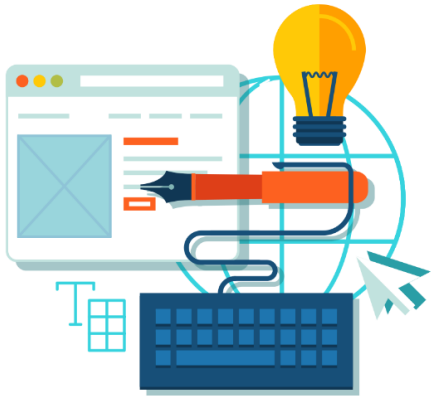


Diseño Web Responsive

UD2: Diseño Web Responsive

Índice

Introducción

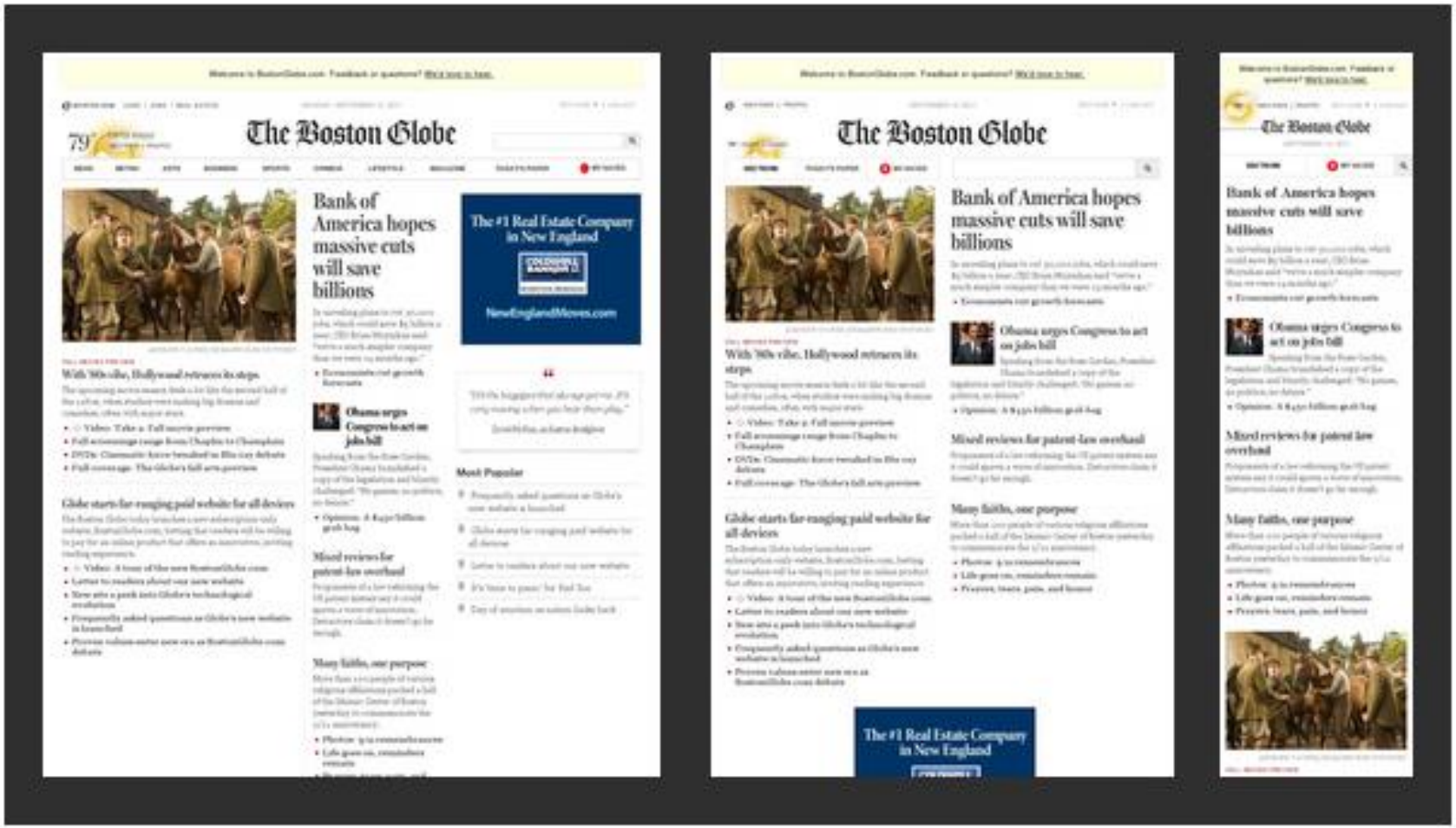
UD2: Diseño Web Responsive

¿Un diseño por dispositivo?

