ISSD

-Asc-
Analista de Sistemas

I2

Informática 2

Módulo didáctico - 2015

CREANDO
ESTILOS CON CS3

Unidad 2

Clase 5

CS3

Al finalizar esta clase ya podrás...

| Comprender los selectores avanzados y su uso.

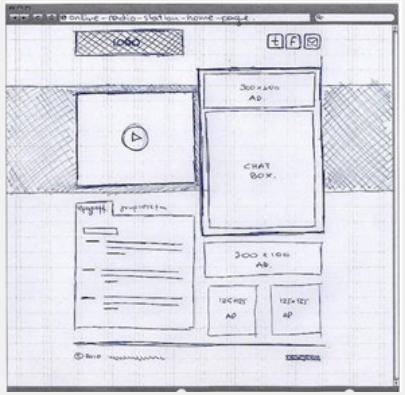
| Entender el uso y la definición de los diferentes prefijos para los navegadores en los estilos CSS3.

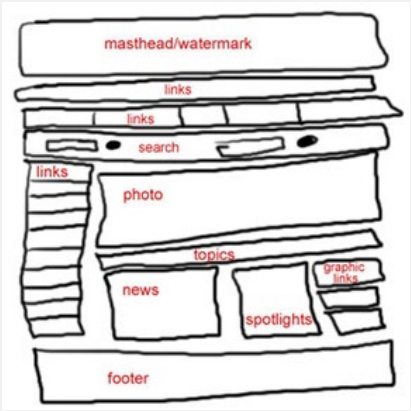
| Comprender la forma de maquetar e implementar con Capas en el diseño de sitios webs.

| Conocer de qué manera se deben aplicar las propiedades y atributos de las capas para lograr sitios de calidad profesional.

Muchas empresas -como habrás visto en las solicitudes laborales- al tomar a sus programadores les solicitan un excelente manejo en la maquetación del sitio y uso de estilos.

Para dominar esto deberás prepararte bien en el uso de estilos. Te dejo con la temática de la clase ¡Espero que la disfrutes!





CSS3... ¡Allá vamos!

En esta clase continuaremos con CSS3, para ello te recomiendo que no dejes de practicar los ejemplos y codificarlos, es la única manera con la que vas a poder adquirir habilidad en la codificación y depuración del código. Te adelantamos algo de lo que veremos:

- Prefijos para los navegadores en los estilos CSS3.
- Maquetación e implementación con Capas para el diseño los sitios.
- Propiedades y atributos de las capas.

Es sumamente importante que leas atentamente -y tantas veces como lo necesites- esta clase, lee hasta que los conceptos te queden bien claros y aprendidos. Otra posibilidad, es que estudies en grupo para que puedas compartir conocimientos con tus compañeros.

Como siempre... **¡te deseo mucha suerte!**

Desempeño de Exploración

Los alumnos nos presentaron sus inquietudes para conocer y profundizar más sobre estilos, por eso propondremos el debate de cuan utilizado son los estilos en la actualidad.

Desempeño de Investigación

Te proponemos aplicar CSS3 a los ejercicios vistos hasta el momento, para eso vamos a armar grupos de investigación a fin de ampliar conocimientos. Luego mostraremos entre todos los trabajos realizados y daremos nuestras opiniones y comentarios. Además expondremos los trabajos explicando los estilos utilizados.

Selectores básicos

¿Cuáles son los selectores básicos?

Selector Universal

A este selector lo usamos cuando queremos que todos los elementos tengan un estilo definido. Por ejemplo, si quisiéramos que todos los elementos que tengan texto tuvieran el mismo tipo de fuente usaríamos * para definir la fuente, y de esa forma todos los textos que tengamos en nuestro sitio tendrán igual fuente.

```
* {  margin: 0;  padding: 0;}
```

Etiqueta

Este selector es el más usado. Cabe aclarar que cuando hablamos de Elemento nos vamos a referir a cualquier etiqueta HTML (span, div, img, etc). Si quisiéramos definir un estilo para todas las imágenes definiríamos un estilo img{} para que afecte a las imágenes, un p{} para los párrafos, y así con cada etiqueta que quisiéramos.

```
h1 {  color: red;}
```

En vez de definir dos veces los mismos estilos para dos etiquetas diferentes podemos agruparlas usando un espacio entre cada nombre de la etiqueta, de esa forma ahorramos espacio. Por ejemplo, si quisiéramos que los H1, H2, y H3 tuvieran el mismo color de texto, lo lógico sería hacer un estilo para H1{}, luego para H2{}, y para H3{}, pero para hacernos la vida más fácil podemos definir un sólo estilo que afecte a las tres etiquetas quedando como H1, H2, H3{}

```
h1, h2, h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
```

Recordá que los selectores le dicen al navegador web qué directivas o directivas de una página tienen que someterse al estilo.

Selectores de Clases

También podemos usar el "." para definir un estilo y que éste aplique a todas las etiquetas HTML que tengan la propiedad 'class="nombreClase"'. Este tipo de estilo se utiliza de manera tal que el programador pueda crear su propio estilo y aplicarlo cuantas veces quiera.

```
.aviso {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6feda;
}
```

```
.error {
  color: #930;
  font-weight: bold;
}
<h1 class="error">...</h1>

<p class="aviso">...</p>
```

Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden utilizar el mismo valor en el atributo class.

Los selectores de clase son imprescindibles para diseñar páginas web complejas, ya que permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores permiten reutilizar los mismos estilos para varios elementos diferentes.

Selectores ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso. El selector de ID permite seleccionar un elemento de

la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página. La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.).

```
#destacado { color: red; }

<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

En una misma página, el valor del atributo id debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de id. Sin embargo, el atributo class no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo class.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML. Podemos tener la siguiente combinación:

```
p.destacado { color: red }
```

Selector descendiente

El selector descendiente se utiliza para encontrar objetos dentro de otros objetos. Por ejemplo, si quisiéramos darle un estilo a cualquier link dentro de algún párrafo, usaríamos:

```
p a {color:black;}
```

Selector de hijos (>)

Este selector funciona de manera similar al selector descendente, sin embargo dista mucho en funcionamiento. La diferencia recae en que, el selector descendiente no tiene en cuenta si es que el objeto a seleccionar está directamente dentro del objeto padre, o si –de hecho– está dentro de un objeto, que está dentro de otro objeto, que está dentro del objeto padre.

Te recomiendo que uses el Selector ID cuando quieras aplicar estilo a un sólo elemento específico de la página y el Selector de Clase cuando desees aplicar estilo a varios elementos diferentes de la página HTML.

En el siguiente código, podemos ver que todos los links que estén dentro del h1, cambiarán a un color negro, incluyendo el que está dentro del elemento span:

```
< h1 >
 < a href="#" / >
 < span >< a href="#" / >< /span >
< /h1 >
h1 a {color:#000;}
```

Sin embargo, en el código de ejemplo de más abajo, podemos ver el uso del selector de hijos. En este caso, el selector solo dará el color negro a los links que estén directamente relacionados con el h1. Es decir, el link que está dentro de un elemento span NO cambiará de color, pues no es un hijo directo de h1.

```
h1 > a {color:#000;}
```

Es realmente importante entender y practicar el buen uso de esta regla, pues puede ser de mucha utilidad para evitar la sobre-utilización de clases e ids.

Selector adyacente

El selector adyacente es –probablemente– el selector más específico de todos los presentes en CSS. Su sintaxis es:

```
elemento1 + elemento2 { ... }
```

El ejemplo expuesto como se muestra más arriba, selecciona todo "elemento2" que sea hijo directo del mismo elemento padre que "elemento1", pero que –además– esté justo después del "elemento1".

```
h1 + h2 {color:#000;}
< body >
 < section >
 < h1 >Elemento1< /h1 >
 < h2 >Elemento2< /h2 >
 < h2 >Elemento3< /h2 >
 < /section >
 < h2 >< /h2 >
< /body >
```

El "elemento2" (h2) debe tener el mismo padre que el "elemento1" (h1), siendo este, el section.

Es muy importante entender y practicar el buen uso de la regla que corresponde al Selector de Hijos, pues puede ser de mucha utilidad para evitar la sobre-utilización de clases e IDs.

Selector de atributos

Este tipo de selector nos permite entregarle estilos a cualquier elemento que comparta algún atributo específico. Los tipos de atributos pueden ser cualquiera de los siguientes:

- **[nombre_atributo]**: selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, sin importar su valor.

- **[nombre_atributo=valor]**: selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor específico, el cual debe ser igual a "valor".

- **[nombre_atributo~=valor]**: selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de sus valores es equivalente a "valor".

- **[nombre_atributo|=valor]**: selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con guiones, pero que comience con "valor".

En el siguiente ejemplo, se seleccionan todos los links que tengan alguna clase (independiente de cual):

```
a[class] { color: blue; }
```

En este otro ejemplo, se seleccionan todos los links que tengan una clase "estaEsUnaClase":

```
a[class="estaEsUnaClase"] { color: blue; }
```

y en este último ejemplo, se seleccionan todos los links que tengan un atributo "href" con el valor de "http://www.ejemplo.com":

```
a[href="http://www.ejemplo.com"] { color: blue; }
```

A continuación, se seleccionan todos los links que contengan al menos una clase llamada "estaEsUnaClase". En este caso no se seleccionan ni el segundo ni el tercer link, pues contienen otras clases. Sin embargo, el cuarto elemento link también se selecciona, pues –pese a contener varias clases–, tiene la que especificamos en la regla CSS.

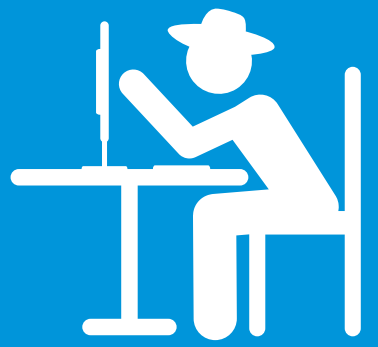
[nombre_atributo]

[nombre_atributo=valor]

[nombre_atributo~=valor]

[nombre_atributo|=valor]

#!*



Prefijo	Familia de navegadores a los que aplica
-webkit-	Chrome, Safari, Android, iOs
-moz-	Firefox
-o-	Opera
-ms-	Microsoft Internet Explorer

```
a[class~="estaEsUnaClase"] { color: blue; }
< a href="#" class="estaEsUnaClase" / >
< a href="#" class="otraClase" / >
< a href="#" / >
< a href="#" class="estaEsUnaClase
otraClase otraClase2 otraClase3" / >
```

Prefijos CSS de los navegadores

Se llaman prefijos de navegador o prefijos comerciales (vendor prefixes) a un prefijo que se antepone a una regla CSS destinado a que dicha regla sea leída y aplicada exclusivamente por un navegador concreto (por ejemplo Chrome) pero no por el resto de navegadores. El uso de prefijos suele aplicarse a propiedades que se encuentran en fase experimental o que aún no se han convertido en un estándar.

Al igual que los comentarios condicionales que se han venido usando específicamente para Microsoft Internet Explorer, los prefijos son un tipo de filtro que permite que una instrucción CSS se aplique específicamente a un navegador o familia de navegadores pero no a los demás. Sin embargo, a diferencia de los comentarios condicionales, existen prefijos específicos para todos los tipos de navegador. En el cuadro **(1)** se indican los prefijos más habituales. Te mostramos un ejemplo:

