

Punto
Digital

Plataforma de
Aprendizaje Virtual

Programación Web



Jefatura de
Gabinete de Ministros
Argentina

Secretaría de
Innovación Pública

Módulo 3

El Modelo de Cajas

Una página web puede ser vista como una sucesión y un conjunto de cajas llamadas

«bloques». La mayoría de los elementos vistos en los capítulos anteriores son bloques: `<header>`, `<article>`, `<nav>`... Pero existen otros bloques: párrafos `<p>`, títulos `<h1>`, etc.

En este capítulo, vamos a aprender cómo interactuar con estos bloques como si fuera cajas reales. Vamos a darles dimensiones, diseñarlos, pero ajustando sus márgenes, y también aprenderemos cómo gestionar su contenido... ¡para evitar que el texto salga de estos bloques!

Estos son los conceptos básicos que necesitamos para diseñar nuestra página web... ¡Así que presta atención!

Bloquear y colocar entre líneas las etiquetas de tipo

En HTML, la mayoría de las etiquetas pueden colocarse en una o dos categorías:

- etiquetas inline: este es el caso, por ejemplo, de los enlaces `<a>`/``.
- etiquetas de bloque: este es el caso, por ejemplo, de los párrafos `<p>`/`</p>`.

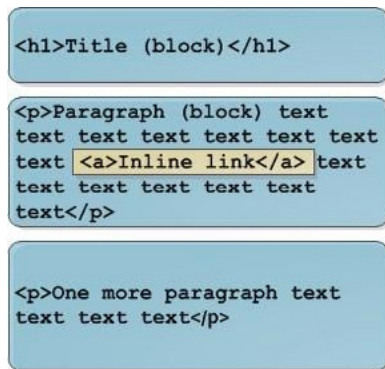
En realidad, hay varias categorías muy específicas, como las celdas de las tablas (`table-cell`) o viñetas (`list-item`). No nos vamos a preocupar por eso de momento, ya que estas etiquetas son una minoría.

¿Pero cómo indicas una etiqueta inline de una etiqueta de bloque?

En realidad, es bastante fácil:

- **block:** una etiqueta de bloque en tu página web automáticamente crea un salto de línea antes y después. Sólo tenés que imaginar un bloque. Tu página consistirá, de hecho, en una serie de bloques, uno después del otro. Pero, además, ¡verás que es posible poner un bloque dentro de otro, lo que aumentará significativamente nuestras habilidades de crear el diseño de nuestra página web!
- **inline:** una etiqueta inline está necesariamente situada en una etiqueta de bloque. Una etiqueta inline no crea una vuelta de línea, así que el texto escrito dentro sigue desde el texto anterior en la misma línea (por eso se conoce como etiqueta «inline»).

Para poder visualizar el concepto correctamente, he creado un pequeño diagrama en la siguiente figura.



Diferencia entre una etiqueta inline y otra de bloque

Las etiquetas de bloque se muestran con fondo azul.

Las etiquetas inline se muestran con fondo amarillo.

Como podés ver, los bloques están debajo uno de otro. Pueden ser anidados unos con otros (recuerda nuestra <section> de bloques contiene bloques <aside>

(Por ejemplo), la etiqueta inline <a> está situada en una etiqueta de bloque y el texto está insertado en la misma línea.

Unos pocos ejemplos

Para ayudarte a entender qué etiquetas son inline y cuáles son de bloque, aquí hay una tabla con algunas etiquetas habituales.

Etiquetas de bloque	Etiquetas de inline
<p>	
<footer>	
<h1>	<mark>
<h2>	<a>
<article>	

Etiquetas para objetivos generales

Estas son etiquetas que no tienen un significado en particular (a diferencia de <p> que significa «párrafo», que significa «importante», etc.).

La principal ventaja de estas etiquetas es que una clase (o un id) puede aplicarse al CSS cuando no hay otra etiqueta adecuada.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Dimensiones

Aquí vamos a trabajar solamente con etiquetas de tipo bloque.

Para empezar, miremos el tamaño de los bloques. Al contrario que una etiqueta inline, un bloque tiene dimensiones específicas. Tiene anchura y altura. Así que tenemos dos propiedades CSS:

- **width:** es la anchura del bloque. Para especificarse en píxeles (px) o como porcentaje (%).
- **height:** es la altura del bloque. Que, de nuevo, se especifica en píxeles (px) o en porcentaje (%).

Para ser exactos, width y height representan la anchura y la altura del contenido de bloques. Si el bloque tiene márgenes (aprenderemos más sobre esto más adelante), se añadirán a la anchura y a la altura.

Por defecto, un bloque abarca el 100 % de la anchura disponible. Esto se puede comprobar aplicando bordes o un color de fondo a nuestros bloques (figura siguiente).

Título

Párrafo

```
347 .widget-area-sidebar {
348     font-size: 13px;
349 }
350
351
352 /* =Menu
353 -----
354
355 #access {
356     display: inline-block;
357     height: 69px;
358     float: right;
359     margin: 11px 28px 0px 0px;
360     max-width: 800px;
361 }
362
363 #access ul {
364     font-size: 13px;
365     list-style: none;
366     margin: 0 0 0 -0.8125em;
367     padding-left: 0;
368     z-index: 99999;
369     text-align: right;
370 }
```

Lenguaje CSS

En el capítulo anterior, hemos aprendido a crear un documento HTML, a organizar su estructura, y a determinar qué elementos son más apropiados para representar su contenido. Esta información nos permite definir el documento, pero no determina cómo se mostrará en pantalla.

Estilos

Desde la introducción de HTML5, esa tarea es responsabilidad de CSS. CSS es un lenguaje que facilita instrucciones que podemos usar para asignar estilos a los elementos HTML, como colores, tipos de letra, tamaños, etc. Los estilos se deben definir con CSS y luego asignar a los elementos hasta que logramos el diseño visual que queremos para nuestra página.

Por razones de compatibilidad, los navegadores asignan estilos por defecto a algunos elementos HTML. La mayoría de estos estilos deben ser reemplazados o complementados con estilos personalizados. En CSS, los estilos personalizados se declaran con propiedades. Un estilo se define declarando el nombre de la propiedad y su valor separados por dos puntos. Por ejemplo, el siguiente código declara una propiedad que cambia el tamaño de la letra a 24 píxeles.

```
font-size: 24px;
```

ejemplo 2-1: Declarando propiedades CSS

Si la propiedad del **ejemplo 2-1** se aplica a un elemento, el texto contenido por ese elemento se mostrará en la pantalla con el tipo de letra definido por defecto, pero con un tamaño de 24 píxeles.

La mayoría de las propiedades CSS pueden modificar un solo aspecto del elemento (el tamaño de la letra en este caso). Si queremos cambiar varios estilos al mismo tiempo, tenemos que declarar múltiples propiedades. CSS define una sintaxis que simplifica el proceso de asignar múltiples propiedades a un elemento. La construcción se llama regla. Una regla es una lista de propiedades declaradas entre llaves e identificadas por un selector. El selector indica qué elementos se verán afectados por las propiedades. Por ejemplo, la siguiente regla:

```
P {  
  color: #FF0000;  
  font-size: 24px;  
}
```

ejemplo 2-2: Declarando reglas CSS

La regla del **ejemplo 2-2** incluye dos propiedades con sus respectivos valores agrupadas por llaves (color y font-size). Si aplicamos esta regla a nuestro documento, el texto dentro de cada elemento <p> se mostrará en color rojo y con un tamaño de 24 píxeles.

Aplicando Estilos

Las propiedades y las reglas definen los estilos que queremos asignar a uno o más elementos, pero estos estilos no se aplican hasta que los incluimos en el documento html. Existen tres técnicas disponibles para este propósito. Podemos usar estilos en línea, estilos incrustados u hojas de estilo.