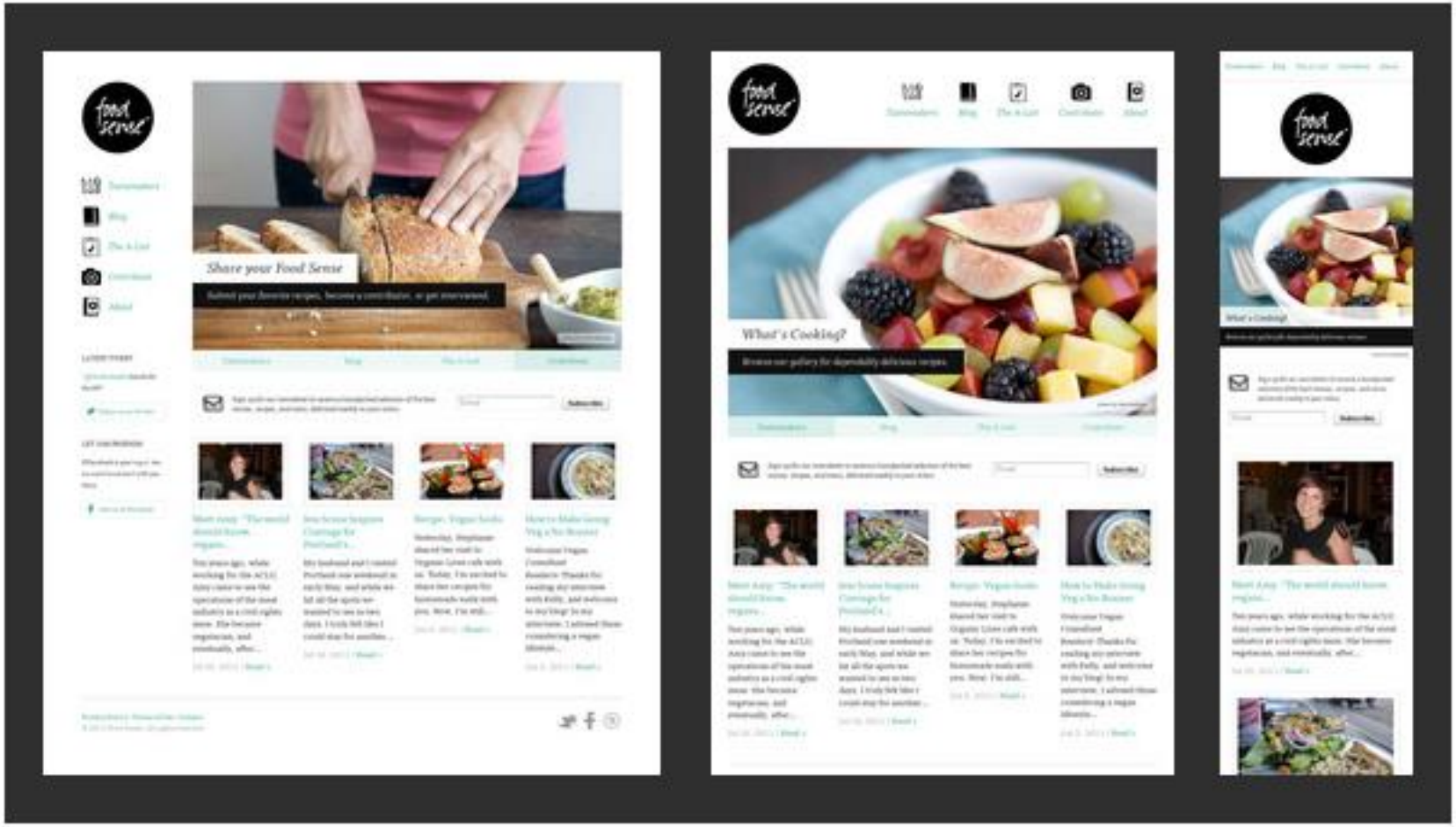
Responsive Web Design

- Consiste en la adaptación de un sitio Web al dispositivo de navegación (**diseño adaptativo**).
  - Apoyado por la tecnología de media queries
  - Busca lograr una única versión de la Web sensible a todas las resoluciones de pantalla.
- Es la tendencia actual en desarrollo
  - *Adaptive Web Design* es otra tendencia que opta por versiones separadas para cada tipo de dispositivo.
- Ejemplo de buenas prácticas

Ejemplo: The Boston Globe



Ejemplo: Food Sense



Valoración

- Ventajas
  - Sitios más económicos
  - Menor tiempo de mantenimiento
  - Se aprovecha óptimamente el tamaño de las pantallas
- Inconvenientes
  - No sólo se deben llenar los espacios, sino transformar diseño y funcionalidad

Ingredientes

- La base del diseño *responsive* es la siguiente:
  - Media queries (CSS3)
  - Estructura flexible basada en rejilla (grid)
  - Imágenes fluidas

Media queries

UD2: Diseño Web Responsive

Atributo media (CSS2)

- CSS2 permite hojas de estilo para medios específicos como impresoras, pantalla de ordenador o móviles.
- Atributo @media
  - Valores más usados: screen, **print**, handheld

