Posición y comportamiento de contenedores en CSS

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Nov - Dic
- iPasen - cjaedia071@g.educaand.es

Carmelo José Jaén Díaz

Índice

OBJETIVO

- Analizar y seleccionar los colores y las tipografías adecuados para la visualización en la pantalla.
- Utilizar marcos y tablas para presentar la información de manera ordenada.
- Identificar nuevos elementos y etiquetas en HTML5.
- Reconocer las posibilidades de modificar etiquetas HTML.
- Valorar la utilidad de las hojas de estilo para conseguir un diseño uniforme en todo el sitio web.

GLOSARIO

Formularios. Documentos interactivos utilizados para recoger información en un sitio web. Esta información es enviada al servidor, donde es procesada. Cada formulario contiene uno o varios tipos de controles que permiten recolectar la información de varias formas diferentes.

Fuentes seguras. Fuentes tipográficas que los usuarios tenían instaladas por defecto en su dispositivo. En la actualidad, gracias a que la mayoría de los navegadores soportan la directiva @font-face, es posible utilizar casi cualquier tipografía a través de Google Fonts.

Guías de estilo. Documentos con directrices que permiten la normalización de estilos. En estas guías se recogen los criterios y normas que debe seguir un proyecto; de esta forma se ofrece una apariencia más uniforme y atractiva para el usuario.

HTML. Lenguaje de marcado de hipertexto utilizado en las páginas web. Este tipo de lenguaje presenta una forma estructurada y agradable, con hipervínculos que conducen a otros documentos y con inserciones multimedia (sonido, imágenes, vídeos...).

GLOSARIO

HTML5. Última versión del lenguaje para la programación de páginas web HTML. Los sitios implementados con este lenguaje solo pueden visualizarse correctamente en los navegadores más actuales.

Legibilidad. Cualidad deseable en una familia tipográfica. Se trata de la facilidad de la lectura de una letra. Esta cualidad puede venir determinada por varios parámetros como el interletrado, el interpalabrado o el interlineado.

Marcos. Son las ventanas independientes incorporadas dentro de la página general. Gracias a ellos, cada página quedará dividida en varias subpáginas, permitiendo realizar un diseño más organizado y limpio a la vista. Con HTML5 ha quedado obsoleto.

Tipografía. Se trata del tipo de letra que se escoge para un determinado diseño. Según la RAE, significa ”modo o estilo en que está impreso un texto” o ”clase de tipos de imprenta”.

INTRODUCCIÓN

En el proceso de creación de una web es imprescindible organizar elementos como imágenes, textos o tablas. Para ello, necesitaremos conocer los elementos de ordenación y las propiedades que nos ayudan a organizar todos los componentes. Las propiedades más importantes se definen en la siguiente tabla y se explican a continuación.

POSICIÓN Y COMPORTAMIENTO DE CONTENEDORES

Propiedad	Descripción	Valores
`display`	Comportamiento del contenedor	inline | block | inline-block | none
`position`	Esquema de posicionamiento	static | relative | absolute | fixed | sticky
`top right bottom left`	Desplazamiento de la caja respecto al borde superior, derecho, inferior o izquierdo	longitud | porcentaje | auto
`float`	Posicionamiento flotante	left | right | none
`clear`	Control de cajas adyacentes a las float	none | left | right | both

POSICIÓN Y COMPORTAMIENTO DE CONTENEDORES

Propiedad	Descripción	Valores
`z-index`	Nivel de la capa	auto | número entero
`box-sizing`	Control de bordes y relleno en el comportamiento del contenedor	content-box | border-box
`visibility`	Muestra u oculta un elemento ocupando el espacio	visible | hidden

POSICIÓN Y COMPORTAMIENTO DE CONTENEDORES

Las propiedades `top`, `bottom`, `left` y `right` se utilizan para posicionar elementos en una página web cuando se está usando posicionamiento mediante las propiedades de CSS como `position: relative`, `position: absolute`, `position: fixed` o `position: sticky`; estas propiedades definen distancias desde los bordes del contenedor de referencia (que puede ser el contenedor padre o la ventana del navegador) y sus valores pueden expresarse en unidades como píxeles (`px`), porcentaje (`%`) en relación al contenedor padre, unidades relativas al tamaño de fuente como `em` o `rem`, entre otras.