El primero, estilos en línea, utiliza un atributo global llamado **style** para insertar los estilos directamente en el elemento. Este atributo está disponible en cada uno de los elementos HTML y puede recibir una propiedad o una lista de propiedades que se aplicarán al elemento al que pertenece. Si queremos asignar estilos usando esta técnica, todo lo que tenemos que hacer es declarar el atributo style en el elemento que queremos modificar y asignarle las propiedades CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
</head>
<body>
  <main>
    <section>
      <p style="font-size: 20px;">Mi Texto</p>
    </section>
  </main>
</body>
</html>
```

ejemplo 2-3: Estilos en Línea

El documento del ejemplo 2-3 incluye un elemento

<p> con el atributo style y el valor font-size: 20px;

Cuando el navegador lee este atributo, le asigna un tamaño de 20 píxeles al texto dentro del elemento <p>.

Ejercitación en Sublime Text: cree un nuevo archivo HTML con el código del ejemplo 2-3 y abra el documento en su navegador. Debería ver el texto "Mi Texto" con letras en el tamaño definido por la propiedad font-size. Intente cambiar el valor de esta propiedad para ver cómo se presentan en pantalla los diferentes tamaños de letra.

Los estilos en línea son una manera práctica de probar estilos y ver cómo modifican los elementos, pero no se recomiendan para proyectos extensos. La razón es que el atributo style solo afecta al elemento en el que se ha declarado. Si queremos asignar el mismo estilo a otros elementos, tenemos que repetir el código en cada uno de ellos, lo cual incrementa innecesariamente el tamaño del documento, complicando su actualización y mantenimiento.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Por ejemplo, si más adelante decidimos que en insertar las reglas CSS en la cabecera del documento lugar de 20 píxeles el tamaño del texto en cada elemento `<p>` deber ser de 24 píxeles, tendríamos que cambiar cada estilo en cada uno de los elementos `<p>` del documento completo y en todos los documentos de nuestro sitio web.

Una alternativa es la de usando selectores que determinan los elementos que se verán afectados. Para este propósito, HTML incluye el elemento `<style>`.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <style>
    p {
      font-size: 20px;
    }
  </style>
</head>
<body>
  <main>
    <section>
      <p>Mi texto</p>
    </section>
  </main>
</body>
</html>
```

ejemplo 2-4: Incluyendo estilos en la cabecera del documento

La propiedad declarada entre las etiquetas `<style>` en el documento del ejemplo 2-4 cumple la misma función que la declarada en el documento del ejemplo 3-3, pero en este ejemplo no tenemos que escribir el estilo dentro del elemento `<p>` que queremos modificar porque, debido al selector utilizado, todos se ven afectados por esta regla.

Declarar estilos en la cabecera del documento ahorra espacio y hace que el código sea más coherente y fácil de mantener, pero requiere que copiemos las mismas reglas en cada uno de los documentos de nuestro sitio web.

Debido a que la mayoría de las páginas compartirán el mismo diseño, varias de estas reglas se duplicarán. La solución es mover los estilos a un archivo CSS y luego usar el elemento `<link>` para cargarlo desde cada uno de los documentos que lo requieran.

Este tipo de archivos se denomina **hojas de estilo**, pero no son más que archivos de texto con la lista de reglas CSS que queremos asignar a los elementos del documento.

El elemento `<link>` se usa para incorporar recursos externos al documento. Dependiendo del tipo de recurso que queremos cargar, tenemos que declarar diferentes atributos y valores. Para cargar hojas de estilo CSS, solo necesitamos los atributos **rel** y **href**. El atributo **rel** significa relación y especifica la relación entre el documento y el archivo que estamos incorporando, por lo que debemos declararlo con el valor **stylesheet** para comunicarle al navegador que el recurso es un archivo CSS con los estilos requeridos para presentar la página. Por otro lado, el atributo **href** declara la URL del archivo a cargar.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <main>
    <section>
      <p>Mi texto</p>
    </section>
  </main>
</body>
</html>
```

ejemplo 2-5: Aplicando estilos CSS desde un archivo externo

El documento del ejemplo 3-5 carga los estilos CSS desde el archivo [misestilos.css](#). En este archivo tenemos que declarar las reglas CSS que queremos aplicar al documento, tal como lo hemos hecho anteriormente dentro del elemento `<style>`.

El siguiente es el código que debemos insertar en el archivo [misestilos.css](#) para producir el mismo efecto logrado en ejemplos anteriores.

```
p {
  font-size: 20px;
}
```

ejemplo 2-6: Definiendo estilos CSS en un archivo externo

La práctica de incluir estilos CSS en un archivo aparte es ampliamente utilizada por programadoras se recomienda para sitios web diseñados con HTML5, no solo porque podemos definir una sola hoja de estilo y luego incluirla en todos los documentos con el elemento `<link>`, sino porque podemos reemplazar todos los estilos a la vez simplemente cargando un archivo diferente, lo cual nos permite probar diferentes diseños y adaptar el sitio web a las pantallas de todos los dispositivos disponibles.

Ejercitación en Sublime Text: cree un archivo HTML con el código del ejemplo 2-5 y un archivo llamado `misestilos.css` con el código del ejercicio 2-6. Abra el documento en su navegador. Debería ver el texto dentro del elemento `<p>` con un tamaño de 20 píxeles. Cambie el valor de la propiedad `font-size` en el archivo CSS y actualice la página para ver cómo se aplica el estilo al documento.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Nombres

Una regla declarada con el nombre del elemento como selector afecta a todos los elementos de ese tipo encontrados en el documento. En ejemplos anteriores usamos el nombre **p** para modificar elementos **<p>**, pero podemos cambiar este nombre para trabajar con cualquier elemento en el documento que deseemos. Si en su lugar declaramos el nombre **span**, por ejemplo, se modificarán todos los textos dentro de elementos ****.

```
span {  
  font-size: 20px;  
}
```

ejemplo 2-7: Referenciando elementos por su nombre

Si queremos asignar los mismos estilos a elementos con nombres diferentes, podemos declarar los nombres separados por una coma. En el siguiente ejemplo, la regla afecta a todos los elementos **<p>** y **** encontrados en el documento.

```
p, span {  
  font-size: 20px;  
}
```

ejemplo 2-8: Declarando reglas con múltiples selectores

También podemos referenciar solo elementos que se encuentran dentro de un elemento en particular listando los selectores separados por un espacio. Estos tipos de selectores se llaman selectores de descendencia porque afectan a elementos dentro de otros elementos, sin importar el lugar que ocupan en la jerarquía.

```
main p {  
  font-size: 20px;  
}
```

ejemplo 2-9: Combinando selectores

La regla en el ejemplo 2-9 afecta solo a los elementos **<p>** que se encuentran dentro de un elemento **<main>**, ya sea como contenido directo o insertados en otros elementos.

Por ejemplo, el siguiente documento incluye una sección principal con una cabecera y una sección adicional. Ambos elementos incluyen elementos **<p>** para representar su contenido. Si aplicamos la regla del ejemplo 2-9 a este documento, el texto dentro de cada elemento **<p>** se mostrará con un tamaño de 20 píxeles porque todos son descendientes del elemento **<main>**.

```

<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>

<body>
  <main>
    <header>
      <h1>Título</h1>
      <p>Esta es la introducción</p>
    </header>
    <section>
      <p>Frase 1</p>
      <p>Frase 2</p>
      <p>Frase 3</p>
      <p>Frase 4</p>
    </section>
  </main>
</body>
</html>

```

ejemplo 2-10: Probando selectores

La regla del ejemplo 2-9 solo afecta a elementos **<p>** que se encuentran dentro del elemento **<main>**. Si, por ejemplo, agregamos un elemento **<footer>** al final del documento

del ejemplo 2-10, los elementos **<p>** dentro de este pie de página no se verán modificados. La Figura 2-1 muestra lo que vemos cuando abrimos este documento en el navegador.



Figura 2-1: Selectores de descendencia

Atributo Id

Las reglas anteriores afectan a los elementos del tipo indicado por el selector. Para seleccionar un elemento HTML sin considerar su tipo, podemos usar el atributo id. Este atributo es un nombre, un identificador exclusivo del elemento y, por lo tanto, lo podemos usar para encontrar un elemento en particular dentro del documento. Para referenciar un elemento usando su atributo id, el selector debe incluir el valor del atributo precedido por el carácter numeral (#).

