el siguiente código lo lograría.

```
body { font-family: sans-serif; }  
em { font-family: serif; }
```

El siguiente gráfico muestra ahora los elementos con la tipografía fijada en sans-serif. Puedes ver que el elemento em ya no está afectado.



4. HOJAS DE ESTILO

Los selectores id y class

Además de establecer un estilo para un elemento HTML, CSS le permite especificar sus propios selectores denominados "id" y "class".

El Selector de Identificación

El selector de id se utiliza para especificar un estilo para un único elemento único. Utiliza el atributo ID del elemento HTML y se define con un "#".

El ejemplo siguiente se aplicará al elemento con id = "para1":

```
#para1 { text-align:center; color:red; }
```

El selector de clase

El selector de clase se utiliza para especificar un estilo para un grupo de elementos. Utiliza el atributo del HTML "class", y se define con un ".".

En el siguiente ejemplo, todos los elementos HTML con class = "center" serán centro alineado:

```
.center {text-align:center;}
```

También se puede especificar que elementos HTML específicos apliquen una clase.

En el siguiente ejemplo, todos los elementos p con class = "center" estarán centrados:

```
p.center {text-align:center;}
```

4. HOJAS DE ESTILO

Propiedades CSS más comunes

Efectos de fondo:

background-color: puede ser un valor HEX (como "# ff0000"); un valor RGB (como "rgb (255,0,0)"); ó un nombre de color (como "red")

background-image: se carga una imagen de fondo de la siguiente manera: url("ejemplo.gif");

background-repeat: sus valores posibles pueden ser repeat, repeat-x, repeat-y, no-repeat, initial, e inherit

background-attachment: establece si una imagen de fondo se fija o se desplaza con el resto de la página. Sus valores posibles son scroll (default), fixed, local, initial, inherit

background-position: establece la posición inicial de una imagen de fondo. Se puede establecer como left, right, o center, ó también se puede especificar su posición con el formato x% y% (por ejemplo, 20% 30%) ó como coordenadas xpos ypos.

Texto

text-align: Puede tomar valores center, right, justify

text-decoration: Puede tomar los valores none, underline, overline, line-through, initial, inherit

text-transform: Puede tomar los valores none, capitalize, uppercase, lowercase, initial, e inherit

text-indent: Especifica la sangría que tiene el elemento. Se puede especificar un valor o un porcentaje.

font-family: La propiedad font-family debe contener varios nombres de fuente como un sistema de "reserva". Si el navegador no es compatible con la primera fuente, que trata la siguiente fuente.

font-style: se utiliza para especificar si el texto se muestra en cursiva (italic).

font-size: Define el tamaño de la fuente. El tamaño de la unidad em es el recomendado por el W3C. El tamaño se puede calcular a partir de píxeles a em usando esta fórmula: pixeles / 16 = em.

Ejemplo

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
```

```
h2 {font-size:1.875em;} /* 30px/16=1.875em */
```

```
p {font-size:0.875em;} /* 14px/16=0.875em */
```

4. HOJAS DE ESTILO

Propiedades CSS más comunes

Tablas:

El aspecto de una tabla HTML se puede mejorar en gran medida con CSS.

Se pueden establecer los bordes:

```
table, th, td { border: 1px solid black;}
```

Se pueden establecer alto y ancho de la tabla:

```
table {width:100%;}  
th { height:50px;}
```

Se puede alinear el texto en una celda:

```
td { text-align:right; }  
td { height:50px; vertical-align:bottom; }
```

Se puede controlar el espacio entre el borde y el contenido de una tabla, utilizando la propiedad padding en td y elementos th:

```
td { padding:15px; }
```

Se puede especificar el color de los bordes, y el texto y el color de fondo de elementos th:

```
table, td, th { border:1px solid green; }  
th { background-color:green; color:white; }
```

4. HOJAS DE ESTILO

El modelo de caja CSS

Todos los elementos HTML pueden ser considerados como cajas. En CSS, el término "modelo de caja" se usa cuando se habla sobre el diseño y el diseño.

El modelo de caja CSS es esencialmente una caja que se envuelve alrededor de los elementos HTML, y se compone de: márgenes, bordes, el relleno y el contenido real.

El modelo de caja nos permite colocar un borde alrededor de elementos y elementos del espacio en relación con otros elementos.

La imagen siguiente muestra el modelo de caja:



4. HOJAS DE ESTILO

Estilo de borde:

border-style: puede tomar los valores none (Sin border), dotted (borde punteado), dashed (borde con guiones), solid (línea sólida), double (línea doble)

border-width: establece el ancho del borde.

border-color: establece el color del borde

Márgenes y relleno:

Ejemplos:

```
margin-top:100px;  
margin-bottom:100px;  
margin-right:50px;  
margin-left:50px;
```

```
padding-top:25px;  
padding-bottom:25px;  
padding-right:50px;  
padding-left:50px;
```

4. HOJAS DE ESTILO

Cómo ocultar un elemento - display: none o visibility: hidden

Se puede hacer estableciendo la propiedad display a "none" o la propiedad de visibilidad a "hidden". Sin embargo, note que estos dos métodos producen resultados diferentes:

visibility: hidden esconde un elemento, pero todavía tendrá el mismo espacio que antes. El elemento se ocultará, pero aún afecta al diseño.

Ejemplo

```
h1.hidden {visibility:hidden;}
```

display: none esconde un elemento, y no ocupará ningún espacio. El elemento se ocultará y la página se mostrará como si el elemento no está ahí:

Ejemplo

```
h1.hidden {display:none;}
```

4. HOJAS DE ESTILO

RESUMEN

- CSS contiene declaraciones simples, llamadas reglas.
- Una regla proporciona el estilo para una selección de elementos HTML.
- Una regla típica consiste en un selector junto con un o más propiedades y valores.
- El selector especifica a que elementos podemos aplicar la regla.
- Cada propiedad finaliza con un punto y coma.
- Todas las propiedades y valores van entre {}.
- Puede seleccionar cualquier elemento usando su nombre como selector.
- Separando mediante comas los nombres de elementos podemos seleccionar varios elementos a la vez.
- Una de las maneras más fácil de incluir estilo en HTML es mediante la etiqueta <style>
- Para portales Web de alguna complejidad, debería enlazar a una hoja de estilos externa.
- El elemento <link> se usa para incluir una hoja de estilos externa.
- Muchas propiedades se heredan. Por ejemplo, si una propiedad que se hereda se define para el elemento <body>, todos los elementos hijos de <body> la heredarían.
- Siempre puede sobrescribir propiedades heredadas creando una regla más específica para el elemento que queremos cambiar.

FIN
Gracias

CODES

Capacitación Web – Bootstrap

ÍNDICE

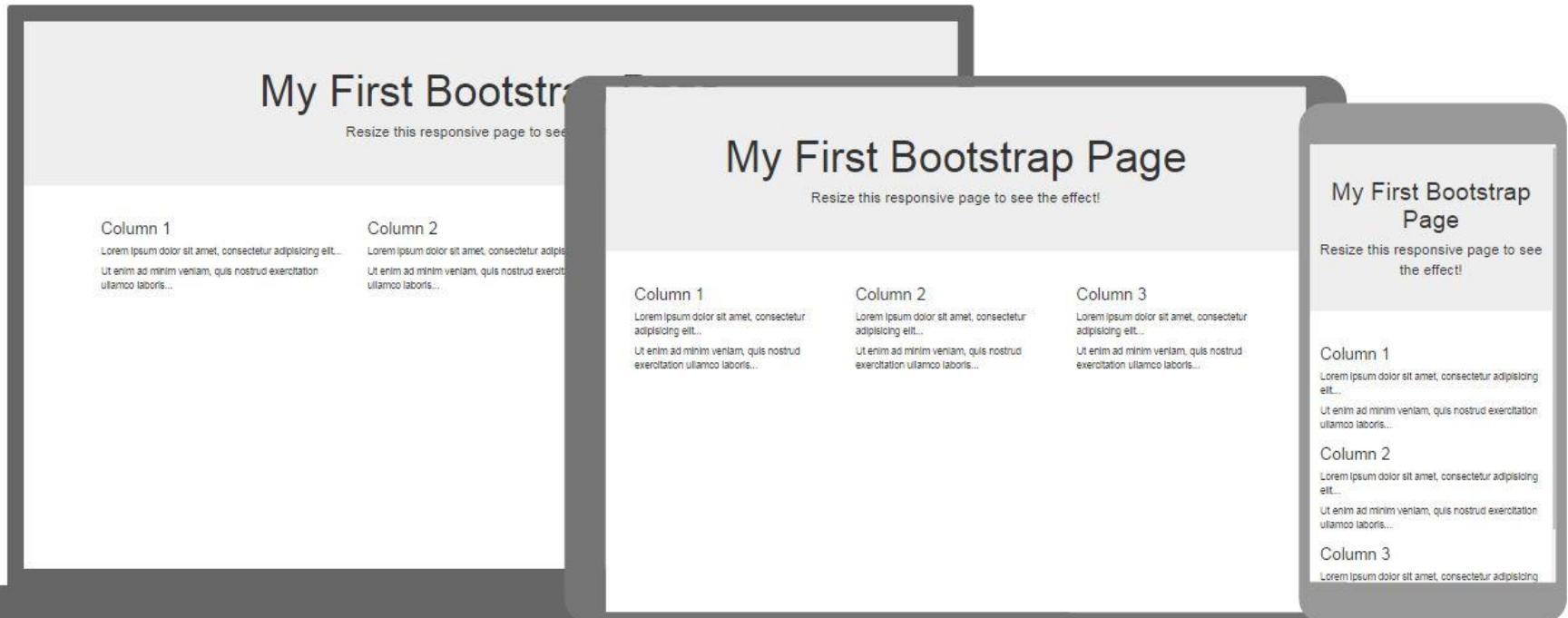
1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULAS O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

1. QUE ES BOOTSTRAP

- Es un **conjunto de librerías JavaScript y archivos de estilos (CSS)** que **colaboran con el desarrollo de aplicaciones Web** para se adapten automáticamente al dispositivo desde donde que se están accediendo.
- Bootstrap es de código abierto, y se puede descargar desde <http://getbootstrap.com/>



1. QUE ES BOOTSTRAP

- Para incluir el uso de las librerías en nuestro proyecto, incluir las mismas en la página:

```
<script src="@Url.Content("~/Scripts/jquery-1.11.3.min.js")"  
type="text/javascript"></script>
```

```
<script  
src="@Url.Content("~/Scripts/bootstrap/bootstrap.min.js")"  
type="text/javascript"></script>
```

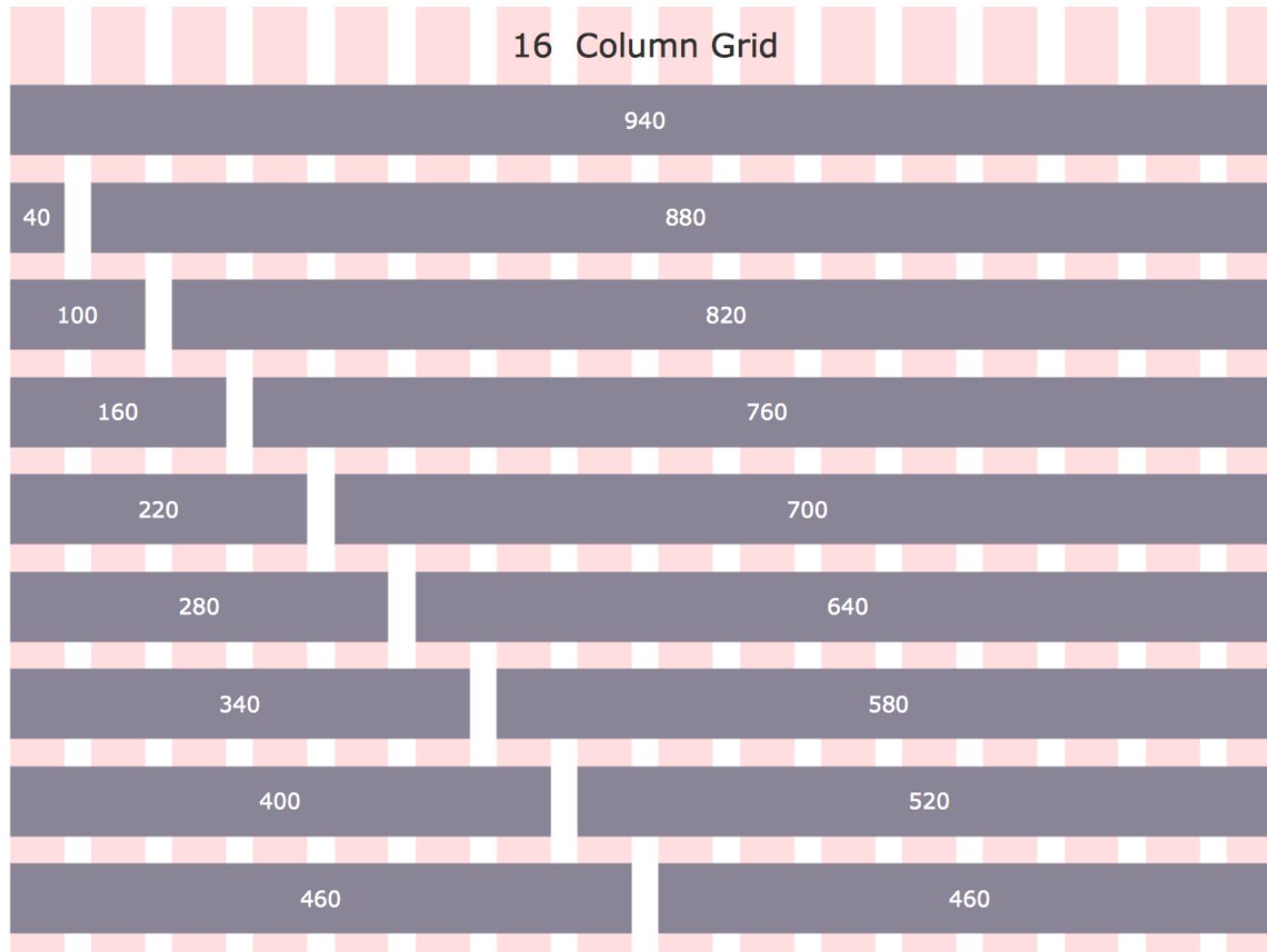
```
<link  
href="@Url.Content("~/Content/bootstrap/css/bootstrap.min.css")"  
rel="stylesheet" type="text/css" />  
<link href="@Url.Content("~/Content/bootstrap/css/bootstrap-  
theme.css")" rel="stylesheet" type="text/css" />
```

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS

- Es una estrategia para maquetar nuestras páginas Web mediante la definición de estilos.



2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS

Refresh Boston

another refreshing city

What is Refresh?

REFRESH IS A COMMUNITY OF DESIGNERS AND DEVELOPERS WORKING TO REFRESH THE CREATIVITY, TECHNOLOGY, & PROFESSIONAL SCENE OF NEW ENGLAND. IN THE BOSTON AREA, WHILE PROMOTING DESIGN, TECHNOLOGY, QUALITY, AND STANDARDS.

Refresh News

Happy Anniversary! Refresh Boston is officially 4 years old, and to celebrate we're having a bigger-than-usual event in the usual place, Microsoft NERD Center. We've got tons of goodies in store, and we'll announce everything soon, so stay tuned.

Our Next Event

Our speaker this time around is [Fred LeBlanc](#), a web developer and co-organizer of the [Salon Build Guild](#), and he will present [Development for the New Mobile Web](#).

What: Development for the New Mobile Web
When: Tuesday, April 2nd at 6pm PT
Where: Microsoft NERD Center
Who: [Fred LeBlanc](#)

[RSVP](#) on the [Upcoming event page](#)

Development for the New Mobile Web

Everyone wants an iPhone app or a mobile website these days, but it's harder than they think. Learn the ins and outs, tips and tricks, on developing for the mobile web at a time when smart phones, iPhone's and small, pocketable devices are becoming the norm for getting online to browse the web.

About Fred

Fred is a web developer, writer, photographer, beer-drinker, husband, dog-dad & more. He also recently built a mobile website called [I Know Fred!](#), which uses location-based services to determine how far you are from Fred at the moment (use it on a mobile phone for best results). He also created [Budabout](#), a jQuery plugin that enhances regular HTML forms with location-based data.

Refresh Boston Events

Keep up-to-date with Refresh Boston and collaborate with other fellow Bostonians through our multiple channels, including:

- Frequent updates from Twitter
- Photos of events on Flickr
- Our event schedule on Upcoming
- Share news, photos, video & more on Mixx
- Provide speaker feedback with SpeakerRate
- Check us out on Facebook
- View presentation slides on Slideshare

STAY IN THE LOOP

Sign up for our occasional newsletter with news about Refresh meetings and other Boston happenings.

Your name, so we know who you are

Your email address, so we can contact you

Prove that you are a human

Sign me up

Note: Our newsletter is provided by Campaign Monitor, which makes it easy for us to send email campaigns and statistics for you to add or remove yourself from our mailing list.

Other Local Groups

There are plenty of other great web & tech communities in the Boston area, including:

Microsoft NERD Center

The NERD Center is Microsoft's research & innovation campus in Cambridge, and hosts many groups of events, including Refresh Boston.

Freshview

Freshview's Campaign Monitor makes it easy to send out beautiful, effective email campaigns and track the results as you go.

Boston Web Studio

Refresh Boston is organized by [Boston Web Studio](#), a little design studio that specializes in attractive, accessible interfaces for websites & mobile.

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA

- El sistema de grilla de Bootstrap está dividido en 12 columnas. Todo el HTML dentro de la maqueta debe estar dentro de un DIV con el estilo «container».

span 1															
span 4				span 4				span 4							
span 4				span 8											
span 6						span 6									
span 12															

- Contiene 4 clases principales:
 - xs (para teléfonos)
 - sm (para tabletas)
 - md (para PC escritorio)
 - lg (para pantallas más grandes)

3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA

- Estructura básica del sistema de cuadrícula

```
<div class="container">
  <div class="row">    fila
    <div class="col-*-*"></div>    columna
  </div>
  <div class="row">
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
    <div class="col-*-*"></div>
  </div>
  <div class="row">
    ...
  </div>
</div>
```

3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA

- Fila de elementos con tres columnas de igual tamaño

.col-sm-4

.col-sm-4

.col-sm-4

```
<div class="container">
    <div class="row">
        <div class="col-sm-4">.col-sm-4</div>
        <div class="col-sm-4">.col-sm-4</div>
        <div class="col-sm-4">.col-sm-4</div>
    </div>
</div>
```

3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA

- Fila de elementos con dos columnas de distinto tamaño

.col-sm-4

.col-sm-8

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

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

4. TABLAS EN BOOTSTRAP

- Bootstrap tiene estilos definidos para acomodar tablas.
 - La clase **.table** agrega el estilo básico a una tabla
 - La clase **.table-bordered** agrega bordes tanto a las celdas como al contorno externo de la tabla.
 - La clase **.table-hover** agrega un efecto "hover" en cada fila de la tabla.

Nombre	Apellido	Correo Electrónico
Default	Defaultson	def@somemail.com
Success	Gustavo	gustavo@example.com
Danger	Juan	juan@example.com
Warning	Pepe	pepe@example.com

```
<table class="table">
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Apellido</th>
      <th>Correo Electrónico</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Default</td>
      <td>Defaultson</td>
      <td>def@somemail.com</td>
    </tr>
    <tr class="success">
      <td>Success</td>
      <td>Gustavo</td>
      <td>gustavo@example.com</td>
    </tr>
    <tr class="danger">
      <td>Danger</td>
      <td>Juan</td>
      <td>juan@example.com</td>
    </tr>
    <tr class="warning">
      <td>Warning</td>
      <td>Pepe</td>
      <td>pepe@example.com</td>
    </tr>
  </tbody>
</table>
```

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

5. BOTONES CON ESTILOS

- Bootstrap incluye estilos para los botones, los cuales pueden ser utilizados tanto para los tags INPUT, A (anchor Hiperlink), y BUTTON. Existen 7 estilos definidos:

Default

Primary

Success

Info

Warning

Danger

Link

Las clases definidas para los botones son:

- .btn-default
- .btn-primary
- .btn-success
- .btn-info
- .btn-warning
- .btn-danger
- .btn-link

Ejemplos:

```
<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>
```

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

6. PANELES

Muchas veces querremos agrupar el contenido de nuestra página utilizando para ello algún tipo de marco. En Bootstrap, existen los estilos para definir «paneles» que contengan información.

Perfil Lote

Volver Grabar

Nombre Lote
06217-B10

Descripción
06217-B10

Campo
CHASCOMÚS



Capas estáticas

Filtro
[Agregar SHP](#)

Nombre	
06217-B10	PERIMETRO

Campañas asociadas

[Crear Campaña]

Amb. Rinde Mapa Rasters (1)

(2017) 17-18

6. PANELES

En Bootstrap, un panel es un DIV con un borde y un padding, con la opción de agregar una cabecera y un pie en el mismo. Además incluye estilos contextuales por si necesitamos sobresalir al contenido del panel.

Las clases básicas para definir el panel son:

- panel
- panel-default
- panel-heading
- panel-body
- panel-footer

Ejemplo:

```
<div class="panel panel-default">
  <div class="panel-heading">Panel Heading</div>
  <div class="panel-body">Panel Content</div>
  <div class="panel-footer">Panel Footer</div>
</div>
```

Panel Heading

Panel Content

Panel Footer

6. PANELES

Las **clases** que se utilizan para colorizar el panel son: **.panel-default**, **.panel-primary**, **.panel-success**, **.panel-info**, **.panel-warning**, o **.panel-danger**

Panel with panel-default class

Panel Content

Panel with panel-primary class

Panel Content

Panel with panel-success class

Panel Content

Panel with panel-info class

Panel Content

Panel with panel-warning class

Panel Content

Panel with panel-danger class

Panel Content

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

7. ESTILOS EN FORMULARIOS

Bootstrap tiene estilos para tres tipos de formularios:

- Estilo en vertical (que es el predeterminado).
- Estilo en horizontal
- Estilo en línea.

Para utilizar estos estilos, cada control dentro del formulario tiene que estar envuelto dentro de un `<div class="form-group">`

Además, los tags de ingreso de datos deben tener la clase `.form-control` (`<input>`, `<textarea>`, y `<select>`).

De manera predeterminada, estos controles de entrada (`<input>`, `<textarea>`, y `<select>`) tendrán un ancho del 100% del contenedor.

7. ESTILOS EN FORMULARIOS

Ejemplo formulario vertical en Bootstrap

```
<form>
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control" id="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd">
  </div>
  <div class="checkbox">
    <label><input type="checkbox" /> Remember me</label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

The screenshot displays a vertical Bootstrap form. It consists of two stacked `div.form-group` elements. The first group contains a label "Email address:" and an `input` field with type="email" and ID="email". The second group contains a label "Password:" and an `input` field with type="password" and ID="pwd". Below these groups is a `div.checkbox` containing a label with an `input` field of type="checkbox" and the text "Remember me". At the bottom is a `button` with type="submit" and class="btn btn-default" labeled "Submit".