DISPLAY

Valores: **none | inline | block | inline-block**

- **none**: los elementos se ocultan y no se muestra el espacio reservado.
- **inline**: los elementos se muestran en la misma línea (respetando el flujo) y no se aceptan las propiedades width, height ni márgenes verticales.
- **block**: los elementos se muestran en líneas independientes y se aceptan las propiedades width, height y márgenes verticales.
- **inline-block**: su comportamiento es una mezcla entre los dos anteriores, los elementos se muestran en la misma línea (respetando el flujo) y se aceptan las propiedades width, height y márgenes verticales.

DISPLAY

CÓDIGO HTML:

```
    <h3>display: none</h3>
        <p class="a">Bloque1 </p>
        <p class="a">Bloque2 </p>
        <p class="a">Bloque3 </p>
     <h3>display: inline</h3>
        <p class="b">Bloque1 </p>
        <p class="b">Bloque2 </p>
        <p class="b">Bloque3</p>
    <h3>display: block</h3>
        <p class="c">Bloque1 </p>
        <p class="c">Bloque2 </p>
        <p class="c">Bloque3 </p>
     <h3>display: inline-block</h3>
        <p class="d">Bloque1 </p>
        <p class="d">Bloque2 </p>
        <p class="d">Bloque3 </p>
```

CÓDIGO CSS:

```
    .a { display: none; }
    .b { display: inline; width: 100px;
height: 50px;}
    .c { display: block; }
    .d { display: inline-block; width:
100px; height: 50px;}
    p  { color: purple; border:
dotted;}
```

DISPLAY

display: none

display: inline

Bloque1 Bloque2 Bloque3

display: block

Bloque1

Bloque2

Bloque3

display: inline-block

Bloque1 Bloque2 Bloque3

POSITION

Valores: **static | relative | absolute | fixed**

- **static**: los elementos se posicionan de acuerdo al flujo normal de la página. Es la **posición natural de los elementos**. No son afectados por las propiedades `top`, `bottom`, `left` y `right`.
- **relative**: los elementos se posicionan de forma relativa a su posición normal.
- **fixed**: los elementos se posicionan de forma relativa a la ventana del navegador. Su posición permanece fija aunque se desplace la ventana.
- **absolute**: los elementos se posicionan de forma relativa al primer elemento padre que tenga una posición distinta a static. Si un elemento con `position: absolute` no tiene un contenedor padre posicionado (con `position: relative`, `absolute`, `fixed` o `sticky`), entonces se posicionará en relación a la ventana del navegador.
- **sticky**: los elementos son posicionados de forma relativa hasta que su bloque contenedor alcanza un límite establecido.

POSITION.
position: static

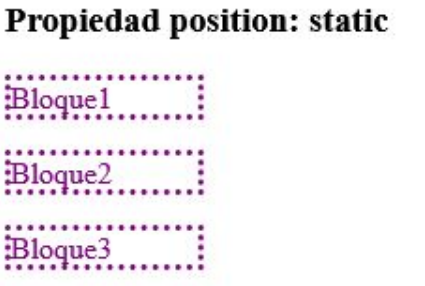
Nota: *las propiedades top, bottom, left y right no están permitidas con «position: static».*

CÓDIGO HTML:

```
<h3>Propiedad position: static</h3>
<p class="a">Bloque1 </p>
<p class="b">Bloque2 </p>
<p class="c">Bloque3 </p>
```

CÓDIGO CSS:

```
   .a { position: static; top: 800px;
left: 400px;} /*Como puedes ver, en
position static no funcionan las
propiedades top, bottom, left y right*/
   .b { position: static; }
   .c { position: static; }
   p  { width: 100px; color: purple;
border: dotted;}
```



