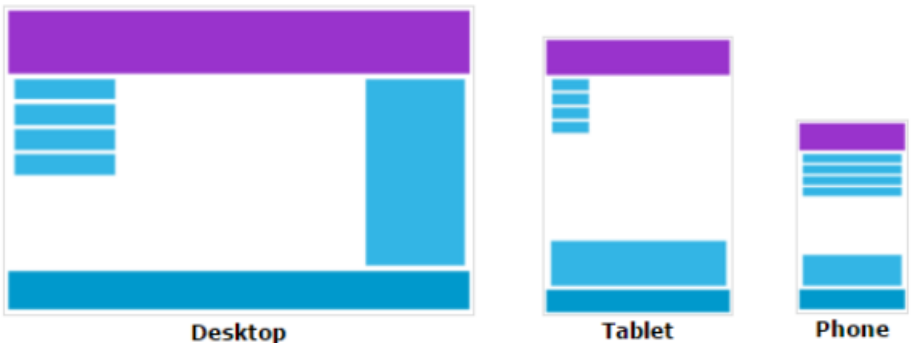
Responsive Web Design (RWD)

1. Introducción

Es una **técnica** que permite que una página Web se visualice ***correctamente independientemente del dispositivo*** en el que se esté visualizando (**Smartphone**, **Tablet** o **PC**).

Para realizar un RWD, ***basta con usar únicamente HTML*** y ***CSS***, no es necesario el uso de otras tecnologías El **objetivo de RWD** es proporcionar una ***mejor experiencia de navegación*** para todos los usuarios:

- Actualmente, las páginas Web pueden ser visualizadas con distintos tipos de dispositivos: Smartphones, Tables y Ordenadores. Nuestra ***página Web deberá adaptarse para quedar bien y ser fácil de usar*** (independientemente del dispositivo)
- Las ***páginas web no deben omitir información para adaptarse a dispositivos*** más pequeños, sino más bien adaptar su contenido para adaptarse a cualquier dispositivo:



Se llama ***Responsive Web Design*** al uso de CSS y HTML para cambiar el tamaño (reducir, ampliar), o mover el contenido de una página Web para que se vea bien en cualquier pantalla.

2. Responsive Web Desing. Unidades.

2.1. Introducción

Para realizar un diseño RWD deberíamos dejar de utilizar las unidades en píxeles (px).

Para trabajar en el entorno responsive debemos emplear proporciones escalables o adaptativas (% y em) e idealmente aplicar también las nuevas proporciones CSS3 para viewport (vh y vw), que escalan el elemento según las propiedades de la pantalla de visualización (viewport).

Tanto em como vw se pueden utilizar en cualquier propiedad CSS que necesitemos hacer responsive. En particular, deberíamos emplearlas para definir todos los elementos /estilos/ que queramos hacer adaptables como margin y padding, y también nos resultarán de gran utilidad en la proporcionalidad height y width de imágenes background y cajas contenedoras con estilo.

2.2. Las proporciones fijas

- **Píxeles (px):** Los píxeles son unidades de tamaño fijo que se utilizan en medios de pantalla (es decir, para leer en la pantalla de la computadora). Un píxel es igual a un punto en la pantalla de la computadora (la división más pequeña de la resolución de su pantalla).
- **Puntos (pt):** Los puntos se usan tradicionalmente en medios impresos (cualquier cosa que se imprima en papel, etc.). Un punto es igual a 1/72 de pulgada. Los puntos son muy parecidos a los píxeles, ya que son unidades de tamaño fijo y no pueden escalar en tamaño.

