



# Unidad 4. Creación de interfaces web utilizando estilos

Diseño web adaptativo

## Contenido

<b>1. OBJETIVO.....</b>	<b>3</b>
<b>2. LOS MEDIA QUERIES.....</b>	<b>3</b>
2.1. LA RECOMENDACIÓN DEL W3C.....	3
2.2. LOS CRITERIOS DE LOS MEDIA QUERIES.....	3
2.3. LA SINTAXIS DE LOS MEDIA QUERIES.....	4
2.4. LOS VALORES MÍNIMOS Y MÁXIMOS.....	4
2.5. LOS OPERADORES LÓGICOS.....	4
<b>3. EL TAMAÑO DE LAS PANTALLAS.....</b>	<b>5</b>
3.1. EL TAMAÑO FÍSICO Y LA VISUALIZACIÓN.....	5
3.2. LOS ZOOMS EN PANTALLA.....	6
3.3. DOS SITIOS DE REFERENCIA PARA EL TAMAÑO DE LAS PANTALLAS.....	7
<b>4. UN EJEMPLO DE UN SITIO SENCILLO.....</b>	<b>8</b>
4.1. LA ESTRUCTURA DEL SITIO.....	8
4.2. REALIZAR UN DISEÑO ADAPTATIVO SENCILLO.....	12
4.3. LOS RESULTADOS.....	15
4.4. DISEÑO ADAPTATIVO CON FLEXBOX LAYOUT.....	17
4.5 DISEÑO ADAPTATIVO CON GRID.....	18
<b>5. IMÁGENES ADAPTATIVAS.....</b>	<b>19</b>

## 1. OBJETIVO

En la actualidad la mayoría de los sitios web adaptan su diseño y su formato al tamaño de la pantalla en la que se muestran. La página se va a adaptar automáticamente en función de si se visualiza en la pantalla de un ordenador, en una tableta o en un teléfono inteligente. La expresión inglesa para designar esta arquitectura web es **Responsive Web Design**, que podemos traducir por **diseño web adaptativo**, aunque también se usa **diseño web reactivo**. Para ello, debemos utilizar los **Media Queries**.

## 2. LOS MEDIA QUERIES

### 2.1. LA RECOMENDACIÓN DEL W3C

Para crear un diseño adaptado a los diferentes tamaños de pantalla, el W3C facilita el módulo **Media Queries**, que actualmente (noviembre 2020) está como **Candidate Recommendation, 21 July 2020**: <https://www.w3.org/TR/mediaqueries-4/>. Con los Media Queries puede crear hojas de estilo que se adapten a los diferentes tamaños de la pantalla. La detección del tamaño de la pantalla se lleva a cabo mediante la «consulta» al medio, tras la cual el navegador utiliza los estilos adaptados.

### 2.2. LOS CRITERIOS DE LOS MEDIA QUERIES

El módulo **Media Queries** propone determinar los medios de difusión mediante criterios precisos y la posibilidad de combinar estos criterios. El valor que devuelve un Media Query es de tipo booleano: verdadero o falso. El navegador escoge la hoja de estilos adaptada en función de las respuestas a los Media Queries.

Estos son los criterios que podemos utilizar en los Media Queries:

- La **anchura de visualización: width**. Podemos testar la anchura del área de visualización del navegador. Por ejemplo: `width: 780px`.
- La **altura de visualización: height**. Podemos testar la altura del área de visualización del navegador.
- La **anchura física: device-width**. Podemos testar la anchura física de la pantalla de difusión.
- La **altura física: device-height**. Podemos testar la altura física de la pantalla de difusión.
- La **orientación de la pantalla: orientation**. Por ejemplo: `orientation: portrait` u `orientation: landscape`. Muy práctico para testar si el usuario usa su tableta táctil en posición vertical (portrait) u horizontal (landscape).
- La **ratio: aspect-ratio**. Para testar el valor de la proporción anchura/altura. Por ejemplo: `aspect-ratio: 16/9`.
- La **ratio física: device-aspect-ratio**. Para testar el valor de la proporción física anchura/altura de la pantalla.
- El **color: color**. Podemos testar si el soporte de difusión usa colores (valor por defecto si no se usan), es en blanco y negro o en escalas de gris. Por ejemplo: `min-color: 8`.
- El **número de colores en la tabla de colores: color-index**.

- El **número de niveles de gris** en los aparatos monocromos: **monochrome**.
- La **resolución** de la pantalla de salida: **resolution**. Se expresa en dpi.
- El **tipo de barrido** para las pantallas de televisión: **scan**.
- Utilice grid para testar si la pantalla de difusión emplea una cuadrícula con un solo tamaño de fuente.

### 2.3. LA SINTAXIS DE LOS MEDIA QUERIES

Veamos ahora un ejemplo concreto. Queremos detectar los periféricos que tengan una anchura exacta de 780 píxeles.

En una página HTML, en el <head>, tenemos el siguiente Query con el elemento <link>:

```
<link rel="stylesheet" href="styles780.css" />
```

En el archivo CSS llamado **styles780.css**, este es el Query con la regla @:

```
@media screen and (width:780px) {  
    ...  
}
```

Indicamos que el tipo de medio es una pantalla: screen.