7. ESTILOS EN FORMULARIOS

Ejemplo formulario horizontal en Bootstrap

```
<form class="form-horizontal">
  <div class="form-group">
    <label class="control-label col-sm-2" for="email">Email:</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="email" placeholder="Enter email">
    </div>
  </div>
  <div class="form-group">
    <label class="control-label col-sm-2" for="pwd">Password:</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="pwd" placeholder="Enter password">
    </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">Submit</button>
    </div>
  </div>
</form>
```

The screenshot shows a horizontal form with the following structure:

- Email:** Input field with placeholder "Enter email".
- Password:** Input field with placeholder "Enter password".
- A checkbox labeled "Remember me".
- A "Submit" button.

7. ESTILOS EN FORMULARIOS

Ejemplo formulario en línea en Bootstrap

```
<form class="form-inline">
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control" id="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" id="pwd">
  </div>
  <div class="checkbox">
    <label><input type="checkbox"> Remember me</label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Email:

Password:

Remember me

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

8. INPUTS

- Bootstrap tiene **clases de estilos** definidos para los siguientes **controles de entrada**:
- **INPUT, TEXTAREA, CHECKBOX, RADIO, y SELECT**

Name:

Password:

1
2
3
4
-

Comment:

- Option 1
- Option 2
- Option 3

- Option 1
- Option 2
- Option 3

Option 1 Option 2 Option 3

Option 1 Option 2 Option 3

Name:

Este es un ejemplo

Comment:

Password:

.....|

```
<div class="form-group">
  <label for="usr">Name:</label>
  <input type="text" class="form-control" id="usr">
</div>
<div class="form-group">
  <label for="pwd">Password:</label>
  <input type="password" class="form-control" id="pwd">
</div>
<div class="form-group">
  <label for="comment">Comment:</label>
  <textarea class="form-
control" rows="5" id="comment"></textarea>
</div>
```

8. INPUTS

Option 1

Option 2

Option 3

Option 1 Option 2 Option 3

```
<div class="checkbox">
  <label><input type="checkbox" value="">Option 1</label>
</div>
<div class="checkbox">
  <label><input type="checkbox" value="">Option 2</label>
</div>
<div class="checkbox disabled">
  <label><input type="checkbox" value="" disabled>Option 3</label>
</div>

<label class="checkbox-inline"><input type="checkbox" value="">Option 1</label>
<label class="checkbox-inline"><input type="checkbox" value="">Option 2</label>
<label class="checkbox-inline"><input type="checkbox" value="">Option 3</label>
```

8. INPUTS

- Option 1
- Option 2
- Option 3

- Option 1
- Option 2
- Option 3

```
<div class="radio">
  <label><input type="radio" name="optradio">Option 1</label>
</div>
<div class="radio">
  <label><input type="radio" name="optradio">Option 2</label>
</div>
<div class="radio disabled">
  <label><input type="radio" name="optradio" disabled>Option 3</label>
</div>
|
<label class="radio-inline"><input type="radio" name="optradio">Option 1</label>
<label class="radio-inline"><input type="radio" name="optradio">Option 2</label>
<label class="radio-inline"><input type="radio" name="optradio">Option 3</label>
```

8. INPUTS

A dropdown menu with a white background and a thin gray border. It contains four items: '1', '2', '3', and '4'. The item '1' is highlighted with a light blue background. A vertical scroll bar is visible on the right side of the menu.

```
<div class="form-group">
  <label for="sel1">Select list:</label>
  <select class="form-control" id="sel1">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
  </select>
</div>
```

Para el caso de SELECT de selección múltiple, usar
`<select multiple class="form-control">`

ÍNDICE

1. QUE ES BOOTSTRAP
2. QUÉ ES UN SISTEMA DE CUADRÍCULA O DE GRILLA EN CSS
3. BOOTSTRAP Y SU SISTEMA DE CUADRÍCULA
4. TABLAS EN BOOTSTRAP
5. BOTONES CON ESTILOS
6. PANELES
7. ESTILOS EN FORMULARIOS
8. INPUTS
9. PANELES MODALES

9. PANELES MODALES

-En Bootstrap, el Modal es un cuadro de diálogo o ventana emergente que aparece en la parte superior de la página actual.



9. PANELES MODALES

```
<!-- Abre el modal con un botón -->
<button type="button" class="btn btn-info btn-lg" data-toggle="modal" data-
target="#miModal">Abre Modal</button>
<!-- Modal -->
<div id="miModal" class="modal fade" role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">&times;</button>
        <h4 class="modal-title">Cabecera MODAL</h4>
      </div>
      <div class="modal-body">
        <p>Cuerpo en el modal.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-
dismiss="modal">Cerrar</button>
      </div>
    </div>
  </div>
</div>
```

En el ejemplo anterior, el cuadro de diálogo se abre a partir del botón "Abre Modal". Éste incluye dos atributos necesarios:

- `data-toggle="modal"` Abre la ventana modal
- `data-target="#miModal"` apunta al ID del modal (DIV)

•La parte "Modal":

La clase `.modal` identifica al DIV como cuadro modal.

La clase `.fade` agrega un efecto cuando aparece/desaparece el modal.

La clase `.modal-dialog` define los estilos del modal.

•La parte "Modal content":

El `<div>` con `class="modal-content"` define los estilos del modal (border, background-color, etc.). Dentro de este `<div>`, se agregan la cabecera, cuerpo y pie.

La clase `.modal-header` define el estilo para la cabecera del modal.

El `<button>` dentro de la cabecera tiene un atributo `data-dismiss="modal"`, El cual cierra el modal.

La clase `.modal-body` define el estilo del cuerpo del modal.

La clase `.modal-footer` define el estilo del pie del modal.

9. PANELES MODALES

En algunos casos, vamos a necesitar manipular el panel modal mediante programación Javascript.

En el ejemplo anterior, para abrir el panel modal ejecutamos

```
$("#miModal").modal()
```

Para ocultar el modal nuevamente:

```
$("#miModal").modal("hide")
```

9. PANELES MODALES

Además, el panel Modal en Bootstrap tiene algunas opciones más que se pueden utilizar:

- **backdrop**: define cómo visualizar el fondo de la ventana. Los valores posibles son true, false, o la palabra «static».
- **show**: indica si el modal se abre una vez inicializada la página. Los valores posibles son true, o false.
- **Keyboard**: indica si se puede cerrar el Modal con la tecla ESC. Los valores posibles son true o false.

Ejemplo:

```
<button class="btn btn-info btn-md" id="btn1">Modal con Overlay</button>
<button class="btn btn-info btn-md" id="btn2">Modal sin Overlay</button>
<button class="btn btn-info btn-md" id="btn3">Modal con backdrop:"static"</button>

<script>
$(document).ready(function(){
    $("#btn1").click(function(){ $("#miModal").modal({backdrop: true}); });
    $("#btn2").click(function(){ $("#miModal").modal({backdrop: false}); });
    $("#btn3").click(function(){ $("#miModal").modal({backdrop: "static"}); });
});
</script>
```

FIN
¿Dudas?
¿Preguntas?

Table of Contents

Contenidos	1.1
Introducción	1.2
Frameworks responsive	1.2.1
Funcionamiento del diseño adaptable	1.2.2
Probando el responsive	1.2.3
Página básica	1.3
Sistema de rejilla	1.4
Columnas de ancho específico	1.4.1
Columnas de ancho automático	1.4.2
Forzar el cambio de fila	1.4.3
Anidamiento de columnas	1.4.4
Márgenes o espaciado entre columnas	1.4.5
Ordenación de columnas	1.4.6
Alineación	1.4.7
Utilidades Responsive	1.5
Media Queries	1.6
Componentes Responsive	1.7
Botones	1.7.1
Desplegables	1.7.2
Grupos de botones	1.7.3
Formularios	1.7.4
Navegación	1.7.5
Barra de navegación	1.7.6
Tablas	1.7.7
Ejercicios 1	1.8
Ejercicios 2	1.9
Bibliografía	1.10

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
 - Frameworks responsive
 - Funcionamiento del diseño adaptable
 - Probando el responsive
- Página básica
- Sistema de rejilla
 - Forzar el cambio de fila
 - Anidamiento de columnas
 - Márgenes o espaciado entre columnas
 - Ordenación de columnas
- Utilidades responsive
- Media Queries
- Componentes Responsive
 - Botones
 - Desplegables
 - Grupos de botones
 - Formularios
 - Navegación
 - Barra de navegación
 - Tablas
- Ejercicios
- Bibliografía

Introducción al diseño "responsive"

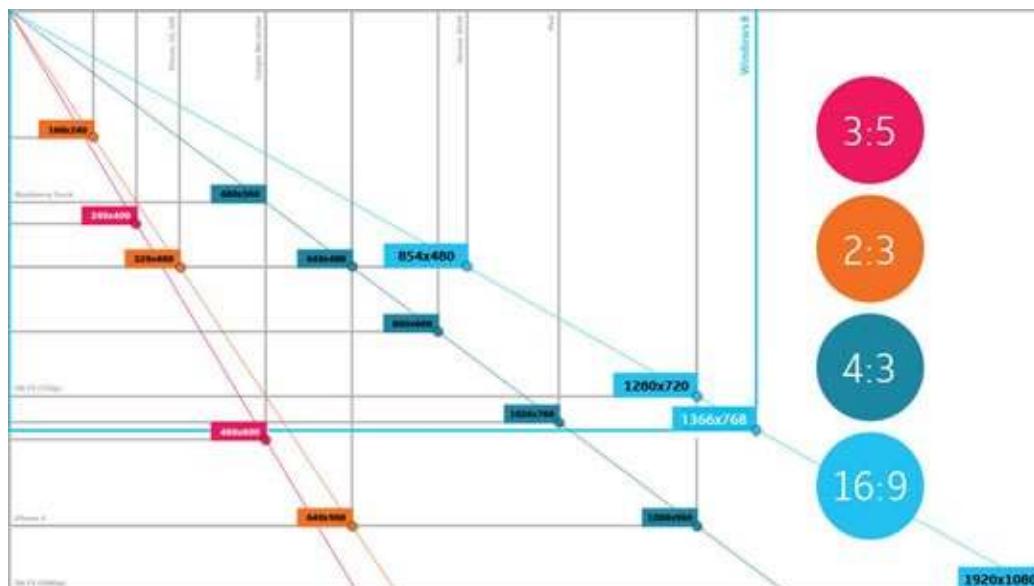
El **diseño web responsive**, adaptable o adaptativo, conocido por las siglas *RWD* (del inglés, *Responsive Web Design*) es una filosofía de diseño y desarrollo cuyo **objetivo** es **adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla**. Hoy día las páginas 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%
800x600	0.5%
< 800x600	0.5%

En la actualidad ya no es 1024x768 la resolución más utilizada, sino que es 1366x768 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.

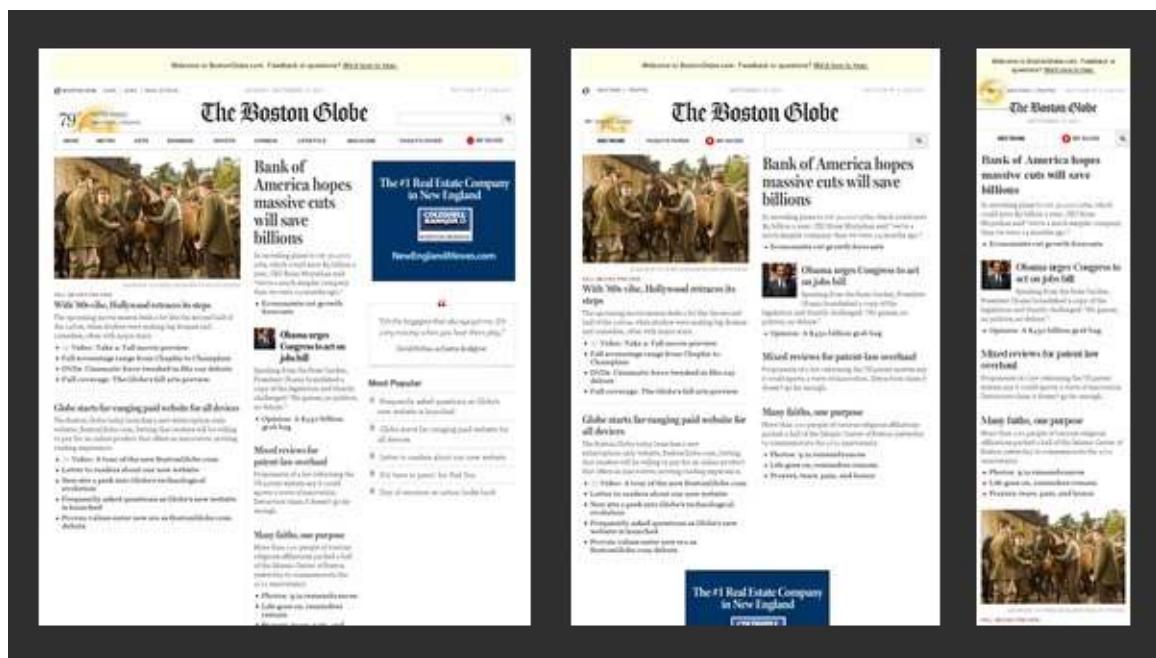
Ejemplos de sitios web creados con tecnología *Responsive*

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:

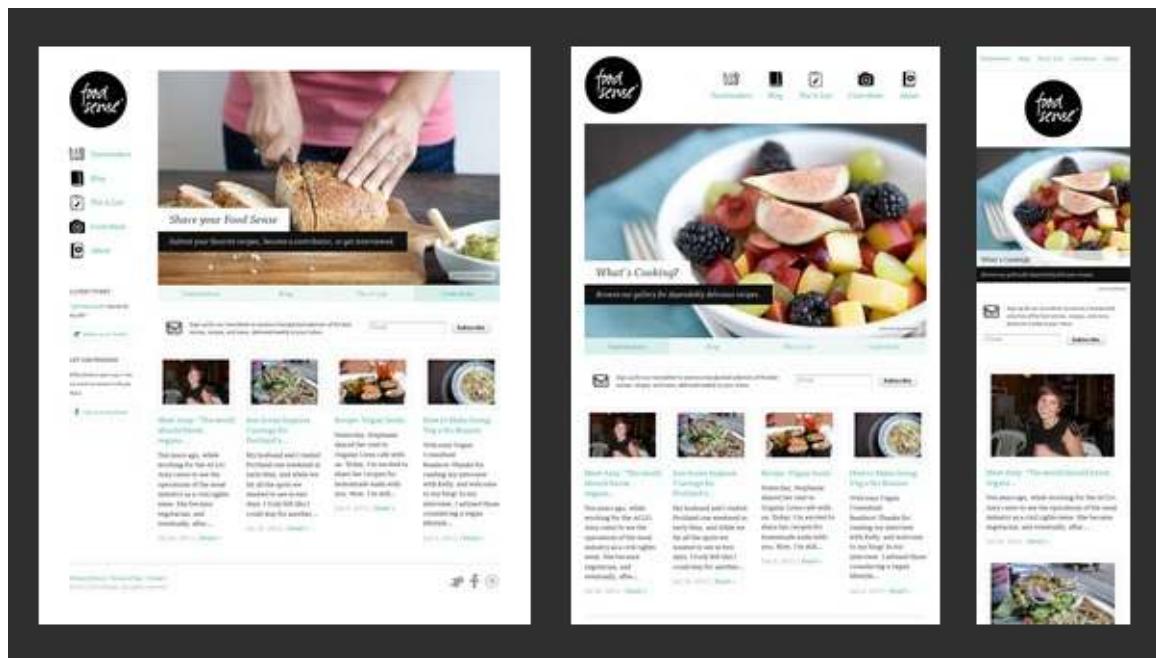
dConstruct 2011



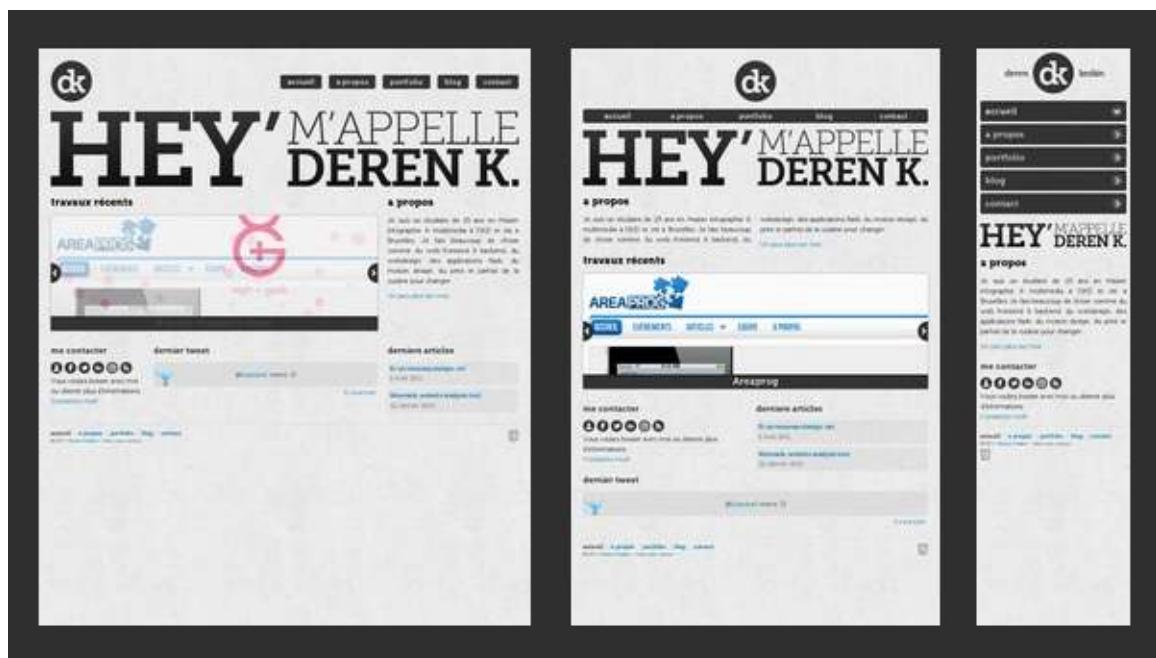
Boston Globe



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).

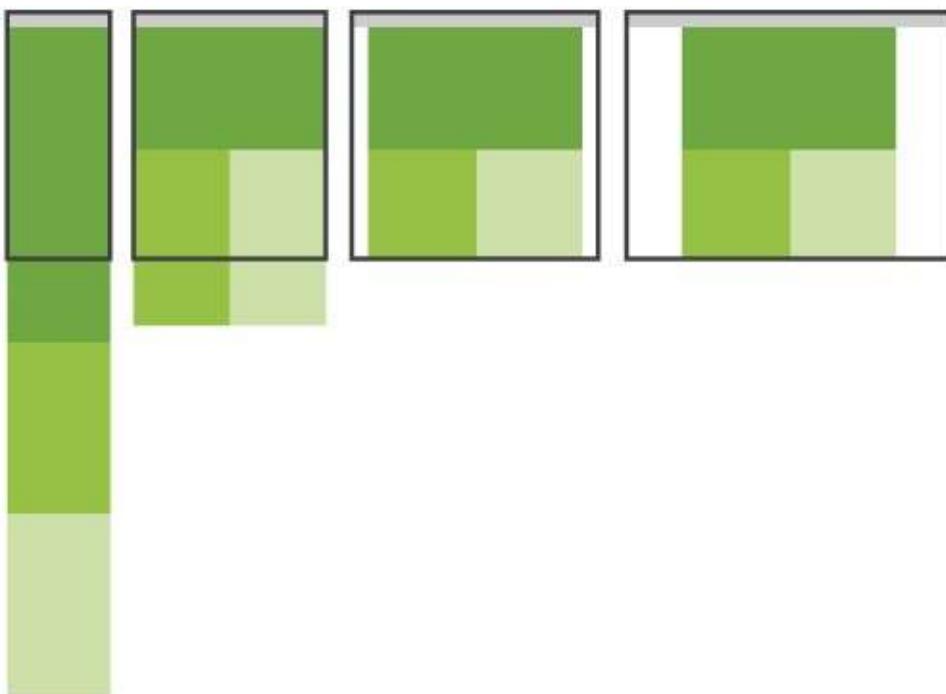
Está preparado para funcionar tanto en navegadores de PCs y portátiles con cualquier tamaño de pantalla así como para *tablets* y *smartphones* de tamaños mucho más reducidos.

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 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;
}
```