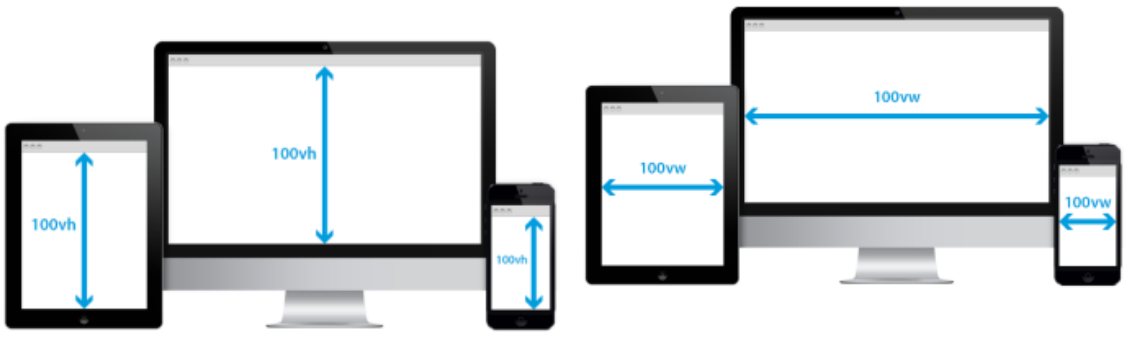
2.3. Las proporciones escalables o adaptativas

- **Porcentaje (%):** La unidad porcentual es muy parecida a la unidad «em», salvo por algunas diferencias fundamentales. En primer lugar, el tamaño de fuente actual es igual al 100% (es decir, 10 puntos = 100%). Mientras usa la unidad de porcentaje, su texto permanece completamente escalable para dispositivos móviles y para accesibilidad.
- **«Ems» (em):** «em» es una unidad escalable que se utiliza en los medios de documentos web. Un em es igual al tamaño de fuente actual, por ejemplo, si el tamaño de fuente del documento es 16px, 1em es igual a 16px. Los ems son de naturaleza escalable, por lo que 2em sería igual a 32px. La unidad em siempre depende de su elemento padre. Por ejemplo, si el elemento body tiene un tamaño de fuente de 16px y un elemento hijo tiene una fuente con tamaño 1.3em, este texto se mostrará de un tamaño un 30% más grande que el del body (20.8px).
- **Rem (rem):** La unidad de medida rem es muy similar a em (que acabamos de ver), con la única diferencia de que rem no depende de la medida del elemento padre, sino del elemento raíz del documento, el elemento HTML. Rem es un em basado en la raíz.

  Esto implica que si el elemento HTML tiene un tamaño de fuente de 16px (como es por defecto), entonces 1rem, sería igual a 16px, y si queremos aplicar un tamaño basado en rem a cualquier elemento de la página, no importará cual sea el tamaño de fuente que tenga asociado ese elemento, ya que 1 rem siempre será igual a 16 pixeles a no ser que se modifique el elemento raíz.

2.4. Proporciones relativas a la medida de la pantalla (viewport)

Viewport units (vh y vw): vh y **vw** son **unidades de longitud relativas a la pantalla de visualización** (también llamada pantalla gráfica o viewport).



Sus valores se expresan en porcentaje (de 0 a 100). Ejemplos:

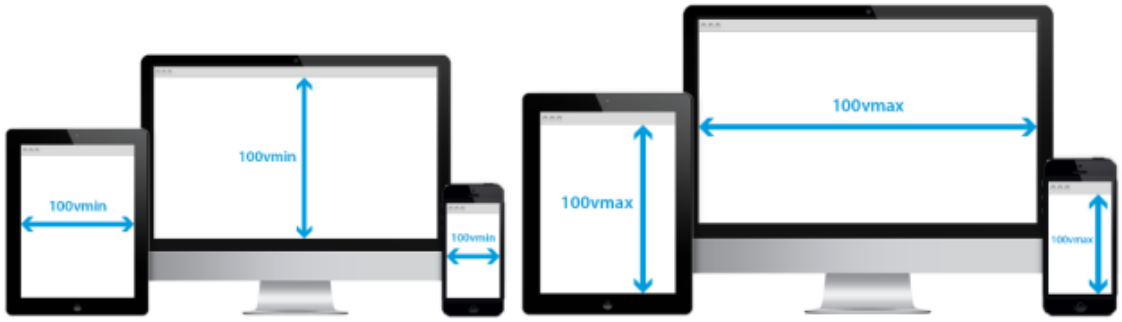
- width:100vw; (anchura del 100% de la anchura de la ventana gráfica)
- height:50vh; (altura del 50% de la altura de la ventana gráfica)
- line-height:3vh; (altura de línea del 3% de altura de la ventana gráfica)

La diferencia entre las unidades porcentuales expresadas con “%” (width:100%) y las viewport-units (width:100vw) es que las primeras heredan los valores de su elemento padre, y las segundas siempre heredan los valores del bloque inicial o ventana gráfica.