Indicamos que hay un segundo criterio: and.

Indicamos que la anchura de estas pantallas debe ser igual a 780 píxeles: width: 780px.

Todos los estilos que se han de usar se ponen entre llaves.

### 2.4. LOS VALORES MÍNIMOS Y MÁXIMOS

Todos los criterios que acabamos de ver, excepto orientation, scan y grid, pueden utilizar los prefijos min- y max- junto a sus valores (de los criterios).

Por ejemplo, vamos a testar las pantallas cuya anchura sea como máximo de 780 píxeles.

```
@media screen and (max-width:780px) {  
    ...  
}
```

Esto es muy útil, ya que resulta raro testar un tamaño fijo para las pantallas de ordenador, por ejemplo.

### 2.5. LOS OPERADORES LÓGICOS

Los Media Queries ofrecen operadores lógicos a fin de precisar y combinar los diferentes criterios.

El operador and permite utilizar el operador lógico Y.

Primer ejemplo: queremos determinar las pantallas que tienen una anchura de 780 píxeles y que son monocromas:

```
@media screen and (width:780px) and monochrome {  
    ...  
}
```

Segundo ejemplo: queremos determinar las pantallas que tienen una resolución comprendida entre un mínimo de 1024 píxeles y un máximo de 1280 píxeles:

```
@media screen and (min-width: 1024px) and (max-width: 1280px) {  
    ...  
}
```

El operador not permite utilizar el operador lógico **NO**.

Por ejemplo: queremos determinar las pantallas cuya resolución no es de 780 píxeles:

```
@media screen and (not width:780px) {  
    ...  
}
```

La coma (,) permite usar el operador lógico **O**.

Por ejemplo, queremos determinar las pantallas de 780 píxeles de ancho o aquellas que son monocromas:

```
@media screen and (width:780px), monochrome {  
    ...  
}
```

La palabra clave **only** permite precisar que el Query debe aplicarse únicamente sobre los criterios indicados. Esto permite ocultar los estilos para navegadores antiguos.

```
@media only screen and (width:780px) {  
    ...  
}
```

### 3. EL TAMAÑO DE LAS PANTALLAS

#### 3.1. EL TAMAÑO FÍSICO Y LA VISUALIZACIÓN

En la actualidad, cada tipo de teléfono inteligente y de tableta tiene sus propias características técnicas relativas al tamaño físico de las pantallas y a la superficie realmente disponible para la visualización de un sitio en la versión móvil de un navegador.

Un segundo problema se deriva de lo que realmente se muestra en función de las características de la pantalla. Cuando salió el iPhone de Apple en 2007, el tamaño de su pantalla se había fijado a 320 x 480 píxeles. Pero Safari Mobile muestra los sitios con una anchura de 980 píxeles, frente a los 320 píxeles de la pantalla, lo que implica una disminución sustancial del área de visualización.

Con el iPhone, Apple ha introducido un nuevo valor para el atributo **name**: **viewport**. Este permite controlar la ratio anchura del sitio (980 píxeles)/anchura de visualización (320 píxeles) aplicada por Safari en su versión móvil.

Si queremos que Safari para iPhone muestre los sitios con una anchura de 320 píxeles (y no con 980 píxeles), debemos utilizar el valor **viewport** para el atributo **name**:

```
<meta name="viewport" content="width=320" />
```

En la actualidad, este atributo ya ha sido adoptado por todos los fabricantes de teléfonos inteligentes y de tabletas. Se ha convertido en un estándar de facto.

El problema que aún queda pendiente es que el tamaño de las pantallas no está en absoluto armonizado entre los distintos teléfonos inteligentes y otras tabletas. Debemos adaptar la visualización en función de las características propias de cada aparato usando el valor **device-width**, que determina la anchura física de la pantalla.

```
<meta name="viewport" content="width=device-width" />
```

De este modo, la visualización se adaptará a la superficie y las capacidades de cada navegador para aparatos móviles.

Vídeo interesante sobre el tema:

<https://www.youtube.com/watch?v=OP2GnAFFeTY>

### 3.2. LOS ZOOMS EN PANTALLA

En **viewport**, dentro del atributo **content**, puede agregar otros valores relacionados con el zoom en pantalla. Estos otros valores se separan de los anteriores con una coma.

El valor **initial-scale** especifica el nivel de zoom inicial de la página cuando esta se carga. Ello permite que los aparatos «respeten» la anchura especificada en los Media Queries.

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

El valor **user-scalable** define el nivel de zoom autorizado por la acción del usuario. Un valor de 0 prohíbe al usuario el uso del zoom. Un valor de 1 autoriza la utilización del zoom.