POSITION.
position: relative

Nota: *las propiedades left: 20px y top: 10px el elemento se desplaza 20px hacia la derecha y 10 px hacia abajo desde su posición por defecto (sin eliminar el hueco de su posición por defecto).*

CÓDIGO HTML:

```
<h3>Propiedad position: relative</h3>
<p>Bloque1 </p>
<p class="bloque2">Bloque2 </p>
<p>Bloque3 </p>
```

CÓDIGO CSS:

```
.bloque2 { position: relative;
left: 20px; top: 10px;  }
p  { width: 100px; color: purple;
border: dotted;}
```

POSITION.
position: fixed

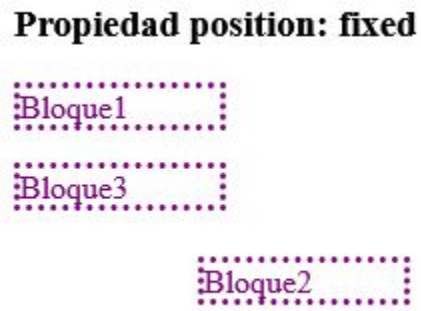
Nota: *los elementos con* `position: fixed` *toman como referencia la ventana del navegador y permanecen fijos.*

CÓDIGO HTML:

```
<h3>Propiedad position: fixed</h3>
<p>Bloque1 </p>
<p class="bloque2">Bloque2 </p>
<p>Bloque3 </p>
```

CÓDIGO CSS:

```
.bloque2 { position: fixed; top: 130px; left: 100px; }
 p { width: 100px; color: purple; border: dotted;}
```



POSITION.
position: absolute

Nota: *el elemento «.b» deja de seguir la posición del flujo normal de la página, sin crearse espacio alguno para el elemento, y se posiciona de forma relativa al primer elemento padre que tiene una posición distinta a static, en este caso el elemento «.a». Si no hubiese ningún elemento padre con propiedad distinta a static, se ubicaría de forma relativa al contenedor inicial (puede ser <body>). Haz la prueba cambiando el valor de position a «static» en el elemento a. Como puedes ver, su posición final está definida por los valores de top, right, bottom, y left.*

CÓDIGO HTML:

```
   <div class="a">
     <p>Contenedor con position:
relative</p>
      <p>Contenedor con position:
relative</p>
      <p>Contenedor con position:
relative</p>
      <p>Contenedor con position:
relative</p>
      <p>Contenedor con position:
relative</p>
      <div class="b"><p>Contenedor con
position: absolute</p></div>
   </div>
```

POSITION.
position: absolute

CÓDIGO CSS:

```css
.a {
  position: relative;
/*cambia este valor a static para
ver la diferencia*/

  width: 300px;
  height: 300px;
  border: 2px dotted purple;
}
```

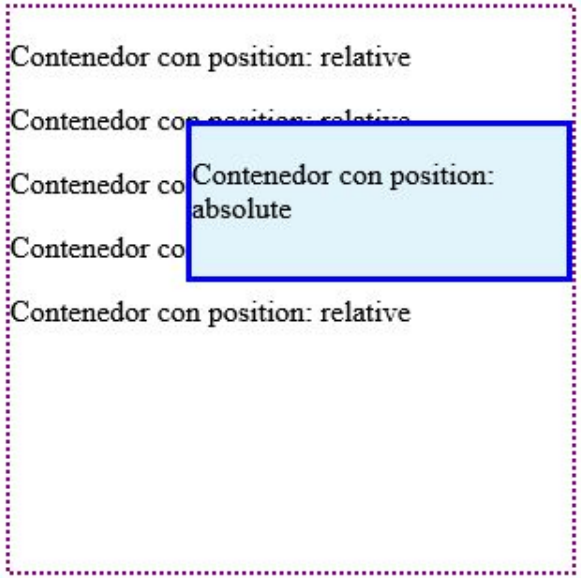
```css
.b {
  position: absolute;
/*Se posiciona de forma relativa al
primer elemento padre que tiene una
posición distinta a static*/

  background-color: #e1f4fc;
  top: 60px;
  right: 0px;
  width: 200px;
  height: 80px;
  border: 3px solid blue;
}
```