2.5. Proporciones relativas a otra proporción

vmin el valor menor en relación a la dimensión pequeña del viewport (anchura o altura).

vmax el valor máximo en relación a la dimensión más grande del viewport (anchura o altura).



2.6. Imágenes

```
<picture>
<source media="(min-width: 800px)" srcset="img/pc.png"> <source
media="(min-width: 480px)" srcset="img/tablet.png"> <img
src="img/movil.jpg" alt="Ejemplo">
</picture>
```

3. RWD. EL VIEWPOR

El ***Viewport*** es el área visible de usuario de una página Web.

Viewport varía con el dispositivo, y será más pequeño en un teléfono móvil que en una pantalla de ordenador.

Antes de las tablets y los smartphones, las ***páginas web eran diseñadas sólo para pantallas de ordenador***, y era común para las páginas web que tuviesen un diseño estático y un tamaño fijo.

La aparición de estos dispositivos y su uso para navegar por la web pusieron de manifiesto un **problema**: las ***páginas webs era demasiado grandes para ser visualizadas en su totalidad por las pantallas de estos dispositivos.***

3.1. Entendiendo el VIEWPORT

Comencemos por entender qué es el **viewport**. Y como veremos, se trata de un concepto bien sencillo, pues corresponde con el ***área que está disponible en la pantalla del navegador***.

El ***viewport en un navegador en cualquier ordenador con sistemas tradicionales es igual al área de la ventana***, o mejor dicho, al área disponible para renderizar el documento web (o sea, le restamos toda la interfaz del navegador, como botones, barra de direcciones, barra de menús, barras de desplazamiento, etc.) Dicho de otro modo, es ***el área útil donde se mostrará la página web***.

Ahora pensemos en móviles o Tablets: Quizás sepas que cuando se ven las páginas en un dispositivo a menudo se reducen los contenidos, para conseguir que se ajusten al reducido espacio de la ventana del navegador. Es decir, para mostrar toda la página en el espacio disponible de la pantalla del dispositivo, se hace un escalado de la web, de modo que se ve todo en pequeñito.

El ***viewport cuando estamos hablando de dispositivos móviles***, ***no corresponde al tamaño real de la pantalla en píxeles, sino al espacio que la pantalla está emulando que tiene***. Por ejemplo, en un iPhone, aunque la pantalla en vertical tiene unas dimensiones de 320 píxeles, en realidad el dispositivo está emulando tener 980 píxeles. Esto hace que ciertas páginas web (optimizadas para navegadores de escritorio) quepan en una pantalla de 320 píxeles, porque en realidad el Safari para iOS está emulando tener un espacio de 980 píxeles (pero se verán muy pequeñas).

El ***viewport en estos casos es el espacio que el dispositivo emula tener, no la resolución real en píxeles que tiene la pantalla***.

Los desarrolladores somos capaces de ***alterar el viewport*** que viene configurado en el navegador, algo que resulta ***totalmente necesario si queremos que nuestra página se vea correctamente en dispositivos de movilidad.***

La foto mide 320 píxeles de ancho. En la parte de la derecha tendríamos la foto a tamaño real, que es como se vería si tuviéramos un viewport configurado a 320 píxeles de ancho. Pero al verla en un iPhone con un viewport configurado a 980 píxeles de ancho, la imagen se verá bastante más pequeña.

3.2. Configurando el VIEWPORT

HTML5 introduce la posibilidad de configuración del Viewport mediante la etiqueta **<meta>**. Esta admite los siguientes parámetros:

- **Width**: anchura virtual (emulada) de la pantalla o anchura del viewport.
- **Height**: altura virtual de la pantalla o altura del viewport.
- **Initial-scale**: la escala inicial del documento.
- **Minimum-scale**: la escala mínima que se puede poner en el documento.
- **Maximum-scale**: la escala máxima configurable en el documento.
- **User-scalable**: si se permite o no al usuario hacer zoom. Si le damos el valor "no", conseguiremos que el usuario no pueda hacer zoom en la página, manteniéndose las medidas y proporciones que nosotros hayamos definido al construir la web.