```
.micapa{
-webkit-transform: rotate(45deg);
-o-transform: rotate(45deg);
-ms-transform: rotate(45deg);
-moz-transform: rotate(45deg);
transform: rotate(45deg);
}
```

Las etiquetas DIV

¡A tener en cuenta para el uso de los Divs!

La etiqueta div sigue funcionando exactamente igual que lo hacía hasta el momento. La usaremos para definir bloques sin ningún tipo de significado, normalmente bloques que usaremos para maquetar correctamente la página o para agrupar elementos en principio sin querer dar un significado específico. Podemos insertar una capa a través de las etiquetas y , que como ya vimos, sirven para agrupar bloques de texto.

Cuando uses Divs deberás tener en cuenta:

| A través del atributo **id** se le da un nombre a la capa, y a través del atributo **style** se establecen el resto de propiedades de la capa.

| A través de las propiedades **left** (izquierda) y **top** (superior) se establece la posición de la capa respecto a los márgenes izquierdo y superior de la página. Pueden tomar un número como valor, acompañado de **px** cuando haga referencia a píxeles, y acompañado de **%** cuando haga referencia a un porcentaje.

| Para que la capa aparezca en la posición establecida, es necesario incluir también la propiedad **position** con el valor **absolute**. Si no se estableciera este valor, la capa se mostraría pegada al margen izquierdo, en la posición en la que hubiera sido insertada dentro del código.

| A través de las propiedades **width** (anchura) y **height** (altura) se establece el tamaño de la capa. Pueden tomar un número como valor, acompañado de **px** cuando haga referencia a píxeles, y acompañado de **%** cuando haga referencia a un porcentaje.

| A través de la propiedad **z-index** puede establecerse el índice de la capa dentro de la página. Una capa podrá ser solapada por aquellas capas cuyo índice sea mayor. Siempre es un valor numérico.

| A través de la propiedad **visibility** puede establecerse la visibilidad de la capa. Puede

tomar los valores **inherit** (se muestra la capa mientras la capa a la que pertenece también se esté mostrando), **visible** (muestra la capa, aunque la capa a la que pertenece no se esté viendo) y **hidden** (la capa está oculta).

| A través de las propiedades **layer-background-image** y **background-image** se puede establecer una imagen de fondo para la capa. La ruta y el nombre de la imagen han de aparecer entre paréntesis, después de la palabra **url**.

| A través de las propiedades **layer-background-color** y **background-color** se puede establecer un color de fondo para la capa. Ha de ser un número hexadecimal.

| A través de la propiedad **overflow** puede establecerse si se mostrará o no el contenido de la capa cuando no pueda ser visualizado en su totalidad, por ser la capa demasiado pequeña. Puede tomar los valores visible (se muestra todo el contenido de la capa, aunque esto implique hacer que la capa sea más grande), hidden (no es posible visualizar el contenido de la capa que no quepa en ella), scroll (se muestra la barra de desplazamiento, aunque el contenido de la capa pueda ser visualizado totalmente) y auto (se muestra la barra de desplazamiento cuando sea necesario).

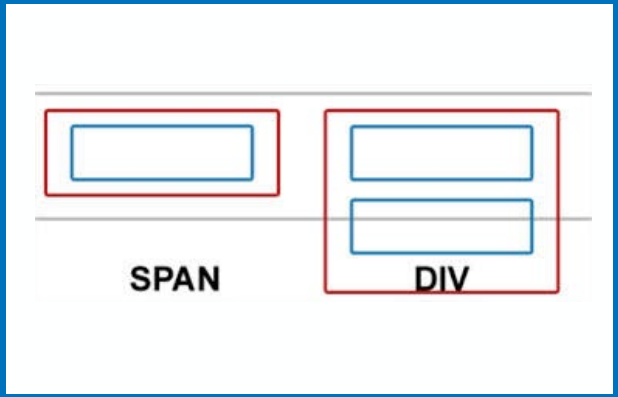
| A través de la propiedad **clip** puede establecerse el área de la capa que podrá ser visualizado. Lo que hace es recortar la capa, haciendo que partes de ella no sean visibles. Ha de especificarse la distancia de los márgenes de la capa entre paréntesis, después de la palabra url.

Más adelante detallaremos cada uno de estos atributos con sus correspondientes ejemplos.

¿Cuáles son las diferencias entre div y span?

Ambas etiquetas tienen las mismas propiedades, pero la diferencia varia en la función que tienen, ya que mientras la etiqueta **span** trabaja sólo como contenedor de línea, la etiqueta **div** trabaja como contenedor de bloque. La imagen 1 ejemplifica esto.

1



Recordá que aunque div y span tengan las mismas propiedades difieren en su función: la primera trabaja sólo como contenedor de línea, mientras que la segunda trabaja como contenedor de bloque.

La etiqueta span sólo funciona como contenedor de una línea, es decir, no se puede ampliar para ocupar un párrafo, por ejemplo, lo que si puede hacer div, puesto que puede expandirse lo que sea necesario, es quizá la razón más clara por la que para crear un sitio web, la estructura básica comienza con capas div.

Entonces ¿cuándo debo usar span?

Hay varios usos en cuanto a la implementación de la etiqueta span, por ejemplo:

| **Edición de frases en párrafos:** a veces queremos aplicar estilos a frases específicas de un párrafo o texto, por ejemplo:

```
Inicio<span class="highlight">Texto iluminado.</span> Fin del texto
```

| **Creación de botones:** esta es una buena idea, usar span para poder dar un padding a un enlace, creando el efecto de botón.

| **Agrupar diferentes elementos en una línea:** Por ejemplo, aunque no es tan común, podemos definir y acomodar elementos como imágenes, textos o enlaces en una sola línea, aplicar estilos para cada uno de ellos, y mostrarlos en línea sin que salten de una línea a otra, esto ayuda a tener una alineación buena y no tener contenido desalineado.

¿Cuándo se usa div?

| **Maquetación web:** para crear la estructura de tu sitio web, al poder contener todos los elementos es fácil crear la forma que va a tener tu sitio usando hojas de estilo.

| **Crear contenido flotante:** es la mejor opción si quieres crear un menú, imagen o algún tipo de elemento flotante ya que se ajusta muy bien a la propiedad CSS float, por lo que si ves algún elemento flotante o que se mueve es porque tiene una capa.

| **Contenedor de más contenedores:** Otros contenedores que podemos colocar con p, span y cualquier tipo de elemento HTML.

El uso de la etiqueta div es sencillo. Observemos este ejemplo:

Para crear un sitio web, la estructura básica comienza con capas div ya que ésta etiqueta puede expandirse lo que sea necesario (ampliarse para ocupar un párrafo por ejemplo).

```
<div>
<h1>Índice</h1>
Página principal<br />
Material multimedia<br />
Autores<br />
</div>
```

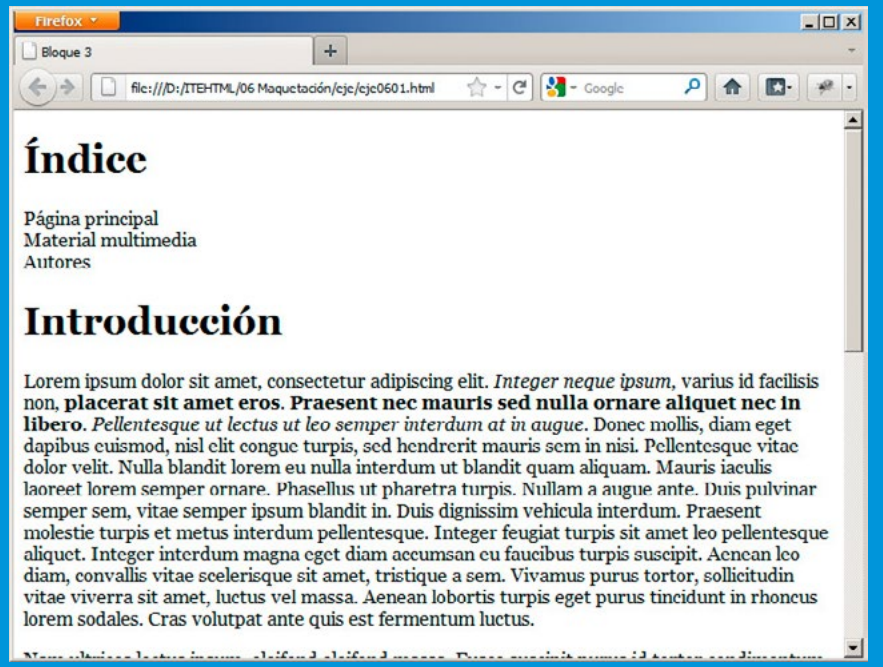
Hemos empleado la etiqueta para crear un bloque, que hará las veces de índice de contenidos. Visualmente la etiqueta no provoca ningún cambio, pero en la estructura interna del documento hemos aplicado una división muy importante. La figura 1 muestra el texto del ejemplo; no hay ninguna diferencia visible.

Probemos ahora a aplicar una modificación a la apariencia de ese bloque. Añadiendo un estilo de borde a la etiqueta ; quedaría así:

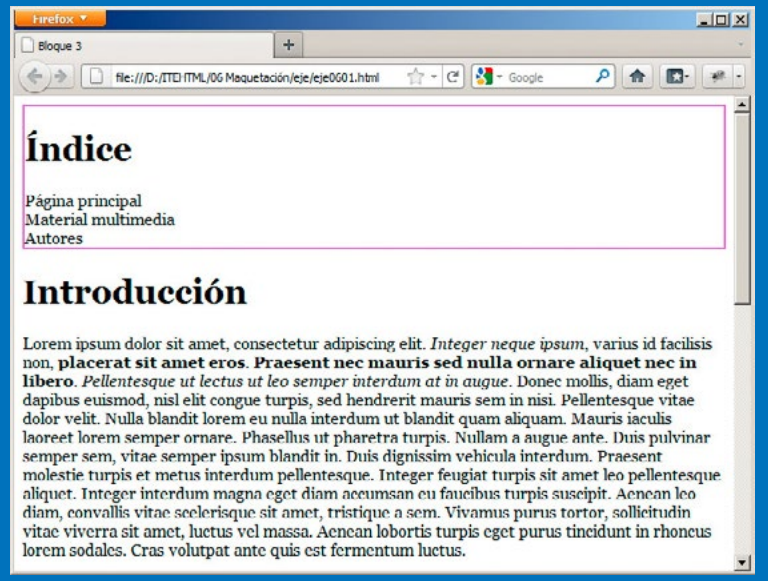
```
<div style="border: 2px solid rgb(204, 102, 204);">
<h1>Índice</h1>
Página principal<br />
Material multimedia<br />
Autores<br />
</div>
```

Observemos la figura 2; ahora sí que aparece un borde sólido de 2 píxeles alrededor de todo el espacio definido por la etiqueta div.

1



2



Las capas (div) no son más que unos recuadros, que pueden situarse en cualquier parte de la página, en los que podemos insertar contenido HTML. Dichas capas pueden ocultarse y solaparse entre sí, lo que proporciona grandes posibilidades de diseño.

Enlaces

Como ya sabés, los enlaces se especifican en HTML con la etiqueta <a>. Por lo tanto, podemos usar **a** como selector en CSS:

```
a {    color: blue;    }
```

Todo enlace tiene diferentes estados. Por ejemplo, **visitado** o **no visitado**. Puedes usar una pseudo-clase para asignar diferentes estilos a los enlaces visitados y no visitados.

```
a:link {color: blue;    }

a:visited {    color: red; }
```

Usa **a:link** y **a:visited** para enlaces visitados y no visitados, respectivamente. A los enlaces activos se les aplica la pseudo-clase **a:active**, y **a:hover** cuando el cursor se coloca o pasa sobre el enlace.
Vamos a repasar ahora cada una de las cuatro pseudo-clases con ejemplos y más explicaciones.

¡Ojo que los dos puntos : son parte de las pseudo clases!

En resumen:

| La pseudo clase **:link** se usa para enlaces que dirigen a páginas que el usuario no ha visitado.

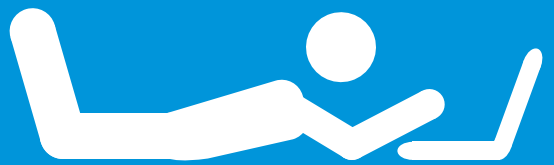
| La pseudo clase **:visited** se usa para enlaces que dirigen a páginas que el usuario ya ha visitado.

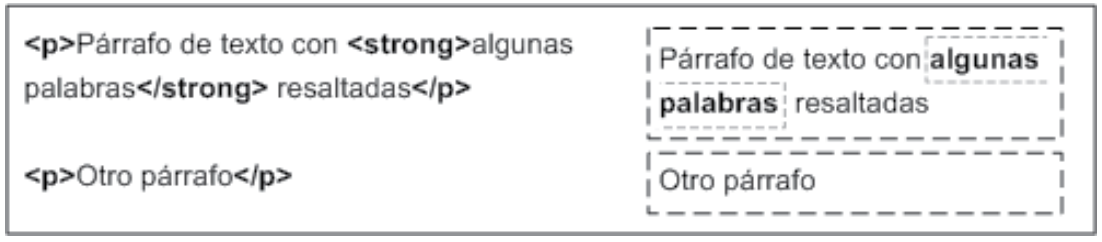
| La pseudo clase **:active** se usa para enlaces que están activos. El código de este ejemplo hace que el color de fondo para los enlaces activos sea amarillo:

```
a:active {background-color: #FFFF00;    }
```

| La pseudo clase **a:hover** se usa cuando el puntero del mouse pasa por encima de un

Resumiendo...las capas (div) no son más que unos recuadros que pueden ubicarse en cualquier lugar de la página, dentro de los cuales podemos insertar contenido HTML. La gran ventaja que tienen es que pueden ocultarse y solaparse entre sí, lo que nos brinda grandes potencialidades de diseño.





enlace. Esta pseudo-clase se puede usar para crear efectos interesantes: si queremos que nuestros enlaces sean de color naranja y estén en cursiva cuando el cursor pase sobre ellos, el código CSS que utilizaremos será el siguiente:

```
a:hover {color: orange;   font-style: italic;}

a:hover {letter-spacing: 10px;  font-weight:bold; color:red; }

a:hover {text-transform: uppercase;  font-weight:bold; color:blue; background-color:yellow; }
```

Cajas y Contenedores

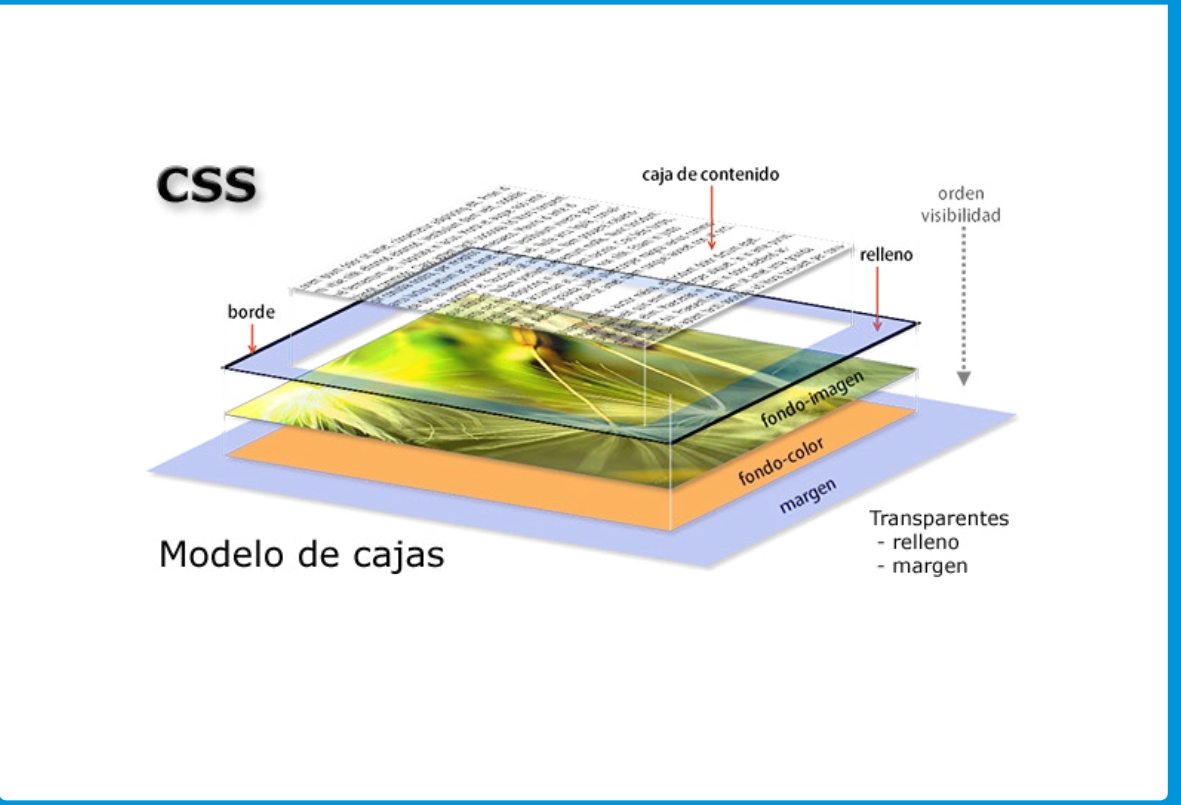
En realidad, todos los elementos de una web (párrafos, enlaces, imágenes, tablas, etc.) son cajas rectangulares. Los navegadores sitúan estas cajas de la forma que nosotros les hayamos indicado para maquetar la página(1) Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- Contenido (***content***): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.).
- Relleno (***padding***): espacio libre opcional existente entre el contenido y el borde.
- Borde (***border***): línea que encierra completamente el contenido y su relleno.
- Imagen de fondo (***background image***): imagen que se muestra por detrás del contenido y el espacio de relleno.
- Color de fondo (***background color***): color que se muestra por detrás del contenido y el espacio de relleno.

| Margen (**margin**): separación opcional existente entre la caja y el resto de cajas adyacentes.

También podes ver el modelo de cajas en la imagen 1

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).
Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.



Anchura y altura

La propiedad CSS que controla la anchura de la caja de los elementos se denomina **width.**

La propiedad **width** no admite valores negativos y los valores en **porcentaje** se calculan a partir de la anchura de su elemento padre. El valor **inherit** indica que la anchura del elemento se hereda de su elemento padre. El valor **auto**, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página (1).

La propiedad CSS que controla la altura de los elementos se denomina **height.**

Al igual que sucede con width, la propiedad height no admite valores negativos. Si se indica un **porcentaje**, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor **auto** a la altura.

El valor **inherit** indica que la altura del elemento se hereda de su elemento padre. El valor **auto**, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página (2).

Padding

Con padding establecemos la distancia de "relleno" entre el límite interior de la caja y el exterior (borde). Si queremos poner un padding de 20 píxeles para toda la caja, lo haríamos así:

```
padding : 20 px;
```

Podemos establecer un padding distinto para cada lado, usando los sufijos -top (superior), -bottom (inferior), left (izquierda) y right (derecha):

```
padding-top: 10px;
padding-bottom: 5px;
padding-left: 30px;
padding-right: 20px;
```

1

Valores de width	unidad de medida | porcentaje | auto | inherit

2

Valores de height	unidad de medida | porcentaje | auto | inherit

Podemos abreviar lo anterior en una sola línea, indicando primero el padding superior y luego siguiendo el orden de las agujas del reloj. Es decir, nos quedaría: arriba, derecha, abajo, izquierda. El ejemplo anterior se acortaría así:

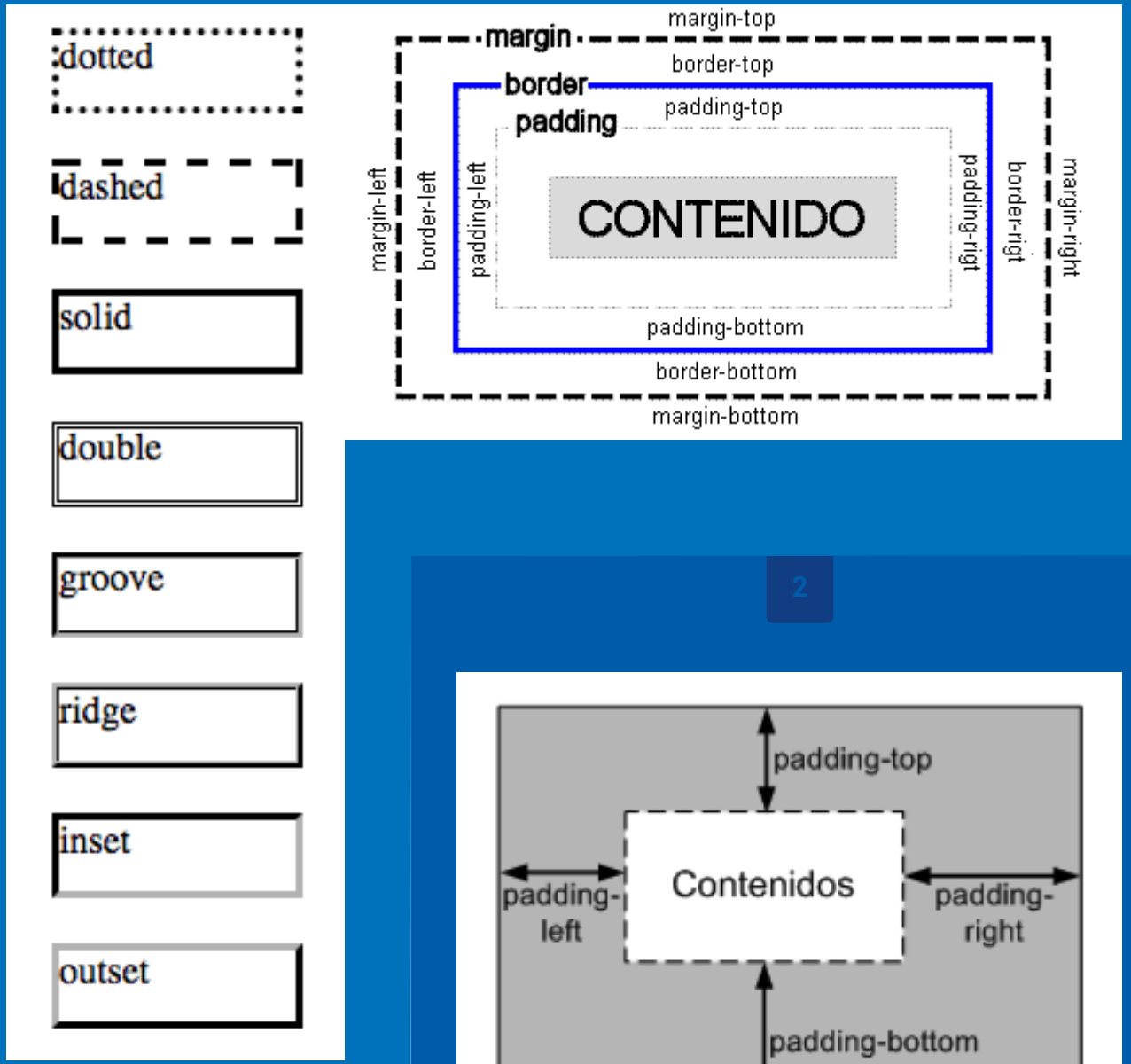
padding: 10px 20px 5px 30px;

Otro atajo útil es especificar sólo dos medidas: una corresponderían al padding superior e inferior, y la otra al lateral. Si queremos que los paddings superior e inferior sean de 10 píxeles, y los laterales (izquierdo y derecho) de 20 píxeles, escribimos:

padding: 10px 20px;

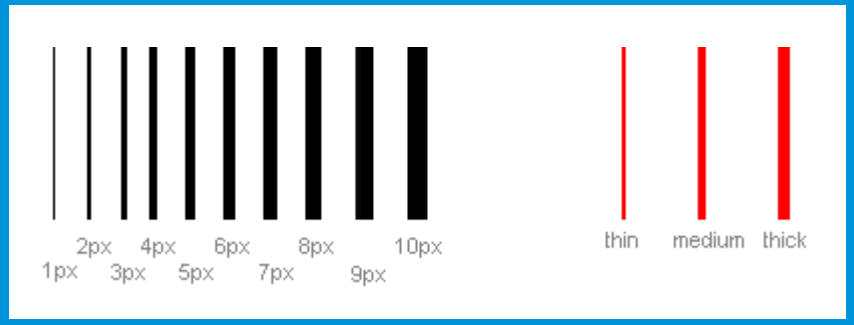
Podés aclarar lo que venimos trabajando observando las imágenes 1 y 2.

1



2

1



2

Valores	color | transparent | inherit

Bordes

Si queremos que nuestra caja tenga **bordes**, lo conseguimos con la propiedad **border**. Esta tiene la siguiente sintaxis:

border: width | style | color

Como habrás supuesto, **width** especifica el grosor del borde. Normalmente es una medida en píxeles, pero también podemos utilizar las palabras thin (fino), medium (normal) y thick (grueso) **(1)**.

En cuanto a **style**, es el tipo de borde. Hay bastantes, pero los más comunes son: solid (línea continua), dashed (línea discontinua), dotted (línea de puntos) y double (línea continua doble). Por último, **color** indica el color del borde.

Podemos escoger un tipo de borde diferente para cada lado con los sufijos -top, -bottom, -left y -right.

border-bottom: 1px dotted #f00;

Para eliminar el borde, simplemente ponemos que tiene de grosor 0 píxeles o que el estilo del borde es none. Esto es muy importante con las imágenes, ya que si tenemos una imagen enlazando a algo, los navegadores la ponen con un reborde muy feo. Así que esto se ha convertido ya en un fijo de las hojas de estilos.

```
img { border: none; }

div {
  border-top-width: 10px;
  border-right-width: 1em;
  border-bottom-width: thick;
  border-left-width: thin;}
```

Color

Los valores del color de los bordes se pueden observar en el cuadro **(2).**

```
div {  border-top-color: #CC0000;  border-right-color: blue;
border-bottom-color: #00FF00;  border-left-color: #CCC;}
```

1

Valores	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit

2

Valores	(none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset) {1, 4} | inherit

3

Valores	(unidad de medida_borde || color_borde || estilo_borde) | inherit

Estilo

Por último, CSS permite establecer el estilo de cada uno de los bordes mediante los valores que se expresan en el cuadro (1). El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. Como el valor por defecto de esta propiedad es **none**, los elementos no muestran ningún borde visible a menos que se establezca explícitamente un estilo de borde.

```
div {
  border-top-style: dashed;
  border-right-style: double;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

Para establecer de forma simultánea los estilos de todos los bordes de una caja, es necesario utilizar la propiedad **"shorthand"** llamada **border-style** (2).

Propiedades shorthand

Como sucede con los márgenes y los rellenos, CSS define una serie de propiedades de tipo **shorthand** que permiten establecer todos los atributos de los bordes de forma simultánea. CSS incluye una propiedad **shorthand** para cada uno de los cuatro bordes y una propiedad **shorthand** global (3).

El significado de cada uno de los valores especiales es el siguiente:

| <**medida_borde**>: thin, medium, thick.
| <**color_borde**>: Color o transparent
| <**estilo_borde**>: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset.