```
#mitexto {  
  font-size: 20px;  
}
```

ejemplo 2-11: Referenciando por medio del valor del atributo id

La regla del ejemplo 3-16 solo se aplica al elemento identificado por el atributo id y el valor "mitexto", como el elemento <p> incluido en el siguiente documento.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <title>Este texto es el título del documento</title>  
  <meta charset="utf-8">  
  <link rel="stylesheet" href="misestilos.css">  
</head>  
<body>  
  <main>  
    <section>  
      <p>Frase 1</p>  
      <p id="mitexto">Frase 2</p>  
      <p>Frase 3</p>  
    </section>  
  </main>  
</body>  
</html>
```

ejemplo 2-12: Identificando un elemento <p> por medio de su atributo id

La ventaja de este procedimiento es que cada vez que creamos una referencia usando el identificador mitexto en nuestro archivo CSS, solo se modifica el elemento con esa identificación, pero el resto de los elementos no se ven afectados.

Atributo Class

En lugar de usar el atributo `id` para asignar estilos, en la mayoría de las ocasiones es mejor hacerlo con el atributo **class**. Este atributo es más flexible y se puede asignar a varios elementos dentro del mismo documento.

```
.mitexto {  
    font-size: 20px;  
}
```

ejemplo 2-13: Referenciando por medio del valor del atributo class

Para referenciar un elemento usando su atributo `class`, el selector debe incluir el valor del atributo precedido por un punto. Por ejemplo, la regla del ejemplo 2-13 afecta a todos los elementos que contienen un atributo `class` con el valor "mitexto", como en el siguiente ejemplo.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <title>Este texto es el título del documento</title>  
    <meta charset="utf-8">  
    <link rel="stylesheet" href="misestilos.css">  
</head>  
  
<body>  
    <main>  
        <section>  
            <p class="mitexto">Frase 1</p>  
            <p class="mitexto">Frase 2</p>  
            <p>Frase 3</p>  
            <p>Frase 4</p>  
        </section>  
    </main>  
</body>  
</html>
```

ejemplo 2-14: Asignando estilos con el atributo class

Los elementos `<p>` de las dos primeras líneas dentro de la sección principal del documento del ejemplo 2-14 incluyen el atributo `class` con el valor "mitexto". Debido a que se puede aplicar la misma regla a diferentes elementos del documento, estos dos primeros elementos son afectados por la regla del ejemplo 2-13. Por otro lado, los dos últimos elementos `<p>` no incluyen el atributo `class` y, por lo tanto, se mostrarán con los estilos por defecto.

Las reglas asignadas a través del atributo `class` se denominan clases. A un mismo elemento se le pueden asignar varias clases. Todo lo que tenemos que hacer es declarar los nombres de las clases separados por un espacio (por ejemplo, `class="texto1 texto2"`).

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Propiedades

Las propiedades son la pieza central de CSS. Todos los estilos que podemos aplicar a un elemento se definen por medio de propiedades. Ya hemos introducido algunas en los ejemplos anteriores, pero hay cientos de propiedades disponibles. Para simplificar su estudio, se pueden clasificar en dos tipos: propiedades de formato y propiedades de diseño.

Las propiedades de formato se encargan de dar forma a los elementos y su contenido, mientras que las de diseño están enfocadas a determinar el tamaño y la posición de los elementos en la pantalla. Así mismo, las propiedades de formato se pueden clasificar según el tipo de modificación que producen. Por ejemplo, algunas propiedades cambian el tipo de letra que se usa para mostrar el texto, otras generan un borde alrededor del elemento, asignan un color de fondo, etc.

En esta sección vamos a introducir las propiedades de formato siguiendo esta clasificación.

Texto

Desde CSS se pueden controlar varios aspectos del texto, como el tipo de letra que se usa para mostrar en pantalla, el espacio entre líneas, la alineación, etc. Las siguientes son las propiedades disponibles para definir el tipo de letra, tamaño, y estilo de un texto.

font-family—Esta propiedad declara el tipo de letra que se usa para mostrar el texto. Se pueden declarar múltiples valores separados por coma para ofrecer al navegador varias alternativas en caso de que algunos tipos de letra no se encuentren disponibles en el ordenador del usuario. Algunos de los valores estándar son Georgia, “Times New Roman”, Arial, Helvetica, “Arial Black”, Gadget, Tahoma, Geneva, Helvetica, Verdana, Geneva, Impact, y sans-serif (los nombres compuestos por más de una palabra se deben declarar entre comillas dobles).

font-size—Esta propiedad determina el tamaño de la letra. El valor puede ser declarado en píxeles (px), porcentaje (%), o usando cualquiera de las unidades disponibles en CSS como em, rem, pt, etc. El valor por defecto es normalmente 16px.

font-weight—Esta propiedad determina si el texto se mostrará en negrita o no. Los valores disponibles son normal y bold, pero también podemos asignar los valores 100, 200, 300, 400, 500, 600, 700, 800, y 900 para determinar el grosor de la letra (solo disponibles para algunos tipos de letra).

font-style—Esta propiedad determina el estilo de la letra. Los valores disponibles son normal, italic, y oblique.

font—Esta propiedad nos permite declarar múltiples valores al mismo tiempo. Los valores deben declararse separados por un espacio y en un orden preciso. El estilo y el grosor se deben declarar antes que el tamaño, y el tipo de letra al final (por ejemplo, font: bold 24px Arial, sans-serif).

Estas propiedades se deben aplicar a cada elemento (o elemento padre) cuyo texto queremos modificar. Por ejemplo, el siguiente documento incluye una cabecera con un título y una sección con un párrafo. Como queremos asignar diferentes tipos de letra al título y al texto, tenemos que incluir el atributo id en cada elemento para poder identificarlos desde las reglas CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Este texto es el título del documento</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="misestilos.css">
</head>
<body>
  <header>
    <span id="titulo">Hojas de Estilo en Cascada</span>
  </header>
  <section>
    <p id="descripcion">Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en HTML.</p>
  </section>
  <footer>
    <a href="http://www.jdgauchat.com">www.jdgauchat.com</a>
  </footer>
</body>
</html>
```