Se deberá incluir en la cabecera de todas las páginas web que constituyen el WebSite, de la forma siguiente:

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

Dónde:

- Con ***width=device-width*** conseguimos que el viewport sea igual a la anchura real de la pantalla del dispositivo, de modo que no se tratará de emular una pantalla mayor de lo que realmente es y veremos los píxeles reales.
- Con ***initial-scale=1*** conseguimos que no se haga zoom sobre el documento. Es bien simple, el contenido de la web no se transformará, ni se agrandará, ni se hará menor.

Como se puede observar, el ***META viewport*** no indica simplemente las ***dimensiones de la pantalla emulada***, sino también el ***nivel de zoom que se puede estar configurando inicialmente y el nivel de zoom que se permitiría*** tener.

4. RWD. EL GRID VIEW

Consiste en considerar que la ***página web está dividida en columnas*** "lógicas".

Esto nos va a servir de gran ayuda ya que nos va a ***facilitar situar elementos en la página web***

Es habitual usar un ***Grid View de 12 columnas***, con una anchura total de 100%, y que se comprimirá y expandirá a medida que cambia el tamaño de la ventana del navegador.

4.1. Construyendo un Grid View “Responsive”

En primer lugar nos aseguraremos de que todos los elementos HTML tengan la propiedad **box-sizing** con el valor **border-box.** De esta forma nos aseguramos de que el ***padding y el border de los elementos se incluyen en el ancho y alto calculado*** para todos ellos (según las cajas del modelo Box-Model de CSS).

Añadiremos a nuestro código la regla

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

A continuación ***creamos el Grid View de 12 columnas*** que nos permitirá tener más control sobre la página web:

- Calculamos en primer lugar el porcentaje de cada columna: 100%/12=**8,33%**
- A continuación, creamos una clase para cada columna, pudiendo elegir (es opcional) el formato de `col-<numero>` para el nombre de la clase que identificará a cada una de ellas:

```
.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%;}
```

- Todas esas columnas, deberán estar flotadas a la izquierda (aquí además podemos especificar características comunes a todas ellas):

```
[class*="col-"] {
float: left; OBLIGATORIAMENTE
padding: 15px;
border: 1pxsolidred;
}
```

- Es aconsejable también incluir todas las capas (`<div>`) definidos como de la clase `col-` dentro de un elemento `<div>` (lo definimos de clase “contenedor” – por ejemplo-).

  Dentro del contenedor estarán todas flotando a la izquierda, con lo que son eliminados del flujo de la página y cualquier otro elemento que les siguiese (por ejemplo un pie de la página) sería colocado como si no existieran (provocando un resultado “no deseado”). La solución es añadir una regla para introducir un elemento que “limpie” el flujo:

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

- Evidentemente, podemos añadir la modificación o modificaciones correspondientes para obtener un diseño más atractivo, como por ejemplo muestra el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>
* {
box-sizing: border-box;
}
.row:after {
content: "";
clear: both;
display: block;
}
[class*="col-"] {
float: left;
padding: 15px;
}
.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%;}
html {
font-family: "Lucida Sans", sans-serif;
}
.header {
background-color: #9933cc;
color: #ffffff;
padding: 15px;
}
.menu ul {
list-style-type: none;
margin: 0;
padding: 0;
}
.menu li {
padding: 8px;
margin-bottom: 7px;
background-color :#33b5e5;
color: #ffffff;
```