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 13px }
}
@media screen, print {
  body { line-height: 1.2 }
}
```

Hojas de estilo específicas

- Problema: La variedad de dispositivos y resoluciones hace que sea imposible generalizar.
- Resoluciones más comunes

Dispositivos y resoluciones

- Podríamos pensar en mostrar páginas dependiendo de la resolución
- Pero, los dispositivos actuales usan resoluciones similares a las usadas en PC

Dispositivo	Resolución
iPhone 6	750 x 1334
Samsung Galaxy S5	1080 x 1920
LG Nexus 5	1080 x 1920
Microsoft Lumia 1020	768 x 1280
iPad Pro	2048 x 2732
Samsung Nexus 10	1600 x 2560
Microsoft Surface Pro 3	1440 x 2160

Pixel ratio

- Habitualmente distinguimos entre una **resolución física** (real en la pantalla) de una **resolución lógica** (que va a interpretar el CSS)
  - La resolución lógica se mide en **píxeles independientes de dispositivo** (dip), aunque habitualmente hablamos de píxeles.
- Evita que veamos las páginas en pantallas pequeñas de alta resolución igual que en pantallas grandes de resolución similar.
- A la relación entre resolución física y la resolución lógica la llamamos **pixel ratio.**
  - Cada dispositivo establece el valor que cree conveniente.
- Emulación con Chrome

Dispositivos y resoluciones (2)

Dispositivo	Resolución (física)	Resolución (CSS)	Pixel ratio
iPhone 6	750 x 1334	375 x 667	2
Samsung Galaxy S5	1080 x 1920	360 x 640	3
LG Nexus 5	1080 x 1920	360 x 640	3
Microsoft Lumia 1020	768 x 1280	320 x 480	2.4
iPad Pro	2048 x 2732	1024 x 1366	2
Samsung Nexus 10	1600 x 2560	800 x 1280	2
Microsoft Surface Pro 3	1440 x 2160	1024 x 1440	1.5 / 1.4

Además..

- Es un concepto que no sólo se aplica a pantallas pequeñas, también a pantallas grandes como las Retina de los Mac Book Pro

Viewport

- Es el área de una página Web visible por el usuario
- Podemos definir una etiqueta `meta` para el viewport para establecer el ancho, alto y escala.
  - No estándar pero soportada por la mayoría de navegadores
- Dos opciones:
  - Número fijo de píxeles
  - Ancho y alto del dispositivo (caculados dinámicamente)

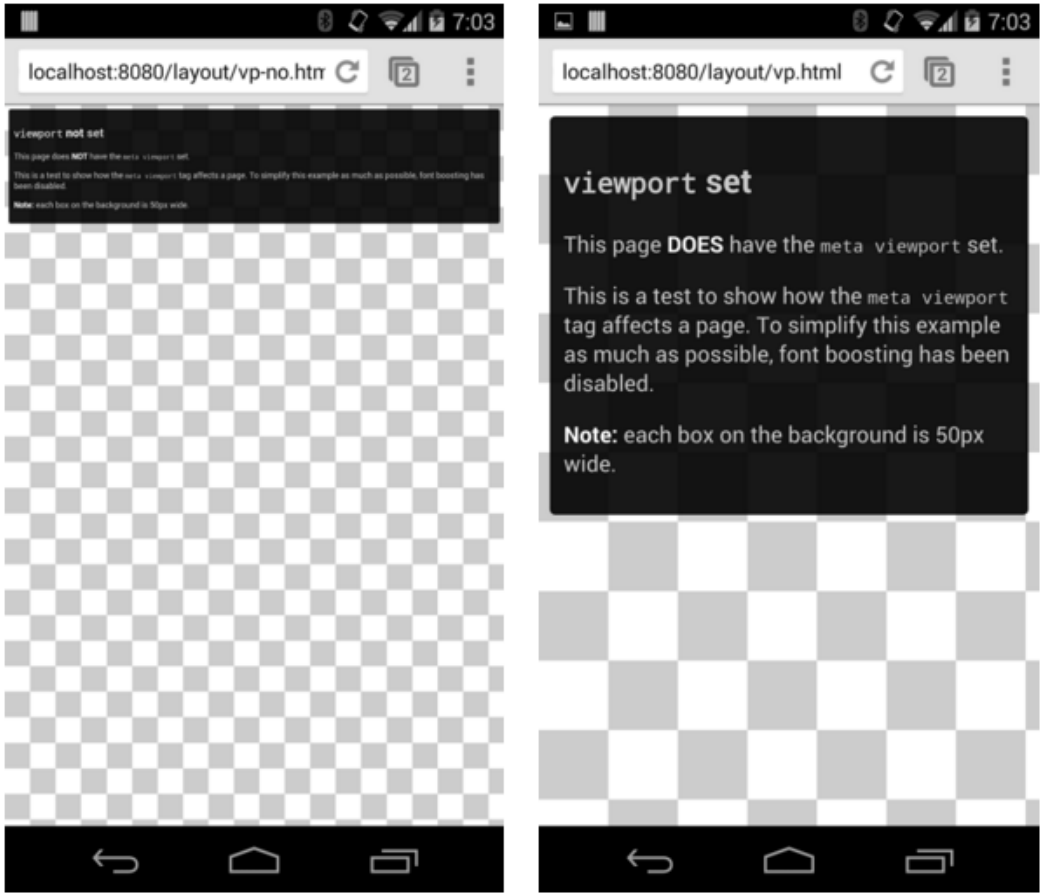
Viewport: Ejemplo

- **width**: Controla el ancho del área de visualización
  - **device-width** es un ancho del 100%
- **initial-scale**: Zoom inicial al cargar la página.