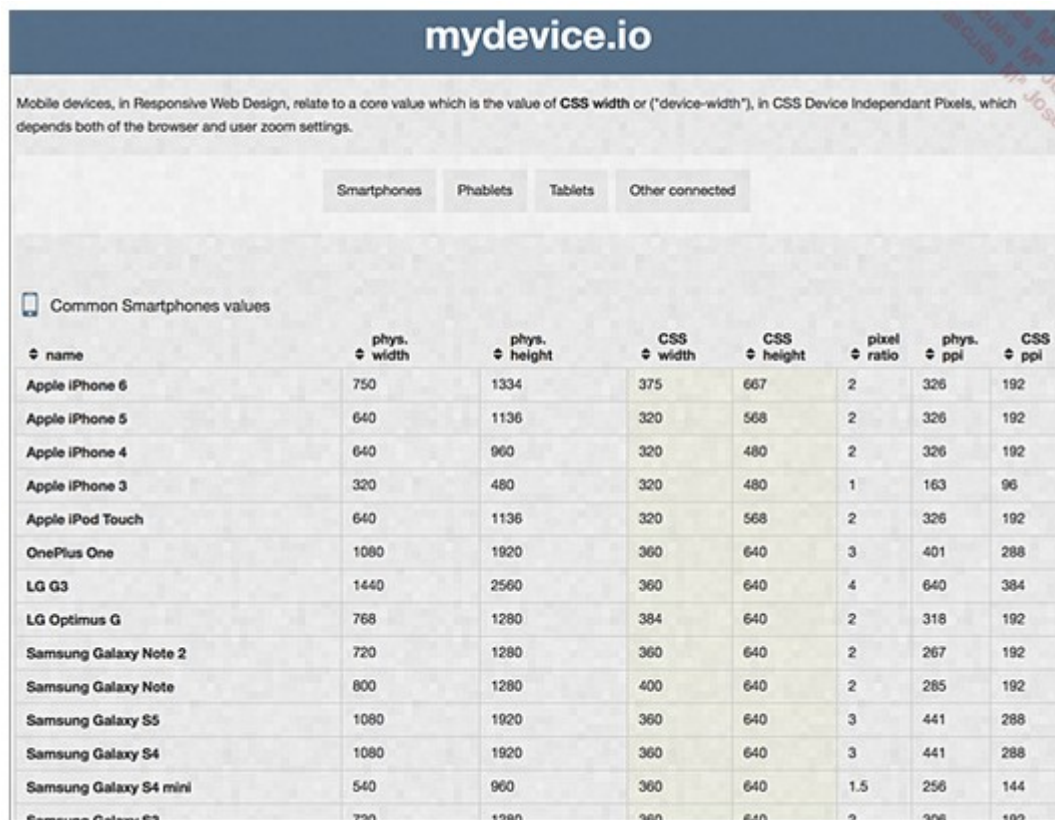
```
<meta name="viewport" content="width=device-width, user-scalable=0" />
```

Los valores **minimum-scale** y **maximum-scale** permiten definir los zooms mínimos y máximos autorizados.

```
<meta name="viewport" content="width=device-width, minimum-scale=0.5, maximum-scale=3.0, initial-scale=1.0, user-scalable=1" />
```

## 3.3. DOS SITIOS DE REFERENCIA PARA EL TAMAÑO DE LAS PANTALLAS

El sitio **mydevice.io** (<http://mydevice.io/devices/>) proporciona los valores width y device-width de los teléfonos inteligentes, las phablets (teléfonos inteligentes con pantallas muy grandes) y las tabletas más corrientes.



mydevice.io

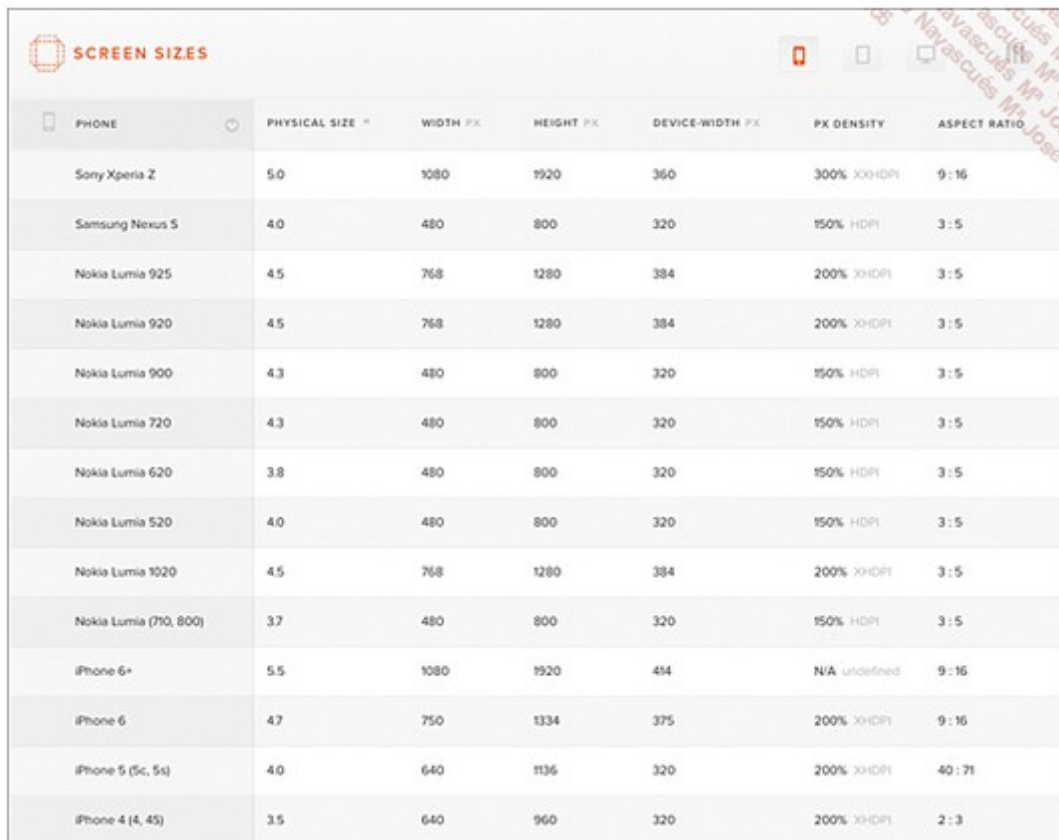
Mobile devices, in Responsive Web Design, relate to a core value which is the value of **CSS width** or ("device-width"), in CSS Device Independent Pixels, which depends both of the browser and user zoom settings.