```
box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px
rgba(0,0,0,0.24);
}
.menu li:hover {
background-color: #0099cc;
}
</style>
</head>
<body>
<div class="header">
<h1>Chania</h1>
</div>
<div class="row">
<div class="col-3 menu">
<ul>
<li>The Flight</li>
<li>The City</li>
<li>The Island</li>
<li>The Food</li>
</ul>
</div>
<div class="col-9">
<h1>The City</h1>
<p>Chania is the capital of the Chania region on the island of
Crete. The city can be divided in two parts, the old town and
the
modern city.</p>
<p>Resize the browser window to see how the content respond to
the resizing.</p>
</div>
</div>
</body>
</html>
```

4.2. Añadiendo un BreakPoint.

Este paso permitirá visualizar correctamente las páginas en pantallas pequeñas (por ejemplo smartphones).

Para ello debemos usar las reglas `@media` vistas ya en clase. Estas permitirán añadir distintos “puntos de ruptura” (breakpoints) que harán que la página se comporte de forma distinta en cada “lado” del mismo.
Para añadir un BreakPoint en 768px (por ejemplo):

```
/* Para monitores de PC de escritorio: */
.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%;}

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

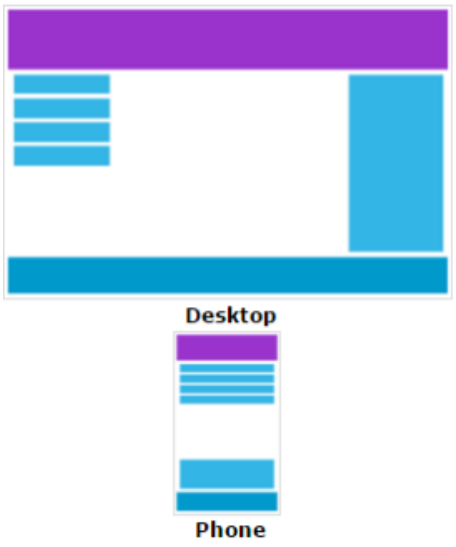
5. RWD. DESIGN FOR MOBILE FIRST

Mobile First es una filosofía, una manera de encarar el trabajo y una forma de facilitarnos la labor durante el diseño responsive, comenzando siempre por los dispositivos, con pantallas menores.

En realidad es un concepto bastante simple: ***diseñar pensando en los móviles primero.***

Cuando diseñas un sitio web responsivo debes tener en la cabeza una ***enorme cantidad de contextos en los que tu contenido va a ser consumido***. Para no perderse y para al final conseguir una solución optimizada para todo el mundo debemos aplicar la filosofía Mobile First.

Esto va a significar (entre otros muchos aspectos como de contenido y de diseño) que deberemos hacer ***cambios en nuestro código CSS***: En lugar de cambiar estilos cuando el ancho del viewport sea inferior a 768px, hay que cambiarlo cuando el ancho sea superior a 768px:



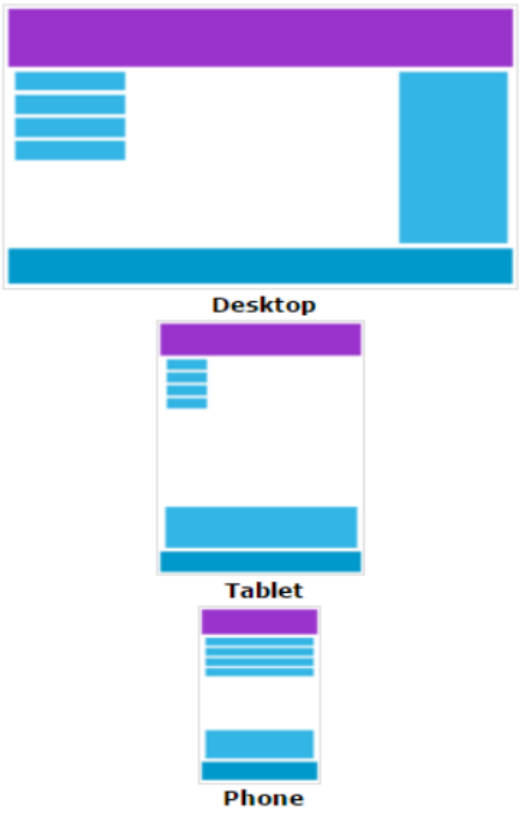

```
/* Para monitores de PC de escritorio: */
[class*="col-"] {width: 100%;}

/* Para SmartPhones: */
@media only screen and (min-width: 768px)
{
.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%;} }
```

Una vez comprendidos los cambios a realizar en nuestro código CSS para adaptarnos a la filosofía Mobile First, estamos en disposición de añadir un nuevo ***BreakPoint***.