ejemplo 2-15: Probando propiedades de formato

Las reglas deben incluir todas las propiedades requeridas para asignar los estilos que deseamos. Por ejemplo, si queremos asignar un tipo de letra y un nuevo tamaño al texto, tenemos que incluir las propiedades **font-family** y **font-size**.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
```

ejemplo 2-16: Cambiando el tipo de letra con la propiedad font

Cuando la regla del ejemplo 2-16 se aplica al documento del ejemplo 2-15, el título se muestra con los valores definidos por la propiedad font y el párrafo se presenta con los estilos por defecto.

Hojas de Estilo en Cascada

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en HTML.

www.jdgauchat.com

Figura 2-2: La propiedad font

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Desde CSS podemos cambiar no solo el tipo de letra, sino también otros aspectos del texto, como el alineamiento, la sangría, el espacio entre líneas, etc. Las siguientes son algunas de las propiedades disponibles para este propósito.

text-align—Esta propiedad alinea el texto dentro de un elemento. Los valores disponibles son `left`, `right`, `center`, y `justify`.

text-align-last—Esta propiedad alinea la última línea de un párrafo. Los valores disponibles son `left`, `right`, `center`, y `justify`.

text-indent—Esta propiedad define el tamaño de la sangría de un párrafo (el espacio vacío al comienzo de la línea). El valor se puede declarar en píxeles (px), porcentaje (%), o usando cualquiera de las unidades disponibles en CSS, como `em`, `rem`, `pt`, etc.

letter-spacing—Esta propiedad define el espacio entre letras. El valor se debe declarar en píxeles (px), porcentaje (%) o usando cualquiera de las unidades disponibles en CSS, como `em`, `rem`, `pt`, etc.

word-spacing—Esta propiedad define el ancho del espacio entre palabras. El valor puede ser declarado en píxeles (px), porcentaje (%) o usando cualquiera de las unidades disponibles en CSS, como `em`, `rem`, `pt`, etc.

line-height—Esta propiedad define el espacio entre líneas. El valor se puede declarar en píxeles (px), porcentaje (%) o usando cualquiera de las unidades disponibles en CSS, como `em`, `rem`, `pt`, etc.

vertical-align—Esta propiedad alinea elementos verticalmente. Se usa frecuentemente para alinear texto con imágenes (la propiedad se aplica a la imagen). Los valores disponibles son `baseline`, `sub`, `super`, `text-top`, `text-bottom`, `middle`, `top`, y `bottom`.

Colores

Existen dos formas de declarar un color en CSS: podemos usar una combinación de tres colores básicos (rojo, verde y azul), o definir el matiz, la saturación y la luminosidad. El color final se crea considerando los niveles que asignamos a cada componente. Dependiendo del tipo de sistema que utilizamos para definir el color, tendremos que declarar los niveles usando números hexadecimales (desde 00 a FF), números decimales (desde 0 a 255) o porcentajes.

Por ejemplo, si decidimos usar una combinación de niveles de rojo, verde y azul, podemos declarar los niveles con números hexadecimales. En este caso, los valores del color se declaran en secuencia y precedidos por el carácter numeral, como en `#996633` (99 es el nivel de rojo, 66 es el nivel de verde y 33 es el nivel de azul). Para definir colores con otros tipos de valores, CSS ofrece las siguientes funciones.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



rgb (rojo, verde, azul) — Esta función define un color por medio de los valores especificados por los atributos (desde 0 a 255). El primer valor representa el nivel de rojo, el segundo valor representa el nivel de verde y el último valor el nivel de azul (por ejemplo, `rgb(153, 102, 51)`).

rgba (rojo, verde, azul, alfa) — Esta función es similar a la función `rgb()`, pero incluye un componente adicional para definir la opacidad (alfa). El valor se puede declarar entre 0 y 1, con 0 como totalmente transparente y 1 como totalmente opaco.

hsl (matiz, saturación, luminosidad) — Esta función define un color desde los valores especificados por los atributos. Los valores se declaran en números decimales y porcentajes.

hsla (matiz, saturación, luminosidad, alfa) — Esta función es similar a la función `hsl()`, pero incluye un componente adicional para definir la opacidad (alfa). El valor se puede declarar entre 0 y 1, con 0 como totalmente transparente y 1 como totalmente opaco.

Como veremos más adelante, son varias las propiedades que requieren valores que definen colores, pero la siguiente es la que se utiliza con más frecuencia:

color — Esta propiedad declara el color del contenido del elemento.

La siguiente regla asigna un gris claro al título de nuestro documento usando números hexadecimales.

```
#titulo {  
  font: bold 26px Verdana, sans-serif;  
  color: #CCCCCC;  
}
```

ejemplo 2-17: Asignando un color al título

Cuando los niveles de rojo, verde y azul son iguales, como en este caso, el color final se encuentra dentro de una escala de grises, desde negro (`#000000`) a blanco (`#FFFFFF`).

Declarar un color con una función es bastante parecido: solo tenemos que reemplazar el valor hexadecimal por la función que queremos utilizar. Por ejemplo, podemos definir el mismo color de grises con la función **rgb()** y valores decimales.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
  color: rgb(204, 204, 204);
}
```

ejercicio 2-18: Asignando un color con la función rgb()

La función **hsl()** es simplemente otra función disponible para generar un color, pero es más intuitiva que **rgb()**. A algunos diseñadores les resulta más fácil crear grupos de colores usando **hsl()**. Los valores requeridos por esta función definen el matiz, la saturación y la luminosidad. El matiz es un color extraído de una rueda imaginaria, expresado en grados desde 0 a 360: alrededor de 0 y 360 se encuentran los rojos, cerca de 120 los verdes, y cerca de 240 los azules. La saturación se representa en porcentaje, desde 0 % (escala de grises) a 100 % (todo color o totalmente saturado), y la luminosidad es también un valor en porcentaje, desde 0 % (completamente negro) a 100 % (completamente blanco); un valor de 50 % representa una luminosidad promedio.

Por ejemplo, en la siguiente regla, la saturación se define como 0 % para crear un color dentro de la escala de grises y la luminosidad se especifica en 80 % para obtener un gris claro.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
  color: hsl(0, 0%, 80%);
}
```

ejercicio 2-19: Asignando un color con la función hsl()



Lo básico: no es práctico encontrar los colores adecuados para nuestro sitio web combinando números y valores. Para facilitar esta tarea, los ordenadores personales incluyen varias herramientas visuales que nos permiten seleccionar un color y obtener su valor. Además, la mayoría de los editores de fotografía y programas gráficos disponibles en el mercado incluyen una herramienta que muestra una paleta desde donde podemos seleccionar un color y obtener el correspondiente valor hexadecimal o decimal. Para encontrar los colores adecuados, puede usar estas herramientas o cualquiera de las disponibles en la Web, como www.colorhexa.com o htmlcolorcodes.com.

Tamaño

Por defecto, el tamaño de la mayoría de los elementos se determina según el espacio disponible en el contenedor. El ancho de un elemento se define como 100 %, lo cual significa que será tan ancho como su contenedor, y tendrá una altura determinada por su contenido.

CSS define las siguientes propiedades para declarar un tamaño personalizado:

width—Esta propiedad declara el ancho de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra clave auto (por defecto). Cuando el valor se especifica en porcentaje, el ancho se calcula según el navegador a partir del ancho del contenedor, y cuando se declara con el valor auto, el elemento se expande hasta ocupar todo el espacio horizontal disponible dentro del contenedor.

height—Esta propiedad declara la altura de un elemento. El valor se puede especificar en píxeles, porcentaje, o con la palabra clave auto (por defecto). Cuando el valor se especifica en porcentaje, el navegador calcula la altura a partir de la altura del contenedor, y cuando se declara con el valor auto, el elemento adopta la altura de su contenedor.

Los navegadores generan una caja alrededor de cada elemento que determina el área que ocupa en la pantalla. Cuando declaramos un tamaño personalizado, la caja se modifica y el contenido del elemento se adapta para encajar dentro de la nueva área, tal como muestra la Figura 2-3.

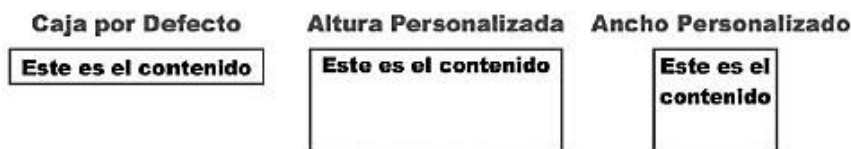


Figura 2-3: Caja personalizada

Ahora añadamos un poco de CSS para cambiar la anchura de los párrafos. El código CSS siguiente dice: «Quiero que todos mis párrafos tengan una anchura del 50 %»

```
p
{
  width: 50%;
}
```

El resultado se muestra en la siguiente figura.



Los porcentajes serán útiles para crear un diseño que automáticamente se adapte a la resolución de pantalla del visitante. Sin embargo, podrían necesitar crear bloques con un tamaño específico en píxeles.