Smartphones Phablets Tablets Other connected

Common Smartphones values

name	phys. width	phys. height	CSS width	CSS height	pixel ratio	phys. ppi	CSS ppi
Apple iPhone 6	750	1334	375	667	2	326	192
Apple iPhone 5	640	1136	320	568	2	326	192
Apple iPhone 4	640	960	320	480	2	326	192
Apple iPhone 3	320	480	320	480	1	163	96
Apple iPod Touch	640	1136	320	568	2	326	192
OnePlus One	1080	1920	360	640	3	401	288
LG G3	1440	2560	360	640	4	640	384
LG Optimus G	768	1280	384	640	2	318	192
Samsung Galaxy Note 2	720	1280	360	640	2	267	192
Samsung Galaxy Note	800	1280	400	640	2	285	192
Samsung Galaxy S5	1080	1920	360	640	3	441	288
Samsung Galaxy S4	1080	1920	360	640	3	441	288
Samsung Galaxy S4 mini	540	960	360	640	1.5	256	144
Samsung Galaxy S3	720	1280	360	640	2	306	192

El sitio **screensiz.es** (<http://screensiz.es/phone>) ofrece el mismo servicio.



PHONE	PHYSICAL SIZE "	WIDTH PX	HEIGHT PX	DEVICE-WIDTH PX	PX DENSITY	ASPECT RATIO
Sony Xperia Z	5.0	1080	1920	360	300% XXHDPI	9:16
Samsung Nexus S	4.0	480	800	320	150% HDPI	3:5
Nokia Lumia 925	4.5	768	1280	384	200% XXHDPI	3:5
Nokia Lumia 920	4.5	768	1280	384	200% XXHDPI	3:5
Nokia Lumia 900	4.3	480	800	320	150% HDPI	3:5
Nokia Lumia 720	4.3	480	800	320	150% HDPI	3:5
Nokia Lumia 620	3.8	480	800	320	150% HDPI	3:5
Nokia Lumia 520	4.0	480	800	320	150% HDPI	3:5
Nokia Lumia 1020	4.5	768	1280	384	200% XXHDPI	3:5
Nokia Lumia (710, 800)	3.7	480	800	320	150% HDPI	3:5
iPhone 6+	5.5	1080	1920	414	N/A undefined	9:16
iPhone 6	4.7	750	1334	375	200% XXHDPI	9:16
iPhone 5 (5c, 5s)	4.0	640	1136	320	200% XXHDPI	40:71
iPhone 4 (4, 4s)	3.5	640	960	320	200% XXHDPI	2:3

## 4. UN EJEMPLO DE UN SITIO SENCILLO

### 4.1. LA ESTRUCTURA DEL SITIO

Vamos a construir un sitio web usando diseño web adaptativo. Se tratará de un sitio (**09\_01.html**) muy simple, con un encabezado (<header>), una barra de navegación (<nav>), una columna a la izquierda (<section>) con tres artículos (<article>), una columna lateral derecha (<aside>) con una imagen y una lista. Finalmente, incluiremos un pie de página (<footer>).