Las propiedades **shorthand** permiten establecer alguno o todos los atributos de cada borde. El siguiente ejemplo establece el color y el tipo del borde inferior, pero no su anchura:

```
h1 {  border-bottom: solid red;}
```

En el ejemplo, la anchura del borde será la correspondiente al valor por defecto (**medium**). Este otro ejemplo muestra la forma habitual utilizada para establecer el estilo de cada borde:

```
div {  border-top: 1px solid #369;
 border-bottom: 3px double #369;}
```

Por último, CSS define una propiedad de tipo **shorthand** global para establecer el valor de todos los atributos de todos los bordes de forma directa. Las siguientes reglas CSS son equivalentes:

```
div {
  border-top: 1px solid red;
  border-right: 1px solid red;
  border-bottom: 1px solid red;
  border-left: 1px solid red;
}
div { border: 1px solid red; }
```

Como el valor por defecto de la propiedad **border-style** es none, si una propiedad **shorthand** no establece explícitamente el estilo de un borde, el elemento no muestra ese borde:

```
/* Sólo se establece el color, por lo que el
estilo es  "none" y el borde no se muestra */

div { border: red; }

/* Se establece el grosor y el color del borde,
pero no  su estilo, por lo que es "none" y el
borde no se muestra */

div { border-bottom: 5px blue; }
```

Cuando los cuatro bordes no son idénticos pero sí muy parecidos, se puede utilizar la propiedad **border** para establecer de forma directa los atributos comunes de todos los bordes y posteriormente especificar para cada uno de los cuatro, sus propiedades particulares:

```
h1 { border: solid #000;  border-top-width:
6px; border-left-width: 8px;}
```

Márgenes

Los **márgenes** se controlan con la propiedad **margin**, y es la distancia entre el borde de la caja y los elementos que la rodean.
En cuanto a la forma de usarla, es igual que con la propiedad padding, así que la forma de escribir y los atajos es exactamente la misma. Por ejemplo, si queremos márgenes superior e inferior de 20 píxeles, y laterales de 5 píxeles:

```
margin: 20px 5px;
```

La propiedad margin se usa de la misma forma que la propiedad padding, por eso, la forma de escribirla y los atajos ¡son exactamente iguales!

Para centrar un elemento de bloque, podemos hacer uso de auto:

```
margin: 0px auto;ç
```

Podemos observar esto en las imágenes 1 y 2

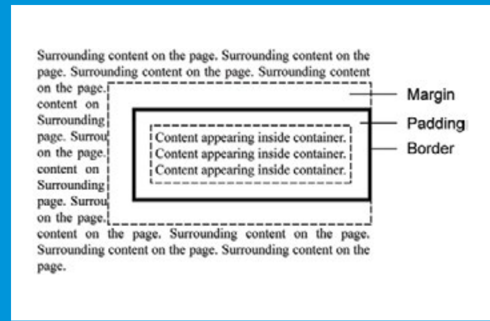
Márgenes negativos

A todas las propiedades de márgenes se les pueden asignar valores negativos. En este caso, un margen adyacente puede ser "cancelado" a cualquier nivel. Si se aplica un valor lo suficientemente negativo a un elemento lo suficientemente grande, el elemento adyacente afectado puede acabar quedando por debajo del otro.

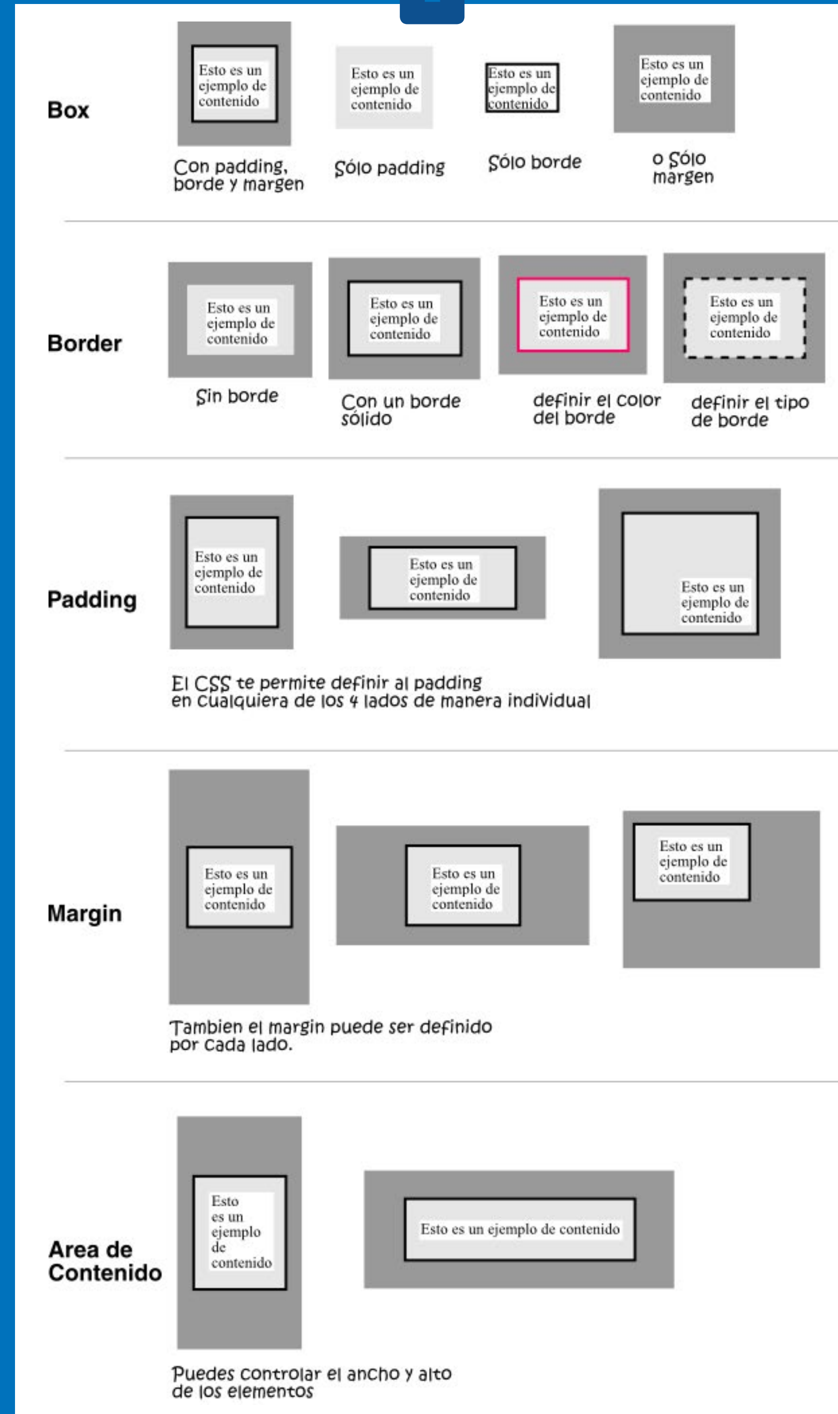
```
body {background-color:White; font-family:Geneva;}
#header { background-color:yellow; }
h1 { color:red; font-size:2em; }
#content {margin-top:-3em;}
```

Esto provoca el efecto visual de cambiar el elemento de manera que se superponga (3).

1

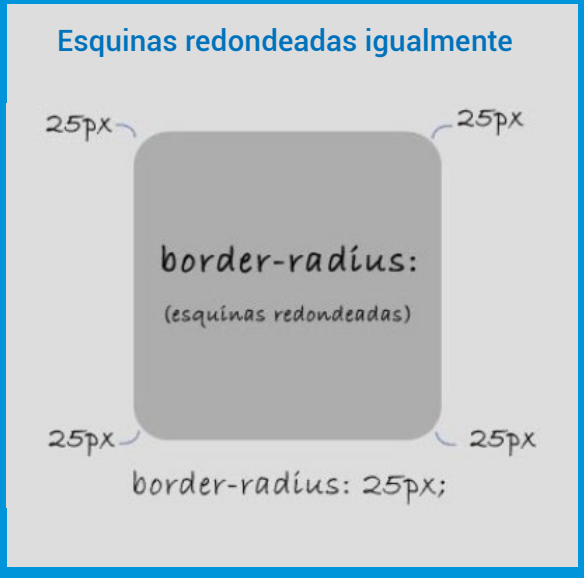


2

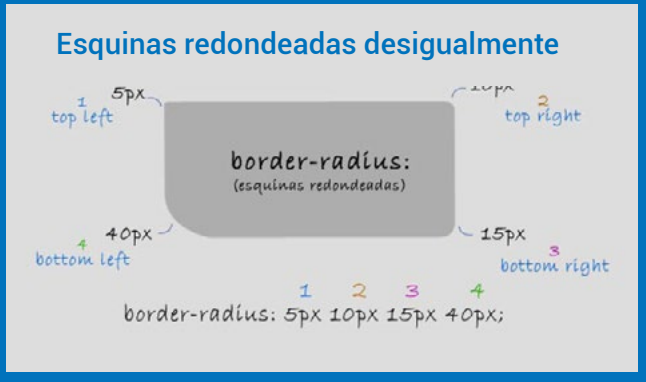


3

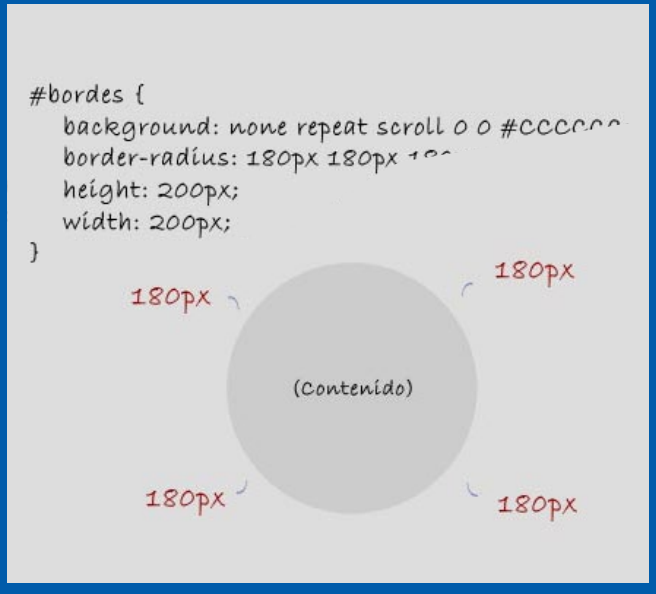
1



2



3



Bordes redondeados

La propiedad **border-radius** nos permite crear esquinas redondeadas sin la necesidad de imágenes o marcas adicionales como lo hacíamos anteriormente (¿te acordás que teníamos que cortar los bordes redondeados en photoshop y crear una cantidad de div para dar un aspecto agradable a un bloque o div?). Para añadir esquinas redondeadas a nuestra caja, simplemente se añade:

```
#ejemplo1{
    border-radius: 25px;
}

#ejemplo2{
    border-top-left-radius: 25px;
    border-top-right-radius: 25px;
    border-bottom-right-radius:25px;
    border-bottom-left-radius: 25px;
}
```

En el código **#ejemplo1** lo que especificamos fue que al elemento que se le aplique este estilo, va a tener un radio de 25 px en las esquinas. También lo podríamos haber escrito de la forma en que se encuentra **#ejemplo2** pero es mejor simplificarlo como el primero **(1)**

También podemos crear deformaciones de este estilo, podemos jugar con los ángulos para crear objetos que antes eran imposibles con solo css, como te muestro en la imagen **2** o en la **3.**

Posicionamiento

Los navegadores crean y posicionan de forma automática todas las cajas o divs que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestran. Podemos encontrar las siguientes posiciones o situaciones:

| **Posicionamiento normal o estático:** se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.

| **Posicionamiento relativo:** variante del posicionamiento normal que consiste en ubicar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.

| **Posicionamiento absoluto:** la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.

| **Posicionamiento fijo:** variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.

| **Posicionamiento flotante:** se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

El posicionamiento de una caja se establece mediante la propiedad **position** (1).

El significado de cada uno de los posibles valores de la propiedad position es el siguiente:

| **static:** corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades **top**, **right, bottom** y **left**.

| **relative:** corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades **top, right, bottom** y **left**.

1

Valores	static | relative | absolute | fixed | inherit

Si bien los navegadores crean y posicionan de forma automática todas las cajas o divs que forman cada una de las páginas HTML, CSS nos permite a los diseñadores modificar la posición en la que éstas se muestran.

| **absolute:** corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.

| **fixed:** corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad **position** no permite controlar el posicionamiento flotante, que se establece con otra propiedad llamada **float** y que se explica más adelante. Además, la propiedad **position** sólo indica cómo se posiciona una caja, pero no la desplaza.

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas **top, right, bottom** y **left** para controlar el desplazamiento de las cajas posicionadas **(1).**

En el caso del posicionamiento relativo, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el posicionamiento es absoluto, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.
En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades **right** y **left**) o altura (propiedades **top** y **bottom**) del elemento.