```
p
{
  width: 250px;
}
```

Por ejemplo, si declaramos un ancho de 200 píxeles para el elemento <p> de nuestro documento, las líneas del texto serán menos largas, pero se agregarán nuevas líneas para mostrar todo el párrafo.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
}
```

ejercicio 2-20: Asignando un ancho personalizado

Hojas de Estilo en Cascada

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en HTML.

www.jdgauchat.com

Figura 2-4: Contenido principal con un tamaño personalizado

La regla del ejercicio 2-20 declara el ancho del elemento <p>, pero la altura queda aún determinada por su contenido, lo que significa que la caja generada por este elemento será más alta para contener el párrafo completo, según ilustra la Figura 2-4.

Si también queremos restringir la altura del elemento, podemos usar la propiedad height. La siguiente regla reduce la altura del elemento <p> de nuestro documento a 100 píxeles.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
}
```

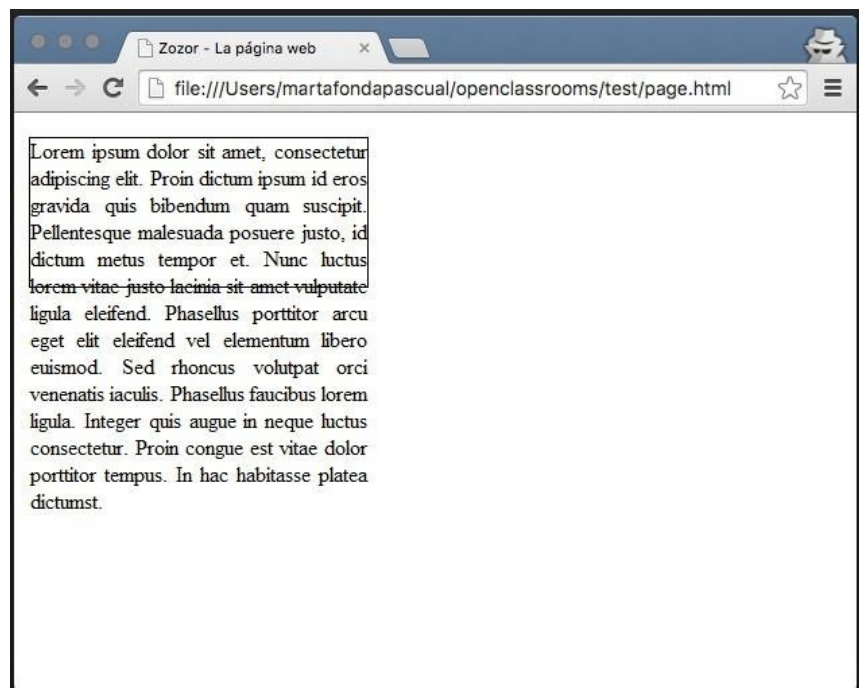
ejercicio 2-21: Asignando una altura personalizada

Cuando las cosas se desbordan...

Overflow: cortar un bloque

Supongamos que tienes un párrafo largo y querés (por cualquier razón) que sea 250px de ancho y 110px de alto. Añadámosle un borde y rellenémoslo con texto... hasta el borde (figura siguiente).

```
p
{
  width: 250px;
  height: 110px;
  text-align: justify;
  border: 1px solid black;
}
```



El texto va más allá del bloque del párrafo

El problema con elementos que tienen un tamaño definido es que a veces el contenido no se puede mostrar en su totalidad. Por defecto, los navegadores muestran el resto del contenido fuera del área de la caja. En consecuencia, parte del contenido de una caja con tamaño personalizado se puede posicionar sobre el contenido del elemento que se encuentra debajo, tal como se ilustra en la Figura 2-5.

Hojas de Estilo en Cascada

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en HTML.

www.jugandoahtml.com

Figura 2-5: Contenido desbordado

En nuestro ejemplo, el texto que se encuentra fuera de la caja determinada por el elemento

<p> ocupa el espacio correspondiente al elemento **<footer>** y, por lo tanto, el contenido de ambos elementos se encuentra superpuesto. CSS incluye las siguientes propiedades para resolver este problema:

overflow—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento. Los valores disponibles son **visible** (por defecto), **hidden** (esconde el contenido que no entra dentro de la caja), **scroll** (muestra barras laterales para desplazar el contenido), **auto** (deja que el navegador decida qué hacer con el contenido).

overflow-x—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento horizontalmente. Acepta los mismos valores que la propiedad **overflow**.

overflow-y—Esta propiedad especifica cómo se mostrará el contenido que desborda el elemento verticalmente. Acepta los mismos valores que la propiedad **overflow**.

overflow-wrap—Esta propiedad indica si una palabra debería ser dividida en un punto arbitrario cuando no hay suficiente espacio para mostrarla en la línea. Los valores disponibles son **normal** (la línea será dividida naturalmente) y **break-word** (las palabras se dividirán en puntos arbitrarios para acomodar la línea de texto en el espacio disponible).

Con estas propiedades podemos determinar cómo se mostrará el contenido cuando no hay suficiente espacio disponible. Por ejemplo, podemos ocultar el texto que desborda el elemento asignando el valor **hidden** a la propiedad **overflow**.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
  overflow: hidden;
}
```

ejercicio 2-22: Ocultando el contenido que desborda el elemento

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Hojas de Estilo en Cascada

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado.
www.jdgauchat.com

ejercicio 2-23: Ocultando el contenido que desborda el elemento

Si queremos que el usuario o la usuaria pueda ver el texto que se ha ocultado, podemos asignar el valor **scroll** y forzar al navegador a mostrar barras laterales para desplazar el contenido.

```
#titulo {
  font: bold 26px Verdana, sans-serif;
}
#descripcion {
  width: 200px;
  height: 100px;
  overflow: scroll;
}
```

ejercicio 2-24: Incorporando barras de desplazamiento

Hojas de Estilo en Cascada

Hojas de estilo en cascada (o CSS, siglas en inglés de Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear
www.jdgauchat.com

Figura 2-6: Barras de desplazamiento

¡Oh, no! ¡El texto va más allá de los límites del párrafo!

¡En efecto! Preguntaste las dimensiones exactas, ¡las conseguiste! Pero... el texto no encaja en ese bloque tan pequeño.

Si querés que el texto quede dentro de los límites del párrafo, tendrás que usar la propiedad `overflow`. Estos son los valores que puede llevar:

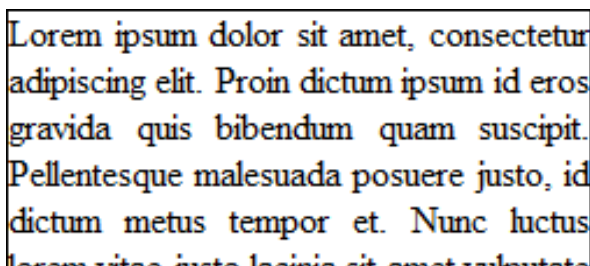
visible (valor por defecto): si el texto excede los límites del tamaño, permanece visible y deliberadamente va más allá de los límites del bloque.

hidden: si el texto excede los límites, simplemente se cortará. No seremos capaces de ver todo el texto.

scroll: de nuevo, el texto será cortado si excede los límites. Excepto esta vez, el navegador creará barras de desplazamiento para que podamos leer el texto completo. Es como un marco dentro de la página.

auto: es el modo «autopiloto». Básicamente, es el navegador el que decide si mostrar o no las barras de desplazamiento (sólo lo hará si es necesario). Este es el valor que recomiendo que uses más a menudo.