POSITION.
position: absolute

SALIDA CON:

```
.a { position: relative;
```

```
.b {
  position: absolute;
/*Se posiciona de forma relativa al primer elemento padre que tiene una posición distinta a static*/
```

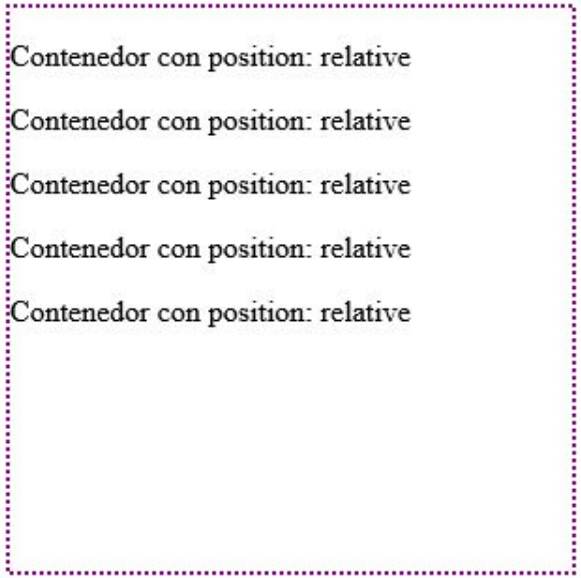


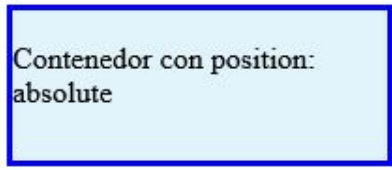
POSITION.
position: absolute

SALIDA CON:

```
.a { position: static;
```

```
.b {
  position: absolute;
/*Se posiciona de forma relativa al primer elemento padre que tiene una posición distinta a static*/
```





POSITION.
position: sticky

La posición **sticky** se usa cuando queremos que un elemento tenga una posición relativa hasta un punto y que luego cambie a una posición fija, usando solo CSS sin necesidad de código JavaScript.

Por ejemplo si tenemos un banner de publicidad flotante en el sidebar y nos interesa que una vez aparezca al hacer scroll se mantenga fijo y visible.

La propiedad `position: sticky` se utiliza junto con valores de desplazamiento (`top`, `right`, `bottom`, `left`) para definir cuándo y dónde debe volverse fijo el menú o contenedor específico que queramos.

A continuación, se muestra otro ejemplo donde los encabezados de una lista alfabética se posicionan en la parte alta del documento HTML.

POSITION.
position: sticky

CÓDIGO CSS:

```css
p {
  position: -webkit-sticky;
  position: sticky;
  top: 0px;
  background-color: #e1f4fc;
  padding: 20px;
  font-weight: bold;
}
ul{
  margin: 20px 0;
}
```

```css
li{
  padding-left: 20px;
}
```

POSITION.
position: sticky

La salida del código anterior puede observarse, tanto en la siguiente imagen como en el siguiente enlace:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/QwLLNXQ

A

aguacate
ahuyama
avena
azucar
arracacha
arequipe
anón
ají
ajonjolí
atún

B

brocoli
berros
banano

FLOAT

Valores: **left | right | none**

Cuando a un elemento HTML se le aplica un estilo con la propiedad float, el elemento sale del flujo normal y aparece posicionado a la izquierda o a la derecha de su contenedor, donde el resto de elementos de la página se posicionarán alrededor.