```
<meta name="viewport"
      content="width=device-width,
               initial-scale=1.0">
```

El uso de ambas combinadas **evita** que el navegador (**móvil**) haga zoom (out) automáticamente para adaptar al ancho de pantalla.

Uso de `viewport`: Comparación



Otros atributos de viewport

- `minimum-scale`
  - Escala mínima
- `maximum-scale`
  - Escala máxima
- `user-scalable`
  - Determina si el usuario puede hacer zoom

Peligrosos!

Otros atributos de viewport: Ejemplo

```
<meta name="viewport"
      content="700px,
          minimum-scale=1.0,
          maximum-scale=1.0,
          user-scalabe=no">
```

Página de ancho fijo donde no puedo hacer zoom ☹

Media Queries

- Permite consultar aspectos acerca del dispositivo y aplicar reglas CSS en función de los mismos.
- Entre los aspectos que podemos consultar están:
  - Tipo de medio
  - Resolución del dispositivo
  - Orientación del dispositivo

Especificación de Media Queries

- **@media [CSS]**

```
@media screen and (min-width: 1024px) {
  body  {
    font-size: 100%;
  }
}
```

- Etiqueta **link** (atributo **media**)

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

Sintaxis

- Podemos utilizar operadores lógicos:
  - `not`, `and`, `only`
- Características:

Característica	Definición	Permite prefijos **`min-`** y **`max-`**
`width`	Ancho del área de visualización	Sí
`height`	Alto del área de visualización	Sí
`device-width`	Ancho de dispositivo	Sí
`device-height`	Alto de dispositivo	Sí
`orientation`	Orientación (**`portrait`**, **`landscape`**)	No

Orientación

- Se entiende:
  - Orientación **`portrait`**
    - La altura del navegador es mayor que la anchura.
  - Orientación **`landscape`**:
    - La anchura del navegador es mayor que la altura.

Ajustando a tamaños pequeños

- Las siguientes reglas se aplican si el **área de visualización** es **menor de 600px**:

```
@media screen and (max-width: 600px) {
  .class {
    /* Reglas CSS */
  }
}
```

- En hoja de estilos separada:

```
<link rel="stylesheet" media="screen and (max-width: 600px)"
                       href="small.css" />
```

Ajustando a tamaños grandes

- Las siguientes reglas se aplican si el **área de visualización** es **mayor de 900px**:

```
@media screen and (min-width: 900px) {
  .class {
    /* Reglas CSS */
  }
}
```

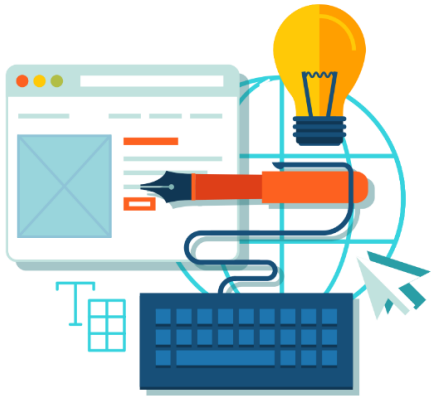
- En hoja de estilos separada:

```
<link rel="stylesheet" media="screen and (min-width: 900px)"
                       href="high.css" />
```

Ajustando a tamaños intermedios

- Las siguientes reglas se aplican si el área de visualización está **entre 600 y 900px**:
  - Combinamos varias queries con el operador lógico and:

```
@media screen and (min-width: 600px)
and (max-width: 900px) {
  .class {
    /* Reglas CSS */
  }
}
```



Rejillas (grid)

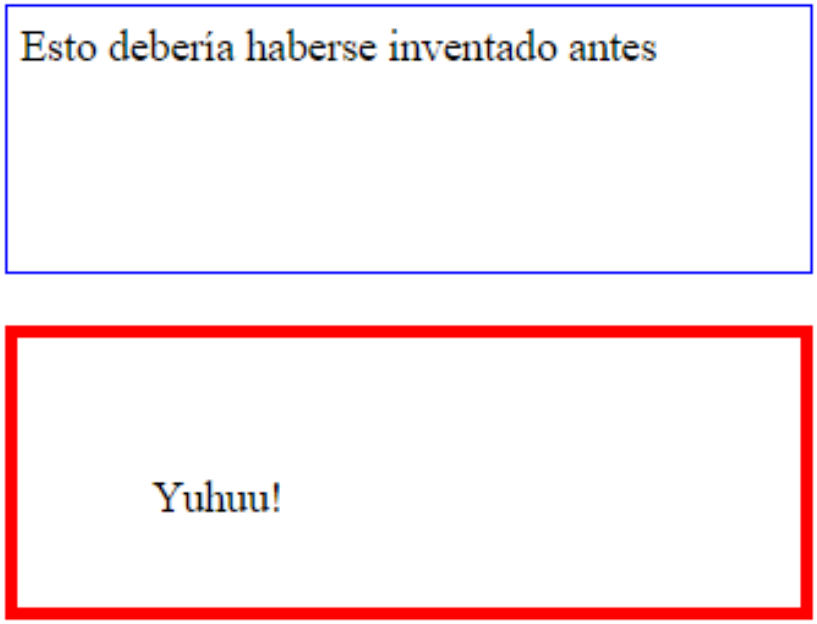
UD2: Diseño Web Responsive

Redefiniendo el modelo de cajas

box-sizing: content-box|border-box|inherit;	
content-box	Comportamiento por defecto. La anchura y altura especificada se aplica al contenido. El padding y borde se calculan aparte.
border-box	El padding y el borde se calculan dentro de la anchura y altura especificada para el elemento. La anchura y la altura del elemento se calculan restando el borde y el padding de la altura/anchura especificada.
inherit	Valor heredado.
Valor por defecto: **content-box**. No se hereda.	

box-sizing: Ejemplo

```
.div1 {
    width: 300px;
    height: 100px;
    padding:5px;
    border: 1px solid blue;
    box-sizing: border-box;
}

.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 5px solid red;
    box-sizing: border-box;
}
```



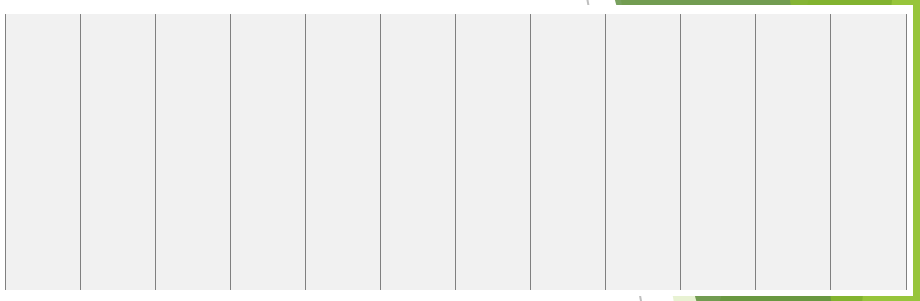
box-sizing: Ejemplo

```
* {
    box-sizing: border-box;
}
```

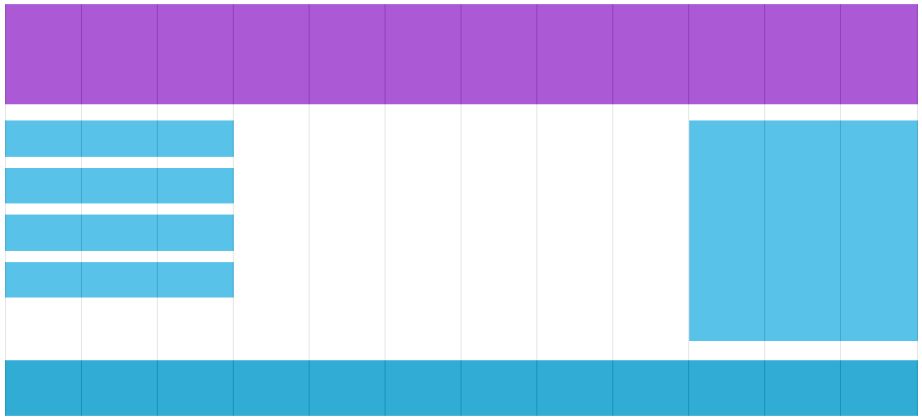
MEJOR APLICARLO A TODOS LOS ELEMENTOS PARA QUE NO HAYA EQUÍVOCOS

Diseño basado en rejillas

- Es útil dividir nuestra página en rejillas (columnas)



- Facilita la colocación horizontal de los elementos en la página



¿Cuántas columnas?

- Un número adecuado de columnas es **12**
- Las columnas se expanden y encogen a medida que redimensionamos la ventana.
- Cada columna ocupará 100 / 12 = **8.33 %**

Definiendo la rejilla

```
* {
    box-sizing: border-box;
}
```

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