He aquí el código HTML5 de la página:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<title>Mi página de inicio</title>
<meta charset="UTF-8" />
<link rel="stylesheet" href="rwd.css" />
</head>
<body>
<div id="contenedor">
  <header id="superior">
    <h1>Mi sitio web</h1>
    <h2>El eslogan de mi sitio web</h2>
  </header>
  <nav id="navegacion">
    <p><a href="#">Vínculo 1</a> | <a href="#">Vínculo 2</a> |
    <a href="#">Vínculo 3</a> | <a href="#">Vínculo 4</a> | <a
```



```

href="#">Vínculo 5</a> | <a href="#">Vínculo 6</a></p>
</nav>
<section id="contenido">
  <article>
    <h1>Mi primer artículo</h1>
    <p>Cum sociis natoque penatibus...</p>
    <p>...</p>
  </article>
  <article>
    <h1>Mi segundo artículo</h1>
    <p>Aenean lacinia bibendum...</p>
    <p>...</p>
  </article>
  <article>
    <h1>Mi tercer artículo</h1>
    <p>Donec ullamcorper nulla...</p>
    <p>...</p>
  </article>
</section>
<aside id="sidebar">
  <p></p>
  <ul>
    <li>Primero</li>
    <li>Segundo</li>
    <li>Tercero</li>
    <li>Cuarto</li>
  </ul>
</aside>
<footer id="pieDePagina">
  <p>Creación y realización: un servidor. Contacto
: <a
href="mailto:yo@micorreo.org">contacto@micorreo.org</a></p>
</footer>
</div>
</body>
</html>

```

Estos son los estilos CSS3 recogidos en el archivo **rwd.css**:

```

/* Estilos generales */
body {
  padding: 0;
  background-color: gray;
  font: .8em Verdana, Arial, Helvetica, sans-serif;
}
body, h1, h2, p {
  margin: 0;
}
div#contenedor {
  width: 940px;
  margin: 20px auto 0 auto;
  padding: 10px;
  background-color: white;
  border-radius: 10px;
}

```

```
/* Encabezado*/
header#superior {
    margin: 0 0 10px 0;
    padding: 5px;
    background-color: gray;
    border-radius: 10px;
}
header#superior h1, header#superior h2 {
    color: white;
}

/* Barra de navegación */
nav#navegacion {
    margin: 0 0 10px 0;
    padding: 5px;
    background-color: silver;
    border-radius: 10px;
}
nav#navegacion p {
    color: white;
}
nav#navegacion a {
    text-decoration: none;
    color: white;
}

/* Contenido principal */
section#contenido {
    float: left;
    width: 720px;
    padding: 5px;
}
article {
    margin: 0 0 20px 0;
    border-top: 1px dotted #333;
}

/* Columna lateral */
aside#sidebar {
    float: left;
    width: 190px;
    margin: 0 0 0 10px;
    padding: 5px;
    border-radius: 5px;
    background-color: #E8E8E8;
}
aside#sidebar p {
    text-align: center;
}

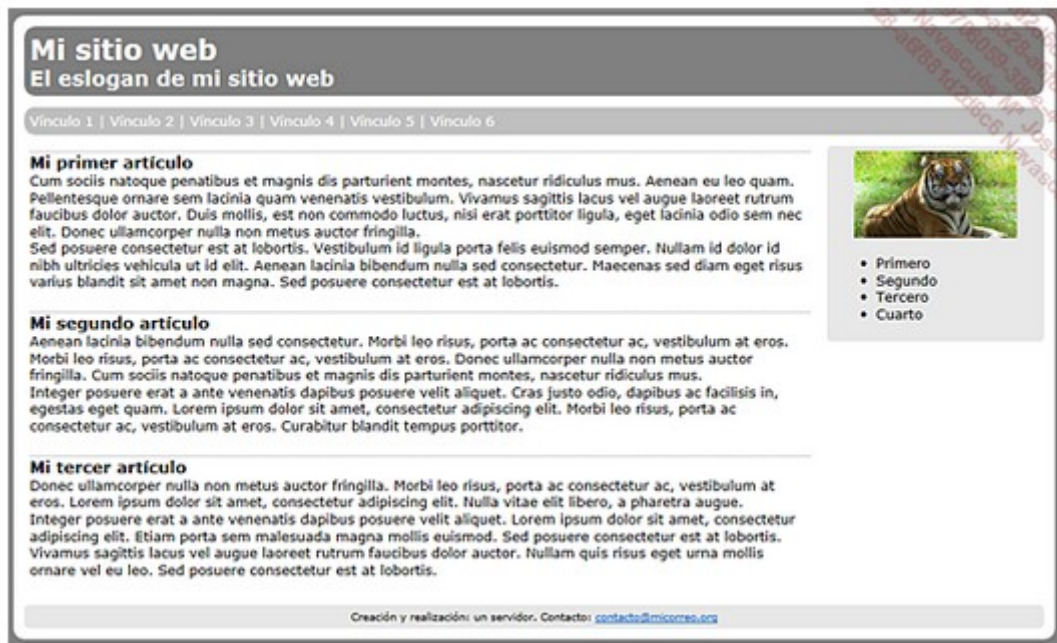
/* Pie de página */
footer#pieDePagina {
    padding: 5px;
    border-radius: 5px;
    background-color: #E8E8E8;
    clear: both;
}
```