FLOAT

Código HTML:

```
<div class="a">
    <div></div>
</div>

<div class="a">
    <div></div>
</div>

<div class="b">
    <div></div>
</div>
<div class="b">
    <div></div>
</div>
```

```
<div class="c">
    <div></div>
</div>
<div class="d">
    <div></div>
</div>
```

FLOAT

Código CSS:

```css
.a {
  float: left;
  padding: 10px;
}
.a div{
  height: 100px;
  width: 100px;
  background-color: #2980B9;
}
```

```css
.b {
  float: right;
  padding: 10px;
}
.b div{
  height: 100px;
  width: 100px;
  background-color: #17A589;
}
```

FLOAT

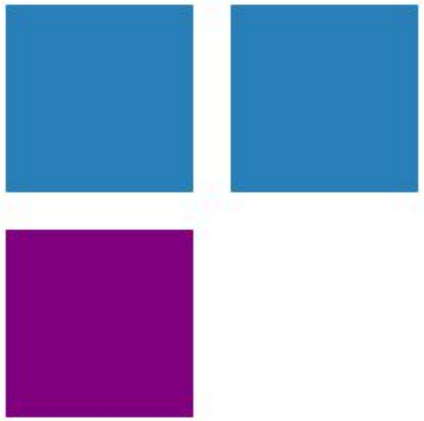
Código CSS:

```css
.c {
  float: right;
  padding: 10px;
}
.c div{
  height: 100px;
  width: 100px;
  background-color: red;
}
```

```css
.d {
  clear: both;
/* Cambia este valor a float:left, float:right para ver resultados */
  padding: 10px;
}
.d div{
  height: 100px;
  width: 100px;
  background-color: purple;
}
```

FLOAT

```
.d { clear: both;
```



```
.d { float: left;
```

CLEAR

Valores: **none | left | right | both**

La propiedad `clear` establece si un elemento debe estar al lado de los elementos flotantes que lo preceden o si debe situarse bajo de ellos. Se suele utilizar para restaurar el flujo normal del documento y así los elementos dejan de flotar hacia la izquierda, la derecha o ambos lados.

En el siguiente ejemplo, se crea un contenedor aplicándole la propiedad float con el valor left.
Después, se sitúa un texto bajo del contenedor creado utilizando la propiedad clear.

CLEAR

Código HTML:

```
<div class="a">
    <div></div>
</div>
<div class="b">
    <div></div>
</div>
<h4>Texto sin propiedad clear</h4>
<p>Texto con propiedad clear:both</p>
```

CLEAR

Código CSS:

```css
.a {
  float: left;
  padding: 10px;
}
.a div{
  height: 100px;
  width: 100px;
  background-color: #2980B9; /*Azul*/
}
```

```css
.b {
  float: right;
  padding: 10px;
}
.b div{
  height: 100px;
  width: 100px;
  background-color: #17A589;
}
p{
  clear: both;
}
```