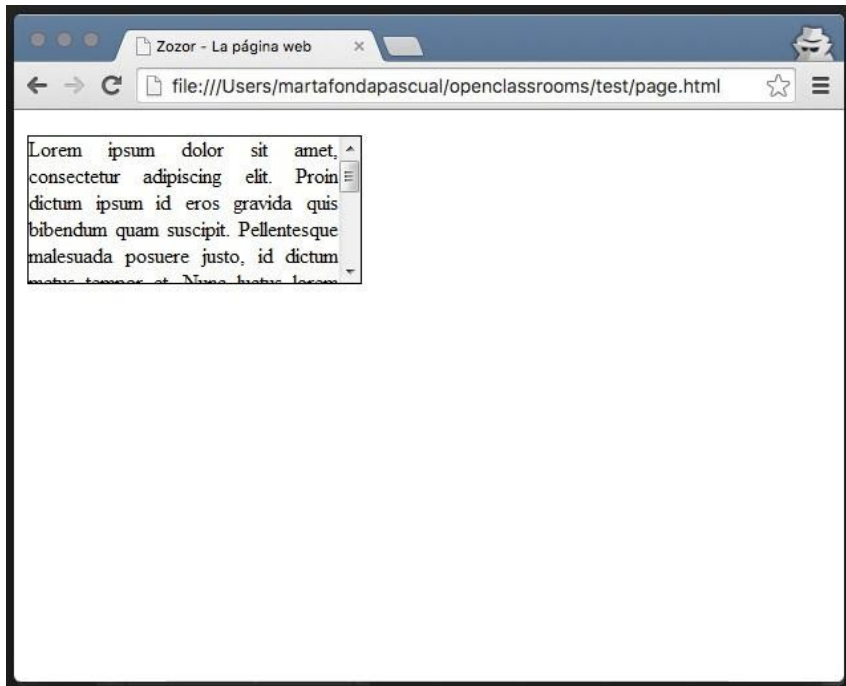
Mediante **overflow: hidden**, el texto queda cortado (no se puede ver qué hay debajo), como en la siguiente ilustración.

Una ilustración que muestra un bloque de texto con un borde negro. El texto es Lorem ipsum y está cortado por los límites del párrafo, mostrando solo las primeras líneas completas dentro del espacio asignado.

El texto queda cortado en los límites del párrafo

Probemos ahora **overflow: auto**; con el código CSS mostrado a continuación (el resultado se puede comprobar en la siguiente imagen):

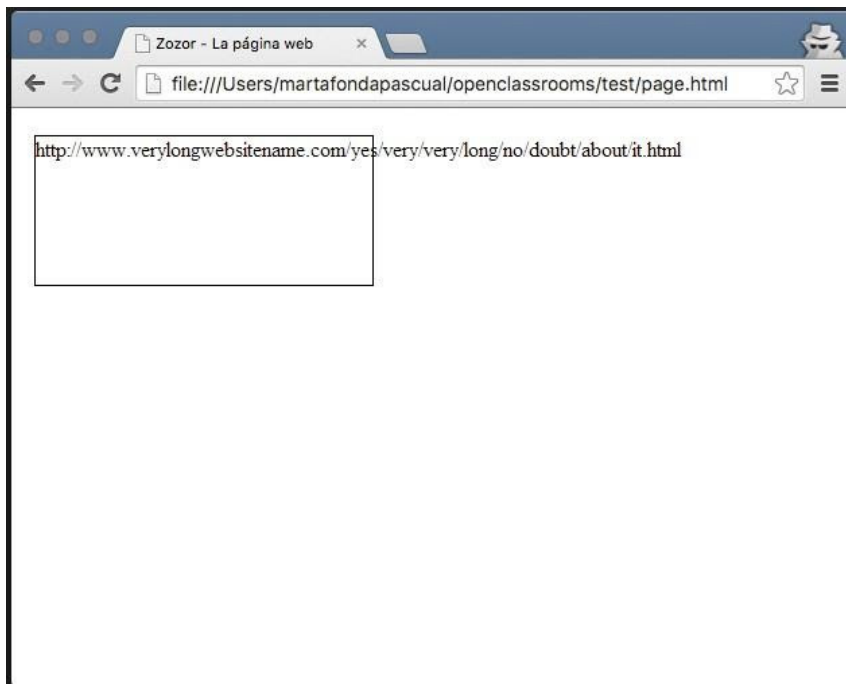
```
p
{
  width: 250px;
  height: 110px;
  text-align: justify;
  border: 1px solid black;
  overflow: auto;
}
```



word-wrap: cortar textos demasiado grandes

Si necesitas colocar una palabra muy larga en un bloque y no cabe por ser demasiado ancha, te encantará word-wrap. Esta propiedad se utiliza para forzar la partición con guion de palabras muy largas (normalmente, direcciones bastante largas).

La siguiente ilustración muestra el posible resultado de introducir una URL bastante larga en un bloque.



El texto excede el ancho disponible

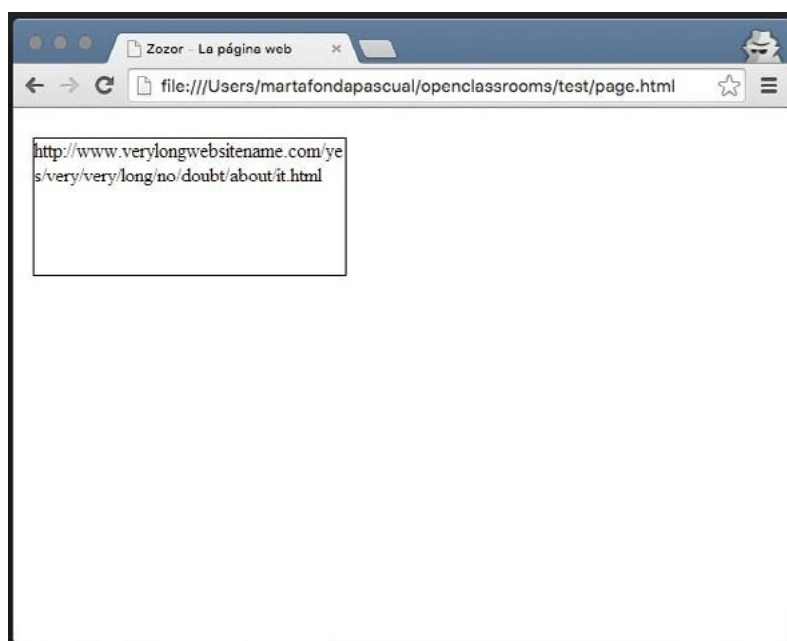
Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



El ordenador no puede «cortar» la dirección ya que no hay espacio o guion, y no sabe cómo dividirla con un guion.

Con el siguiente código, se forzará una división con guiones cuando el texto sea susceptible de exceder los límites (siguiente ilustración).

```
p
{
  word-wrap: break-word;
}
```



El texto se corta para que no exceda el espacio

Recomiendo usar esta función cuando un bloque pueda contener texto introducido por usuarios (por ejemplo, en los foros de tu futura página web).

Mínimo y máximo

Podés especificar las dimensiones máximas y mínimas de un bloque. Esto es muy útil ya que nos permite definir dimensiones «límite» para permitirle a nuestra página adaptarse a las diferentes resoluciones de las pantallas de los visitantes.

`min-width:` anchura mínima;

`min-height:` altura mínima;

`max-width:` anchura máxima;

`max-height:` altura máxima.

Por ejemplo, podés especificar que los párrafos deben ocupar en 50% del ancho y tener al menos 400 píxeles de ancho en todos los casos.

```
p
{
  width: 50%;
  min-width: 400px;
}
```

Ve el resultado cambiando la anchura de la ventana de tu navegador. Verás que, si es muy pequeño, el párrafo se ve forzado a tener una anchura de al menos 400 píxeles.

Resumen

Hay dos tipos principales de etiquetas en HTML:

- Etiquetas de bloque (`<p>`, `<h1>`...): estas etiquetas crean un salto de línea por defecto y ocupan todo el ancho disponible. Se suceden de arriba a abajo.
- Etiquetas en línea (`<a>`, ``...): estas etiquetas definen el texto en medio de una línea. Se suceden de izquierda a derecha.