```

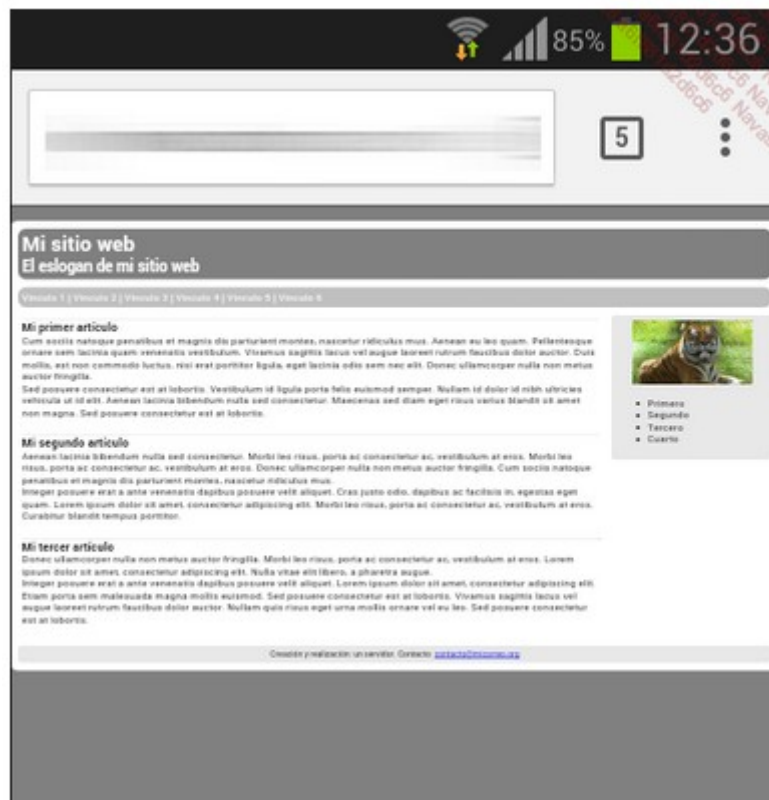
}
footer#pieDePagina p {
    text-align: center;
    font-size: .75em;
}

```

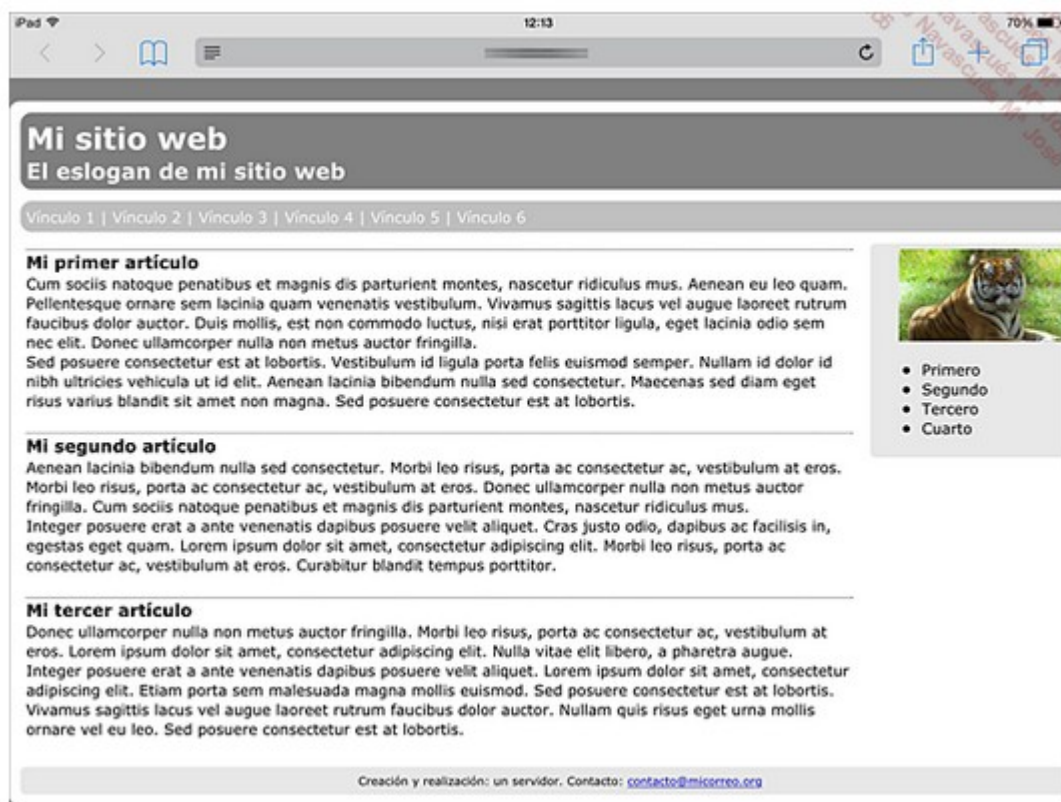
Este es el resultado en una pantalla de ordenador:



Así se ve en un móvil con Android:



Y así se ve en un iPad:



## 4.2. REALIZAR UN DISEÑO ADAPTATIVO SENCILLO

En primer lugar, debemos añadir la línea de viewport en el encabezado de la página HTML, en el `<head>`.

```
<head>
...
<meta name="viewport" content="width=device-width" />
...
</head>
```

En el archivo CSS, añadimos un Media Query al final de la página. Determinamos la difusión en una pantalla con tamaño máximo.