Posicionamiento normal o estático

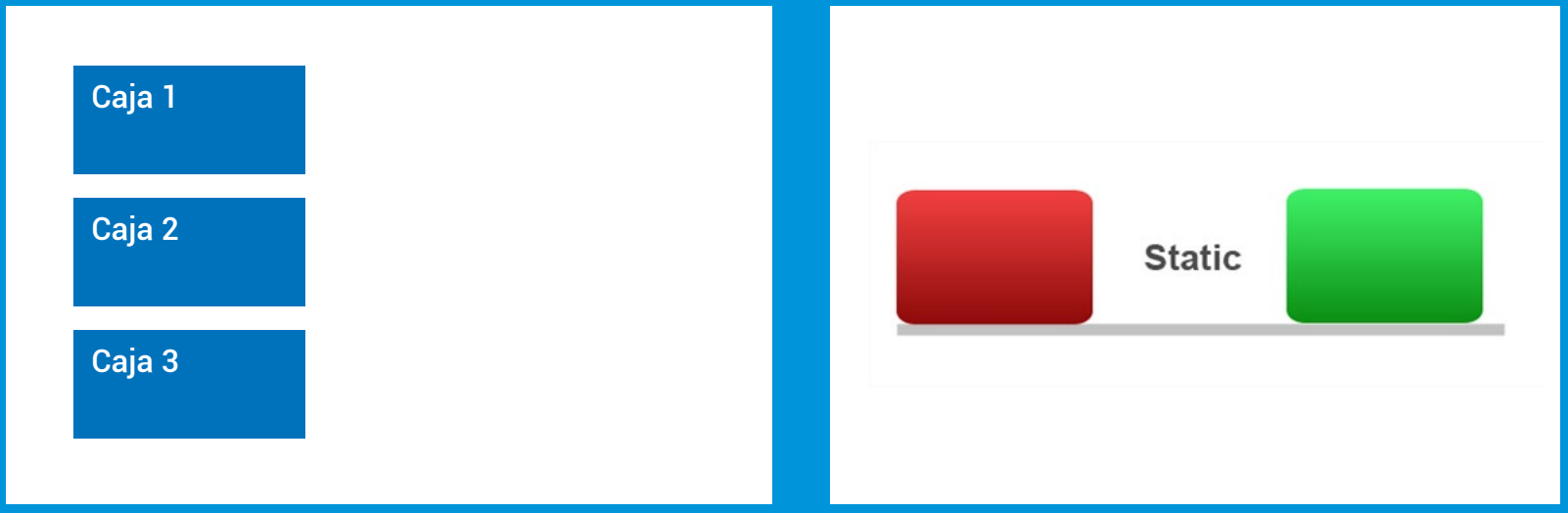
El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades **width** y **height** y su contenido.

Los elementos de bloque forman lo que CSS denomina **"contextos de formato de bloque"**. En este tipo de contextos, las cajas

1

Valores	**unidad de medida | porcentaje | auto | inherit**

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas, mientras que el posicionamiento relativo desplaza una caja respecto de su posición original establecida mendiante el posicionamiento normal.



se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.

Siempre que creen una caja nueva, les aparecerá en la esquina arriba-izquierda de su contenedor. De esta forma, las cajas (divs, párrafos, etc) se ubicarán uno debajo del otro, tal como lo muestra la imagen (1)

Si un elemento se encuentra dentro de otro, el elemento padre se llama "elemento contenedor" y determina tanto la posición como el tamaño de todas sus cajas interiores.

Posicionamiento relativo

El posicionamiento relativo desplaza una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades **top, right, bottom** y **left**.

El valor de la propiedad **top** se interpreta como el desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original. De la misma forma, el valor de las propiedades **left, right** y **bottom** indica respectivamente el desplazamiento entre el borde izquierdo/derecho/inferior de la caja en su posición final y el borde izquierdo/derecho/inferior de la caja original.

Por tanto, la propiedad **top** se emplea para mover las cajas de forma descendente, la propiedad **bottom** mueve las cajas de forma ascendente, la propiedad **left** se utiliza para desplazar las cajas hacia la derecha y la propiedad **right** mueve las cajas hacia la izquierda. Este comportamiento parece poco intuitivo y es causa de errores cuando se empiezan a diseñar páginas con CSS. Si se utilizan valores negativos en las propiedades **top, right, bottom** y **left,** su efecto es

1

2

justamente el inverso. Observá la imagen 1 para comprender esto mejor.

Ahora se le aplicará un posicionamiento relativo para desplazar la primera imagen hacia abajo por ejemplo:

```
img.desplazada {
  position: relative;
  top: 8em;
}
```

```
<img class="desplazada" src="imagenes/
imagen.png" alt="Imagen genérica" />
<img src="imagenes/imagen.png"
alt="Imagen genérica" />
<img src="imagenes/imagen.png"
alt="Imagen genérica" />
```

Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento. La nueva posición de la caja se indica mediante las propiedades **top, right, bottom** y **left**. Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas, como se muestra en la figura 2.

Este atributo es como **relative** pero mucho más poderoso, nos permite hacer romper el flujo total del elemento, no sólo de la línea donde esté, sino en todo el documento o sitio. Una vez fijado **absolute**, podemos definir **top, bottom, left, right, width, height;** a diferencia de **relative** que empieza a contar a partir de la línea donde esté, **absolute** empieza desde el inicio del sitio web, en la esquina superior izquierda. Este atributo es el más usado en la mayoría de aplicaciones y sitios web.

```
<div style="position: absolute; width:
300px; height: 140px; top: 100px; left: 30px;
background-color: #ff8800; color: #fff;
padding: 15px;z-index: 2;"> Esta capa tiene
posicionamiento absoluto. <br> <br>
Me permite especificar top y left para
colocarla con respecto a la esquina superior
izquierda.
</div>
```

El posicionamiento absoluto se emplea para establecer de forma exacta la posición en la que se muestra la caja de un elemento. Si el posicionamiento es aboluto y fijo, estamos frente a un caso particular que se diferencia del anterior en el comportamiento de las cajas posicionadas. La principal característica de una caja posicionada de forma fija es que su ubicación es inamovible dentro de la ventana del navegador.

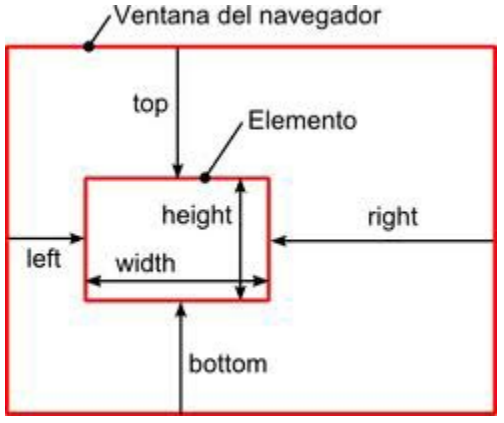
Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades **top, right, bottom** y **left**. A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de la caja posicionada.

Si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar este último. Para ello, sólo es necesario añadir la propiedad **position: relative**, por lo que no es obligatorio desplazar el elemento contenedor respecto de su posición original. Como ejemplo de posicionamiento absoluto, vamos a colocar 4 cajas en cada esquina del documento:

```
#box1 {
    position:absolute;
    top: 50px;
    left: 50px;
    }
#box2 {
    position:absolute;
    top: 50px;
    right: 50px;
    }
#box3 {
    position:absolute;
    bottom: 50px;
    right: 50px;
    }
    #box4 {
    position:absolute;
    bottom: 50px;
    left: 50px;
    }
```

Posicionamiento absoluto fijo

Es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas. Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto
La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que



Valores	left | right | none | inherit

las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador (1)

```
<!DOCTYPE html>
 <head>
  <meta content="text/html;
charset=ISO-8859-1"
     http-equiv="content-type" />
   <title>Posicionamiento fijo</title>
   <style type="text/css">
    div#fijo_izquierda {
     position: fixed;
     left: 10%;
     top: 50px;
     width: 100px;
```

Posicionamiento flotante

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una ***caja flotante***, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.
Cuando se posiciona una caja de forma flotante ésta deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por aquella. La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina ***float*** (2).

Si se indica un valor ***left***, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y ***fluyen*** alrededor de la caja flotante.
El valor ***right*** tiene un funcionamiento idéntico,

salvo que en este caso, la caja se desplaza hacia la derecha. El valor ***none*** permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original **(1)**.

Por ejemplo:

```
img {  float: left;}
```

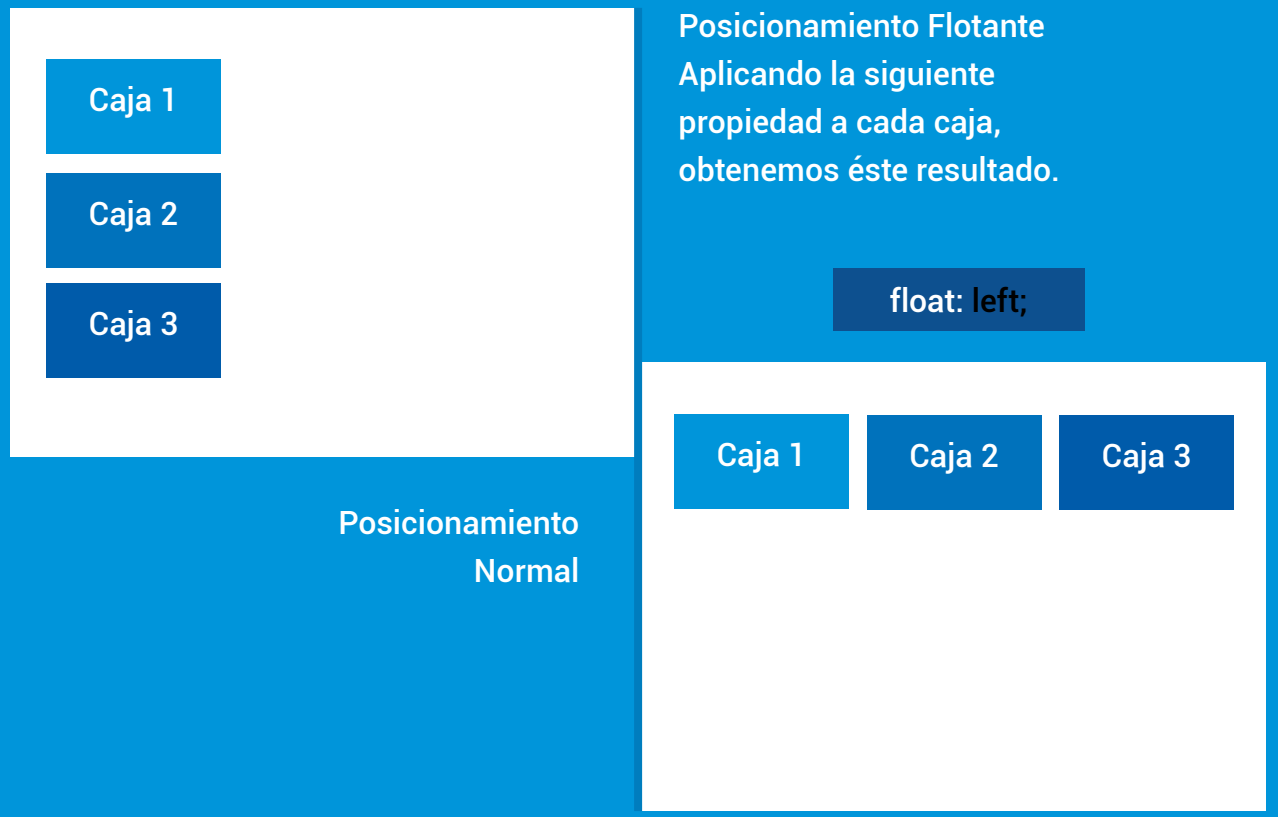
Uno de los principales motivos para la creación del posicionamiento **float** fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

La propiedad **clear** permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante.

```
<p style="clear: left;">...</p>
```

Su sintaxis es la que podemos observar en el cuadro **2.**

1



2

Valores	none | left | right | both | inherit

La propiedad **clear** indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor **left**, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo. El valor **both** despeja los lados izquierdos y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha. Por ejemplo:

```
#paginacion {  border: 1px solid #CCC;
  background-color: #E0E0E0;  padding:
.5em; }
 .derecha   { float: right; }
.izquierda { float: left;  }

<div id="paginacion">
  <span class="izquierda">&laquo; Anterior</
span>
  <span class="derecha">Siguiente &raquo;</
span>
</div>
```

Visualización

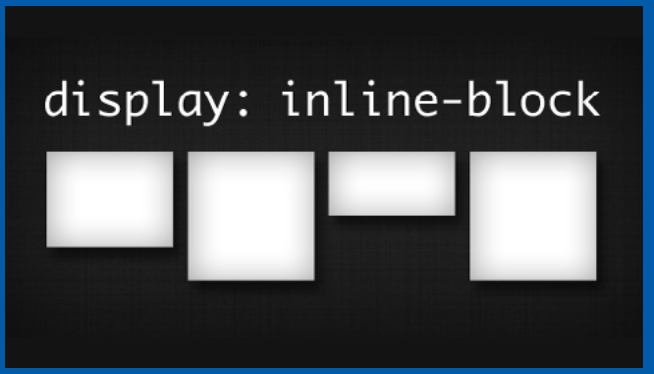
Las propiedades display y visibility controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página.