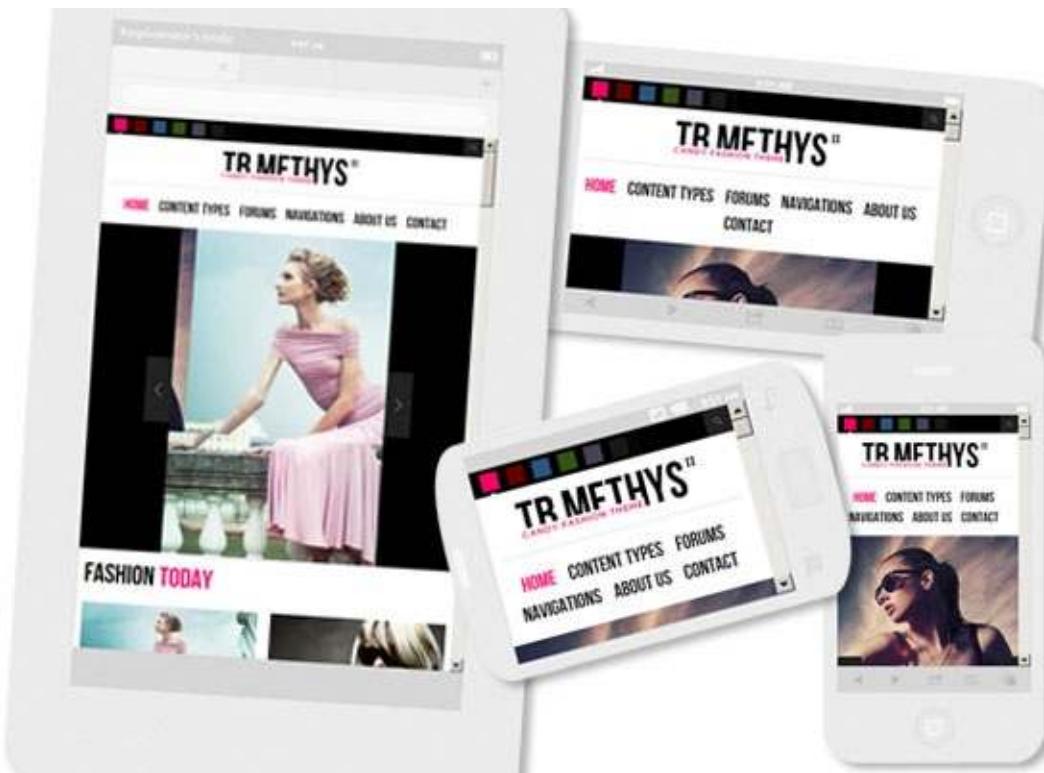
Para más información podéis consultar: <http://www.w3.org/TR/css3-mediaqueries/>

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>)

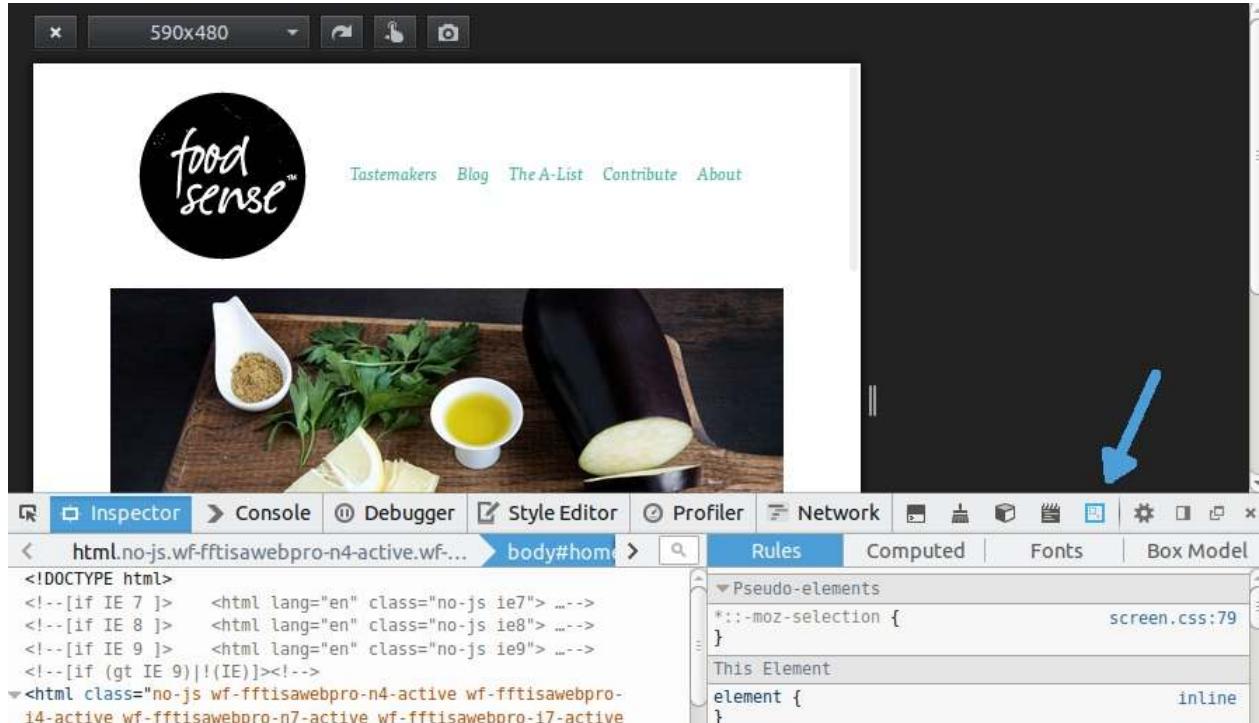
Esta herramienta está disponible solamente de forma *online*, pero nos permite ver de un solo vistazo como se mostraría nuestra web con el tamaño de los *smartphones* y *tablets* más populares, como por ejemplo las diferentes versiones de iPhone, iPad, Kindle y algunas versiones de teléfonos Android.



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, shrink-to-fit=no">
```

La propiedad `width` controla el tamaño del viewport. Puede definirse con un número en pixeles como `width=600` o con un valor especial como `device-width` que es el equivalente al ancho de la pantalla en pixeles del dispositivo que cargue la web. El atributo `shrink-to-fit="no"` configura este mismo comportamiento para los navegadores Safari anteriores a la versión 9.

La propiedad `initial-scale` del viewport controla el nivel de zoom cuando la página se carga por primera vez. Las propiedades `maximum-scale`, `minimum-scale`, y `user-scalable` controlan la forma en cómo se permite a los usuarios aumentar o disminuir el zoom en la página. Si añadimos a la etiqueta `meta` del `viewport` el atributo `user-scalable=no` (como se puede ver en el ejemplo inferior) podemos deshabilitar el zoom para dispositivos móviles. 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, shrink-to-fit=no",
      maximum-scale=1, user-scalable=no">
```

A continuación se incluye la 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>
    <title>Hello, world!</title>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=
no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css" integrity="sha384-PSh8R72JQ3S0dhVi3uxftmaW6Vc51MKb0q5P2rRUUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIKvYIK3UENzm7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5Kkn" crossorigin="anonymou
s"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.3/dist/umd/popper.m
in.js" integrity="sha384-vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwickWPJf9c9IpF
gh" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.m
in.js" integrity="sha384-alpBpkh1PF0epccYYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNKCK9SaVuEZ
flJ" crossorigin="anonymous"></script>
  </body>
</html>
```

Como se puede ver hemos añadido la etiqueta meta de cabecera y hemos cargado la hoja de estilo de Bootstrap y las librerías Javascript que vamos a necesitar (JQuery, Bootstrap y Popper). También hemos añadido la etiqueta h1 con "Hello world!" dentro del cuerpo de la web, que será donde podremos empezar a escribir el contenido de nuestro sitio web responsive.

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**.

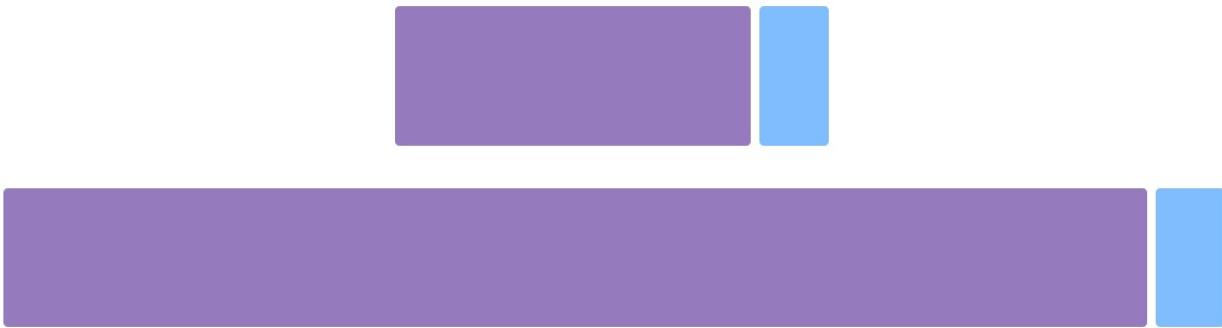
Si lo que queremos es que el contenido de nuestra web aparezca centrado y con un ancho fijo entonces podemos utilizar la etiqueta `.container`, de la forma:

```
<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>
```

En las siguientes imágenes se ejemplifica el resultado obtenido para un mismo ancho al aplicar los dos tipos de contenedores, `container` en el primer caso y `container-fluid` en el segundo. El comportamiento de estos elementos ante distintos tamaños de pantalla es el siguiente: el elemento "`container-fluid`" siempre se adapta al 100% del tamaño de la pantalla, mientras que el tipo "`container`" tiene un tamaño máximo, por lo que si el ancho de la pantalla es superior a este ancho el contenido aparecerá centrado, dejando un margen a cada lado, y si el ancho de la pantalla es igual o inferior al tamaño máximo del contenedor, entonces se adaptará al ancho disponible.



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	Dimensiones	Prefijo de la clase	Ancho del contenedor
Tamaño extra pequeño	< 576 px	.col-	Ninguno (automático)
Tamaño pequeño	≥ 576 px	.col-sm-	540px
Tamaño medio	≥ 768 px	.col-md-	720px
Tamaño grande	≥ 992 px	.col-lg-	960px
Tamaño extra grande	≥ 1200 px	.col-xl-	1140px

Es importante destacar que 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). Es decir, nos tenemos que fijar que en la tabla anterior el tamaño se indica con el símbolo de mayor o igual (\geq) (o de menor para el caso de xs) a un tamaño dado, y por lo tanto se aplicará esa disposición a partir de ese tamaño, a no ser que se indique otra cosa.

Bootstrap está diseñado pensando en los dispositivos móviles primero (o como ellos indican: siguiendo la estrategia mobile first). Por lo tanto todos los tamaños y dimensiones están pensadas para los dispositivos móviles, y para tamaños más grandes lo que hacen es adaptar o escalar estos tamaños.

Si nos fijamos en la tabla anterior podremos ver que para el tamaño extra pequeño el prefijo de la clase que se define es ".col-" (a diferencia de los demás que añaden un sufijo para el tamaño de pantalla). Cuando indiquemos el tamaño de las columnas usando esta clase se aplicará para todos los tamaños, a no ser, como ya hemos dicho, que se indique otra clase para otro tamaño mayor que defina otra disposición.

A continuación veremos diferentes formas de indicar el número de columnas que conforman cada fila, usando el sistema automático, especificando el ancho o bien usando un sistema mixto.

Columnas de ancho específico

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 medianas

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

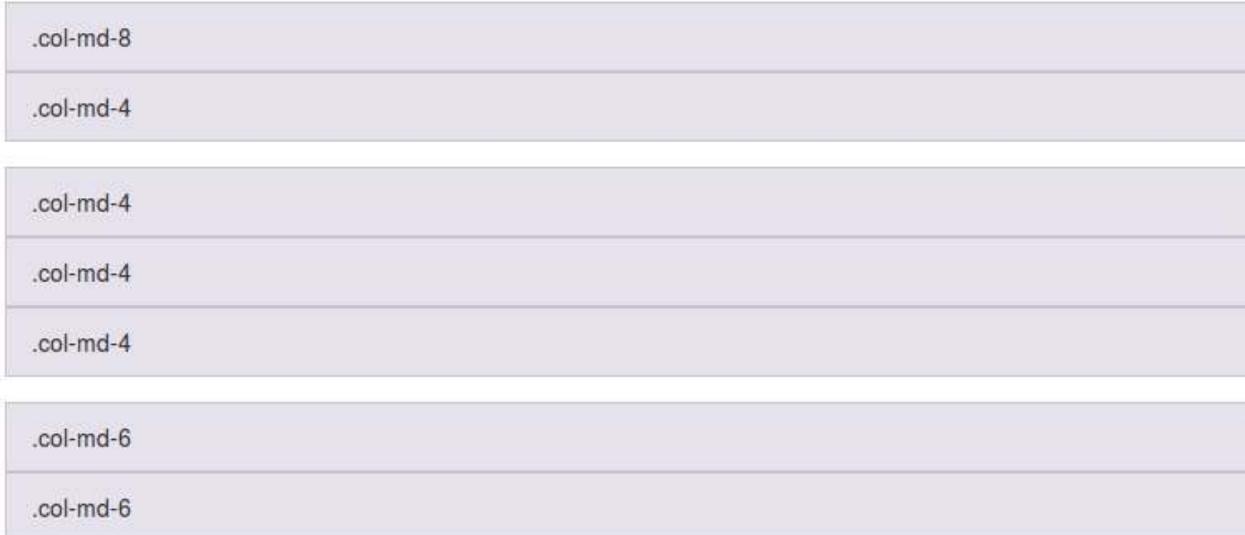
```
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6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

En la siguiente imagen se puede ver el resultado para pantallas de tamaño mediano (tamaños de md en adelante):

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

Para poder visualizar las columnas se ha añadido una clase CSS que establece color para el borde y el fondo de las cajas. Por defecto, ni la etiqueta DIV ni las etiquetas .col-* establecen apariencia (ni color de borde ni de fondo), solamente establecen la anchura, y por lo tanto al renderizar el código anterior tal cual las cajas se verán transparentes.

Dado que las columnas se han especificado únicamente mediante las clases `.col-md-*` esto creará estas divisiones solo para las pantallas medianas y grandes, pero no para los tamaños de pantalla más pequeños. En este último caso las columnas se ampliarán para ocupar todo el ancho y por lo tanto se mostrarán apiladas de la forma:



Selección de dos tamaños de columna: pequeño y mediano

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-*` además de las que ya teníamos con `.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-12 col-md-8">.col-12 .col-md-8</div>  
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>  
</div>  
  
<!-- En pantallas medianas se indica que cada columna ocupe la mitad  
del ancho disponible -->  
<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>  
  
<!-- Como no se indica el tamaño para pantallas grandes las columnas  
siempre ocuparán el 50% -->  
<div class="row">  
  <div class="col-6">.col-6</div>  
  <div class="col-6">.col-6</div>  
</div>
```

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

.col-12 .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	

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

.col-12 .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

Selección de tres tamaños: extra pequeño, pequeño y mediano

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

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

A continuación se incluye una previsualización de este código de ejemplo para pantallas medianas y grandes (tamaños md, lg y xl):

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

El mismo código pero en pantallas tipo *small* (tamaño sm) se mostraría como:

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

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

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

Además de los tres tamaños indicados en este último ejemplo para la primera columna (`.col-12 .col-sm-6 .col-md-8`) podríamos añadir también, si lo necesitamos, el tamaño para pantallas grandes y extra grandes con `col-lg` y `col-xl`. Por ejemplo, podríamos haber definido la siguiente columna:

```
<div class="col-12 col-sm-6 col-md-8 col-lg-9 col-xl-10">...</div>
```

En resumen, podemos indicar **para cada columna** todos los tamaños que queramos de entre los disponibles (con `.col-`, `.col-sm-`, `.col-md-`, `.col-lg-` y `.col-xl-`). Sin embargo, **esto solo lo tendremos que hacer** cuando necesitemos establecer un ancho de columna distinto para cada tamaño de pantalla. Si para todos los tamaños de pantalla necesitamos el mismo ancho entonces utilizaremos solamente la clase `.col-`. Es decir, no tendría sentido escribir algo como "col-6 col-sm-6 col-md-6 col-lg-6 col-xl-6", ya que se conseguiría el mismo efecto que si hubieramos puesto solamente "col-6".

Por lo tanto, solo añadiremos más de una clase cuando necesitemos establecer anchos distintos entre esos tamaños, y además si solo vamos a diferenciar entre 2 tamaños solo será necesario usar 2 etiquetas. Por ejemplo si queremos diferenciar solamente entre móvil y escritorio solamente tendríamos que añadir la clase `.col-` y la clase `.col-md-`.

Columnas de ancho automático

A partir de la versión 4 de Bootstrap podemos utilizar las columnas de ancho automático, es decir, indicar únicamente el número de columnas que queremos y el sistema calculará automáticamente su anchura. Para esto podremos usar la clase "`.col`", sin número de columnas ni tamaño de pantalla, por ejemplo:

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

Con este código obtendríamos un resultado similar al de la siguiente figura, donde en primer lugar se crea una fila con dos columnas de igual ancho, y a continuación se añade una segunda fila con tres columnas de igual ancho.



El número de columnas del ejemplo anterior se mantendrá para todos los tamaños de pantalla, adaptando el ancho de las columnas para cada uno de ellos.

A esta clase podemos añadir el sufijo para establecer el tamaño de pantalla, por lo que disponemos de las clases `.col`, `.col-sm`, `.col-md`, `.col-lg`, `.col-xl`. En todos los casos estaremos indicando que queremos una columna de ancho automático desde el tamaño de pantalla indicado en adelante. Debemos de tener en cuenta que si indicamos algo como `<div class="col col-sm col-md">` el resultado que obtendríamos sería el mismo en todos los tamaños, una columna de ancho automático, equivalente a haber indicado únicamente `<div class="col">`. Por lo tanto, el uso de estos sufijos solo se justificará cuando queramos una columna de ancho automático solamente de un tamaño en adelante, y que por lo tanto, para los tamaños inferiores se cree una columna que ocupe todo el ancho. Por ejemplo, veamos el siguiente código:

```
<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>
```

Estamos indicando que queremos una fila con tres columnas de ancho automático para los tamaños desde sm en adelante, y por lo tanto, para el tamaño extra pequeño estas tres columnas pasarán a ocupar todo el ancho, transformándose en tres filas completas.

Modo mixto

Estas columnas de ancho automático se pueden mezclar **en una misma fila** con las columnas de ancho específico que hemos visto antes. La forma de calcular el ancho de cada columna será el siguiente: En primer lugar se calculará el tamaño de las columnas de ancho específico y a continuación se llenará el espacio restante usando las columnas de ancho automático. Por ejemplo, a continuación vamos a definir dos filas mezclando ambos tipos de columnas:

```
<div class="container">
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-6">2 of 3 (wider)</div>
    <div class="col">3 of 3</div>
  </div>
  <div class="row">
    <div class="col-5">1 of 3 (wider)</div>
    <div class="col">2 of 3</div>
    <div class="col">3 of 3</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura:

1 of 3	2 of 3 (wider)	3 of 3
1 of 3 (wider)	2 of 3	3 of 3

Como podemos ver en este ejemplo, los sistemas para de definir las columnas **se pueden mezclar como queramos**, por ejemplo usando el de ancho fijo entre dos columnas de ancho automático, o estableciendo una columna de ancho específico al principio y después dos de ancho automático.

Ancho de columna variable

Con Bootstrap 4 también se introdujeron las columnas de ancho variable, las cuales ocuparán el ancho justo que se necesite según el contenido de la columna. Para utilizarlas disponemos de las clases " `.col-*-auto` ", donde * puede ser cualquiera de los sufijos de tamaño de pantalla que hemos visto antes `sm`, `md`, `lg`, `xl`, o ningún sufijo (`.col-auto`) para indicar todos los tamaños.

Estas etiquetas, igual que las de ancho automático, se pueden mezclar con las de ancho específico, por ejemplo:

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">1 of 3</div>
    <div class="col-md-auto">Variable width content</div>
    <div class="col col-lg-2">3 of 3</div>
  </div>
  <div class="row">
    <div class="col">1 of 3</div>
    <div class="col-md-auto">Variable width content</div>
    <div class="col col-lg-2">3 of 3</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura:

1 of 3	Variable width content	3 of 3
1 of 3	Variable width content	3 of 3

En el código anterior se ha usado la clase " `.justify-content-md-center` " para alinear el contenido dentro de una fila, estas etiquetas las veremos en la sección "Alineación". A continuación vamos a ver otras utilidades del sistema de rejilla, como el anidamiento de columnas, o cómo forzar el cambio de fila.