```
/* Con los corchetes selecionamos
en función de atributo
 * El asterisco es un metacaracter
 */
[class*="col-"] {
    float: left;
    padding: 15px;
    border: 1px solid red;
}
```

Añadiendo datos a la rejilla

- Cuando tenemos varios elementos en una fila del grid los agruparemos en un div de mayor nivel (fila)
- Dado que todos sus elementos hijos son `float`, podemos hacer el efecto del `limpia` con una nueva regla:

```
.fila:after {
    content: "";
    clear: both;
    display: block;
}
```

Marcado

```
<div class="fila col-12“>
 <h1>Titulo</h1>
</div>
<div class="fila">
    <div class="col-3">
        Menu
    </div>
    <div class="col-9">
        Contenido
    </div>
</div>
```

Marcado (2)

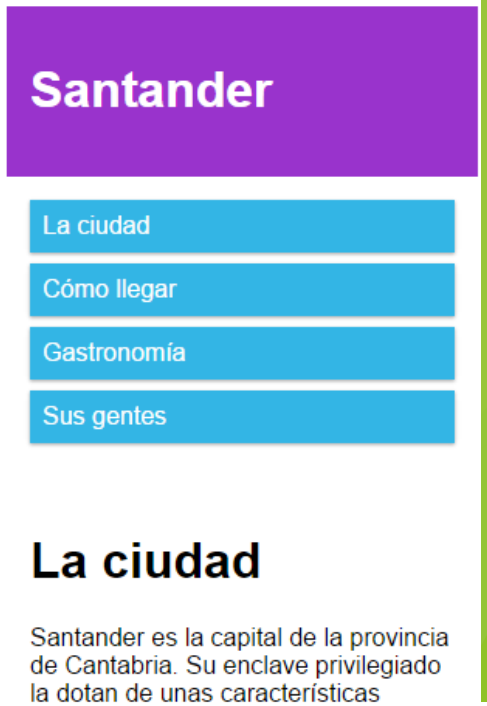
- Podemos combinar con nuestras reglas de estilo:

```
<div class="fila col-12" id="cabecera">
 Cabecera
</div>
<div class="fila" id="contenedor">
     <div class="col-3" id="menu">
       Menu
     </div>
     <div class="col-9" id="contenido">
      Contenido
     </div>
</div>
```

Rejilla para móvil

- Ejemplo: Una única columna:

```
@media only screen and (max-width: 768px) {
    /* Para resoluciones pequeñas (móvil): */
    [class*="col-"] {
        width: 100%;
}
```

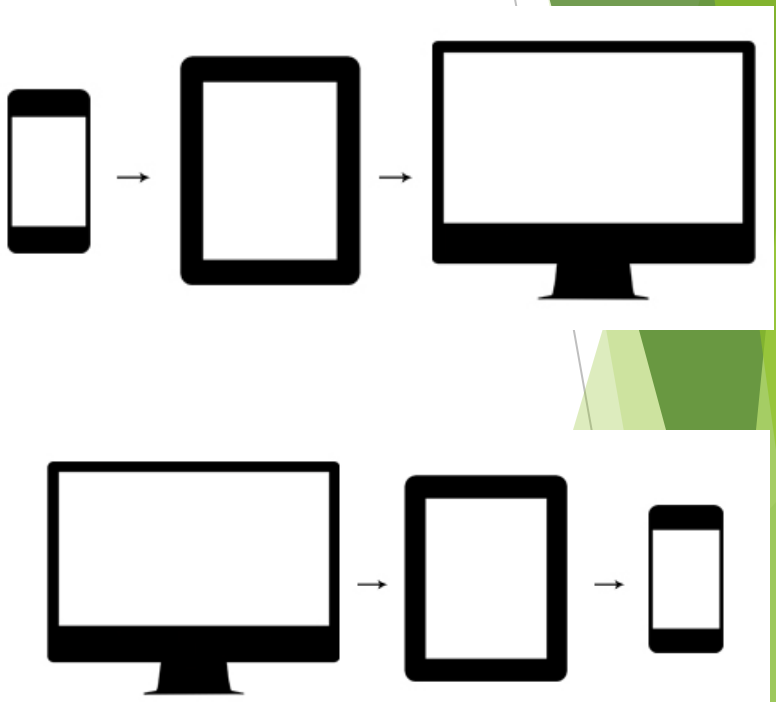


Puntos de rotura más habituales

- 320px para móvil
- 768px para tablet
- 1024px para portátil
- 1200px para sobremesa

Mobile First vs Desktop First

- Mobile First
  - Diseñamos primero para el móvil y luego adaptamos a PC
- Desktop First
  - Diseñamos primero para PC y luego adaptamos a móvil



Ventajas de Mobile First

- A nivel de diseño:
  - Determinamos de manera clara qué es lo más importante de nuestra Web.
- A nivel de rendimiento:
  - Los estilos cargarán más rápidamente en móvil (menor ancho de banda).
- A niel de implementación:
  - La hoja de estilos resultante suele ser más fácil

Adaptando a *Mobile First*