```
/* Media Query para las pantallas inferiores a 768 píxeles */
@media screen and (max-width: 768px) {
    ...
}
```

¿Por qué hemos elegido como tamaño máximo 768 píxeles? Sencillamente, porque es la anchura de un iPad.

Deseamos que el fondo de la página sea blanco, y no gris como aparece en la versión para pantalla de ordenador.

```
@media screen and (max-width: 768px) {  
  body {  
    background-color: #fff;  
  }  
}
```

Deseamos que la caja `<div id="contenedor">` no tenga un tamaño fijo sino relativo, para que ocupe todo el espacio disponible en su elemento padre, `<body>`.

```
@media screen and (max-width: 768px) {  
  body {  
    background-color: #fff;  
  }  
  div#contenedor {  
    width: 100%;  
    margin: 0;  
  }  
}
```

El color de fondo de la banda superior debe ser negro, con el texto blanco.

```
@media screen and (max-width: 768px) {  
  body {  
    background-color: #fff;  
  }  
  div#contenedor {  
    width: 100%;  
    margin: 0;  
  }  
  header#superior {  
    background-color: #000;  
    color: #fff;  
  }  
}
```

Los dos contenedores (`<section>` y `<aside>`) no deben ser flotantes, para ganar espacio y disponer, de este modo, de contenedores que se organizan verticalmente.

```
@media screen and (max-width: 768px) {  
  body {  
    background-color: #fff;  
  }  
  div#contenedor {  
    width: 100%;  
    margin: 0;  
  }  
  header#superior {  
    background-color: #000;  
    color: #fff;  
  }  
  section#contenido {  
    float: none;  
    width: auto;  
  }  
  aside#sidebar {
```

```
float: none;
width: auto;
margin: 0 0 10px 0;
}
}
```

No mostraremos la imagen (<img>) en la sidebar para ganar más espacio y concentrarnos en el contenido.

```
@media screen and (max-width: 768px) {
  body {
    background-color: #fff;
  }
  div#contenedor {
    width: 100%;
    margin: 0;
  }
  header#superior {
    background-color: #000;
    color: #fff;
  }
  section#contenido {
    float: none;
    width: auto;
  }
  aside#sidebar {
    float: none;
    width: auto;
    margin: 0 0 10px 0;
  }
  aside#sidebar img {
    display: none;
  }
}
```

Finalmente, adaptaremos el tamaño de los títulos <h1> en los artículos.

```
@media screen and (max-width: 768px) {
  body {
    background-color: #fff;
  }
  div#contenedor {
    width: 100%;
    margin: 0;
  }
  header#superior {
    background-color: #000;
    color: #fff;
  }
  section#contenido {
    float: none;
    width: auto;
  }
  aside#sidebar {
    float: none;
  }
}
```

```

    width: auto;
    margin: 0 0 10px 0;
}
aside#sidebar img {
    display: none;
}
article h1 {
    font-size: 1.5em;
}
}

```

### 4.3. LOS RESULTADOS

Como es lógico, el diseño de página en una pantalla de ordenador no cambia, a no ser que el usuario disminuya el tamaño de la ventana del navegador a menos de 768 píxeles.

Esto es lo que se ve en un móvil con Android (parte superior de la página):



Y así se muestra la parte inferior de la página con el mismo dispositivo:



Todas las modificaciones aplicadas al formato de la página se muestran tal y como queríamos.

Esto es lo que obtenemos en un iPad en modo vertical:





#### 4.4. DISEÑO ADAPTATIVO CON FLEXBOX LAYOUT

En los siguientes enlaces tienes algunos ejemplos de diseños flexibles usando Flexbox Layout.

<https://webdesign.tutsplus.com/es/tutorials/how-to-build-a-full-screen-responsive-page-with-flexbox--cms-32086>

<https://carlosazaustre.es/los-5-patrones-del-responsive-design/>

Esta es la versión original en inglés de la anterior:

<https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns?hl=en>

<https://codepen.io/enriquearkas/pen/XJwBEr>

<https://codepen.io/cchambers/pen/KdBpa>

<https://webdesign.tutsplus.com/es/tutorials/how-to-build-a-responsive-navigation-bar-with-flexbox--cms-33535>

## 4.5 DISEÑO ADAPTATIVO CON GRID

<https://blog.10pines.com/2020/06/08/disenho-responsive-con-css-grid-y-gatitos/>

Ejemplo de página web adaptativa con Grid (explicado en <https://www.youtube.com/watch?v=BHtSWxuMESM>):

```
<!DOCTYPE html>
```

```
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>DISEÑO CSS GRID</title>
  <style>
    body{ margin: 0; background: cadetblue;}
    .pagina{
      background: white;
      display: grid;
      height: 100vh;
      grid-template:
        "header" 100px
        "nav" 50px
        "contenido" auto
        "sidebar" 100px
        "footer" 100px

    }
    .pagina > div { border: 1px solid black}
    .header{ grid-area: header;}
    .nav{ grid-area: nav;}
    .contenido{ grid-area: contenido;}
    .sidebar{ grid-area: sidebar;}
    .footer{ grid-area: footer;}
    @media (min-width:900px){
      .pagina{
```