Forzar el cambio de fila

Mediante la clase `.w-100` podemos forzar el cambio de fila cuando nosotros queramos:

```
<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>
```

Con lo que obtendríamos dos filas con dos columnas cada una:

.col-6 .col-sm-3	.col-6 .col-sm-3
.col-6 .col-sm-3	.col-6 .col-sm-3

Esta clase también nos puede ser útil para forzar el cambio de fila solo para determinados tamaños de pantalla. Para esto tenemos que combinarla con otras clases de Bootstrap que nos permiten mostrar u ocultar elementos según el tamaño de pantalla. A continuación se incluye un ejemplo:

```
<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 at md breakpoint and up -->
  <div class="w-100 d-none d-md-block"></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>
```

Donde la clase "`.d-none`" significa que no se muestre ese elemento (para ningún tamaño), y la clase "`.d-md-block`" indica que se muestre a partir del tamaño de pantalla "md" en adelante. Por lo tanto, el campo div marcado con "w-100" permanecerá oculto para los tamaños extra pequeño y pequeño, y por lo tanto no se activará el cambio de fila para esos dos tamaños, pero sí para los tamaños desde "md" en adelante.

En la sección "Utilidades Responsive" se explicarán las etiquetas ".d-*" que nos permitirán controlar la visibilidad de cualquier elemento HTML en función del tamaño de pantalla.

Anidamiento de columnas

Una característica muy potente del sistema de rejilla es que se pueden anidar columnas dentro de otras columnas. Para esto solamente tenemos que crear una nueva fila dentro de una columna, y dentro de esta nueva fila podremos subdividirla usando también hasta 12 columnas.

Por ejemplo, en el siguiente código se crea una primera fila con una columna de tamaño 9, dentro de la cual se añade una segunda fila con dos columnas:

```
<div class="row">
  <div class="col-sm-9">
    Level 1: .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>
```

Al visualizar este código obtendríamos:

Level 1: .col-sm-9	
Level 2: .col-8 .col-sm-6	Level 2: .col-4 .col-sm-6

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: `.offset-*`. Por ejemplo `.offset-4` creará un espacio a la izquierda de la columna de tamaño 4 (como si se creara una columna oculta de tipo `.col-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 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>
```

El cual se renderizaría de la forma:



Como se puede ver en el ejemplo anterior, también podemos especificar el *offset* según el tamaño de pantalla. Si usamos, por ejemplo, la clase "`offset-4`" estaremos indicando que se cree un espacio de 4 para **todos** los tamaños de pantalla; mientras que con "`offset-md-4`" se creará este espacio a partir del tamaño de pantalla "md" en adelante.

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 medianas ni grandes tendríamos que hacer:

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

Márgenes

Además de la clase `offset` también disponemos de las clases para crear márgenes de espacio variable tanto al lado izquierdo (con "`.ml-auto`") como al lado derecho (con `.mr-auto`) de una columna. A continuación se incluye un ejemplo:

```
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4 ml-auto">.col-md-4 .ml-auto</div>
</div>
<div class="row">
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
  <div class="col-md-3 ml-md-auto">.col-md-3 .ml-md-auto</div>
</div>
<div class="row">
  <div class="col-auto mr-auto">.col-auto .mr-auto</div>
  <div class="col-auto">.col-auto</div>
</div>
```

Con lo que obtendríamos el siguiente resultado:



Como se puede ver, en la primera fila se crea un marge automático por la izquierda (y para todos los tamaños) de la segunda columna, lo que provoca el desplazamiento de esta hasta alinearla a la derecha. En la segunda fila se añade margen por la izquierda a las dos columnas para tamaños de pantalla de "md" en adelante. Y en la última fila se crea un margen automático por la derecha de la primera columna (para todos los tamaños), esto provoca un efecto similar al obtenido en la primera fila.

Ordenación de columnas

También podemos modificar el orden visual de las columnas mediante la clase `.order-`. Esta clase permite indicar la posición a la cual queremos desplazar la columna (del 1 al 12, por ejemplo `.order-1`, `.order-2`, etc.). También podemos especificar el tamaño de pantalla para el que queremos que se aplique (por ejemplo `.order-md-12`). A continuación se incluye un ejemplo:

```
<div class="container">
  <div class="row">
    <div class="col">First, but unordered</div>
    <div class="col order-12">Second, but last</div>
    <div class="col order-1">Third, but first</div>      <!-- ¡¡CUIDADO!! -->
  </div>
</div>
```

Obteniendo como resultado:

First, but unordered	Third, but first	Second, but last
----------------------	------------------	------------------

Si nos fijamos en el resultado obtenido podemos ver que **no** se obtiene el resultado esperado, la tercera columna (en color rojo) aparece en la segunda posición en lugar de en la primera como se había indicado con "`order-1`". Esto es debido a un pequeño error al cambiar el orden de derecha izquierda. En los casos en los que simplemente queramos mover una columna hacia la derecha no se producirá este error, **pero si queremos mover hacia la izquierda será necesario que establezcamos el order de todas las columnas**. Por lo tanto, para que funcione correctamente el ejemplo anterior tendríamos que escribir el siguiente código:

```
<div class="container">
  <div class="row">
    <div class="col order-2">First, but unordered</div>      <!-- Añadimos el orden de esta columna -->
    <div class="col order-12">Second, but last</div>
    <div class="col order-1">Third, but first</div>
  </div>
</div>
```

Obteniendo ahora sí el resultado esperado:

Third, but first	First, but unordered	Second, but last
------------------	----------------------	------------------

Como se puede ver también en este ejemplo, no es necesario que los números de columna para la ordenación sean consecutivos, simplemente se ordenarán de mayor a menor.

Hay que tener cuidado con estas clases si hay un salto de línea dentro de una misma fila (debido a que el número de columnas ocupe más de 12), ya que en estos casos el orden no funcionarán correctamente.

También disponemos de la clase "`.order-first`", la cual nos permitirá situar cualquier elemento en primer lugar. Además, esta clase sí que funciona aunque haya un salto de línea.

Alineación

Con la nueva versión de Bootstrap también han aparecido nuevas clases que nos permiten especificar la alineación de las columnas tanto en horizontal como en vertical.

Alineación vertical

Para indicar la alineación en vertical tenemos dos opciones: indicar la misma alineación para todos los elementos de una fila o indicar la alineación a nivel de columna, lo que nos permitirá establecer distintas alineaciones para cada columna.

En el primer caso la clase CSS para la alineación la tendremos que añadir a la fila usando la etiqueta "`.align-items-*`", donde "*" podrá ser "`start`" (al principio o pegada a la parte superior de la fila), "`center`" (alineación centrada en vertical) o "`end`" (alineación pegada al final o a la parte inferior de la fila). A continuación se incluye un ejemplo de los tres tipos de alineación:

```
<div class="container">
  <div class="row align-items-start">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
  <div class="row align-items-center">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
  <div class="row align-items-end">
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
    <div class="col">One of three columns</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado como el de la siguiente figura:

One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns

En el segundo caso, si queremos indicar por separado la alineación vertical de cada una de las columnas de una fila, tendremos que usar la clase CSS `.align-self-*`, donde "*" podrá adoptar los mismos valores: `start`, `center` o `end`. A continuación se incluye un ejemplo en el que se indican los tres tipos de alineaciones dentro de una misma fila:

```
<div class="container">
  <div class="row">
    <div class="col align-self-start">One of three columns</div>
    <div class="col align-self-center">One of three columns</div>
    <div class="col align-self-end">One of three columns</div>
  </div>
</div>
```

Con lo que obtendremos el siguiente resultado:

One of three columns		
	One of three columns	One of three columns

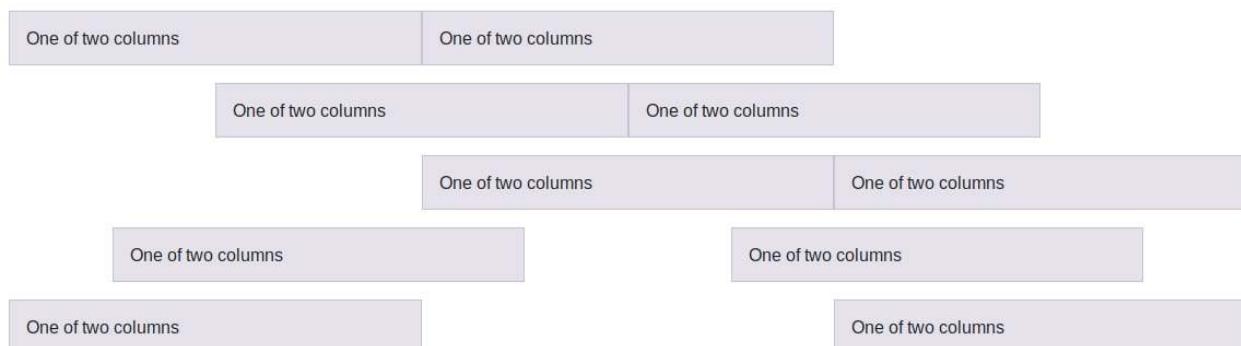
Es importante destacar que al utilizar cualquiera de estas etiquetas de alineación, la altura de las columnas **se ajustará al contenido**, mientras que si no utilizamos ninguna etiqueta de alineación, la altura de la celda se **extenderá hasta ocupar todo el espacio disponible** en la fila.

Alineación horizontal

También podemos especificar la alineación horizontal de los elementos de una fila. Para esto disponemos de la clase ".justify-content-*", donde "*" podrá ser "start" (izquierda), "center" (centrado), "end" (derecha), "around" (añadirá el mismo espacio a **ambos lados** de la columna) y "between" (añade espacio **entre** las columnas). A continuación se incluye un ejemplo de cada uno de estos tipos de alineación horizontal:

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">One of two columns</div>
    <div class="col-4">One of two columns</div>
  </div>
</div>
```

Con lo que obtendríamos un resultado similar al de la siguiente figura, con la fila alineada a la izquierda, la segunda centrada, la tercera alineada a la derecha, la cuarta con el espaciado "al rededor" (o a ambos lados) de las columnas, y la última con el espaciado entre las columnas.



Alineación responsive

En caso de que lo necesitemos podremos añadir también el tamaño de pantalla a las distintas clases de alineación que hemos visto: `align-items-*`, `align-self-*` y `justify-content-*`. Para esto tendremos que añadir primero el tamaño de pantalla (sm, md, lg o xl), a continuación un guión (-), y después el tipo de alineación deseado (de entre los que hemos visto), por ejemplo: `align-items-md-center`, `align-self-sm-end`, `justify-content-lg-end`, etc.

Al indicar el tamaño de pantalla dicha alineación se aplicará solamente a partir de dicho tamaño en adelante, aunque también podemos indicar distintas alineaciones para un mismo campo según el tamaño de la pantalla, por ejemplo:

```
<div class="row justify-content-center justify-content-md-start">
  ...
</div>
```

En el ejemplo anterior el contenido se alinearía de forma centrada para los tamaños de pantalla extra pequeños y pequeños, y cambiará a alineación izquierda a partir del tamaño de pantalla "md".

Utilidades Responsive

Bootstrap también incluye una serie de clases para ayudarnos a mostrar u ocultar contenidos según el tamaño del dispositivo. En primer lugar vamos a ver las clases base que utilizaremos para estas acciones:

- `.d-none` : **Oculta** el elemento sobre el que se aplique.
- `.d-inline` : **Muestra** el elemento de forma "inline", es decir, permitiendo otros elementos por los lados y ocupando el ancho justo.
- `.d-block` : **Muestra** el elemento en forma de bloque, ocupando todo el ancho disponible y sin permitir otros elementos por los lados.
- `.d-inline-block`: **Muestra** el elemento en forma de bloque, pero ocupando el ancho justo y permitiendo otros elementos por los lados.

A continuación podemos ver un ejemplo del efecto obtenido al aplicar las distintas etiquetas de las que disponemos para mostrar elementos:



La diferencia entre las etiquetas "`d-inline`" y "`d-inline-block`" es el comportamiento de bloque que adopta el elemento, el cual respetará todos los márgenes y algunas que le indiquemos.

Al aplicar estas etiquetas sobre un elemento lo mostraremos u ocultaremos **para todos los tamaños**, sin embargo, si queremos podemos añadirles modificadores para indicar el tamaño **a partir del cual** queremos que se muestren u oculten. En este último caso tendremos que añadir el tamaño de pantalla entre el prefijo "`d-`" y el sufijo `none` , `inline` , `block` O `inline-block` , es decir, siguiendo el patrón "`d-*-*`". Por ejemplo, podremos indicar `d-sm-none` para que se oculte a partir del tamaño pequeño de pantalla, `d-xl-none` para que se oculte para las pantallas extra grandes, o `d-md-block` para que se muestre en forma de bloque a partir del tamaño md.

Es importante que nos fijemos que estas utilidades *responsive* se aplicarán a partir del tamaño indicado en adelante, sin embargo, ¿cómo podríamos hacer para que solamente se oculte o se muestre para un tamaño de pantalla? Para esto podemos **combinar varias clases**, por ejemplo, para que solo se oculte para el tamaño extra pequeño tendríamos que poner "`d-none d-sm-block`", o para que solo se muestre para el tamaño pequeño usaríamos "`d-none d-sm-block d-md-none`".

A continuación se incluye una tabla resumen de las etiquetas que tendríamos que aplicar para mostrar u ocultar solamente para un tamaño de pantalla:

Tamaños de pantalla	Mostrar	Ocultar
Solo para tamaños extra pequeños	d-block d-sm-none	d-none d-sm-block
Solo para tamaños pequeños (sm)	d-none d-sm-block d-md-none	d-sm-none d-md-block
Solo para tamaños medianos (md)	d-none d-md-block d-lg-none	d-md-none d-lg-block
Solo para tamaños grandes (lg)	d-none d-lg-block d-xl-none	d-lg-none d-xl-block
Solo para tamaños extra grandes (xl)	d-none d-xl-block	d-xl-none

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 (colores, alineación interna, 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 del tamaño mínimo, también podemos indicar el tamaño máximo o el rango de tamaño en el que se aplicarán los estilos, 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) < 576px
- Pantallas pequeñas (_sm, tablets _en vertical) \geq 576px
- Pantallas medianas (md, para tablets en horizontal) \geq 768px
- Pantallas grandes (lg, tamaño escritorio) \geq 992px
- Pantallas extra grandes (xl, escritorio grande) \geq 1200px

Es importante tenerlos en cuenta a la hora de definir nuestras propias *media queries*, para crear los puntos de ruptura o cambio con las mismas dimensiones. Si no lo hicieramos así estaríamos añadiendo todavía más casos a los cinco tamaños de pantalla que contempla Boostrap, y por lo tanto complicando tanto la programación como el mantenimiento del código.

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-primary`, `.btn-secondary`, `.btn-success`, `.btn-danger`, `.btn-warning`, `.btn-info`, `.btn-light`, `.btn-dark` O `.btn-link` para crear botones para diferentes estados o acciones en nuestros formularios:

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```

Con lo que obtendríamos el siguiente resultado:

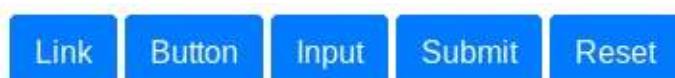


Elementos tipo botón

Estas clases no son exclusivas para las etiquetas `button` sino que también funcionarán de la misma forma con `<a>` y `<input>`:

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

Obteniendo en todos los casos botones con la misma apariencia:

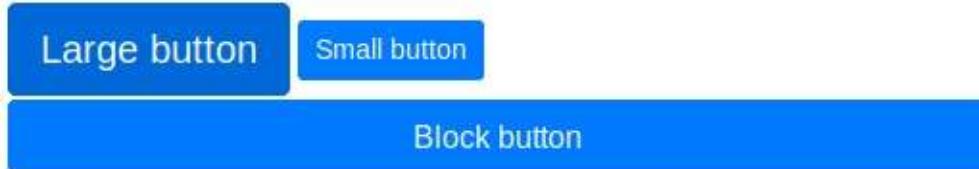


Tamaño de los botones

Podemos variar el tamaño de los botones simplemente añadiendo las clases `.btn-lg` , `.btn-sm` o `.btn-block` , para obtener botones con un tamaño más grande, más pequeño, o un botón que ocupe todo el ancho. Por ejemplo, con el siguiente código:

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
<button type="button" class="btn btn-primary btn-sm">Small button</button>
<button type="button" class="btn btn-primary btn-block">Block button</button>
```

Obtendríamos el siguiente resultado:



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-secondary dropdown-toggle" type="button" id="dropdownMenuButton"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropdown button
  </button>
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</div>
```

Con lo que obtendríamos el siguiente resultado:



Sobre el botón principal podemos aplicar todos los colores de botones que hemos visto en la sección titulada "Botones", por ejemplo "`.btn-success`" o "`.btn-danger`". También podemos añadir los modificadores de tamaño "`.btn-lg`" y "`.btn-sm`" para aumentar o disminuir el tamaño del botón del desplegable.

Si nos fijamos en el código anterior, para el botón principal se ha usado la etiqueta "`button`" y para los elementos del desplegable la etiqueta "`a`", sin embargo podríamos haber usado solamente la etiqueta "`a`" o solamente la etiqueta "`button`", es decir, funcionan exactamente igual y su apariencia es la misma.

Los atributos que empiezan con "aria" son para crear contenido accesible, para que los lectores de pantalla pueden encontrar las etiquetas correctas a la hora de interpretar el contenido. Para más información consultar la documentación sobre HTML 5 ARIA.

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

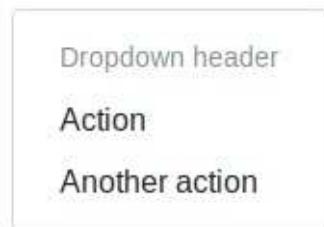
```
<div class="dropdown-menu dropdown-menu-right">
```

Encabezados en un desplegable

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

```
<div class="dropdown-menu">
  <h6 class="dropdown-header">Dropdown header</h6>
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
</div>
```

Con lo que obtendremos:



Separadores en un desplegable

También podemos añadir separadores en un desplegable mediante la clase `.dropdown-divider` de la forma:

```
<div class="dropdown-menu">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">Separated link</a>
</div>
```

Con lo que obtendríamos el siguiente listado con separador:

Action
Another action
Something else here
Separated link

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="Basic example">
  <button type="button" class="btn btn-secondary">Left</button>
  <button type="button" class="btn btn-secondary">Middle</button>
  <button type="button" class="btn btn-secondary">Right</button>
</div>
```

Con lo que obtendríamos el siguiente resultado:

Left Middle Right

Los atributos "`role`" que utiliza Bootstrap provienen también de HTML 5 ARIA y sirven para indicar el rol o función de un elemento, en este caso se indica la agrupación de los botones.

Mediante la librería JavaScript de Bootstrap podemos añadir comportamientos tipo *checkbox* o *radio button* a un grupo de botones, para que al pulsarlos se quede marcados. Para más información consultad la documentación oficial.

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="Toolbar with button groups">
  <div class="btn-group mr-2" role="group" aria-label="First group">
    <button type="button" class="btn btn-secondary">1</button>
    <button type="button" class="btn btn-secondary">2</button>
    <button type="button" class="btn btn-secondary">3</button>
    <button type="button" class="btn btn-secondary">4</button>
  </div>
  <div class="btn-group mr-2" role="group" aria-label="Second group">
    <button type="button" class="btn btn-secondary">5</button>
    <button type="button" class="btn btn-secondary">6</button>
    <button type="button" class="btn btn-secondary">7</button>
  </div>
  <div class="btn-group" role="group" aria-label="Third group">
    <button type="button" class="btn btn-secondary">8</button>
  </div>
</div>
```

Con lo que obtendríamos el siguiente resultado:

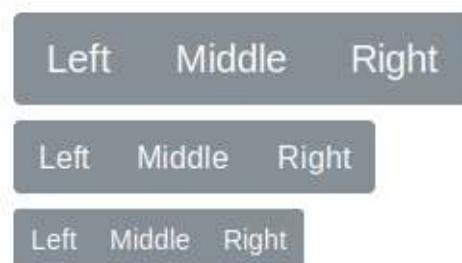


Tamaños de los grupos de botones

Los grupos de botones también nos permiten indicar el tamaño de los botones que contienen mediante las etiquetas `.btn-group-*`, donde "*" puede ser "`sm`" o "`lg`", por ejemplo:

```
<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>
```

Con lo que podríamos obtener grupos de botones de diferentes tamaños:

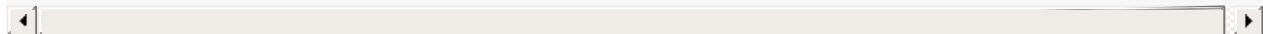


Grupo de botones con desplegables

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 group with nested dropdown">
  <button type="button" class="btn btn-secondary">1</button>
  <button type="button" class="btn btn-secondary">2</button>

  <div class="btn-group" role="group">
    <button id="btnGroupDrop1" type="button" class="btn btn-secondary dropdown-toggle"
      data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
      Dropdown
    </button>
    <div class="dropdown-menu" aria-labelledby="btnGroupDrop1">
      <a class="dropdown-item" href="#">Dropdown link</a>
      <a class="dropdown-item" href="#">Dropdown link</a>
    </div>
  </div>
</div>
```



El resultado visual obtenido sería el siguiente:



Como se puede observar en el código de ejemplo anterior, 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 convertirlos en elementos responsive, mejorar su apariencia y permitirnos crear diferentes alineaciones. La estructura básica de un formulario es la siguiente:

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

Con lo que obtendríamos un formulario **en vertical** como el de la siguiente figura, es decir, los elementos del formulario se dispondrán en vertical, unos debajo de otros.