Podés cambiar el tamaño de una etiqueta de bloque mediante las propiedades CSS **width** y **height**.

Podés establecer el mínimo y máximo permitido para el ancho y el alto: **min-width**, **max-width**, **min-height**, **max-height**.

Los elementos de página tienen márgenes internos (**padding**) y márgenes externos (**margin**).

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Márgenes

El tamaño del elemento no queda solo determinado por el ancho y la altura de su caja, sino también por el relleno y los márgenes. CSS nos permite designar espacio alrededor de la caja para separar el elemento de otros elementos a su alrededor (margen), además de incluir espacio entre los límites de la caja y su contenido (relleno).

La Figura 2-7 ilustra cómo se aplican estos espacios a un elemento.



Figura 2-7: Márgenes, rellenos y bordes

CSS incluye las siguientes propiedades para definir márgenes y rellenos para un elemento.

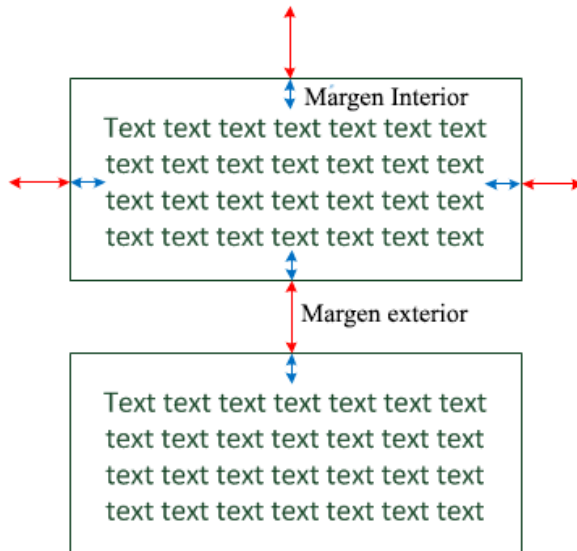
margin—Esta propiedad declara el margen de un elemento. El margen es el espacio que hay alrededor de la caja. Puede recibir cuatro valores que representan el margen superior, derecho, inferior, e izquierdo, en ese orden y separados por un espacio (por ejemplo, `margin: 10px 30px 10px 30px;`). Sin embargo, si solo se declaran uno, dos o tres valores, los otros toman los mismos valores (por ejemplo, `margin: 10px 30px` asigna 10 píxeles al margen superior e inferior y 30 píxeles al margen izquierdo y derecho). Los valores se pueden declarar independientemente usando las propiedades asociadas `margin-top`, `margin-right`, `margin-bottom` y `margin-left` (por ejemplo, `margin-left: 10px;`). La propiedad también acepta el valor `auto` para obligar al navegador a calcular el margen (usado para centrar un elemento dentro de su contenedor).

padding—Esta propiedad declara el relleno de un elemento. El relleno es el espacio entre el contenido del elemento y los límites de su caja. Los valores se declaran de la misma forma que lo hacemos para la propiedad `margin`, aunque también se pueden declarar de forma independiente con las propiedades `padding-top`, `padding-right`, `padding-bottom` y `padding-left` (por ejemplo, `padding-top: 10px;`).

Hay dos tipos de márgenes:

- márgenes interiores (padding)
- márgenes exteriores (margin)

Echá un vistazo en el diagrama en la figura siguiente.



Márgenes interiores y exteriores

He colocado un borde alrededor de este bloque para ver mejor dónde acaba.

El espacio entre el texto y el borde es el margen interior (mostrado en **azul**).

El espacio entre el borde y el siguiente bloque es el margen exterior (mostrado en **rojo**).

Otros ejemplos:

```
p
{
    width 350px;
    border: 1px solid black;
    text-align: justify;
}
```



Márgenes por defecto para los párrafos

Como podés ver, por defecto no hay margen interior (**padding**). Sin embargo, hay un margen exterior (**margin**). Es debido a este margen por lo que dos párrafos no están pegados y parecen estar separados por una «línea de separación».

Los márgenes por defecto no son los mismos para todas las etiquetas del bloque. Intenta aplicar este CSS a las etiquetas <div> que contienen texto, por ejemplo: ¡verás que en este caso por defecto no hay ningún margen interior ni exterior!

Supongamos que quiero añadir un margen interno de 12px a los párrafos (figura siguiente):

```
p
{
  width: 350px;
  border: 1px solid black;
  text-align: justify;
  padding: 12px; /* Margen interior de 12px */
}
```



Un margen interior añadido a los párrafos

Ahora quiero que mis párrafos estén más separados. Añado la propiedad **margin** para especificar un margen de 50 px entre dos párrafos (siguiente figura):

```
p
{
  width: 350px;
  border: 1px solid black;
  text-align: justify; padding: 12px;
  margin: 50px; /* Margen Exterior de 50px */
}
```

¿Pero? ¡Un margen también se añade a la izquierda!



Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



En efecto, **margin** (como **padding**) se aplica a los cuatro lados del bloque. Si querés especificar un margen superior, inferior, izquierdo y derecho diferente, tenés que usar unas propiedades más específicas... El principio es el mismo que para la propiedad **border**, como verás.

arriba, derecha, izquierda, abajo...

Tenés que recordar los siguientes términos:

top;
bottom;
left;
right;

Para que puedas mantener todas las propiedades en tu cabeza.

Igual, te haré una lista de propiedades para **margin** y **padding** para que entiendas el principio.

Aquí está la lista para **margin**:

- **margin-top**: margen exterior en la parte superior;
- **margin-bottom**: margen exterior en la parte inferior;
- **margin-left**: margen exterior en la izquierda;
- **margin-right**: margen exterior en la derecha.

Y la lista para **padding**:

- **padding-top**: margen interior en la parte superior;
- **padding-bottom**: margen interior en la parte inferior;
- **padding-left**: margen interior en la izquierda;
- **padding-right**: margen interior en la derecha.

Centrar bloques

Los bloques pueden centrarse perfectamente. Y un diseño centrado es cómodo cuando no tienes la resolución del visitante.

Hay que seguir las siguientes reglas para centrar:

dale una anchura al bloque (con la propiedad **width**);

especifica que quieres unos márgenes exteriores automáticos así: **margin: auto**; Intentemos esta técnica en nuestros pequeños párrafos (líneas 3 y 4):

```
p
{
  width: 350px; /* Especificamos una anchura (obligatorio) */
  margin: auto; /* Podemos pedir que el bloque se centre con el auto */
  border: 1px solid black;
  text-align: justify;
  padding: 12px;
  margin-bottom: 20px;
}
```

Y en la siguiente figura, el resultado.



¡Así el navegador centra automáticamente nuestros párrafos!

Un bloque no puede centrarse verticalmente con esta técnica. Solo el centrado horizontal está permitido.

Fondo

Los elementos pueden incluir un fondo que se muestra detrás del contenido del elemento y a través del área ocupada por el contenido y el relleno.

CSS define varias propiedades para generarlo.

background-color—Esta propiedad asigna un fondo de color a un elemento.

background-image—Esta propiedad asigna una o varias imágenes al fondo de un elemento. La URL del archivo se declara con la función `url()` (por ejemplo, `url("ladrillos.jpg")`). Si se requiere más de una imagen, los valores se deben separar por una coma.

background-position—Esta propiedad declara la posición de comienzo de una imagen de fondo. Los valores se pueden especificar en porcentaje, píxeles o usando una combinación de las palabras clave `center`, `left`, `right`, `top`, y `bottom`.

background-size—Esta propiedad declara el tamaño de la imagen de fondo. Los valores se pueden especificar en porcentaje, píxeles, o usando las palabras clave `cover` y `contain`.

La palabra clave `cover` expande la imagen hasta que su ancho o su altura cubren el área del elemento, mientras que `contain` estira la imagen para ocupar toda el área del elemento.