```

        max-width: 1000px;
        margin: auto;
        grid-template:
            "header      header"      100px
            "nav         nav"         50px
            "sidebar     contenido"    auto
            "footer      footer"      100px
            / 200px      1fr
    }
}

</style>
</head>
<body>
    <div class="pagina">
        <div class="header"> header </div>
        <div class="nav"> menu </div>
        <div class="contenido"> contenido </div>
        <div class="sidebar"> SIDEBAR </div>
        <div class="footer">PIE</div>
    </div>
</body>
</html>

```

Más artículos de CSS Grid:

<https://www.ionos.es/digitalguide/paginas-web/creacion-de-paginas-web/css-grid-layout/>

<https://www.youtube.com/watch?v=T4t00Hd3qZc>

## 5. IMÁGENES ADAPTATIVAS

Las imágenes también pueden comportar problemas de tamaño en relación con su elemento padre. Si la imagen es más ancha que su contenedor, se desborda.

Veamos un ejemplo sencillo (**09\_02.html**):

```

<!DOCTYPE HTML>
<html lang="es">
<head>

```

```

<title>Mi página de inicio</title>
<meta charset="UTF-8" />
<style>
    #miCaja {
        float: left;
        width: 300px;
        border: 1px solid #333;
        padding: 5px;
    }
    p {
        margin: 0;
    }
</style>
</head>
<body>
<div id="miCaja">
    <p><br />Curabitur
blandit tempus...</p>
</div>
</body>
</html>

```

Esto es lo que obtenemos: la foto de la cebra desborda la caja porque es más ancha que su elemento padre.



La solución para evitar este problema (y el de los cambios consecutivos de tamaño en los Media Queries) consiste en imponer una anchura máxima en relación con el elemento contenedor, el elemento padre de la imagen. Para ello, es preciso utilizar la propiedad max-width.

Añadamos este estilo:

```
#miCaja img {  
    max-width: 100%;  
}
```

Lo que pedimos es que la anchura máxima de las imágenes insertadas en la caja <div id="miCaja"> sea del 100 %. De este modo, las imágenes nunca serán más anchas que su elemento padre.

Así se muestra ahora la página:



También podemos imaginar que la caja tenga una anchura flexible. En este ejemplo, indicamos una anchura del 25 % en relación con el <body>.

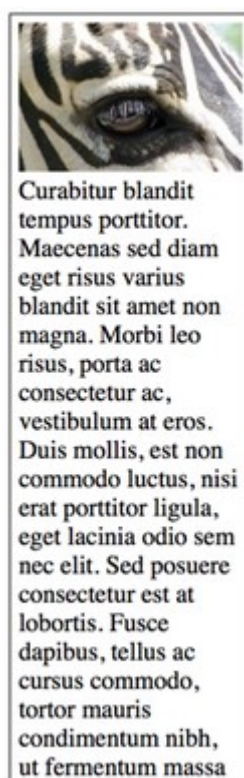
```
#miCaja {  
    float: left;  
    width: 25%;  
    border: 1px solid #333;  
    padding: 5px;  
}
```

La imagen siempre tendrá una anchura máxima del 100 % en relación con su elemento padre, sea cual sea la anchura de este.

Así se ve la página en una pantalla ancha:



Y así, en una pantalla con una anchura más pequeña:







justo sit amet risus.  
Etiam porta sem  
malesuada magna  
mollis euismod.  
Etiam porta sem  
malesuada magna  
mollis euismod.  
Etiam porta sem  
malesuada magna  
mollis euismod. Cras  
mattis consectetur  
purus sit amet  
fermentum.

Puedes ampliar información en el siguiente artículo: <https://disenowebakus.net/disenoweb-responsive.php>

Algunos ejercicios con flex: <https://webdesign.tutsplus.com/es/tutorials/exercises-in-flexbox-simple-web-components--cms-28049>

Interesante web para comprender flex: <https://flexboxfroggy.com/#es>

5 técnicas flexbox que deberías conocer:

[https://programacion.net/articulo/5\\_tecnicas\\_flexbox\\_que\\_deberias\\_conocer\\_1418](https://programacion.net/articulo/5_tecnicas_flexbox_que_deberias_conocer_1418)

Más casos en los que emplear flexbox:

[https://developer.mozilla.org/es/docs/Web/CSS/CSS\\_Flexible\\_Box\\_Layout/Casos\\_de\\_uso\\_tipicos\\_de\\_Flexbox](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Casos_de_uso_tipicos_de_Flexbox).

Más para entender flex:

<https://desarrolloweb.com/articulos/css3-flexbox.html>

<https://desarrolloweb.com/articulos/conceptos-principales-flexbox.html>

<https://desarrolloweb.com/articulos/propiedades-contenedor-flexbox.html>

<https://desarrolloweb.com/articulos/propiedades-item-flexbox.html>

<https://desarrolloweb.com/articulos/practica-justifycontent-flexbox.html>

<https://desarrolloweb.com/articulos/flexbox-align-items.html>