```
/* Móviles: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 768px) {
    /* Escritorio: */
    .col-1 {width: 8.33%;}
        ...
    .col-12 {width: 100%;}
}
```

Añadiendo otro punto de rotura

```
@media only screen and (min-width: 600px) {
    /* Tablets: */
    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
    .col-m-4 {width: 33.33%;}
    .col-m-5 {width: 41.66%;}
    .col-m-6 {width: 50%;}
    .col-m-7 {width: 58.33%;}
    .col-m-8 {width: 66.66%;}
    .col-m-9 {width: 75%;}
    .col-m-10 {width: 83.33%;}
    .col-m-11 {width: 91.66%;}
    .col-m-12 {width: 100%;}
}
```

Añadiendo otro punto de rotura (2)

- Parece que hemos creado clases idénticas a las anteriores, pero en el HTML podemos establecer las clases oportunas

```
<div class="col-3 col-m-3" id="menu">
...
</div>

<div class="col-6 col-m-9" id="contenido">
...
</div>

<div class="col-3 col-m-12" id="lateral">
...
</div>
```

3 columnas siempre

6 columnas en PC
9 columnas en tablet

3 columnas en PC
12 columnas en tablet

Soporte para márgenes

- Para implementar los márgenes vamos a definir clases denominadas `margin-N`
- En principio vamos a definirlos solo para resoluciones grandes (las añadimos al media query correspondiente):

```
.margin-1 {margin-left: 8.33%;}
.margin-2 {margin-left: 16.66%;}
.margin-3 {margin-left : 25%;}
.margin-4 {margin-left: 33.33%;}
.margin-5 {margin-left: 41.66%;}
.margin-6 {margin-left: 50%;}
.margin-7 {margin-left: 58.33%;}
.margin-8 {margin-left: 66.66%;}
.margin-9 {margin-left: 75%;}
.margin-10 {margin-left: 83.33%;}
.margin-11 {margin-left: 91.66%;}
```

Soporte para márgenes: Ejemplo

- Queremos un elemento que ocupe 4 columnas y margen izquierdo de dos columnas (resoluciones grandes) y 12 columnas en resoluciones medianas:

```
<div class="col-4 margin-2 col-m-12">
```

LOS MÁRGENES DERECHOS NO SE IMPLEMENTAN, SIMPLEMENTE DEJAMOS EL HUECO CORRESPONDIENTE

Imágenes fluidas

UD2: Diseño Web Responsive

Imágenes fluidas

- En dispositivos pequeños nos interesa que las imágenes ocupen porcentajes grandes de la página para que se vean bien.
- Usando simplemente la propiedad `width` a 100%, la imagen será *responsive* y escalará correctamente:

```
img {
    width: 100%;
    height: auto;
}
```

Imágenes fluidas (2)

- El problema del enfoque anterior es que en dispositivos grandes la imagen puede mostrarse a un tamaño mayor que su propia resolución.
- Usando `max-width` a 100% la imagen escalará a tamaños menores pero **nunca mayores que su tamaño original**:

```
img {
    max-width: 100%;
    height: auto;
}
```

Imágenes y dispositivos

- Una imagen grande puede ser válida en un dispositivo con pantalla grande, pero inútil en un dispositivo pequeño.
  - Para qué queremos una imagen grande si va a escalarse? → Ancho de banda desaprovechado ☹
- Podemos usar media queries para mostrar distintas imágenes en distintos dispositivos.

Uso de <picture>

- El elemento **<picture>** de HTML5 permite describir el modo en que queremos cargar las imágenes.
  - Permite añadir etiquetas <source> que muestran contenido en función de un media query.
  - Atributos de <source>:
    - **srcset**. Origen de la imagen.
    - **media** (opcional): Media query
  - Es obligatorio añadir una etiqueta <img> al final de la lista de fuentes para navegadores que no aceptan <picture>.

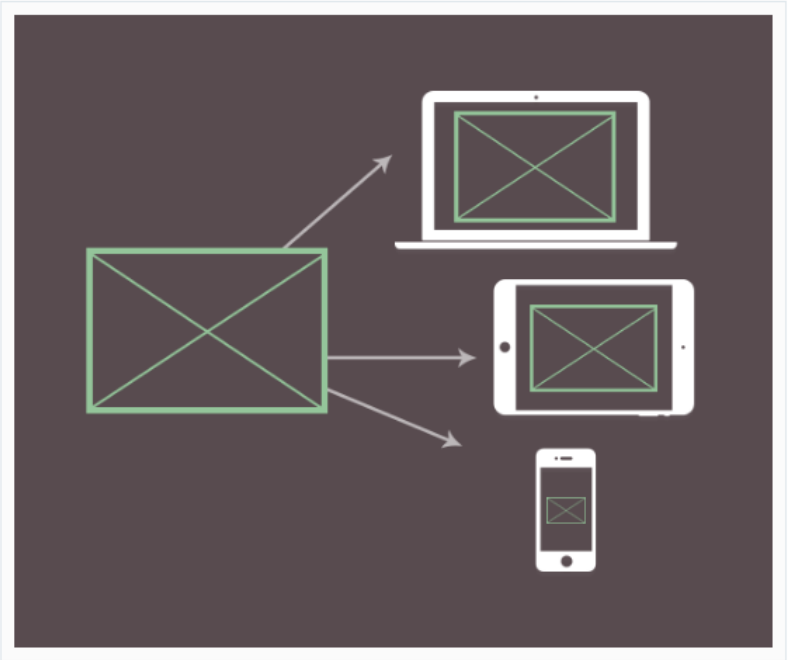
Ejemplo de <picture>