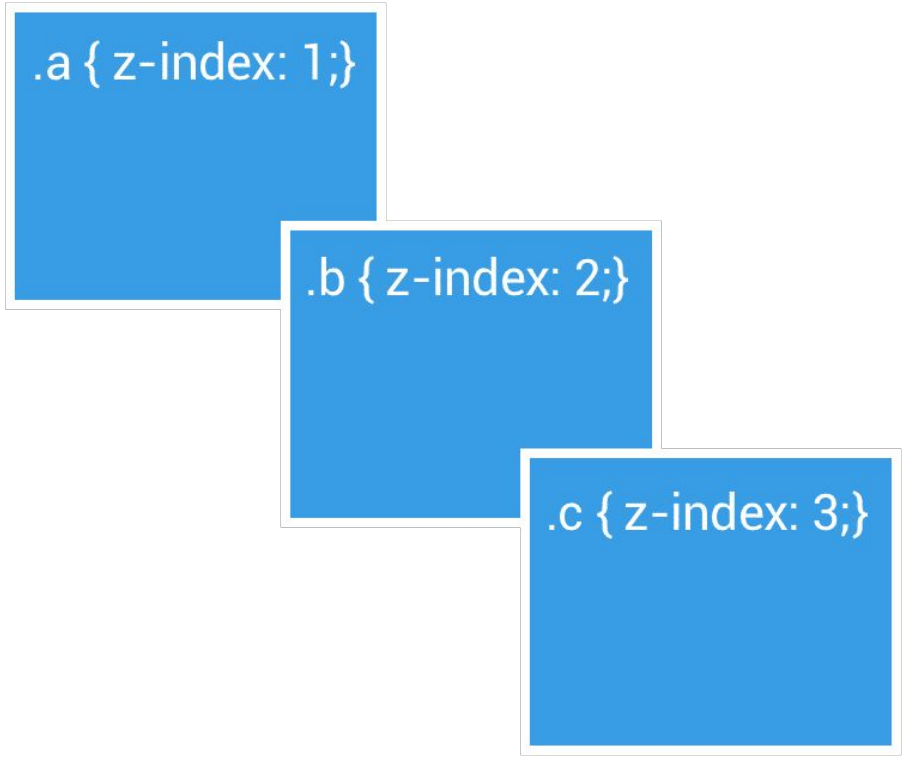
Texto sin propiedad clear

Texto con propiedad clear:both

Z-INDEX

Valores: **auto | número entero**

Mediante el atributo `z-index` podemos organizar cada uno de los elementos del contenido de una página web. Como se puede apreciar en la imagen, cuando varios elementos se superponen, **los elementos con mayor valor z-index se sitúan por encima de los que tienen menor valor.**



Z-INDEX

Código HTML:

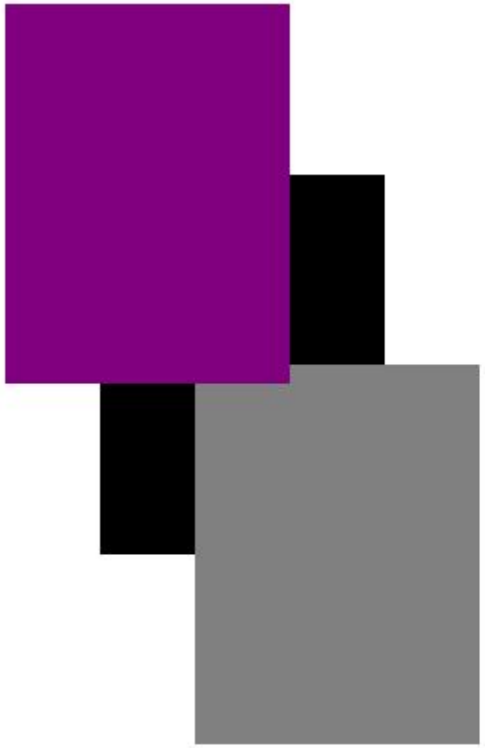
```
<h3>Propiedad z-index</h3>
<div class="a"></div>
<div class="b"></div>
<div class="c"></div>
```

Código CSS:

```
.a { width: 150px; height: 200px;
background-color: purple;
position: relative; z-index: 3;}

.b { width: 150px; height: 200px;
background-color: black;
position: relative; left: 50px;
top: -110px; z-index: 1;}

.c { width: 150px; height: 200px;
background-color: grey;
position: relative; left: 100px;
top: -210px; z-index: 2;}
```

Z-INDEX

Propiedad z-index



BOX-SIZING

Por defecto en el modelo de cajas de CSS, el ancho y alto asignado a un elemento es aplicado solo al contenido de la caja del elemento. Si el elemento tiene algún borde (border) o relleno (padding), este es entonces añadido al ancho y alto del tamaño de la caja o contenedor. Esto significa que cuando se define el ancho y alto, se tiene que ajustar el valor para permitir cualquier borde o relleno que se pueda añadir.