Email address

Enter email

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">
  <div class="form-group mx-sm-3">
    <label for="inputUser" class="sr-only">User</label>
    <input type="password" class="form-control" id="inputUser" placeholder="User">
  </div>
  <div class="form-group mx-sm-3">
    <label for="inputPass" class="sr-only">Password</label>
    <input type="password" class="form-control" id="inputPass" placeholder="Pass">
  </div>
  <button type="submit" class="btn btn-primary">Confirm</button>
</form>
```

Obteniendo el siguiente resultado:



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*).

Esta alineación tipo "inline" solo será visible para pantallas grandes. En pantallas pequeñas los elementos cambiarán a alineación vertical.

En el ejemplo se ha añadido la etiqueta ".mx-sm-3" para crear un pequeño margen en los laterales de cada elemento del formulario. Para más información sobre este tipo de etiquetas consultad la documentación oficial.

Formulario horizontal

Mediante el uso del sistema de rejilla de Bootstrap podemos crear formularios con disposición en horizontal. Para esto tendremos que crear una fila (`.row`) por cada grupo, y situar la etiqueta y el input cada uno en una columna. A continuación se incluye un ejemplo:

```
<form>
  <div class="form-group row">
    <label for="inputEmail3" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="inputEmail3" placeholder="Email">
    </div>
  </div>
  <div class="form-group row">
    <label for="inputPassword3" class="col-sm-2 col-form-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 row">
    <div class="col-sm-2">Checkbox</div>
    <div class="col-sm-10">
      <div class="form-check">
        <label class="form-check-label">
          <input class="form-check-input" type="checkbox"> Check me out
        </label>
      </div>
    </div>
  </div>
  <div class="form-group row">
    <div class="col-sm-10">
      <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
  </div>
</form>
```

Con lo que obtendríamos:

Email	<input type="text" value="Email"/>
Password	<input type="password" value="Password"/>
Checkbox	<input type="checkbox"/> Check me out
Sign in	

Es importante que nos fijemos que la etiqueta `.row` se añade al `div` de cada grupo, manteniendo también la etiqueta "`.form-group`". 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: `.is-valid` o `.is-invalid` sobre el propio input. De esta forma, el color de los elementos del formulario cambiará. A continuación podemos ver un ejemplo:

```
<form>
  <div class="form-group">
    <label for="validation01">First name</label>
    <input type="text" class="form-control is-valid" id="validation01"
           placeholder="First name" value="Mark" required>
  </div>
  <div class="form-group">
    <label for="validation02">City</label>
    <input type="text" class="form-control is-invalid" id="validation02" placeholder=
"City" required>
  </div>
</form>
```

Que se mostraría de la forma:

First name

Mark

City

City

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>
```

Con lo que obtendríamos el siguiente resultado:

The image displays three separate input fields, each enclosed in a light gray border. Each input field has a small gray box to its left containing a character: '@' for the first, '.' for the second, and '\$' for the third. To the right of each input field is a larger gray box containing the text 'Username', '.00', and '.00' respectively.

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:

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Es importante que nos fijemos en cómo se usan las clases CSS `.nav` , `.nav-tabs` , `.nav-item` y `.nav-link` . Cada elemento del menú será un `.nav-item` , los cuales contienen un enlace tipo `.nav-link` a la sección a mostrar. Para marcar el elemento del menú que está activo o seleccionado se utiliza la clase `.active` . Además disponemos de la clase `.disabled` para deshabilitar elementos del menú.

Si visualizamos el código de ejemplo anterior obtendríamos un menú en forma de pestañas como el siguiente:



Píldoras

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

```
<ul class="nav nav-pills">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

En este caso el aspecto del menú sería el siguiente:



También podemos crear un menú vertical o apilado añadiendo la clase `.flex-column` a la etiqueta contenedora:

```
<ul class="nav nav-pills flex-column">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Obteniendo:

[Active](#)[Link](#)[Link](#)[Disabled](#)

Ancho 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-fill` a la etiqueta contenedora, de la forma:

```
<ul class="nav nav-pills nav-fill">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Longer nav link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Con lo que obtendríamos:

[Active](#)[Longer nav link](#)[Link](#)[Disabled](#)

Este estilo no funcionará para pantallas con un ancho menor a 768px, que son las pantallas definidas como extra pequeñas o de *smartphone*. Para estos tamaños cada elemento del menú ocupará el ancho justo que necesite.

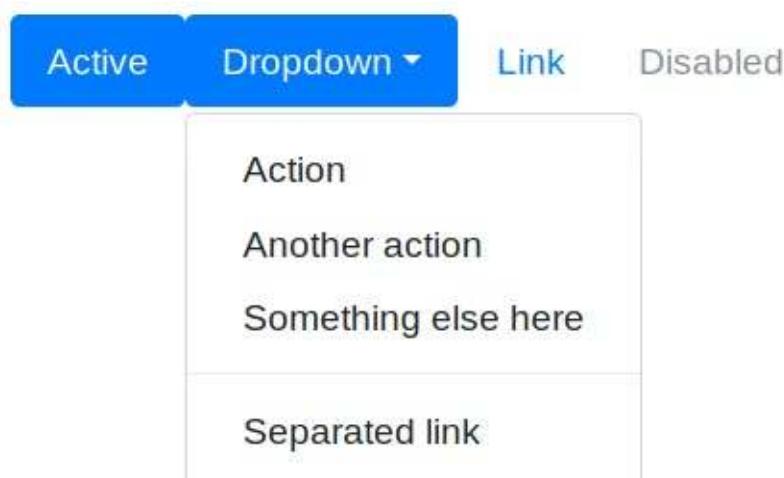
Elementos de navegación con desplegables

También podemos añadir elementos desplegables a nuestros menús de navegación, tanto al de tipo tabs como al de píldoras. Para esto simplemente añadiremos el dropdown como un elemento del menú más, usando la notación que vimos en la sección "Desplegables", pero llevan cuidado de que para la etiqueta incial (que en el dropdown normal era "

class="dropdown"> ") se utilice el propio elemento ".nav-item" del menú, añadiendo la clase ".dropdown" de la forma: "<li class="nav-item dropdown"> ". A continuación se incluye un ejemplo completo:

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button"
       aria-haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Con lo que obtendríamos un resultado como el de la siguiente figura:



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.

Para añadir esta barra a nuestro sitio web utilizaremos la etiqueta "`<nav>`", que es la etiqueta de HTML 5 que identifica un elemento de navegación. En caso de no usar esta etiqueta también podemos crear la barra de navegación usando un "`<div>`", pero en este caso tendremos que añadir el atributo `role="navigation"` por cuestiones de accesibilidad. Además, en esta etiqueta también añadiremos dos etiquetas para indicar el estilo y los colores a aplicar con "`.navbar-light .bg-light`" (más adelante veremos qué otros colores podemos usar), y la etiqueta `.navbar-expand-lg` para indicar el tamaño a partir del cual la barra se mostrará de forma expandida. La etiqueta `.navbar-expand-lg` indica que la barra se mostrará en su tamaño completo a partir del tamaño de pantalla grande (lg), colapsándose para tamaños más pequeños. Este sería el comportamiento por defecto, pero si queremos lo podemos modificar cambiando el tamaño "lg" por otro de los posibles tamaños definidos por Bootstrap: "`sm`", "`md`", "`lg`" o "`xl`".

Dentro de la etiqueta "`<nav>`" el contenido de la barra estará dividido en tres secciones:

- Nombre o logotipo de la web, marcado con la etiqueta "`.navbar-brand`".
- Botón toggler marcado con "`.navbar-toggler`", que se mostrará únicamente cuando el menú se colapse y se ocultará cuando el menú aparezca expandido. Cuando sea visible podremos pulsar sobre él para mostrar u ocultar el menú.
- Las opciones de menú, las cuales las añadiremos dentro de una lista tipo "``" con la clase "`.navbar-nav`". Además, esta lista la tendremos que meter dentro de una caja "`<div>`" con las clases "`.collapse .navbar-collapse`", que definirá la zona que se colapsará (u ocultará) para pantallas pequeñas.
 - Cada elemento de la lista de menú `` se definirá mediante una etiqueta "`<a>`" sobre la que aplicaremos la clase "`.nav-item`". Además, como ya veremos más adelante, podremos añadir otros elementos dentro de las opciones de menú, como por ejemplo un formulario.

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

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
         data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
         aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
           data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Disabled</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
    </form>
  </div>
</nav>

```

Si añadimos este código a nuestro sitio Web y lo visualizamos con un navegador, obtendremos el siguiente resultado cuando lo visualicemos en pantallas medianas y grandes:



En las pantallas pequeñas los elementos de navegación se colapsarían en un botón (*toggler*), de la forma:

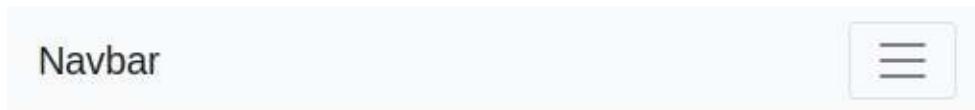


Imagen en la barra de navegación

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

```
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    
  </a>
  ...
</nav>

<!-- O si queremos incluir un logotipo y texto... -->
<nav class="navbar navbar-light bg-light">
  <a class="navbar-brand" href="#">
    
    Bootstrap
  </a>
</nav>
```

Con lo que obtendríamos los siguiente resultados, en el primer caso se mostraría el logotipo



Bootstrap

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

Barra de navegación con formulario

Podemos añadir formularios a nuestra barra de navegación utilizando el tipo de formulario inline, definido con "`.form-inline`" como vimos en la sección "Formularios", por ejemplo:

```
<nav class="navbar navbar-light bg-light">
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Obteniendo:



Alineación

Para modificar la alineación del formulario podemos utilizar las clases que vimos en la sección "Alineación horizontal" dentro del "Sistema de rejilla", como por ejemplo ".justify-content-between":

```
<nav class="navbar navbar-light bg-light justify-content-between">
  <a class="navbar-brand">Navbar</a>
  <form class="form-inline">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</nav>
```

Que produciría que el formulario se alejase a la derecha:



Anclajes de la barra de navegación

Bootstrap nos permite anclar o fijar la posición de la barra de tres formas distintas: fijarla a la parte superior añadiendo la clase `.fixed-top` a la etiqueta nav, fijarla a la parte inferior con `.fixed-bottom`, o usar el modo sticky (o pegajoso) con la etiqueta `.sticky-top`, el cual anclará la barra a la parte superior mientras se realiza scroll y cuando se alcanza el tope permanecerá fija. A continuación se incluye un ejemplo de cada uno de estos modos:

```
<!-- Fixed top -->
<nav class="navbar fixed-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed top</a>
</nav>

<!-- Fixed bottom -->
<nav class="navbar fixed-bottom navbar-light bg-light">
  <a class="navbar-brand" href="#">Fixed bottom</a>
</nav>

<!-- Sticky top -->
<nav class="navbar sticky-top navbar-light bg-light">
  <a class="navbar-brand" href="#">Sticky top</a>
</nav>
```

En los modos " .fixed-top " y " .fixed-bottom ", dado que la barra se colocará de forma "flotante" sobre el contenido, es posible que oculte una parte del mismo. Para solucionar esto es necesario añadir un pequeño espaciado superior o inferior 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; } /* En el caso de .fixed-top */
body { padding-bottom: 70px; } /* En el caso de .fixed-bottom */
```

Contenedores

Aunque no es completamente necesario, sí que se recomienda añadir un elemento contenedor a la barra. El cual podrá añadirse de dos formas: un contenedor externo que incluya toda la barra, o un contenedor interno que incluya solo los elementos de la barra. A continuación se incluyen ambos ejemplos:

```
<!-- Contenedor externo -->
<div class="container">
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Navbar</a>
  </nav>
</div>

<!-- Contenedor interno -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Navbar</a>
  </div>
</nav>
```

El resultado obtenido solo difiere para los tamaños de pantalla grandes, en los cuales, en el primer caso la barra aparecerá centrada (y el color de la barra solo se aplicará en el espacio central), y en el segundo caso la barra ocupará todo el ancho posible (por lo que el color de aplicará en todo el ancho) pero los elementos de la barra aparecerán centrados. A continuación se incluye un ejemplo de los dos casos:



Colores de la barra de navegación

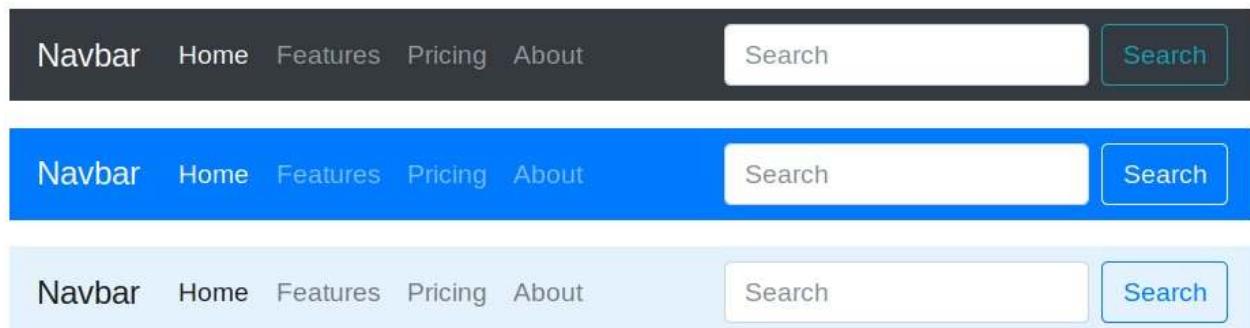
Podemos personalizar el color de la barra y los elementos que la componen de una forma muy sencilla. En primer lugar tendremos que elegir entre el tema claro (`.navbar-light`) o el tema oscuro (`.navbar-dark`), y además asignar un color de fondo con las clases `.bg-*` para personalizar el color (a continuación se incluye la lista de colores posibles). Por ejemplo podríamos modificar la barra de navegación de las siguientes formas:

```
<nav class="navbar navbar-dark bg-dark">
  <!-- ... -->
</nav>

<nav class="navbar navbar-dark bg-primary">
  <!-- ... -->
</nav>

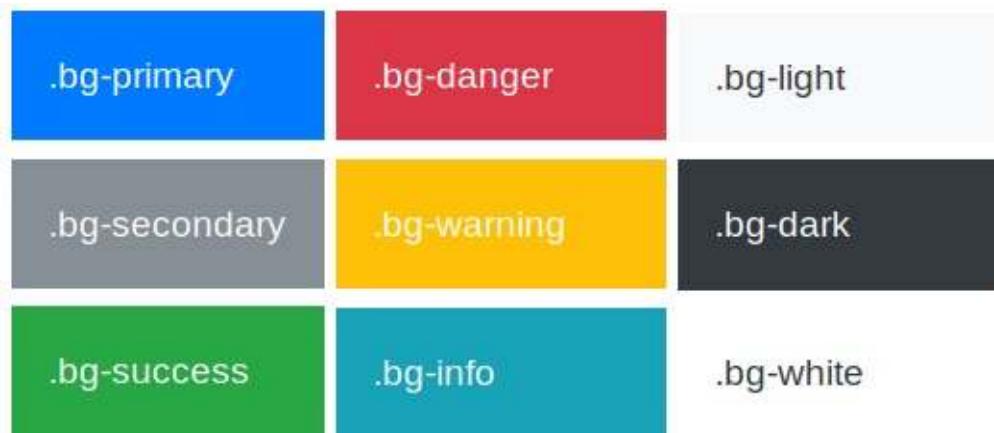
<nav class="navbar navbar-light" style="background-color: #e3f2fd;">
  <!-- ... -->
</nav>
```

Con lo que obtendríamos los siguientes resultados:



Los posibles colores que podemos elegir como fondo para la barra de navegación son los siguientes: `.bg-primary` , `.bg-secondary` , `.bg-success` , `.bg-danger` , `.bg-warning` , `.bg-info` , `.bg-light` , `.bg-dark` y `.bg-white` . Además de poder aplicarlo sobre la barra de

navegación también se pueden utilizar para definir el color de fondo de cualquier otro elemento. A continuación se incluye una imagen de estos colores:



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>
```

La cual configura los estilos de las tablas básicas de HTML para que adopten el siguiente aspecto:

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

En la tabla anterior las celdas de la primera fila estarían marcadas con "th" y el resto de celdas con "td".

Tablas pequeñas

Si queremos compactar el tamaño de la tabla para que deje un padding (o espacio interior) inferior, podemos aplicar la clase `.table-sm` de la forma:

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

Obteniendo:

#	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>
```

Con lo que obtendríamos una tabla con el siguiente aspecto:

#	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>
```

Obteniendo el siguiente resultado:

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Mark	Otto	@TwBootstrap
3	Jacob	Thornton	@fat
4	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 simplemente tenemos que añadir la etiqueta `.table-responsive` a la propia tabla:

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

Obteniendo:

| # | Table heading |
|---|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 1 | Table cell |
| 2 | Table cell |
| 3 | Table cell |

Este efecto se aplicará únicamente sobre dispositivos pequeños (`<576px`) mientras que en el resto de dispositivos no se notará la diferencia. Si queremos que el punto de ruptura a partir del cual se aplique el responsive sobre la tabla sea un tamaño mayor podemos indicar

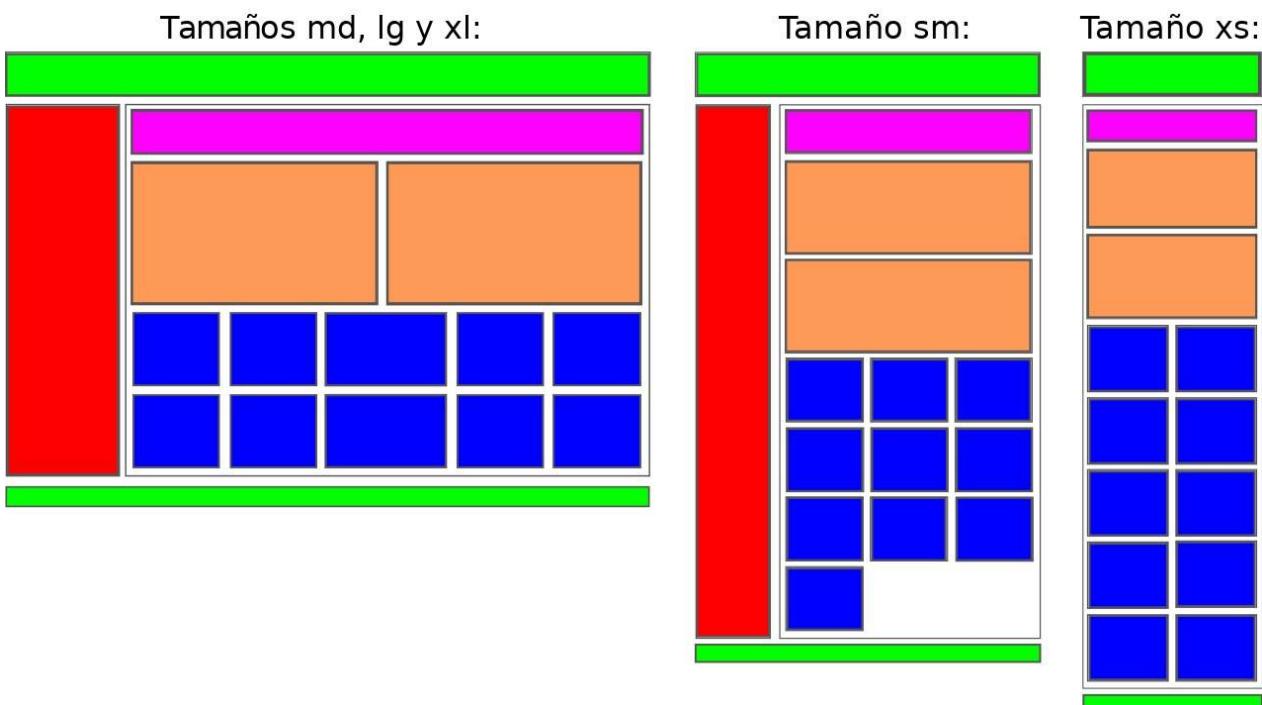
un sufijo de tamaño sobre esta etiqueta, de la forma `.table-responsive-*`, donde "`*`" podrá ser `sm`, `md`, `lg` o `xl`.

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:



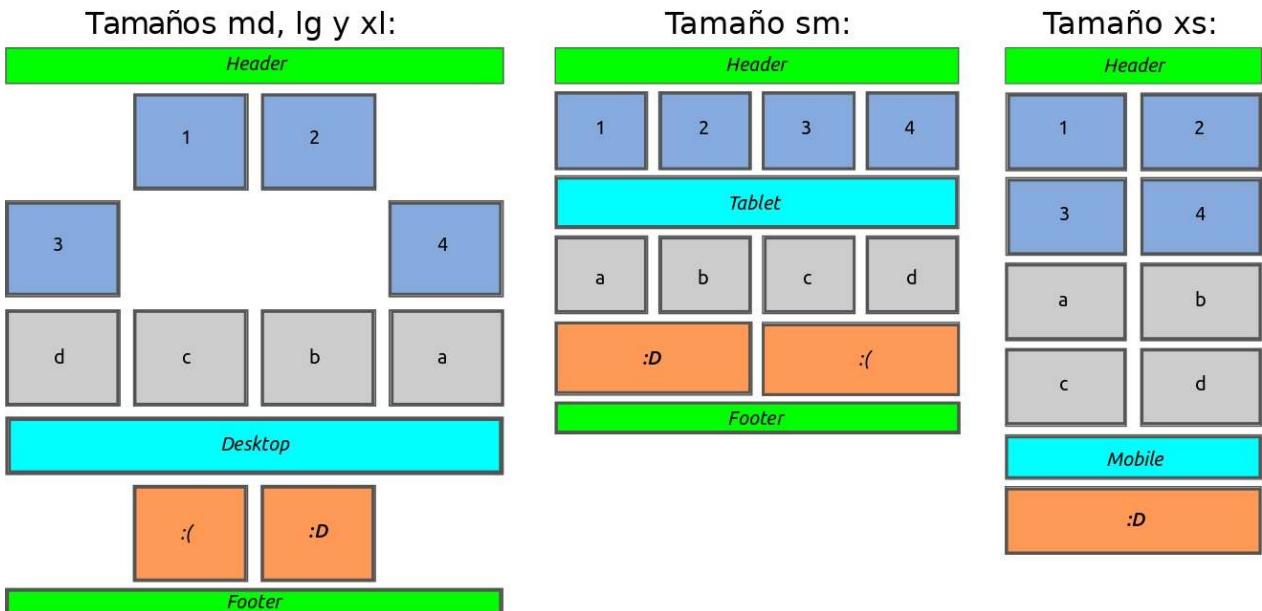
En el esquema de la figura se pueden ver tres disposiciones de la misma web, la de la izquierda se refiere a los tamaños grandes (xl y lg) y medianos (md), la disposición central al tamaño pequeño o de *tablets* (sm) y la de la derecha la correspondiente a móviles (xs).

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 contenedor 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:

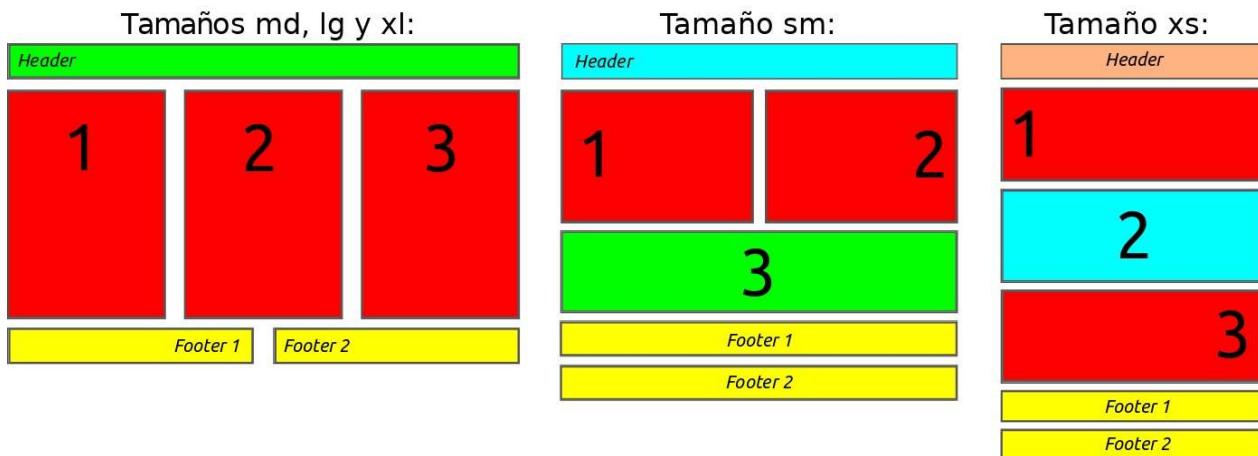
- La segunda fila (que contiene 4 columnas con los números 1, 2, 3 y 4) es solamente una fila a la que se le han añadido *offsets*. Para forzar el cambio de fila se puede añadir un elemento entre la 2^a y la 3^a columna que solo sea visible cuando la pantalla sea mediana o grande (md, lg o xl) y que aplique la clase `.w-100` 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, lg y xl) usando las clases de bootstrap `order-*`.
- En la 5^a fila naranja se ha aplicado un cambio de orden y un offset para las pantallas grandes y medianas (md, lg y xl). 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 md, lg y xl), *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 (*Footer*) 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 y xl):

- 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 del 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 sobre diseño *responsive*, se propone como ejercicio la creación de un pequeño sitio Web estático que use los estilos y componentes 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 enviar 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.

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 código 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.

i This page was translated from English by the community. Learn more and join the MDN Web Docs community.

¿Cuáles son las herramientas de desarrollo del navegador?

Todos los navegadores web modernos incluyen un potente conjunto de herramientas para desarrolladores. Estas herramientas hacen una variedad de cosas, desde inspeccionar HTML, CSS y JavaScript actualmente cargados, hasta mostrar qué activos ha solicitado la página y cuánto tiempo tardaron en cargarse. Este artículo explica cómo utilizar las funciones básicas de las herramientas de desarrollo de tu navegador.

i **Nota:** Antes de ejecutar los siguientes ejemplos, abre el [sitio de ejemplo para principiantes](#) que creamos durante la serie de artículos [Introducción a la Web](#). Lo deberías tener abierto mientras sigues los pasos que explicamos a continuación.

Cómo abrir devtools en tu navegador

Las herramientas para desarrolladores (`devtools`) viven dentro de tu navegador en una subventana que se ve más o menos así, dependiendo del navegador que estés utilizando:

The screenshot shows the Mozilla Firefox browser window with the developer tools open. The main content area displays a yellow background with the text "Mozilla is cool, Irene" and the Firefox logo. The developer tools panel is visible at the bottom, divided into several tabs: Inspector, Console, Debugger, Style Editor, Performance, Memory, Network, Storage, and three others. The 'Inspector' tab is active, showing the HTML structure and associated CSS styles. The CSS pane shows the following rules:

```

<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>Mozilla is cool, Irene</h1>
    
  <p>

```

body {

- ✓ width: 600px;
- ✓ margin: 0 auto;
- ✓ background-color: #FF9500;
- ✓ padding: 0 20px 20px 20px;
- ✓ border: 5px solid black;

Inherited from html

html {

- font-size: 10px;

¿Cómo la levantas? Existen tres distintas maneras:

- **Teclado:** **Ctrl + Mayús + I**, excepto en
 - Internet Explorer y Edge: **F12**
 - macOS: **⌘ + ⌘ + I**
- **Barra de menú:**
 - Firefox: Menú



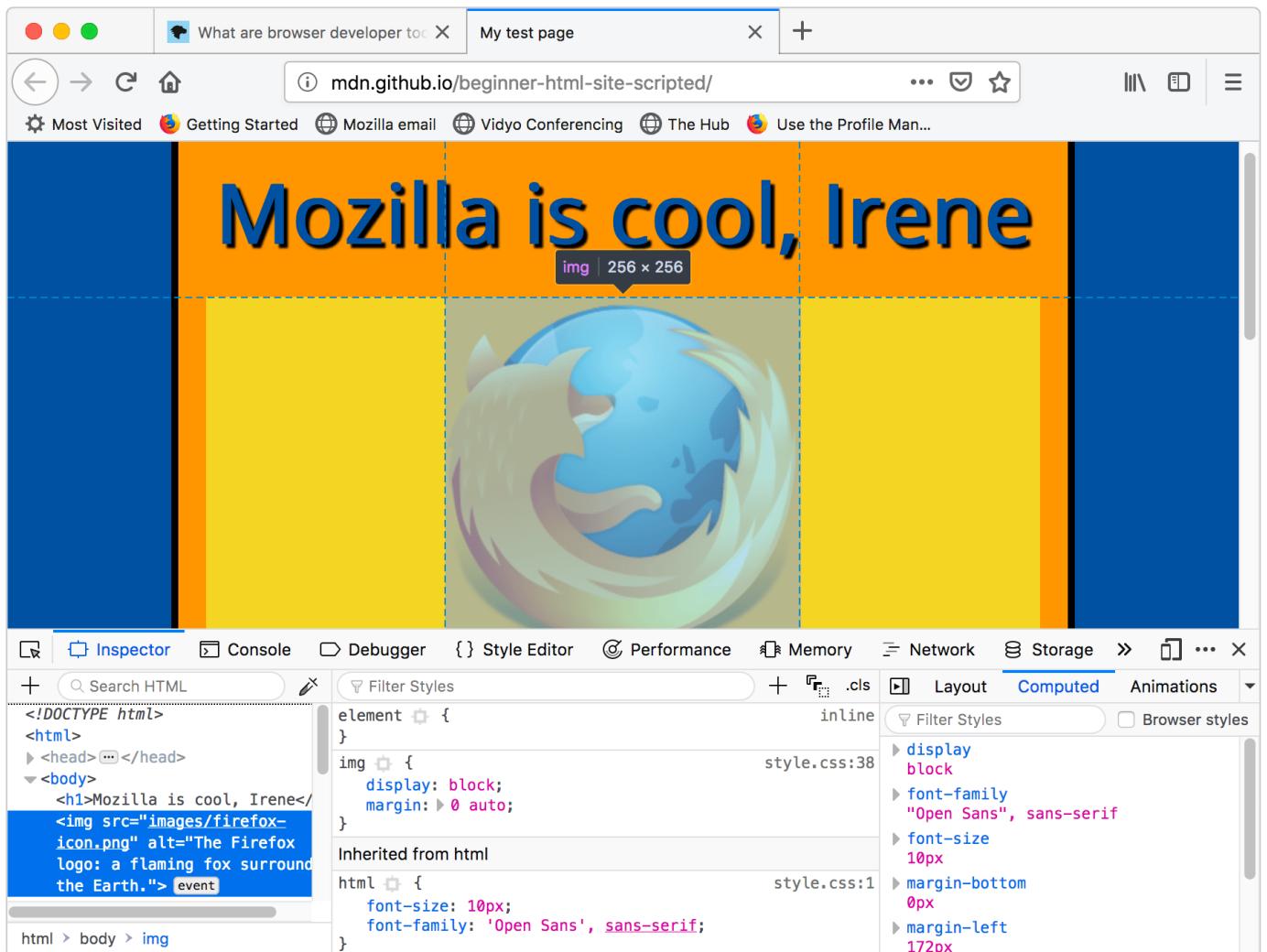
- Desarrollador web ► Alternar herramientas, o ► Herramientas ► Alternar herramientas del desarrollador web
- Chrome: Más herramientas ► Herramientas del desarrollador

- **Safari:** Desarrollador ► Mostrar el inspector web. Si no puedes ver el menú Desarrollar, ve a Safari ► Preferencias ► Avanzado y marca la casilla de verificación Mostrar menú desarrollador en la barra de menú.
- **Opera:** Desarrollador ► Herramientas para desarrolladores
- **Menú contextual:** Presiona y mantén presionado / haz clic con el botón derecho en un elemento en una página web (Ctrl-clic en Mac) y elige Inspeccionar elemento en el menú contextual que aparece. (*Una ventaja adicional:* este método, inmediatamente resalta el código del elemento en el que hiciste clic con el botón derecho).



El inspector: explorador del DOM y editor CSS

Las herramientas del desarrollador, generalmente se abren de forma predeterminada en el inspector, parecido a la siguiente captura de pantalla. Esta herramienta muestra cómo se ve el HTML en tu página en tiempo de ejecución, así como qué CSS se aplica a cada elemento de la página. También te permite modificar instantáneamente el HTML y CSS y ver los resultados de tus cambios reflejados en vivo en la ventana del navegador.

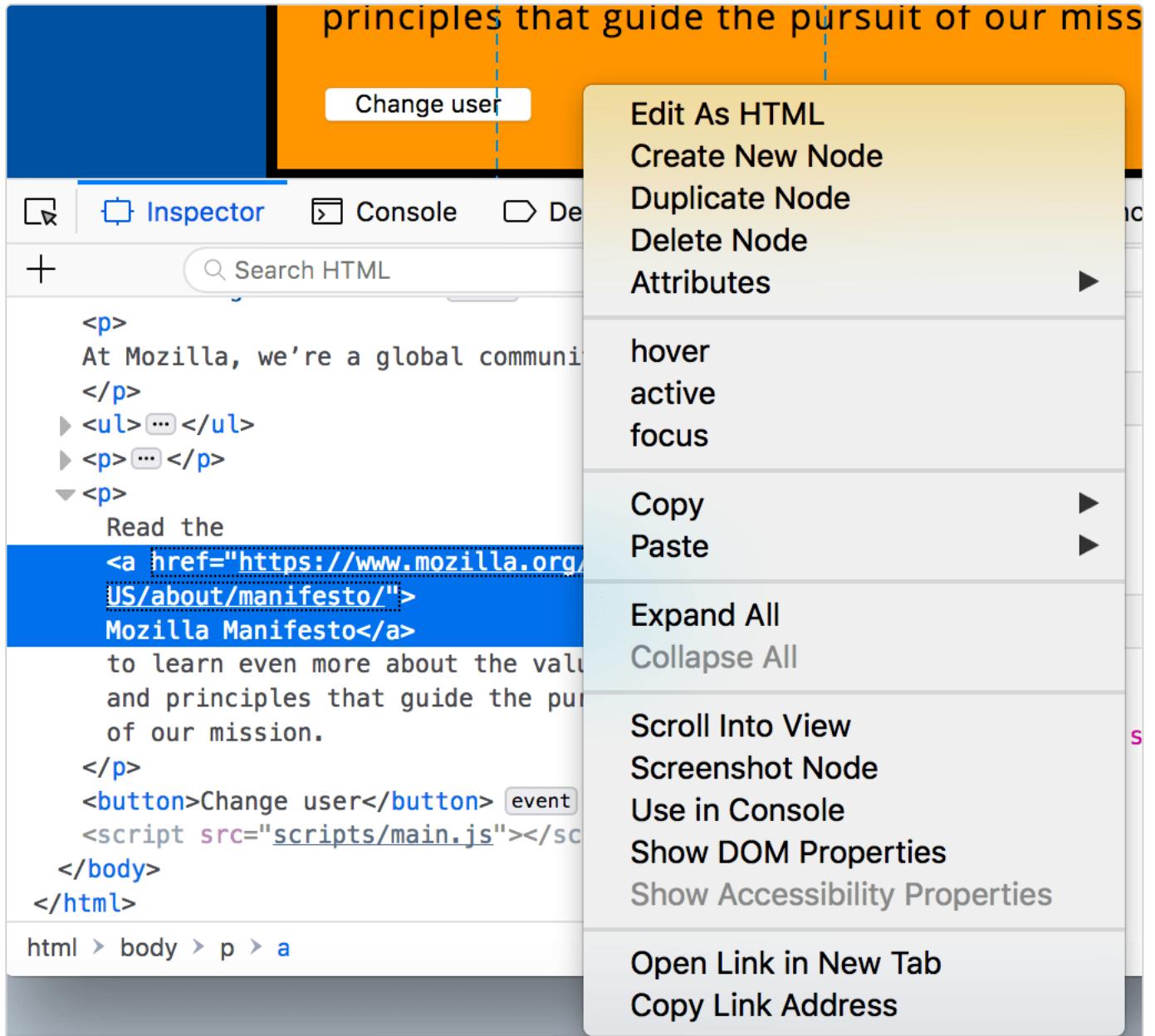


Si no ves al inspector,

- Toca o haz clic en la pestaña *Inspector*.
- En Internet Explorer, toca/haz clic en *Explorador del DOM* o presiona **Ctrl** + **1**.
- En Microsoft Edge u Opera, toca/haz clic en Elementos.
- En Safari, los controles no se presentan con tanta claridad, pero deberías ver el HTML si no has seleccionado otra cosa para que aparezca en la ventana. Presiona el botón *Estilo* para ver el CSS.

Explorando el DOM con el inspector

Para empezar, haz clic con el botón derecho (Ctrl+clic) en un elemento HTML en el inspector del DOM y observa el menú contextual. Las opciones disponibles en el menú varían según el navegador, pero en su mayoría, las más importantes son las mismas:



- **Eliminar nodo** (A veces *Eliminar elemento*). Elimina el elemento actual.
- **Editar como HTML** (A veces *Agregar atributo/Editar texto*). Te permite cambiar el HTML y ver los resultados en tiempo real. Muy útil para depurar y probar.
- **:hover/:active/:focus**. Obliga a que se activen los estados de los elementos, para que puedas ver cómo se vería su estilo.
- **Copiar/Copiar como HTML**. Copie el HTML seleccionado actualmente.
- Algunos navegadores también disponen de *Copiar ruta CSS* y *Copiar XPath*, para permitirte copiar el selector CSS o la expresión XPath que seleccionaría el elemento HTML actual.

Intenta editar algo de tu DOM ahora. Haz doble clic en un elemento o haz clic con el botón derecho del mouse y selecciona *Editar como HTML* en el menú contextual. Puedes realizar los cambios que deseas, pero no los puedes guardar.

Explorar el editor CSS

De manera predeterminada, el editor CSS muestra las reglas CSS aplicadas al elemento seleccionado actualmente:

The screenshot shows the Web Inspector's CSS tab. At the top, there are tabs for 'Layout' (which is selected), 'Computed', and 'Animations'. Below the tabs, the 'Applied Styles' section lists the following rules:

- element { inline }
- Inherited from p
- p, li { style.css:12 }
 - font-size: 16px;
 - line-height: 2;
 - letter-spacing: 1px;
- Inherited from html
- html { style.css:1 }
 - font-size: 10px;
 - font-family: 'Open Sans', sans-serif;

On the right side, the 'Box Model' panel displays the element's dimensions and padding. The element has a width of 148.367 and a height of 22.5. The padding is set to 0. The overall box model is labeled 'content-box'.

Estas características son especialmente útiles:

- Las reglas aplicadas al elemento actual se muestran en orden de mayor a menor especificidad.
- Haz clic en las casillas de verificación junto a cada declaración para ver qué pasaría si eliminaras la declaración.
- Haz clic en la pequeña flecha al lado de la abreviatura de cada propiedad para mostrar los nombres completos equivalentes de la propiedad.
- Haz clic en el nombre o valor de una propiedad para que aparezca un cuadro de texto, donde puedes ingresar un nuevo valor para obtener una vista previa en vivo de un cambio de estilo.

- Junto a cada regla está el nombre del archivo y el número de línea en el que se define la regla. Al hacer clic en esa regla, las herramientas de desarrollo saltan para mostrarlas en su propia vista, donde generalmente puedes editar y guardar.
- También puedes hacer clic en la llave de cierre de cualquier regla para que aparezca un cuadro de texto en una nueva línea, donde puedes escribir una declaración completamente nueva para tu página.

Notarás una serie de pestañas en las que se puede hacer clic en la parte superior del Visor CSS:

- *Calculado*: Muestra los estilos calculados para el elemento seleccionado actualmente (los valores finales normalizados que aplica el navegador).
- *Diseño*: En Firefox, esta área incluye dos secciones:
 - *Modelo de caja*: representa visualmente el modelo de caja del elemento actual, por lo que de un vistazo puedes identificar qué relleno, borde y margen se le aplica, y qué tan grande es su contenido.
 - *Cuadrícula*: Si la página que estás inspeccionando utiliza Grid CSS, esta sección te permite ver los detalles de la cuadrícula.
- *Fuentes*: En Firefox, la pestaña *Fuentes* muestra los tipos de letra aplicados al elemento actual.

Conocer más

Obtén más información sobre el Inspector en diferentes navegadores:

- [Inspector de páginas de Firefox.](#)
- [Explorador del DOM de Edge.](#)
- [inspector del DOM de Chrome.](#) (el inspector de Opera funciona igual que este)
- [Safari inspector y explorador de estilos.](#)

El depurador de JavaScript

El depurador de JavaScript te permite observar el valor de las variables y establecer puntos de interrupción, lugares en tu código en los que deseas pausar la ejecución e identificar los problemas que impiden que tu código se ejecute correctamente.

The screenshot shows the Firefox Developer Tools interface. At the top, there's a browser-like header with tabs, back/forward buttons, and a URL bar showing "127.0.0.1:8080". Below the header, the main content area displays a "JavaScript Example" page with the title "Created for Debugger Article". A section titled "Here is a sample list:" contains the text "Buy eggs, milk, and bread" and "Order birthday cake". Below this, there's a form with a text input field containing "Order birthday cake" and a button labeled "Add Item". The bottom half of the screenshot shows the "Debugger" tab of the developer tools. On the left, the "Sources" panel lists files like "example.js", "(index)", and "JS example.js" (which is selected). The "Script" panel shows the code for "example.js" with line numbers 11 through 21. Line 18 is highlighted. On the right, the "Watch expressions" panel shows variables "ulList: ul#myList" and "listItems: (1) [...]". There's also a "Breakpoints" section and a "Pause on exceptions" checkbox.

Para llegar al depurador:

Firefox: Selecciona



► *Desarrollador Web* ► *Depurador* o presiona **Ctrl + Mayús + S** para abrir el depurador de JavaScript. Si ya estás viendo las herramientas, haz clic en la pestaña **Depurador**.

Chrome: Abre las herramientas para desarrolladores y luego selecciona la pestaña **Fuentes**. (Opera funciona de la misma manera).

Edge e Internet Explorer 11: presiona **F12** y luego **Ctrl + 3**, o si ya estás viendo las herramientas, haz clic en la pestaña Depurador.

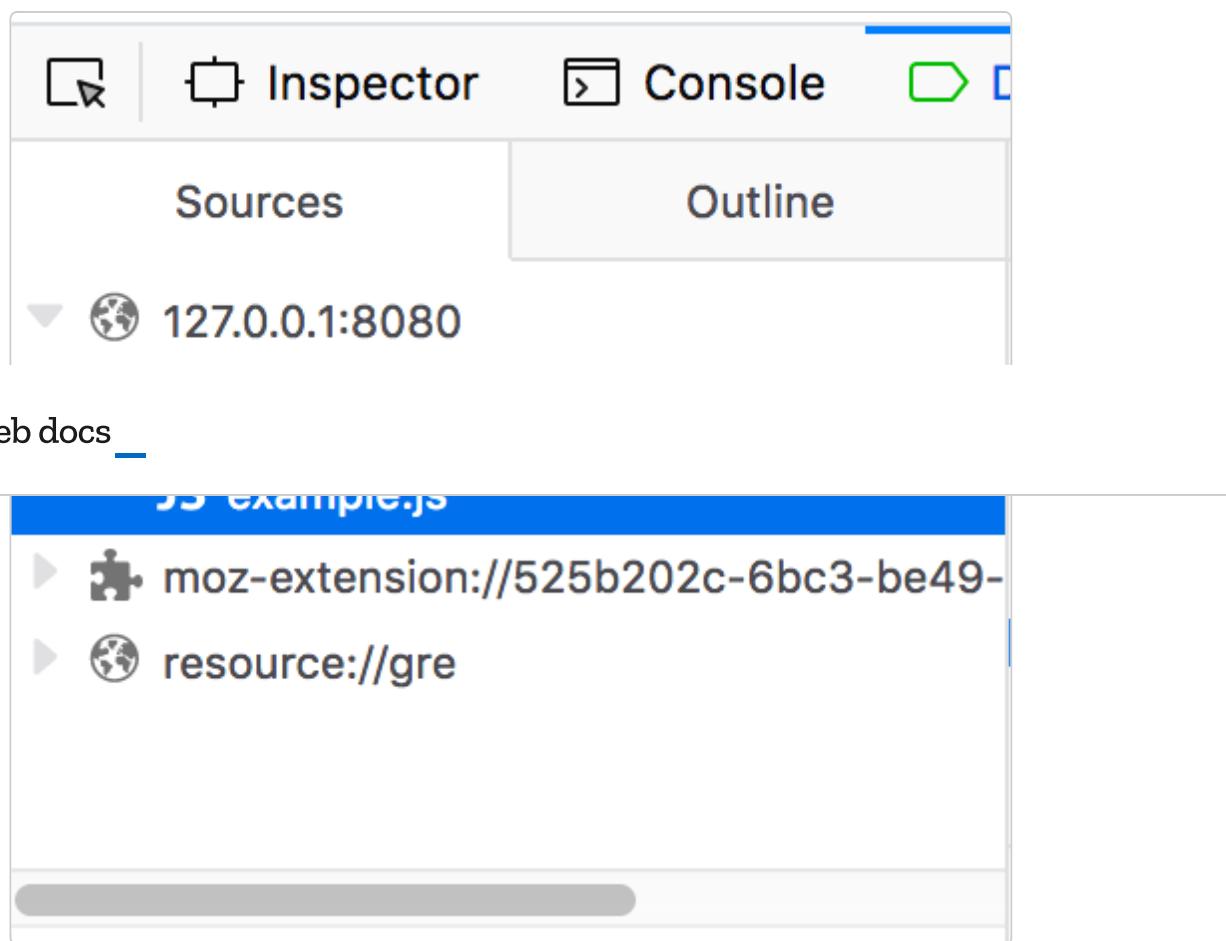
Safari: Abre las herramientas para desarrolladores y luego selecciona la pestaña Depurador.

Explorando el depurador

En Firefox hay tres paneles en el depurador de JavaScript.

Lista de archivos

El primer panel de la izquierda contiene la lista de archivos asociados con la página que estás depurando. Selecciona el archivo con el que deseas trabajar de esta lista. Haz clic en un archivo para seleccionarlo y ver su contenido en el panel central del depurador.



Código fuente

Establece puntos de interrupción donde deseas pausar la ejecución. En la siguiente imagen, el resultado del número 18 muestra que la línea tiene un punto de interrupción establecido.

The screenshot shows the Chrome DevTools Debugger panel. At the top, there are tabs for Debugger, Style Editor, Performance, and Network. Below the tabs, there's a file list with 'example.js x'. The code editor shows a script with several lines of code. Line 18, which contains the instruction `listItems.push(inputNewItem.value);`, has a blue arrow pointing to it, indicating it is the current line of execution. The code is as follows:

```
11 // Input new item value length > 0
12 // Create a new list item
13 var newItem = document.createElement("li")
14 newItem.appendChild(document.createTextNode(inputNewItem.value))
15
16 // Append it to myList
17 myList.appendChild(newItem);
18 listItems.push(inputNewItem.value);
19
20 // Finally clear the text box and set focus
21
```

At the bottom of the code editor, there are two buttons: a brace icon and an eye icon.

Ver expresiones y puntos de interrupción

El panel de la derecha muestra una lista de las expresiones en observación que has agregado y los puntos de interrupción que has establecido.

En la imagen, la primera sección, **Ver expresiones**, muestra que se ha agregado la variable `listItems`. Puedes expandir la lista para ver los valores del arreglo.

La siguiente sección, **Puntos de interrupción**, enumera los puntos de interrupción establecidos en la página. En `example.js`, se ha establecido un punto de interrupción en la instrucción `listItems.push(inputNewItem.value);`

Las dos últimas secciones solo aparecen cuando el código se está ejecutando.

La sección **Pila de llamadas** muestra qué código se ejecutó para llegar a la línea actual. Puedes ver que el código está en la función que maneja un clic del mouse y que el código está actualmente en pausa en el punto de interrupción.

La sección final, **Alcances**, muestra qué valores son visibles desde varios puntos dentro de tu código. Por ejemplo, en la siguiente imagen, puedes ver los objetos disponibles para el código en la función `addItemClick`.

Network >> ... X

▶ ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹

▼ Watch expressions C +
▶ listItems: (1) [...] X

▼ Breakpoints
 Pause on exceptions

example.js

listItems.push(inputNewItem) 18

▼ Call stack

addItemClick example.js: 18

onclick (index): 1

Paused on breakpoint

▼ Scopes

addItemClick
▶ <this>: Window
▶ arguments: Arguments
▶ newListItem: li
 accessKey: ""
 accessKeyLabel: ""
 assignedSlot: null
 ▶ attributes: NamedNodeMap []
 baseURI: "http://127.0.0.1:8080/"
 childElementCount: 0



Conocer más

Obtén más información sobre el depurador de JavaScript en diferentes navegadores:

- [Depurador de JavaScript en Firefox.](#)
- [Depurador de Microsoft Edge.](#)
- [Depurador de Chrome.](#)
- [Depurador de Safari.](#)

La consola de JavaScript

La consola de JavaScript es una herramienta increíblemente útil para depurar JavaScript que no funciona como se esperaba. Te permite ejecutar líneas de JavaScript en la página actualmente cargada en el navegador e informa los errores encontrados cuando el navegador intenta ejecutar tu código. Para acceder a la consola en cualquier navegador:

Si las herramientas para desarrolladores ya están abiertas, haz clic o presiona la pestaña Consola.

De lo contrario, Firefox te permite abrir la consola directamente usando **ctrl + Mayús + K** o usando el comando del menú: Menú



► Desarrollador web ► Consola web, o Herramientas ► Desarrollador web ► Consola web. En otro navegador, abre las herramientas para desarrolladores y luego haz clic en la pestaña Consola.

Esto te dará una ventana como la siguiente:

A screenshot of the Firefox Developer Tools interface, specifically the Console tab. The tabs at the top include Inspector, Console (which is selected), Debugger, Style Editor, Performance, Memory, Network, Storage, and more. The console output shows the following:

```
>> alert('Hello!');  
← undefined  
>> document.querySelector('html').style.backgroundColor = 'purple';  
← "purple"  
>> |
```

Para ver qué sucede, intenta ingresar los siguientes fragmentos de código en la consola uno por uno (y luego presiona Intro):

```
JS  
alert("hello!");
```

```
JS  
document.querySelector("html").style.backgroundColor = "purple";
```

```
JS  
const myWordmark = document.createElement("img");  
myWordmark.setAttribute(  
  "src",  
  "https://blog.mozilla.org/press/wp-content/themes/OneMozilla/img/mozilla-wordmark.png",  
);  
document.querySelector("h1").appendChild(myWordmark);
```

Ahora intenta ingresar las siguientes versiones incorrectas del código y ve lo que obtienes.

```
JS  
alert('hello!');
```

```
JS  
document.cheeseSelector("html").style.backgroundColor = "purple";
```

```
JS
```

```
const myWordmark = document.createElement("img");
myBanana.setAttribute(
  "src",
  "https://blog.mozilla.org/press/wp-content/themes/OneMozilla/img/mozilla-wordmark.png",
);
document.querySelector("h1").appendChild(myWordmark);
```

Comenzarás a ver el tipo de errores que devuelve el navegador. A menudo, estos errores son bastante crípticos, ¡pero debería ser bastante sencillo resolver estos problemas!

Conocer más

Obtén más información sobre la consola de JavaScript en diferentes navegadores:

- [Consola Web de Firefox.](#)
- [Consola de JavaScript Edge.](#)
- [Consola JavaScript de Chrome.](#) (el inspector de Opera funciona de la misma manera)
- [Consola en Safari.](#)

Ve también

- [Depurar HTML.](#)
- [Depurar CSS. \(en-US\)](#)

This page was last modified on 2 ago 2023 by [MDN contributors](#).

¿Qué es el Modelo de Objetos del Documento?

Redactores:

Philippe Le Hégaret, W3C

Lauren Wood, SoftQuad Software Inc. (for DOM Level 2)

Jonathan Robie, Texcel (for DOM Level 1)

Introducción

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones ([API](#)) para documentos válidos [HTML](#) y bien construidos [XML](#). Define la estructura lógica de los documentos y el modo en que se accede y manipula. En la especificación DOM, el término "documento" es utilizado en un sentido amplio - the term "document" is used in the broad sense - cada vez más XML es utilizado como un medio de representar muchas clases diferentes de información que puede ser almacenada en sistemas diversos, y mucha de esta información se vería, en términos tradicionales, más como datos que como documentos. Sin embargo, XML presenta estos datos como documentos, y se puede utilizar DOM para manejar estos datos.

Con el Modelo de Objetos del Documento, los programadores pueden construir documentos, navegar por su estructura, y añadir, modificar, o eliminar elementos y contenido. Se puede acceder a cualquier cosa que se encuentre en un documento HTML o XML, modificando, borrando o añadiendo utilizando el Modelo de Objetos del Documento, con algunas excepciones - en particular, aún no se han especificado [aplicaciones](#) DOM para los subconjuntos internos y externos de XML.

Como una especificación de W3C, un objetivo importante para el Modelo de Objetos del Documento es proporcionar un interfaz estándar de programación que puede ser utilizado en una amplia variedad de entornos y [aplicaciones](#). El DOM se diseña para ser utilizado en cualquier lenguaje de programación. Para proporcionar una especificación de las aplicaciones DOM precisa e independiente del lenguaje, hemos decidido definir las especificaciones en Grupo de Dirección de Objeto (GDO) (OMG -- Object Management Group) IDL [[OMG IDL](#)], según se define en la especificación CORBA 2.3.1 [[CORBA](#)]. Además de la especificación GDO (OMG) IDL, proporcionamos [correspondencias con los lenguajes](#) Java [[Java](#)] y ECMAScript [[ECMAScript](#)] (un lenguaje de scripts industrial basado en JavaScript [[JavaScript](#)] y JScript [[JScript](#)]). Por restricciones de correspondencias del lenguaje, se ha de aplicar un mapeado entre GDO (OMG) IDL y el lenguaje de programación utilizado. Por ejemplo, mientras que DOM utiliza atributos IDL en la definición de la interfaz, Java no permite que las aplicaciones contengan atributos:

```
// Ejemplo 1: Quitar el primer hijo de un elemento utilizando ECMAScript  
mySecondTrElement.removeChild(mySecondTrElement.firstChild);  
  
// Ejemplo 2: Quitar el primer hijo de un elemento utilizando Java  
mySecondTrElement.removeChild(mySecondTrElement.getFirstChild());
```

Nota: GDO (OMG) IDL se utiliza únicamente como un medio de especificar las aplicaciones independiente de la plataforma y del lenguaje. Se podría hacer utilizando otros IDLs ([\[COM\]](#), [\[Java IDL\]](#), [\[MIDL\]](#), ...). En general, los IDLs se diseñan para entornos de computación específicos. El Modelo de Objetos del Documento puede implementarse en cualquier entorno de computación, y no requiere las librerías de enlazado de objetos (object binding runtimes) generalmente asociadas con tales IDLs.

Lo que el Modelo de Objetos es

DOM es un API de programación para documentos. Está basado en una estructura de objeto muy parecida a la estructura del documento que modela. Por ejemplo, considere esta tabla, tomada de un documento XHTML:

```
<table>
  <tbody>
    <tr>
      <td>Shady Grove</td>
      <td>Aeolian</td>
    </tr>
    <tr>
      <td>Sobre el río, Charlie</td>
      <td>Dorian</td>
    </tr>
  </tbody>
</table>
```

Una representación gráfica DOM de la tabla del ejemplo, con espacios en blanco en el contenido del elemento (amenudo abusivamente llamado "espacios en blanco ignorables") eliminados, es:

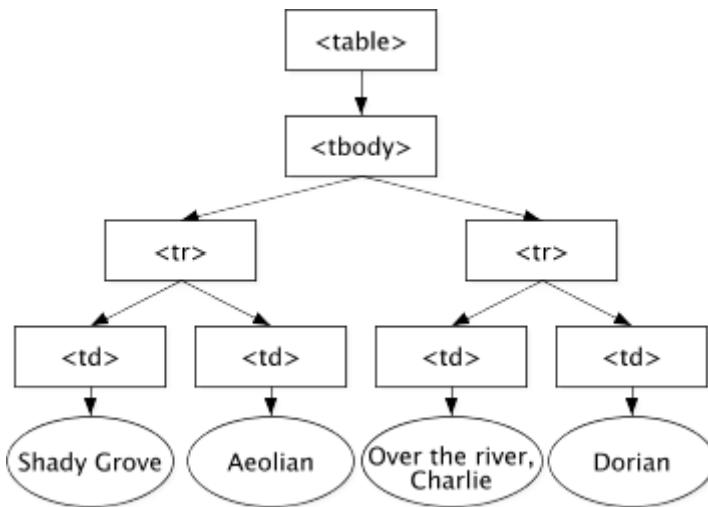


Figura: representación gráfica DOM de la tabla del ejemplo [[SVG 1.0 versión](#)]

Un ejemplo de manipulación de DOM utilizando ECMAScript podría ser:

```
// Tener acceso sobre el elemento tbody del elemento tabla
var myTbodyElement = myTableElement.firstChild;

// Tener acceso al segundo elemento tr
// La lista de hijos empieza en 0 (no en 1).
var mySecondTrElement = myTbodyElement.childNodes[1];
```

```

// Quitar su primir elemento td
mySecondTrElement.removeChild(mySecondTrElement.firstChild);

// Cambiar el contenido del texto del elemento restante td
mySecondTrElement.firstChild.firstChild.data = "Peter";

```

En DOM, los documentos tienen una estructura lógica que es muy parecida a un árbol; para ser más preciso, es más bien como un "bosque" o una "arboleda", que puede contener más de un árbol. Cada documento contiene cero o un nodo doctype, uno nodo de elemento de documento, y cero o más comentarios o instrucciones de tratamiento; el elemento del documento sirve como la raíz del árbol para el documento. Sin embargo, DOM no especifica que los documentos deban ser *implementados* como un árbol o un bosque, ni tampoco especifica como deben implementarse las relaciones entre los objetos. DOM es un modelo lógico que puede implementarse de cualquier manera que sea conveniente. En esta especificación, usamos el término *modelo de estructura* para describir la representación en forma de árbol de un documento. También utilizamos el termino "árbol" cuando refiriendose al arreglo de aquellos artículos de información que pueden ser alcanzados utilizando métodos "exploració por árbol"; (esto no incluye atributos). Una propiedad importante de los modelos de estructura de DOM es su *isomorfismo estructural*: si dos implementaciones cualesquiera del Modelo de Objetos del Documento se usan para crear una representació n del mismo documento, ambas crearán el mismo modelo de estructura, en concordancia con la Información XML Establecida [[Información XML Establecida](#)].

Nota: Puede haber variaciones según el programa de análisis utilizado para contruir el DOM. Por ejemplo, DOM puede no contener espacios en blanco en el elemento contenido si el análisis los descarta.

El nombre "Modelo de Objetos del Documento" se eligio porque es un "[modelo de objeto](#)" en el sentido tradicional del diseño orientado a objetos: los documentos se modelizan utilizando objetos, y el modelo comprende no solamente la estructura de un documento, sino también el comportamiento de un documento y los objetos de los cuales se compone. En otras palabras, los nodos del diagrama anterior no representan una estructura de datos, representan objetos, los cuales pueden tener funciones e indentidad. Como modelo de objeto, DOM identifica:

- las aplicaciones y objetos utilizados para representar y manipular el documento
- la semantica de estas aplicaciones y objetos - incluyendo comportamientos y atributos
- las relaciones y colaboraciones entre estas aplicaciones y objetos

Tradicionalmente, la estructura de documentos SGML se ha representado mediante un [modelo de datos](#) abstractos, no por un modelo de objeto. en un [modelo de datos](#) abstracto, el modelo se centra en los datos. En los lenguajes de programación orientados a objetos, los datos se encapsulan en objetos que ocultan los datos, protegiéndolos de su manipulación directa desde el exterior. Las funciones asociadas con estos objetos determinan como pueden manipularse los objetos, y son parte del modelo de objeto.

Lo que el Modelo de Objetos del Documento no es

Esta secció es diseñada para dar un entendimiento más preciso del DOM distinguiendolo de otros sistemas que aparentemente pueden resultar similares a él.

- El Modelo de Objetos del Documento no es una especificación binaria. Los programas DOM escritos en el mismo lenguaje serán compatibles entre

plataformas a nivel de código fuente, pero DOM no define ninguna forma de interoperabilidad binaria.

- El Modelo de Objetos del Documento no es una manera de ofrecer objetos persistentes para XML o HTML, DOM especifica los documentos como XML y HTML son representados como objetos, de modo que pueden ser utilizados por programas orientados a objetos.
- El Modelo de Objetos del Documento no es un conjunto de estructuras de datos, es un [modelo de objeto](#) que especifica aplicaciones. Aunque este documento contiene diagramas que muestran relaciones padres/hijos, estas son relaciones lógicas definidas por la aplicación de programación, no representaciones de ninguna estructura interna de datos particular.
- El Modelo de Objetos del Documento no define qué información en un documento es relevante o qué información en un documento es estructurado. Para XML, esta es especificada por el Conjunto de Información XML [[Conjunto de Información XML](#)]. DOM es simplemente un [API](#) de este conjunto de información.
- El Modelo de Objetos del Documento, a pesar de su nombre, no es un competidor del Modelo de Objetos de Componentes (MOC (Component Object Model -- COM) [[MOC \(COM\)](#)]). MOC (COM), al igual que CORBA, es una manera independiente de lenguaje de especificar aplicaciones y objetos diseñado para manipular documentos HTML y XML. DOM puede implementarse utilizando sistemas independientes del lenguaje como MOC (COM) o CORBA; también se puede implementar usando enlaces específicos con lenguajes, como los especificados en este documento para Java o ECMAScript.

De donde viene el Modelo de Objetos del Documento

El DOM se originó como una especificación para permitir que los scripts de JavaScript y los programas Java fueran portables entre los navegadores Web. El "HTML Dinámico" fue el antecesor inmediato del Modelo de Objetos del Documento, y originalmente se pensaba en él principalmente en términos de navegadores. Sin embargo, cuando se formó el Grupo de Trabajo DOM en W3C, también se unieron a él compañías de otros ámbitos, incluyendo redactores y depositos de documentos HTML o XML. Varios de estos redactores han trabajado con SGML antes de ser desarrollado XML; como resultado de ello, DOM ha recibido influencias de los "Bosques" y del estándar HyTime. Algunas de estas compañías también habían desarrollado sus propios modelos de objetos para documentos a fin de proporcionar un API para los editores o los archivos de documentos SGML/XML, y estos modelos de objetos también han influido en el DOM.

Las entidades y el núcleo del DOM

En las aplicaciones fundamentales del DOM, no hay objetos que representen entidades. Las referencias numéricas de caracteres y las referencias a entidades predefinidas en HTML y en XML, son reemplazadas por el carácter individual que constituye la sustitución de la entidad. Por ejemplo, en:

```
<p>Esto es un perro & un gato</p>
```

El "&" será reemplazado por el carácter "&", y el texto del elemento P formará una única secuencia continua de caracteres. Debido a que las referencias numéricas de caracteres y las entidades predefinidas no son reconocidas como tales en las secciones CDATA, o en los elementos SCRIPT y STYLE de HTML, no son reemplazadas por el carácter individual al que aparentemente se refieren. Si el ejemplo anterior estuviera contenido en una sección CDATA, el "&" no sería reemplazado por "&"; y el <p> tampoco sería reemplazado como etiqueta inicial. La representación de entidades

generales, tanto internas como externas, está definida dentro de las aplicaciones extendidas (XML) del [Núcleo del Modelo de Objetos del Documento](#).

Nota: Cuando una representación DOM de un documento es serializada como texto XML o HTML, las aplicaciones necesitarán comprobar cada carácter de los datos del texto para ver si necesita ser convertido en una secuencia de escape utilizando una entidad numérica o predefinida. De lo contrario se podría obtener un HTML o XML no válido. Además, las [implementaciones](#) deberían ser conscientes del hecho de que la serialización en una codificación de caracteres ("charset") que no cubra completamente la ISO 10646 puede fallar si hay caracteres en el código o en las secciones CDATA que no estén presentes en esa codificación.

Arquitectura del DOM

La especificación del DOM proporciona un conjunto de APIs que forman la API del DOM. Cada especificación del DOM define uno o mas módulos y cada módulo esta asociado con un nombre de funcionalidad. Por ejemplo, la especificación del Núcleo de DOM (esta especificación) define dos módulos:

- El módulo Núcleo, el cual contiene las aplicaciones fundamentales que deben ser implementadas por todas las aplicaciones DOM conformadas, está asociado con el funcionalidad "Core";
- El módulo XML, el cual contiene las aplicaciones que deben ser implementadas por todos las aplicaciones XML 1.0 [[XML 1.0](#)] (y superiores) DOM conformadas, está asociado con el funcionalidad "XML".

La siguiente representación contiene todos los módulos de DOM, representado utilizando sus nombres de funcionalidades, definidos a lo largo de las especificaciones DOM:

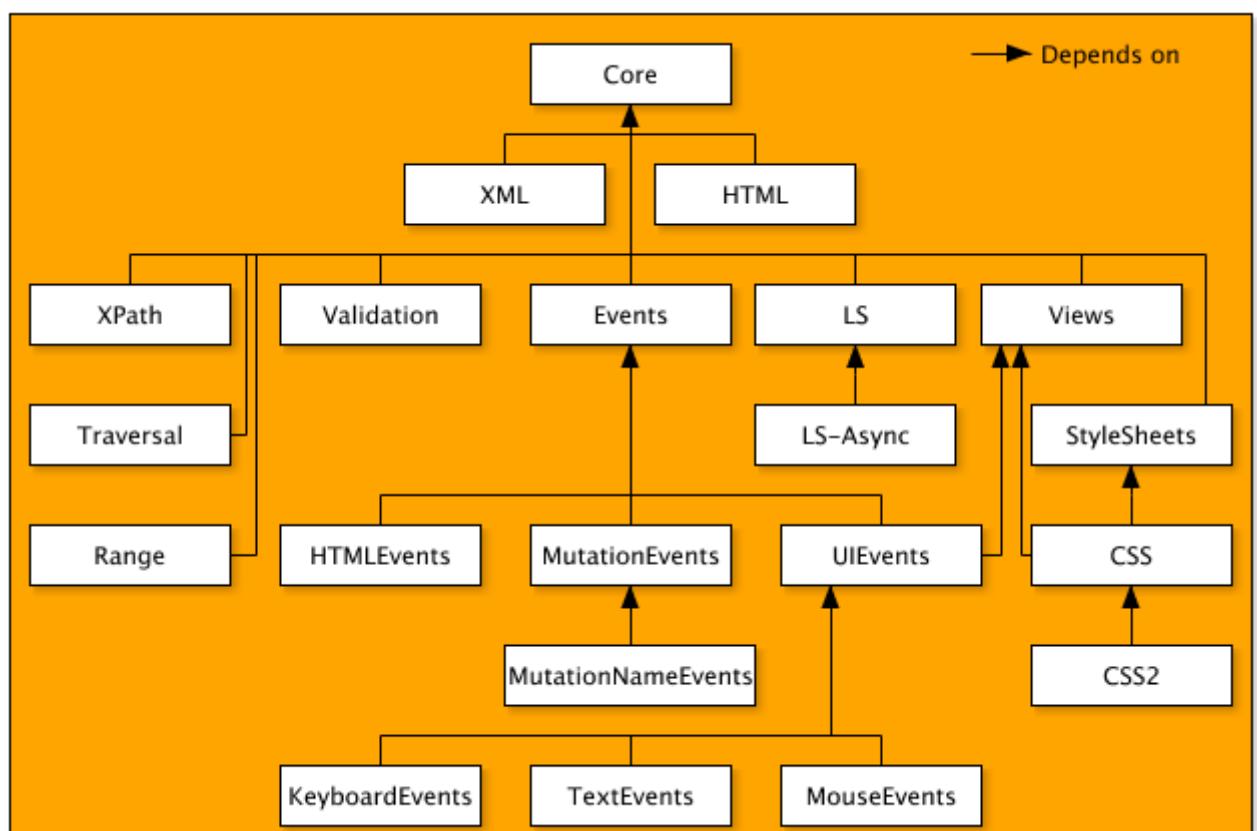


Figura: Una visión de la arquitectura de DOM [[SVG 1.0 versión](#)]

Una implementación DOM puede entonces implementar un (es decir, solo el módulo Núcleo) o más módulos dependiendo del uso de la aplicación. Una aplicación de usuario para Web es muy probable que implemente el módulo "MouseEvents" (eventos del ratón), mientras que una aplicación del lado del servidor no tendría que utilizar este módulo y probablemente no lo implementaría.

Conformidad

Esta sección explica los diferentes niveles de conformidad del Nivel 3 de DOM. DOM Nivel 3 consiste en 16 módulos. Es posible ser conformes de DOM Nivel 3, o al módulo de DOM Nivel 3.

Una aplicación es conforme a DOM Nivel 3 si soporta el módulo Núcleo definido en este documento conformant if it supports the Core (vea [Aplicaciones Fundamentales: Módulo Núcleo](#)). Una aplicación es conforme al módulo DOM Nivel 3 si soporta todas las aplicaciones para ese módulo y la semántica asociada.

Aquí está la lista completa de los módulos de DOM Nivel 3.0 y las funcionalidades utilizadas por ellos. Los nombres de las funcionalidades son insensibles a mayúsculas y minúsculas (case-insensitive).

Módulo Núcleo

Define la funcionalidad "[Core](#)".

Módulo XML

Define la funcionalidad "[XML](#)".

Módulo Eventos

Define la funcionalidad "[Events](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos de la Aplicación del Usuario

Define la funcionalidad "[UIEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos del Ratón

Define la funcionalidad "[MouseEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos del Texto

Define la funcionalidad "[TextEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos del Teclado

Define la funcionalidad "[KeyboardEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos de Mutación

Define la funcionalidad "[MutationEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos de nombre Mutación

Define la funcionalidad "[MutationNameEvents](#)" en [[Eventos de DOM Nivel 3](#)].

Módulo de Eventos HTML

Define la funcionalidad "[HTMLEvents](#)" en [[Eventos de DOM Level 3](#)].

Módulo Cargar y Guardar

Define la funcionalidad "[LS](#)" en [[Cargar y Guardar de DOM Nivel 3](#)].

Módulo de carga Asincronica

Define la funcionalidad "[LS-Async](#)" en [[Cargar y Guardar de DOM Nivel 3](#)].

Módulo de Validación

Define la funcionalidad "[Validation](#)" en [[Validación de DOM Nivel 3](#)].

Módulo XPath

Define la funcionalidad "[XPath](#)" en [[XPath de DOM Nivel 3](#)].

Una implementación no debe devolver verdadero al `DOMImplementation.hasFeature(funcionalidad, versión)` método de una aplicación `DOMImplementation` para ese funcionalidad a menos que la implementación tenga conformidad con ese módulo. El número de `versión` para todos los de las funcionalidades utilizados en DOM Nivel 3.0 es "3.0".

DOM especifica aplicaciones que pueden utilizarse para manipular documento XML o HTML. Es importante darse cuenta de que estas aplicaciones son una abstracción - comparables a las "clases de base abstractas" en C++, constituyen un medio para especificar una forma de acceder y manipular la representación interna que una aplicación hace de un documento. Las aplicaciones no implican una implementación particular concreta. Cada aplicación DOM es libre de mantener los documentos según una representación cualquiera, siempre y cuando soporte las aplicaciones mostradas en esta especificación. Algunas implementaciones del DOM serán programas existentes que usen las interfaces del DOM para acceder a programas escritos mucho antes de que existiera la especificación DOM. Por tanto, DOM se ha diseñado para evitar dependencias de implementación; en particular,

1. Los atributos definidos en el IDL no implican objetos concretos que deban tener miembros de datos específicos - en las correspondencias con cada lenguaje, se transforman en pares de funciones get()/set(), no en un miembro de datos. Los atributos de solo lectura tendrán una función get() en la correspondencia con el lenguaje.
2. Las aplicaciones DOM pueden proporcionar interfaces adicionales y objetos que no se encuentren en esta especificación y seguir siendo consideradas como conformes con el DOM.
3. Debido a que especificamos interfaces y no los objetos reales que deben ser creados, DOM no puede saber a qué constructores llamará una implementación. En general, los usuarios de DOM llamarán a métodos createX() de la clase Document para crear estructuras del documento, y las implementaciones DOM crearán sus propias representaciones internas de dichas estructuras en sus implementaciones de las funciones createX().

Las interfaces de Nivel 2 fueron aplicadas para proveer funcionalidad tanto al Nivel 2 como Nivel 3.

Las implementaciones DOM en otros lenguajes como Java o ECMAScript pueden elegir los enlaces que sean apropiados y naturales para sus lenguajes y las bibliotecas necesarias (run time environment). Por ejemplo, algún sistema puede necesitar crear una clase Document3 que hereda de una clase Document y contiene los nuevos métodos y atributos.

DOM Nivel 3 no especifica mecanismos de multiinserciones.

Document Object Model

Document Object Model o **DOM** ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de plataforma, o API de programación para documentos HTML, que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML, XML y SVG,^{nota 1} un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.¹ A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos mencionados, que es para lo que se diseñó principalmente.

De esta manera el DOM permite acceso dinámico a través de la programación para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (JavaScript). De hecho DOM como interfaz de plataforma multilingüe (cross-language API) para acceder y modificar documentos XML es independiente del lenguaje, de esta manera una variedad de lenguajes de programación utilizan DOM cotidianamente —como C++ y Python entre otros— a medida que interactúan con XML para diversas tareas.^{2 3}

La estandarización principal del DOM corre a cargo del World Wide Web Consortium (W3C).

Historia

El DOM nació como un conjunto de objetos que representan a un documento HTML, y que actúan como una interfaz entre JavaScript y el propio documento, su origen es Netscape Navigator de la compañía Netscape Communications. Pensado para detectar eventos generados por el usuario y modificar al documento HTML, a su primera generación se le conoce como DOM Nivel 0 o legado (DOM Level 0 o Legacy DOM).

Surgió originalmente bajo la influencia de al menos dos desarrollos que han dado forma significativa al mundo de la informática en el pasado reciente. Ambos se basan en la necesidad de poder acceder de manera fácil y uniforme a los datos estructurados en los documentos HTML y XML.⁴

A mediados de la década de 1990, a medida que aumentaba la popularidad de la World Wide Web, se inventó el lenguaje de secuencias de comandos JavaScript y, desde entonces, los navegadores web incluyen intérpretes que ejecutan dichas secuencias de comandos.

Internet Explorer de Microsoft le siguió a Netscape Navigator, estos dos navegadores incluían soporte para HTML dinámico (DHTML)⁵, que requería extensiones que entonces ofrecía el DOM rudimentario de la época, permitiendo a los desarrolladores web crear páginas con interactividad



Jerarquía de DOM.

del lado del cliente, así el documento ahora podría manipularse a través del DOM; no obstante, el mismo documento no fue representado de la misma manera por los dos navegadores.

Los intérpretes de JavaScript o motor para JavaScript, fueron ampliamente implementados por ambos navegadores. Microsoft en aquel momento decidió implementar, para el entonces Internet Explorer, JScript y VBScript, lo cual dio origen a la llamada "guerras de navegadores" de finales de la década de 1990 entre Netscape Navigator e Internet Explorer.

Entonces JavaScript definió formas rudimentarias de acceder al documento HTML y manejar eventos. Más tarde, diferentes fabricantes de navegadores inventaron diferentes modelos de HTML dinámico (DHTML) que permitían una modificación más extensa de la estructura y apariencia del documento mientras se mostraba en el navegador. Sin embargo, estas diferencias hicieron que el trabajo de los desarrolladores web que querían usar HTML dinámico fuera extremadamente tedioso, ya que a menudo se veían prácticamente obligados a escribir una versión separada para cada navegador. Los primeros estándares DOM del W3C (Consorcio WWW) son, por lo tanto, intentos de fusionar, estandarizar y finalmente reemplazar las diversas técnicas patentadas de JavaScript y DHTML que surgieron durante la guerra de los navegadores. Esto ha tenido tanto éxito que el DOM ahora juega un papel central en la programación de JavaScript.⁶

Para septiembre del 2008, nuevos navegadores aparecen gradualmente en escena bajo el proyecto Chromium, cuyo código base es ampliamente utilizado, tanto por el navegador Chrome como por el más reciente Microsoft Edge. Netscape Navigator desaparece como navegador comercial⁷ dejando como una especie de sucesor a Mozilla Firefox.

Estos últimos acontecimientos y la intervención del Consorcio WWW (W3C), marcaron una nueva tendencia en continua evolución, desapareciendo gradualmente los problemas de incompatibilidad con la adopción del DOM estandarizado por el W3C. El DOM (nivel 4) ahora está incorporado en el estándar HTML 5, fue lanzado en diciembre de 2015 y su última actualización fue en diciembre de 2020.⁶

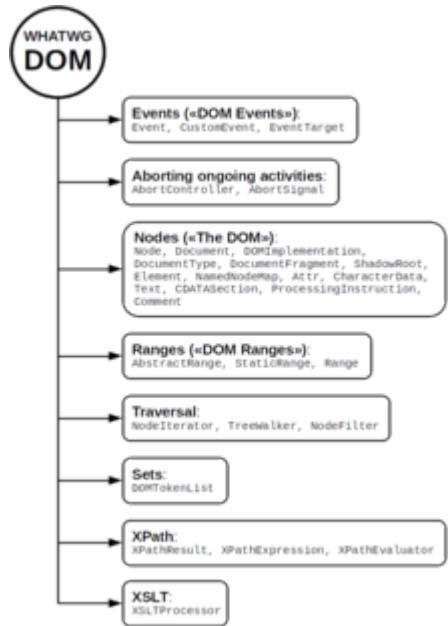
En la actualidad DOM ha permeado todos los lenguajes de programación, independiente del lenguaje, una variedad de lenguajes de programación utilizan DOM cotidianamente, como C++ y Python entre otros, a medida que interactúan con aplicaciones de XML para diversas tareas.^{2 3}

Establecer referencias a objetos

El DOM define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento.

Puede utilizarse cualquier lenguaje de programación adecuado para el diseño web. En el caso de JavaScript, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existe más de un objeto del mismo tipo en un documento web, estos se organizan en un vector.

Es posible asignarle una identificación a un objeto, y luego usarlo para hacer referencia a este, por ejemplo:



Esquema de DOM hecho por WHATWG (Web Hypertext Application Technology Working Group)

```
<!- documento html -->
<div id="IdiomasWiki">
  <li>Deutsch</li>
  <li>English</li>
  <li>Français</li>
</div>
```

Para hacer referencia a este elemento se puede usar la función `getElementById`

```
// JavaScript
document.getElementById("IdiomasWiki")
```

Y realizar alguna operación sobre el mismo, en este caso agregamos un nuevo elemento:

```
document.getElementById("IdiomasWiki").innerHTML += "<li>Português</li>"
```

Manipular las propiedades y funciones de objetos

Los objetos computacionales, de la misma forma que cualquier objeto de la vida real, tienen propiedades. Algunos ejemplos de propiedades de objetos de la vida real son dimensiones, color y peso.

En la mayoría de los objetos computacionales algunas propiedades se pueden determinar de la siguiente manera:

```
Objeto.propiedad = valor;
//por ejemplo para el objeto «Vaso»

Vaso.color = rojo;
```

La manipulación de objetos sigue los mismos principios que en el lenguaje de programación que se esté utilizando. Una de las características de estos objetos es la función para la cual están diseñados, de hecho en la mayoría de ocasiones tienen más de una función. En JavaScript, muchas funciones para cada uno de los objetos, incluyendo el navegador y la ventana que lo contiene, han sido definidas previamente; adicionalmente, el usuario puede definir funciones de acuerdo a sus necesidades, por ejemplo el código:

```
function comeLaLetraA(Texto){
  var TextoNuevo = "";
  while(letras in Texto){
    //Lee la siguiente letra
    //si esta Letra no es «a» añádela al nuevo texto
  }
  return TextoNuevo;
}
```

Añade una nueva función al documento utilizado para crear una página web.

Eventos

Un evento desde el punto de vista computacional ocurre cuando cambia alguna situación en la computadora como, por ejemplo, la posición del ratón, la pulsación de alguna tecla, los contenidos de alguna de las memorias, la condición de la pantalla, etc. En la creación de páginas web estos eventos representan la interacción del usuario con la computadora.

Cuando alguno de estos eventos ocurre, como por ejemplo la presión de algún botón del ratón, es deseable que la computadora responda de alguna manera. Esta es la razón por la que existen *event handlers* ('encargados de manejar eventos') los cuales son objetos que responden a eventos. Una manera de añadir eventos en el DOM utilizando JavaScript es:

```
<element onevent="script">....</element>
```

Por ejemplo:

```
<div id="midivision" onClick="javascript:miFuncion('bar');">  
Aquí va otro texto  
</div>
```

Siendo el indicador de pseudo-protocolo *javascript*: un agregado opcional, y solo usado por Microsoft Internet Explorer, donde podía configurarse por defecto su lenguaje alternativo VBScript.⁸

Otra forma de manipular eventos en JavaScript, al crear páginas web, es tratándolos como propiedades de los elementos que forman la página, por ejemplo:

```
objeto.evento = funcion;  
//Por ejemplo:  
document.getElementById("midivision").onclick = hazAlgo;
```

En DOM se considera que un evento se origina en el exterior de la página web y se propaga de alguna manera hasta los elementos internos de la página. Un posible ejemplo de esta propagación es:

```
EVENTO → Ventana → Document → HTML → BODY → DIV → DESTINO  
RESPUESTA → DIV → BODY → HTML → Document → Ventana → EVENTO
```

Siguiendo esta idea, se establecen tres etapas: *captura*, la cual se da cuando el evento se está trasladando a su destino. *Blanco*, que ocurre cuando llega al blanco, o sea que llega a su destino. Este destino es el objeto en el cual se va a crear una reacción a este evento. Finalmente la etapa de *burbujeo* que ocurre cuando el evento «regresa» a su posición original.

Ciertos objetos pueden estar pendientes de ciertos eventos. Para hacer esto el objeto añade un «oyente de eventos» con la función addEventListener. Cuando el evento ocurra, alguna función determinada se lleva a cabo. En este proceso se indica en qué momento la función se lleva a cabo, ya sea en la etapa de *captura* o en la etapa de *burbujeo*. Este momento se indica con la palabra *true* si debe ocurrir en la etapa de captura o *false* si debe ocurrir en la etapa de burbujeo. En JavaScript se escribe de la siguiente manera:

```
objeto.addEventListener(evento, funcion, momento);  
//por ejemplo:  
document.getElementById("mydivision").addEventListener("click", hazAlgo, false);
```

Véase también

- HyperText Markup Language (HTML)
- Extensible Markup Language (XML)

- JavaScript (JS)

Notas

1. Tanto XHTML como SVG son aplicaciones de XML —un formato de lenguaje de marcado extensible— así XHTML es esencialmente HTML expresado como XML válido, mientras que SVG es un formato de imagen vectorial basado en XML para definir gráficos bidimensionales.

Referencias

1. «Document Object Model (DOM)» (<http://www.w3.org/DOM/#what>). <http://www.w3.org/>: W3C. «*Document Object Model* es una plataforma, y una interfaz independiente del lenguaje, que permite a programas y scripts, acceder y actualizar el contenido, la estructura y el estilo de documentos en forma dinámica. »
2. «The Document Object Model API» (<https://docs.python.org/3/library/xml.dom.html>). Python Software Foundation. «The Python Standard Library: Structured Markup Processing Tools. »
3. «Program with DOM in C-C++» ([https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms759192\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms759192(v=vs.85))). Documentación técnica de Microsoft. «This tutorial is intended for C/C++ developers interested in writing XML applications using the DOM. »
4. Weiss-Engelsberger, Roman (2016). *Diseño web universal compatible con W3C*. AV AkademikerVerlag. p. 521. ISBN 9783330506824.
5. «Working with DHTML and the DHTML Object Model» ([https://docs.microsoft.com/en-us/previous-versions/aa651135\(v=vs.71\)](https://docs.microsoft.com/en-us/previous-versions/aa651135(v=vs.71))). Documentación técnica de Microsoft. «DHTML uses standard HTML tags to render and manipulate content on a page. »
6. Wang, Paul S. (2012). *Dynamic Web Programming and HTML5* (<https://www.taylorfrancis.com/books/mono/10.1201/b13928/dynamic-web-programming-html5-paul-wang>). Chapman & Hall CRC. p. 664. ISBN 9780429169625.
7. «Netscape desaparece y pasa el testigo a su «heredero», el navegador Firefox» (https://www.abc.es/tecnologia/redes/abci-netscape-desaparece-y-pasa-testigo-heredero-navegador-firefox-200803030300-1641691782143_noticia.html). Diario ABC, S.L. «*Netscape*, el programa gracias al que millones de personas tuvieron sus primeras experiencias en internet está condenado sin remedio a desaparecer. »
8. «The useless javascript: pseudo-protocol - Crisp's blog - Tweakblogs - Tweakers» (<https://crisp.tweakblogs.net/blog/the-useless-javascript-pseudo-protocol.html>). crisp.tweakblogs.net. Consultado el 28 de julio de 2021.

Bibliografía

- Wang, Paul S (2013). «*Dynamic Web Programming and HTML5*». New York: Chapman and Hall/CRC. ISBN 9780429169625.
- Weiss-Engelsberger, Roman (2016). «*Diseño web universal compatible con W3C*». Berlín: AV AkademikerVerlag. ISBN 9783330506824.

Enlaces externos

- DOM según el W3C (<https://www.w3.org/DOM/>)
- DOM según Mozilla (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)
- Especificación de DOM Level 1 (<http://html.conclase.net/w3c/dom1-es/cover.html>) (en español)

■