La diferencia entre ambas propiedades es que, mientras display oculta por completo un elemento haciendo que los demás elementos ocupen su lugar, la propiedad visibility hace que los otros elementos que componen la página respeten la posición de éste y reserven su espacio.

La propiedad “display”

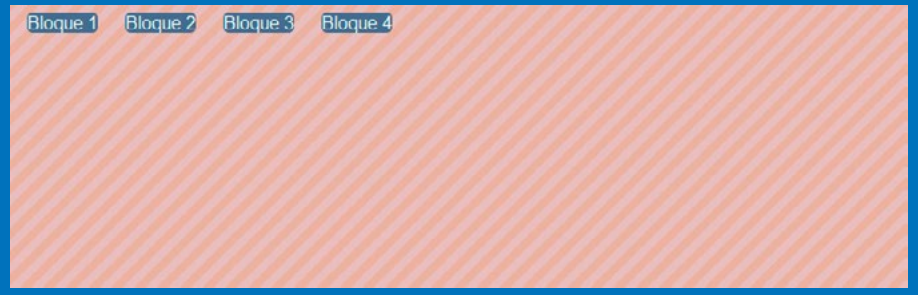
En realidad, la propiedad display modifica la forma en la que se visualiza un elemento. Los valores más utilizados son inline, block y none.

- **block:** Muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate.
- **inline:** Visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.
- **none:** Oculta un elemento y hace que desaparezca de la página **(1).**



La diferencia entre Display y Visibility radica en que la primera propiedad oculta por completo un elemento haciendo que los demás ocupen su lugar, mientras la segunda hace que los otros elementos respeten la posición y reserven el espacio del elemento oculto.

1



2

Código	Resultado
<head> < style type="text/css"> h3.se_ve{visibility:visible} h3.no_se_ve{visibility:hidden} </style> < /head> < body> <h3 class="se_ve">Este texto es visible.</h3> <h3 class="no_se_ve">Este texto es invisible.</h3>	Este texto es visible.


```
.en-linea{display: inline;}
.bloque{display: block;}

<div>DIV normal</div>
<div class="en-linea">DIV con display:inline</div>

<a href="#">Enlace normal</a>
<a href="#" class="bloque">Enlace con display:block</a>
```

| **La propiedad display:** inline se puede utilizar en las listas (<ul>, <ol>) que se quieren mostrar horizontalmente **(1)**
| **display:** block se emplea frecuentemente para los enlaces que forman el menú de navegación.

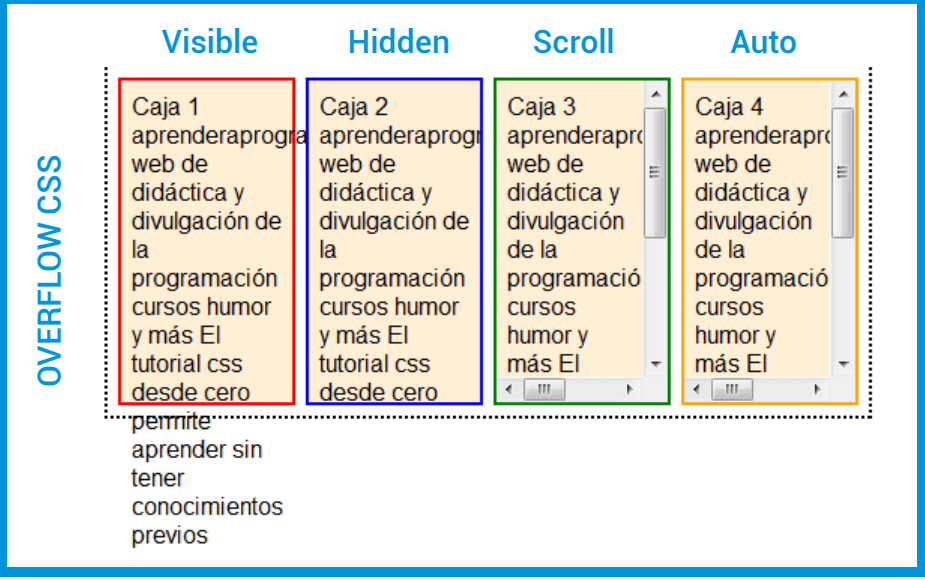
```
div { play:inline;}
```

La propiedad "visibility"

Visibility es una propiedad mucho más sencilla que display, pues sus posibilidades son mucho más limitadas y únicamente permite hacer visibles o invisibles los elementos de una página. Los valores posibles para visibility son:

| **visibility: normal:** Es el valor por defecto.
| **visibility hidden:** convierte una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío

| **visibility: collapse:** sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad **display**, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor **collapse** sobre cualquier otro tipo de elemento, su efecto es idéntico al valor **hidden**.



¡Tomá nota! La propiedad Overflow sirve para controlar la forma en la que se visualizan los contenidos que sobresalen de un elemento.

La propiedad overflow (contenido sobrante)

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda. La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad **width** y/o **height**.

CSS define la propiedad **overflow** para controlar la forma en la que se visualizan los contenidos que sobresalen (sobrantes) de un elemento. Los posibles valores para la propiedad overflow son los siguientes:

- **overflow: visible:** Es el valor por defecto.
- **overflow: hidden:** el contenido sobrante se oculta y únicamente se visualizará la parte del contenido que cabe dentro de la zona reservada para el elemento.
- **overflow: scroll:** solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero **se muestran barras de scroll que permiten visualizar el resto del contenido.**

- **overflow: auto: el comportamiento depende del navegador**, aunque normalmente es el mismo que la propiedad scroll. Obviamente, es un comportamiento **totalmente desaconsejado.**

Observá la imagen **1** para que te sea más claro comprender estas propiedades.

Por ejemplo:

```
<div style="overflow: auto; width: 300px;
height: 100px; background-color:#ededed;
border: 1px solid #990000;">
CONTENIDO....
< /div>
```

Superposición de cajas

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas, determinando el orden de superposición de éstas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas. Es útil cuando se producen solapamientos. Utilizando la propiedad z-index es posible crear páginas complejas con varios niveles o capas.

Profundidad

El valor más común de la propiedad **z-index** es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo. Por lo tanto, cuanto más alto sea el valor, más "cerca" del usuario se mostrará la capa (una caja con valor z-index:10 se mostrará por encima que otra con valor z-index:9).
La propiedad z-index sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad position.

Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (position: relative).

A continuación se aplican las reglas necesarias para posicionar las cajas de tal forma que la caja 2 quede por encima de caja 1 pero por debajo de caja 3:

```
<! Doctype html>
<head>
<style type="text/css">

#caja1,#caja2,#caja3 {  background-color:
#f99;  font-size:1.3em;
  margin-bottom:10px;  width:200px;
height:200px; }
```

CSS permite controlar la posición tridimensional de las cajas, determinando de ese modo el orden de superposición de éstas. Con ésta propiedad es posible indicar las cajas que se muestran delante o detrás de otras cajas.

Clip

Usamos clip cuando queremos recortar una imagen a una medida determinada. Los valores de **rect** aparecen en este orden: **superior, derecha, inferior, izquierda**.

Background y sus propiedades

Css3 nos trae nuevas formas de controlar las imágenes de fondo de cajas, div o de sitios web completos, se trata de **multiple background images**. Esto nos va a permitir colocar varias imágenes de fondo, **background-size** que nos va a permitir personalizar por medio de css el tamaño de la imagen y adaptarla a su contenedor y **background-origin** que nos va a permitir especificar el posicionamiento de una imagen de acuerdo a su área, solo por dar una ejemplo. A continuación veremos cómo se trabaja con ellos.

Código

```
<head>
< style type="text/css">
img.recortada {
position:absolute;
clip: rect(0px 50px 100px 0px);
}
</style>
< /head>

< body>
<img class="recortada" src="/foto3.
jpg" style="border:0; width: 100px;
height: 150px">
<img src="/foto3.jpg" style="border:0;
width: 100px; height: 150px">
</body>
```

Resultado

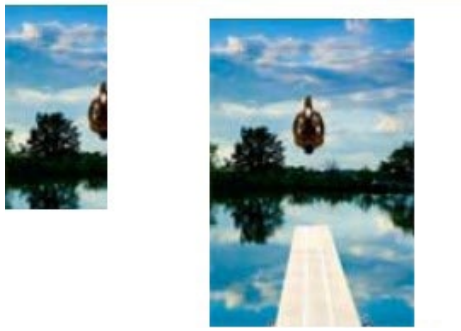


Imagen completa

En el ejemplo, el tamaño real de la imagen es de 100x150 píxeles y con la propiedad clip la cortamos a la medida deseada.

Múltiples imágenes de fondo

La sintaxis para esta operación es la siguiente:

```
background-image:   url(sol.jpg), url(vaca.
png), url(paisaje.jpg);
```

Podemos crear composiciones formadas por varias imágenes y ubicarlas en diferentes posiciones para lograr lo deseado. Por ejemplo, vamos a ver un paisaje compuesto por 3 imágenes diferentes pero que vamos a integrar en un solo div utilizando múltiples imágenes de fondos.

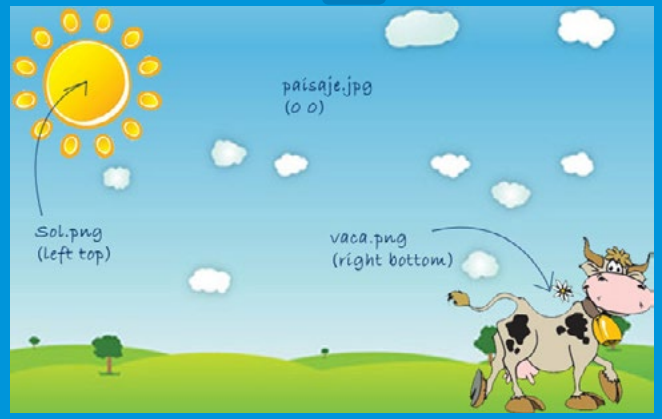
```
<html>
 <head>
   <style text="text/css">

      div{
        height:400px;     width: 635px;
        background-image: url(sol.png),
url(vaca.png), url(paisaje.jpg);
        background-repeat: no-repeat;
        background-position: left top, right
bottom, 0 0;
      }
   </style>
 </head>
 <body>
   <div></div>
 </body>
</html>
```

Lo que dará como resultado lo que muestra la imagen 1

Las posiciones que asignamos a cada imagen dependerán del orden en que tengamos las imágenes, desde la primera hasta la última. Podemos abreviar este código de css para aplicar múltiples fondos de imágenes de la siguiente forma.

```
div{
   background:
     url(sol.png) left top no-repeat,
     url(vaca.png) right bottom no-repeat,
     url(paisaje.jpg) 0 0 no-repeat;
   }
```

1



En resumen, Multiple Background Images nos va a permitir colocar varias imágenes de fondo, Background-size nos va a permitir personalizar el tamaño de la imagen y adaptarla a su contenedor y Background-origin nos permitirá especificar el posicionamiento de una imagen de acuerdo a su área,

Valor	Descripción
background-position: 2cm 2cm	La imagen se posiciona a 2 cm del margen izquierdo y a 2 cm del margen superior de la página
background-position: 50% 25%	La imagen se posiciona en el centro de la página y un 25 % del margen superior de la misma
background-position: top right	La imagen se posiciona en la esquina superior derecha de la página

Ubicación de la imagen de fondo [background-position]

Por defecto, una imagen de fondo se posiciona en la esquina superior izquierda de la pantalla. La propiedad **background-position** te permitirá cambiar este valor por defecto y posicionar la imagen de fondo en cualquier lugar de la pantalla que quieras. Hay muchas formas diferentes de establecer los valores de la propiedad **background-position.** Sin embargo, todas ellas se formatean como un conjunto de coordenadas. Por ejemplo, el valor '100px 200px' posiciona la imagen de fondo a 100 píxeles del margen izquierdo y a 200 píxeles del margen superior de la ventana del navegador.

Las coordenadas se pueden indicar como porcentajes del ancho de la pantalla, como unidades fijas (píxeles, centímetros, etc.) o pueden usar las palabras "top" (superior), "bottom" (inferior), "center" (centro), "left" (izquierda) y "right" (derecha) **(1)**.