```
<picture>
   <source srcset="img/santander-small.jpg"
           media="(max-width: 768px)">

   <source srcset="img/santander-medium.jpg"
           media="(min-width:768px) and (max-width:1024px)">

   <source srcset="img/santander-big.jpg"
           media="(min-width:1024px)">

    <img src="img/santander" alt="Santander"/>
 </picture>
```

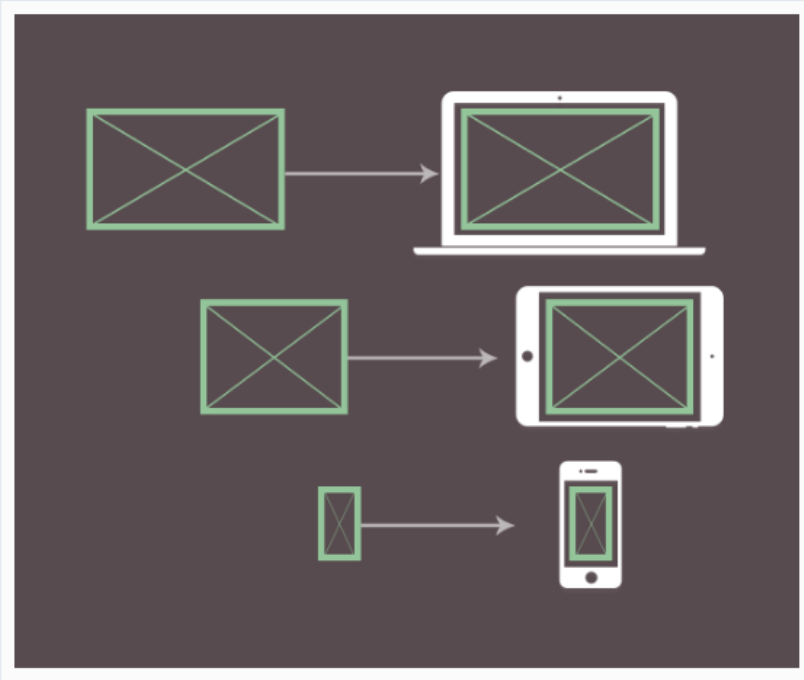
```
picture img{
    width:100%;
    height:auto;
}
```

Imágenes fluidas vs picture

- Imágenes fluídas



- Picture



Adaptando imágenes

- Las media queries pueden incluir la orientación (`orientation`) para utilizar la imagen más adecuada en cada tipo de dispositivo.

Imágenes de fondo

- También pueden responder a escalado usando la propiedad `background-size`
  - `contain`
    - Mantiene la proporción pero puede no llenar el fondo.
  - `100%`
    - No mantiene proporción y se ajusta al fondo.
  - `cover`
    - Mantiene proporción y se ajusta, pero parte de la imagen puede quedar fuera.

Imágenes de fondo y dispositivos: Ejemplo

```
/* Para anchuras menores a 400px: */
body {
    background-repeat: no-repeat;
    background-image: url('img_flores_thumb.jpg');
}

/* Para anchuras de 400px y mayores: */
@media only screen and (min-width: 400px) {
    body {
        background-image: url('img_flores.jpg');
    }
}
```

Sondeando en función del dispositivo

- Para imágenes de fondo, podemos establecer el media query en función de min-device-width (que comprueba ancho de dispositivo) y no de device-width (que comprueba el navegador)

```
/* Para anchuras de 400px y mayores: */
@media only screen and (min-device-width: 400px) {
    body {
        background-image: url('img_flores.jpg');
    }
}
```

Vídeo responsive

- Seguimos el mismo enfoque que con imágenes para adaptar el vídeo el tamaño de la pantalla:

```
video {
    width: 100%;
    height: auto;
}
```

```
video{
    max-width: 100%;
    height: auto;
}
```

- Antes estaba permitido el uso de media queries en etiquetas `source` de `video`, pero actualmente se recomienda el uso de protocolos de streaming en servidor como DASH

Frameworks CSS

- Ofrecen ayuda para hacer de manera automática ciertas de las tareas que solemos hacer con CSS (botones, formularios, sliders..)
- Muchos de ellos ofrecen ayuda para implementar la rejilla (grid)
  - 960gs
  - **Bootstrap**
  - Foundation
  - YAML
  - ...