5.1. Añadiendo otro BreakPoint.

Se pueden ***añadir tantos Breakpoints como se estimen necesarios***. En este apartado vamos a añadir un breakpoint para adaptar contenidos de la página a dispositivos entre Smartphones y Tablets.

Para ello, se añadirá ***una regla @media por cada breakpoint*** que intentemos implementar. Para el caso que nos ocupa, añadiremos @media para contemplar dispositivos con viewport en el rango `[600px, 758px [`




```
 /* Para SmartPhones*/
[class*="col-"] {width: 100%;}
@media only screen and (min-width: 600px)
{

/* Para 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%;}
}
/* Para monitores de PC */
@media only screen and (min-width: 768px)
{
.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%;}
}
```

Podría parecer extraño que tenemos dos conjuntos con clases iguales, pero es esto lo que nos da la posibilidad en nuestro código **HTML** de ***decidir y controlar qué va a pasar con las columnas en cada punto de interrupción***:

```
<div class="row">
<div class="col-3 col-m-3">...</div>
<div class="col-6 col-m-9">...</div>
<div class="col-3 col-m-12">...</div>
</div>
```

6. RWD. IMÁGENES

Si la propiedad ***width*** se establece al ***valor 100%***, la ***imagen automáticamente será responsive*** y se podrá escalar sin problemas.

El posible problema que esto presenta (porque para algunos usuarios puede que no lo sea) es que la imagen ***será escalada para ocupar el espacio de su elemento contenedor hasta el 100%***, y ***para ello puede ser "agrandada" con respecto a su tamaño original (con el posible problema de la pixelación), o bien "disminuida".***

Si la propiedad ***max-width*** se establece al ***valor 100%***, la ***imagen automáticamente será responsive*** y se podrá escalar sin problemas.

La diferencia con el apartado anterior es que ***la imagen nunca será agrandada por encima de su resolución original***, es decir, no tendrá por qué ocupar el ancho total el elemento contenedor que la incluye.

7. RWD. IMÁGENES DE FONDO DE UNA CAJA DEL BOX-MODEL

Las imágenes de fondo de una caja en el modelo de Box-Model también pueden responder a la hipotética posibilidad de ser escaladas y redimensionadas para su visualización en distintos dispositivos.

Podemos encontrar estas posibilidades:

- Si la propiedad ***background-size*** está establecida al valor "***contain***", esta ***podrá ser escalada para adaptarse al tamaño de su elemento contenedor pero manteniendo su relación de aspecto.***
- Si la propiedad ***background-size*** está establecida al valor "***100% 100%***", esta ***se extenderá hasta cubrir todo el área de contenido de su elemento contenedor aunque para ello no se mantenga su relación de aspecto***
- Si la propiedad ***background-size*** está establecida al valor "***cover***", ***la imagen de fondo se escalará para cubrir toda el área de contenido.*** Observe que se ***mantiene la relación de aspecto***, y ***alguna parte de la imagen de fondo puede ser cortada***

8. RWD. DIFERENTES IMÁGENES PARA DISTINTOS DISPOSITIVOS

Una imagen de gran tamaño puede ser perfecta para una pantalla de PC, pero inservible (por sus dimensiones) en un dispositivo pequeño. ¿Por qué cargar una imagen de gran tamaño cuando se tiene que escalar de alguna forma de todos modos? Una solución es usar `@media` para mostrar imágenes diferentes en diferentes dispositivos:

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

```
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>
/* Para width menores que 400px: */
body {
background-repeat: no-repeat;
background-image: url('img_smallflower.jpg');
}
/* Para width de 400px y superiores: */
@media only screen and (min-width: 400px) {
body {
background-image: url('img_flowers.jpg');
}
}
</style>
</head>
<body>
<p style="margin-top:360px;">Resize the browser width and the
background image will change at 400px.</p>
</body>
</html>
```