BOX-SIZING

Valores: **content-box | border-box**

- `content-box` es el comportamiento CSS por defecto para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 pixeles, la caja del contenido del elemento tendrá 100 pixeles de ancho, y el ancho de cualquier borde o relleno será añadido al ancho final desplegado.
- `border-box` toma en cuenta cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 pixeles. Esos 100 pixeles incluirán cualquier borde o relleno que se añada, y la caja de contenido se encogerá para absorber ese ancho extra. Esta propiedad es especialmente útil para redimensionar cualquier elemento.

BOX-SIZING

En el siguiente ejemplo, se crea un contenedor que ocupe el 100% del ancho de la pantalla.
A continuación, se posicionan tres imágenes en línea y se define que cada imagen ocupe el 33,333%.
Observa que el conjunto ocupa el 100% de la pantalla.

Si a continuación dotamos a las imágenes de un padding o relleno, el conjunto ocupará más del 100%. En este punto podríamos establecer un "`box-sizing: border-box`" para incluir en el conjunto el relleno definido.

Importante: hay que añadir los prefijos para los navegadores (se estudian en el *Capítulo 4: CSS avanzado*).

```
box-sizing: border-box;
-webkit-box-sizing: border-box;
```

BOX-SIZING

Código HTML:

```
    <div class="img-container">
      <img
src="https://cdn.pixabay.com/photo/2019/11
/02/21/45/maple-leaf-4597501_960_720.jpg">
      </div>
      <div class="img-container">
      <img
src="https://cdn.pixabay.com/photo/2019/11
/02/21/45/maple-leaf-4597501_960_720.jpg">
      </div>
      <div class="img-container">
      <img
src="https://cdn.pixabay.com/photo/2019/11
/02/21/45/maple-leaf-4597501_960_720.jpg">
      </div>
```

Código CSS:

```
.img-container {
  box-sizing: border-box;
  -webkit-box-sizing: border-box;
  float: left;
  width: 33.33%;
  padding: 5px;
}
.img-container img{
  width: 100%;
}
```

BOX-SIZING



VISIBILITY

La propiedad `visibility` indica si un elemento es visible o permanece oculto (ocupando el mismo espacio).

Valores: **visible | hidden**

En el siguiente ejemplo, se puede observar la diferencia principal entre `display: none` que no reserva el espacio del elemento y `visibility: hidden` que sí lo hace.

VISIBILITY

Código HTML:

```
<div>
  <div id="hide-me">¿Me ves?
  </div>

  <p>Esconder/mostrar con
   <button id="displayNone">display: none / block</button>
   <button id="visibilityHidden">visibility: hidden / visible</button>
  </p>

<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  </p>
<div>
```

VISIBILITY

Código CSS:

```css
body {
  background: #a64b00;
  color: #eee;
  font-family: verdana;
}

p {
  padding: 0;
  margin: 0 0 15px 0;
  font-size: 12px;
  width: 1000px;
}
```

```css
#hide-me {
  width: 100px;
  height: 100px;
  background: #bf7130;
  display: block;
  padding: 10px;
  margin: 0 15px 0 0;
  font-size: 14px;
  font-weight: bold;
  float: left;
  text-align: center;
  background: rgba(0, 0, 0, .15);
}
```

VISIBILITY

Código JS:

```
$('#displayNone').click(function(e) {

// Resetear, por si acaso has estado
jugando con la otra propiedad
      $('#hide-me').css('visibility',
'visible');

      if( $('#hide-me').is(":visible") ) {
        $('#hide-me').css('display',
'none');
      } else {
        $('#hide-me').css('display',
'block');
      }
    });
```

```
$('#visibilityHidden').click(function(e) {

      // Resetear, por si acaso has estado
jugando con la otra propiedad
      $('#hide-me').css('display',
'block');

      if( $('#hide-me').css('visibility')
!= 'hidden' ) {
        $('#hide-me').css('visibility',
'hidden');
      } else {
        $('#hide-me').css('visibility',
'visible');
      }
    });
```

VISIBILITY

La salida del código anterior puede observarse, tanto en la siguiente imagen como en el siguiente enlace:

https://codepen.io/Carmelo-Jos-Ja-n-D-az/pen/yyBBJMG