El ejemplo de código siguiente posiciona la imagen de fondo en la esquina inferior derecha.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.
gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-position: right bottom;
        }

        h1 {color: #990000;
        background-color: #FC9804;}
```

Propiedades [background-size]

Esta propiedad nos permite escalar una imagen dispuesta en el fondo. La sintaxis es:

Elemento { background-size: ancho alto; }

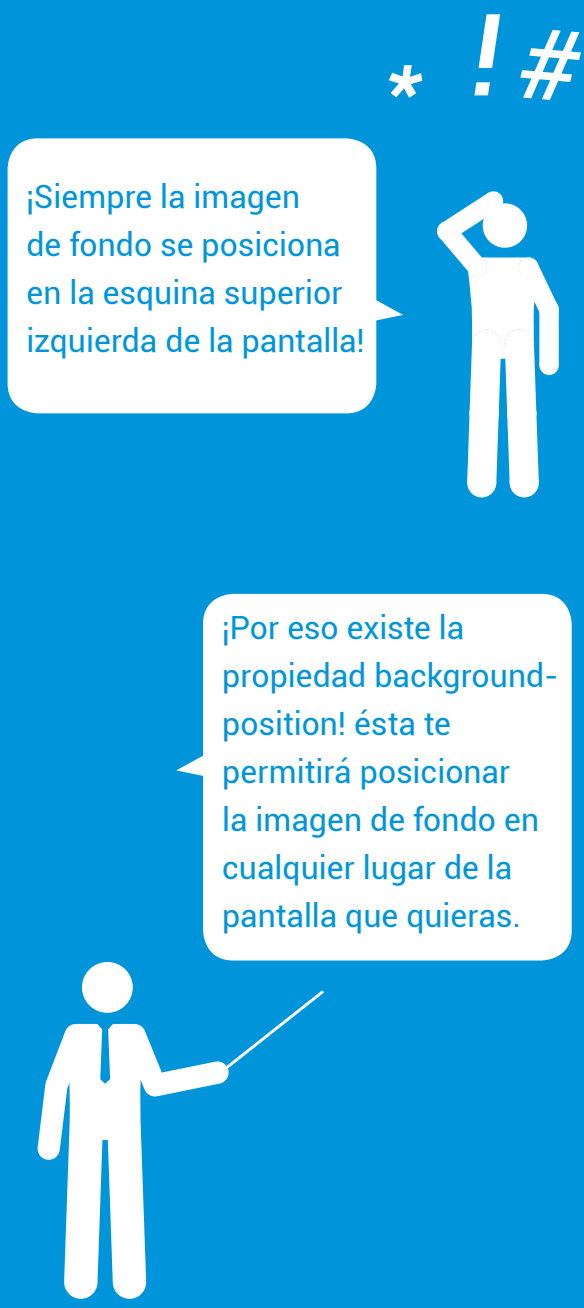
Estas longitudes pueden ser especificadas en píxeles, porcentajes, etc. Por ejemplo si queremos mostrar tres imágenes dentro de un div con distintos tamaño:

```
<!DOCTYPE  html>
<html>
<head>
<title>Prueba</title>

<style type="text/css">

#recuadro1{
  background-image:    url("logo1.png"),
                       url("logo1.png"),
                       url("foto1.jpg");

  background-position: 0% 0%,
                       50% 50%;

  background-size:     30px 30px,
                       250px 250px,
                       700px 450px;
```

Si queremos que una imagen tome el tamaño por defecto que tiene la imagen sin re-escalar podemos utilizar la palabra clave "auto". Por ejemplo, para mostrar la primera imagen con el tamaño original podemos escribir:

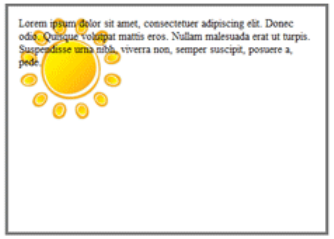
```
#recuadro1{
  background-image:    url("logo1.png"),
                       url("logo1.png"),
                       url("foto1.jpg");

  background-position: 0% 0%,
                       50% 50%;

  background-size:     30px 30px,
                       250px 250px,
                       auto auto;

  background-repeat: no-repeat;
  width:700px;
  height:450px;
}
```

Si utilizamos como unidad porcentajes, la dimensión se basa en el elemento contenedor. Así un ancho y un alto de 50%, por ejemplo, hará que el fondo de la imagen llene el recipiente 1/4

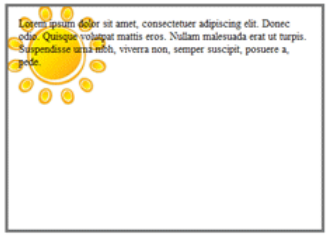


Background - size: contain;

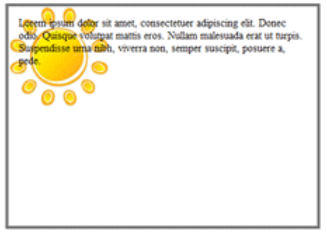
Background - size: cover;



Background - origin: content-box (situamos a la imagen deacuerdo al contenido de la caja)



Background - origin: border-box (situamos a la imagen deacuerdo al borde de la caja)



Background - origin: padding-box (situamos a la imagen deacuerdo al padding de la caja)


```
#recuadro1{
  background-image:   url("logo1.png"),
                      url("logo1.png"),
                      url("foto1.jpg");

  background-position: 0% 0%,
                      50% 50%;

  background-size:    50% 50%,
                      250px 250px,
                      auto auto;

  background-repeat: no-repeat;
  width:700px;
  height:450px;
}
```

Al tamaño también lo podemos definir por medio de porcentajes (como se indicó antes) o bien mediante **cover** y **contain** (1).

| **Cover:** Escala la imagen para el tamaño más pequeño de tal manera que su anchura y su altura puede caber dentro del área de contenido.

| **Contain:** Escala la imagen al tamaño más grande de tal manera que su anchura y su altura puede caber dentro del área de contenido

```
div{   height:400px;   width: 635px;
  background:url(paisaje.jpg) 0 0 no-repeat;
  background-size: 50% 400px;
  border: 1px solid #777;}
```

Esta propiedad nos permite posicionar el **background** o imagen de fondo a relación del contenido o área de la caja contenedora, podemos hacer que sea relativa al **padding** de la caja por medio de **padding-box**, también por medio del borde de la caja por medio de **border-box** y también la podemos ubicar relativamente al contenido de la caja por medio de **content-box**. Veamos esto en el código siguiente y en la imagen **2**

```
div{   height:300px;   width: 435px;
  background:url(sol.png) 0 0 no-repeat;
  background-origin: content-box;   border:
5px solid #777;
  padding: 15px;}
```

1

Valor	Descripción
Background-repeat: repeat-x	La imagen se repite en el eje horizontal
background-repeat: repeat-y	La imagen se repite en el eje vertical
background-repeat: repeat	La imagen se repite en el eje horizontal y vertical
background-repeat: no-repeat	La imagen no se repite

2

Valor	Descripción
Background-attachment: scroll	La imagen se desplaza con la página - no está fija
Background-attachment: fixed	La imagen está fija

Repetir la imagen de fondo [background-repeat]

La tabla **(1)** resume los cuatro valores diferentes para la propiedad **background-repeat.**

Por ejemplo, para evitar que se repita una imagen de fondo, el código que tendríamos que usar sería el siguiente:

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.
gif");
background-repeat: no-repeat;
        }
h1 {color: #990000;  background-color:
#FC9804;}
```

Fijar la imagen de fondo [background-attachment]

La propiedad **background-attachment** especifica si una imagen está fija o se desplaza con el elemento contenedor. Una imagen de fondo fija no se moverá con el texto cuando el lector se desplace por la página, mientras que una imagen de fondo no fija se desplazará con el texto de la página web **(2)**.

Por ejemplo, el siguiente código fijará la imagen de fondo.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.
gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
        }
h1 {color: #990000; background-color:
#FC9804;}
```

Combinación de propiedades [background]

Con la propiedad background se pueden comprimir varias propiedades, y así escribir una hoja de estilo de forma más abreviada, lo que facilitará su lectura. Por ejemplo, observa estas cinco líneas de código:

```
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Usando **background** se puede lograr el mismo resultado con una única línea de código:

```
background: #FFCC66 url("butterfly.gif") no-repeat fixed right bottom;
```

El orden en que deben aparecer las propiedades individuales es el siguiente:

[background-color] | [background-image] | [background-repeat]| [background-attachment] | [background-position]

Si se omite alguna propiedad, de forma automática ésta se establecerá con su valor por defecto. Por ejemplo, si se omiten las propiedades **background-attachment** y **background-position** del ejemplo anterior, el código quedará de la siguiente manera:

```
background: #FFCC66 url("butterfly.gif") no-repeat;
```

```
/* Color e imagen de fondo de la página mediante una propiedad shorthand (o atajo) */
body { background: #222d2d url(./graphics/colores.gif) repeat-x 0 0; }
```

```
/* La propiedad shorthand anterior es equivalente a las siguientes propiedades*/
body {
  background-color: #222d2d;
  background-image: url("./graphics/colores.gif");
  background-repeat: repeat-x;
  background-position: 0 0;
}
```

Recordá que con la propiedad background se pueden comprimir varias propiedades, y así escribir una hoja de estilo de forma más abreviada ¡Esto volverá más fácil su lectura!

Alturas/anchuras máximas y mínimas

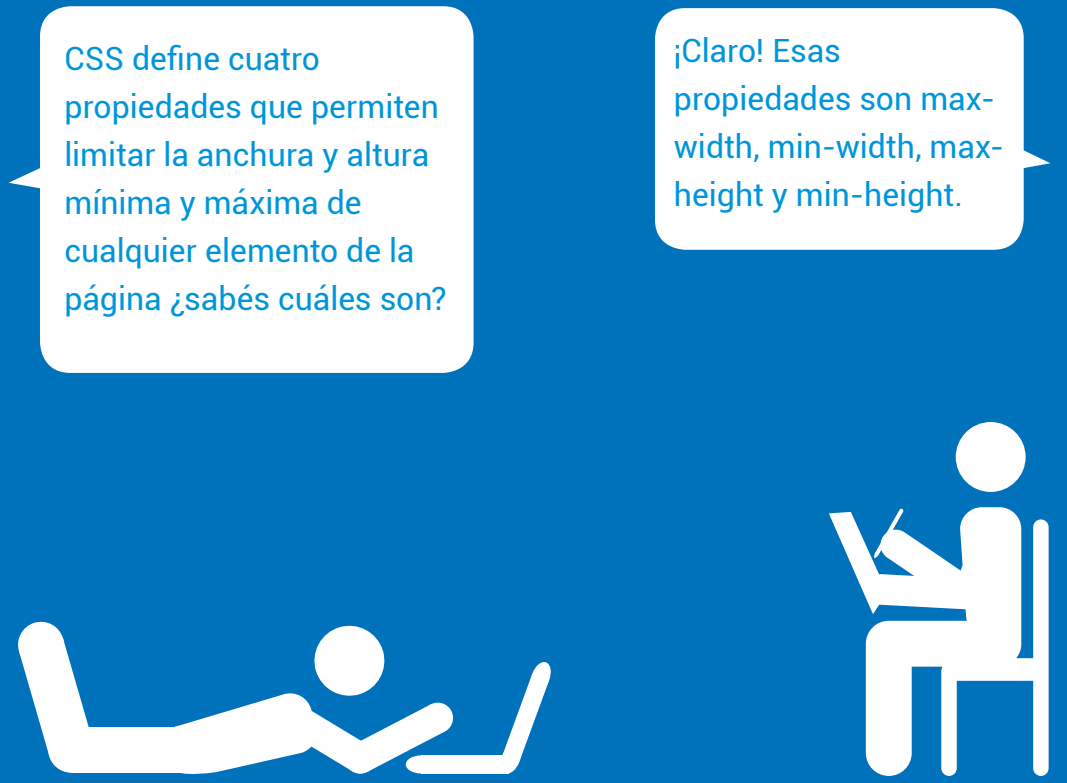
CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son max-width, min-width, max-height y min-height **(1)**.

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

#contenedor { min-width: 500px; max-width: 900px;}

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son max-width y min-width, ya que no es muy habitual definir alturas máximas y mínimas.

Valores	unidad de medida | porcentaje | none | inherit



Desempeños

Desempeño 1

Maquetá la siguiente estructura con HTML5 y CSS3

Desempeño 2

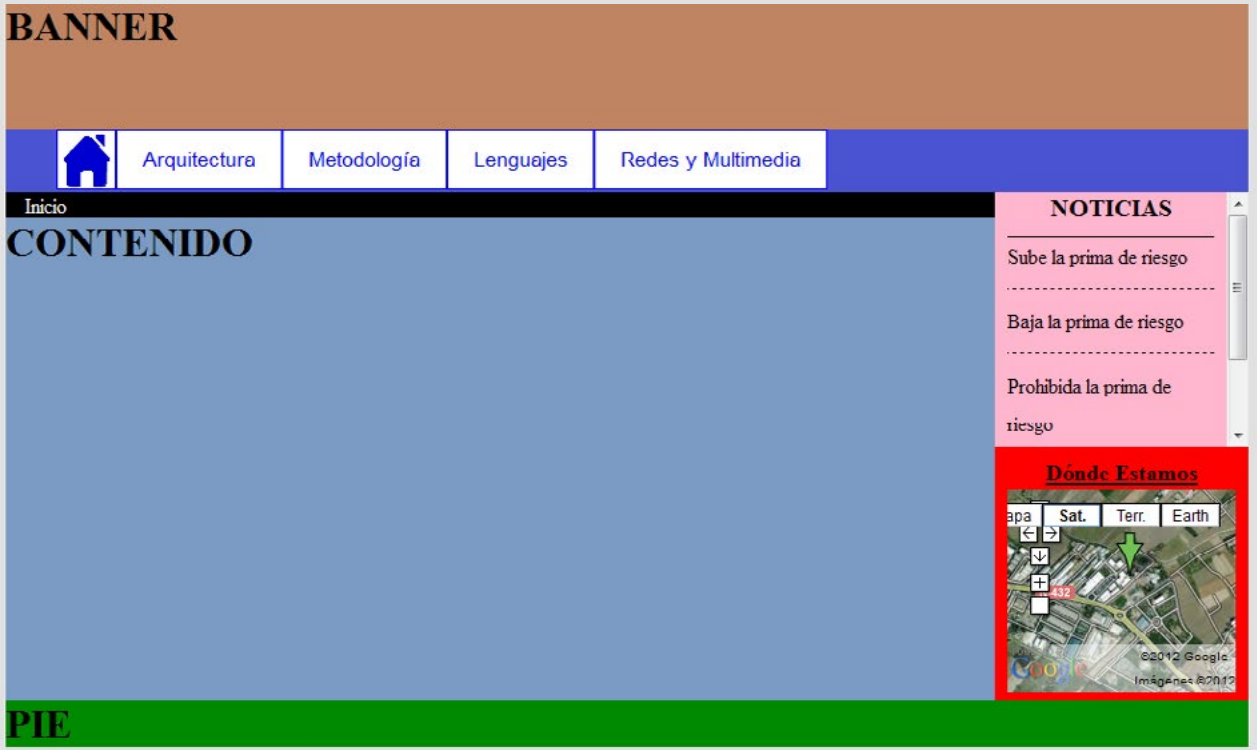
Utilizando Divs, maquetá la siguiente estructura con HTML5 y CSS3

Desempeño 3

Utilizando Divs, maquetá la siguiente estructura con HTML5 y CSS3, eligí la temática que más te guste y completá con imágenes y texto según la elección que realizaste.

Desempeño 4

Utilizando Divs, maquetá la siguiente estructura con HTML5 y CSS3, eligí la temática que más te guste y completá con imágenes y texto según la elección que realizaste.

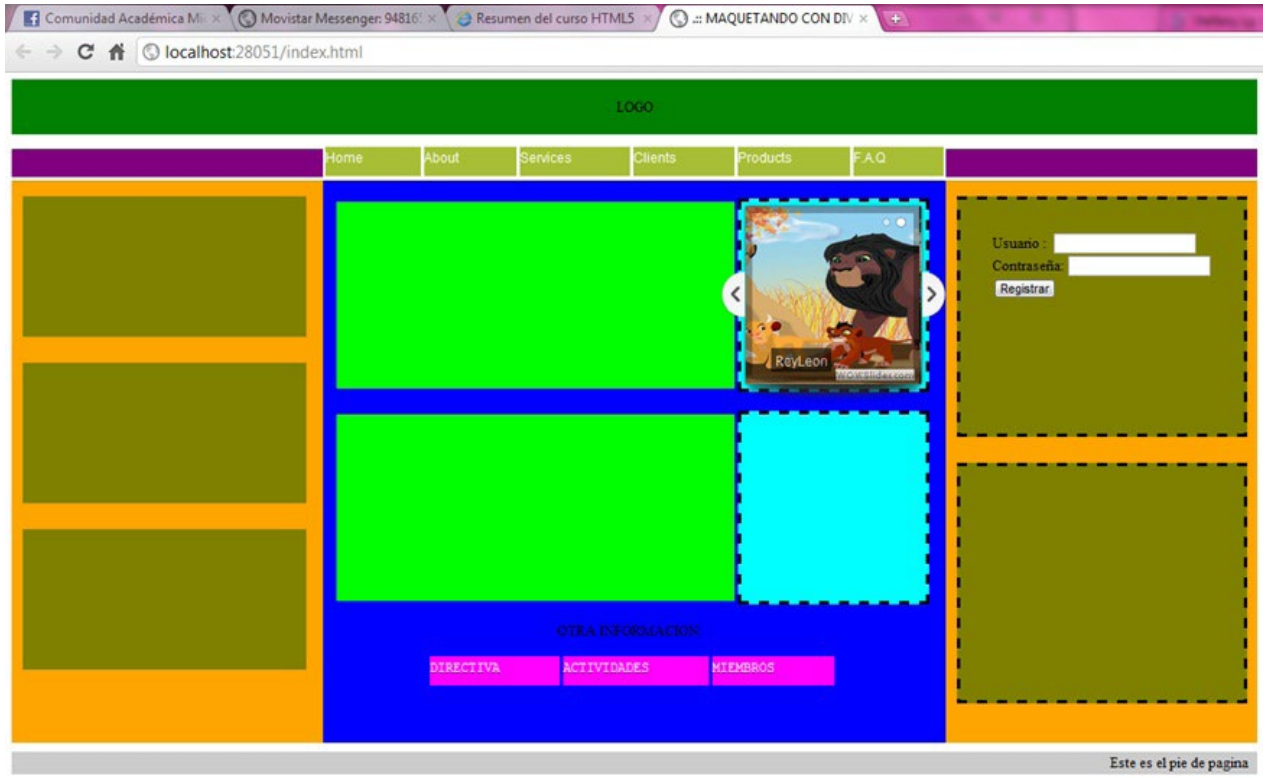


Desempeño 5

Al igual que en el desempeño anterior, utilizando Divs, maquetá la siguiente estructura con HTML5 y CSS3, eligí la temática que más te guste y completá con imágenes y texto según la elección que realizaste.

Desempeño 6

Nuevamente, utilizando Divs, maquetá la siguiente estructura con HTML5 y CSS3, eligí la temática que más te guste y completá con imágenes y texto según la elección que realizaste.



Autoevaluación

1 | Indicá el ámbito de aplicación de los selectores que te mostramos en el cuadro y respondé a las preguntas de la página siguiente.

Valor	Descripción
.header { … } .find form .search .button { … } .find, form, .search, .button { … }	Afecta a elementos cuya clase sea “header”. Afecta a elementos cuya clase sea “button” ó “search” ó “find” ó estén dentro de un <form> … </form>.
.header label { … } #cohe, #cobo { … } .syndicate-module img { … } #form-login p { … } #login br { … } #login .button { … } label.invalid { … } div.itemCommentsForm form input#submitCommentButton #s5_bottom_menu_wrap ul.menu li { … } .module_round_box ul.menu ul a { … }	

Autoevaluación

1 | ¿Qué sucede cuando dos márgenes adyacentes de un contexto de formato estático se superponen y uno o ambos márgenes son negativos?

2 | ¿Cómo se pueden utilizar los elementos flotantes para mostrar miniaturas de imágenes en una galería, como "celdas" del mismo tamaño, sin utilizar una tabla para la distribución?

3 | ¿Cómo podes tener un menú de navegación vertical a la izquierda de la página y una columna de contenido a la derecha sin que el texto del contenido rodee el menú por debajo?

4 | ¿En qué circunstancias es mejor utilizar el valor abreviado como margin o una única propiedad de margen como margin-top?

5 | Cuando las propiedades abreviadas de margin, padding y border-width se presentan con los cuatro valores ¿en qué orden se aplican estos valores a los cuatro lados de un elemento?

6 | ¿Qué valor de border-style utilizarías para hacer que un elemento parezca un botón de interfaz?

7 | Si querés aplicarle un valor float a un elemento ¿qué otra propiedad también tenés que establecer en este elemento?

8 | Si quisieras estar del todo seguro de que un elemento siempre se ampliará para llenar la anchura de su contenedor ¿qué parejas de propiedad/valor establecerías?

9 | Si una imagen es demasiado grande para el elemento contenedor ¿qué pareja de propiedad/valor utilizarías para garantizar que no se creen errores en la composición de la página? ¿por qué?

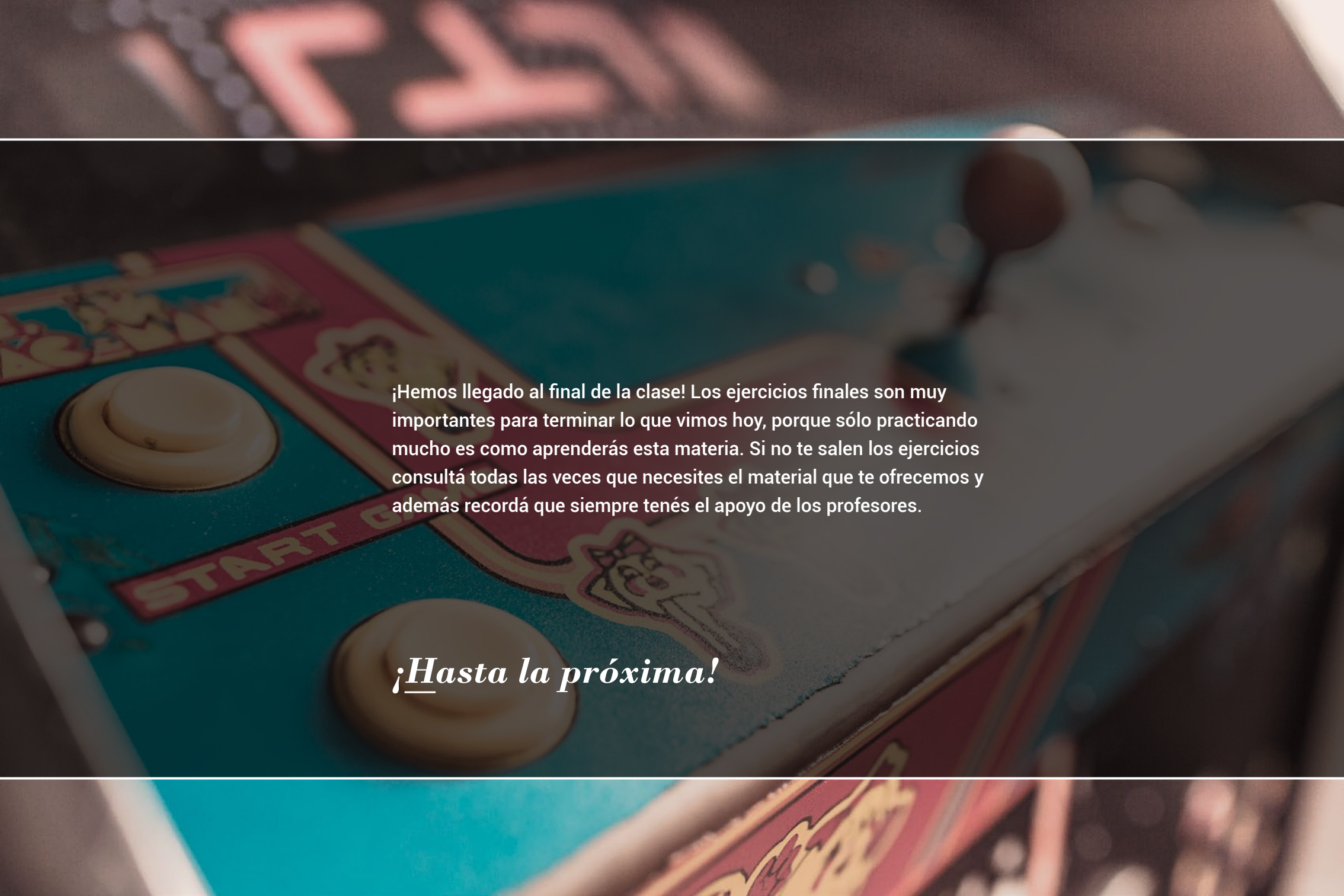
10 | En circunstancias normales, un elemento de bloque se amplía hasta llenar la anchura de su contenedor (menos márgenes, bordes y relleno). Por defecto, ¿cambia realmente este comportamiento cuando este elemento va precedido de un elemento con float, o sólo parece que cambia?

Autoevaluación

11 | Si querés aplicar un valor float a un elemento ¿qué otra propiedad también deberías establecer en este elemento?

12 | Sin ejecutar código, indicá cuáles deberían ser los resultados obtenidos al añadir las líneas que se indican en la tabla al archivo CSS. Completá la tabla primero sin ejecutar el código. Luego, compará tu solución con la expuesta.

Fragmento de código añadido	Resultado
li {display: inline;}	----------
img {display: block;}	----------
#menu1 {display:none;}	----------
h2 {margin-left:30px; display:list-item;}	----------
li {display: inline-block;}	----------
ul {display: table; } li {display: table-cell; padding:10px;}	----------



¡Hemos llegado al final de la clase! Los ejercicios finales son muy importantes para terminar lo que vimos hoy, porque sólo practicando mucho es como aprenderás esta materia. Si no te salen los ejercicios consultá todas las veces que necesites el material que te ofrecemos y además recordá que siempre tenés el apoyo de los profesores.

¡Hasta la próxima!