Los fondos más comunes se crean con colores. El siguiente código CSS implementa la propiedad `background-color` para agregar un fondo gris a la cabecera de nuestro documento.

background-color

```
header {  
  margin: 30px;  
  padding: 15px;  
  text-align: center;  
  background-color: #CCCCCC;  
}  
#titulo {  
  font: bold 26px Verdana, sans-serif;  
}
```

ejemplo 2-25: Agregando un color de fondo

Bordes

Los elementos pueden incluir un borde en los límites de la caja del elemento. Por defecto, los navegadores no muestran ningún borde, pero podemos usar las siguientes propiedades para definirlo.

border-width—Esta propiedad define el ancho del borde. Acepta hasta cuatro valores separados por un espacio para especificar el ancho de cada lado del borde (superior, derecho, inferior, e izquierdo, es ese orden). También podemos declarar el ancho para cada lado de forma independiente con las propiedades `border-top-width`, `border-bottom-width`, `border-left-width` y `border-right-width`.

border-style—Esta propiedad define el estilo del borde. Acepta hasta cuatro valores separados por un espacio para especificar los estilos de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). Los valores disponibles son `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, y `outset`. El valor por defecto es `none`, lo que significa que el borde no se mostrará a menos que asignemos un valor diferente a esta propiedad. También podemos declarar los estilos de forma independiente con las propiedades `border-top-style`, `border-bottom-style`, `border-left-style`, y `border-right-style`.

border-color—Esta propiedad define el color del borde. Acepta hasta cuatro valores separados por un espacio para especificar el color de cada lado del borde (superior, derecho, inferior, e izquierdo, en ese orden). También podemos declarar los colores de forma independiente con las propiedades `border-top-color`, `border-bottom-color`, `border-left-color`, y `border-right-color`.

border—Esta propiedad nos permite declarar todos los atributos del borde al mismo tiempo. También podemos usar las propiedades `border-top`, `border-bottom`, `border-left`, y `border-right` para definir los valores de cada borde de forma independiente.

Para asignar un borde a un elemento, todo lo que tenemos que hacer es definir el estilo con la propiedad `border-style`. Una vez que se define el estilo, el navegador usa los valores por defecto para generar el borde. Si no queremos dejar que el navegador determine estos valores, podemos usar el resto de las propiedades para configurar todos los atributos del borde. El siguiente ejemplo asigna un borde sólido de 2 píxeles de ancho a la cabecera de nuestro documento.

```
header {  
  margin: 30px;  
  padding: 15px;  
  text-align: center;  
  border-style: solid;  
  border-width: 2px;  
}  
#titulo {  
  font: bold 26px Verdana, sans-serif;  
}
```

ejemplo 2-26: Agregando un borde a un elemento



Figura 2-7: Borde sólido

Posicionamiento en CSS

Y aquí llega el ansiado momento: vamos a aprender cómo cambiar la posición de los elementos en nuestra página. Necesitaremos la teoría que veremos aquí para el próximo módulo, donde diseñaremos nuestra primera página web paso a paso.

Veremos que hay varias técnicas para definir el diseño de página de un sitio web. Cada uno tiene sus pros y sus contras, así que dependerá de vos elegir el que creas más conveniente en función del caso.

Cajas

Como hemos mencionado en el capítulo anterior, los navegadores crean una caja virtual alrededor de cada elemento para determinar el área que ocupan. Para organizar estas cajas en la pantalla, los elementos se clasifican en dos tipos básicos: Block (bloque) e Inline (en línea).

La diferencia principal entre estos dos tipos es que los elementos Block tienen un tamaño personalizado y generan saltos de línea, mientras que los elementos Inline tienen un tamaño determinado por su contenido y no generan saltos de línea. Debido a sus características, los elementos Block se colocan de uno en uno en las distintas líneas, y los elementos Inline se colocan uno al lado del otro en la misma línea, a menos que no haya suficiente espacio horizontal disponible, como lo ilustra la **Figura 3-1**.

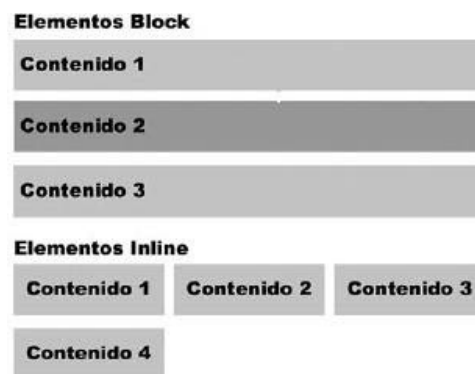


Figura 3-1: Elementos Block e Inline

Debido a sus características, los elementos Block son apropiados para crear columnas y secciones en una página web, mientras que los elementos Inline son adecuados para representar contenido. Esta es la razón por la que los elementos que definen la estructura de un documento, como `<section>`, `<nav>`, `<header>`, `<footer>`, o `<div>`, se declaran como elementos Block por defecto, y otros como ``, ``, o ``, que representan el contenido de esos elementos, se declaran como elementos Inline.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Display

El que un elemento sea del tipo Block o Inline lo determina el navegador, pero podemos cambiar esta condición desde CSS con la siguiente propiedad.

display—Esta propiedad define el tipo de caja usado para presentar el elemento en pantalla. Existen varios valores disponibles para esta propiedad. Los más utilizados son none (elimina el elemento), block (muestra el elemento en una nueva línea y con un tamaño personalizado), inline (muestra el elemento en la misma línea), e inlineblock (muestra el elemento en la misma línea y con un tamaño personalizado).

Los elementos estructurales se configuran por defecto con el valor block, mientras que los elementos que representan el contenido normalmente se configuran como inline. Si queremos modificar el tipo de elemento, solo tenemos que asignar la propiedad display con un nuevo valor.

Si queremos asegurarnos de que estos elementos se interpreten como elementos Block en todos los navegadores, podemos declarar la siguiente regla en nuestras hojas de estilo.

```
header, section, main, footer, aside, nav, article, figure, figcaption
{
  display: block;
}
```

ejemplo 3-1: Definiendo los elementos HTML5 como elementos Block

Posicionamiento flotante

Como ya mencionamos, los elementos Block se colocan unos sobre otros y los elementos Inline se posicionan de izquierda a derecha en la misma línea. El modelo de caja tradicional establece que los elementos pueden flotar a cada lado de la ventana y compartir espacio en la misma línea con otros elementos, sin importar su tipo. Por ejemplo, si tenemos dos elementos Block que representan columnas en el diseño, podemos posicionar una columna a la izquierda y la otra columna a la derecha haciendo que los elementos floten hacia el lado que queremos.

Las siguientes son las propiedades que ofrece CSS para este propósito.

float—Esta propiedad permite a un elemento flotar hacia un lado u otro, y ocupar el espacio disponible, incluso cuando tiene que compartir la línea con otro elemento. Los valores disponibles son none (el elemento no flota), left (el elemento flota hacia la izquierda) y right (el elemento flota hacia la derecha).

clear—Esta propiedad restaura el flujo normal del documento, y no permite que el elemento siga flotando hacia la izquierda, la derecha o ambos lados. Los valores disponibles son none, left, right, y both (ambos).

La propiedad float hace que el elemento flote a un lado u otro en el espacio disponible.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Cuando se aplica esta propiedad, los elementos no siguen el flujo normal del documento, se desplazan a la izquierda o a la derecha del espacio disponible, respondiendo al valor de la propiedad float y hasta que especifiquemos lo contrario con la propiedad clear.

En el siguiente ejercicio vamos a usar el código HTML estructurado que escribimos en el capítulo 1 y luego usando las reglas CSS, iremos cambiando las ubicaciones de cada una de las estructuras (cajas) que definimos en el código HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="estilo.css" />
    <title>Zozor: la página web</title>
  </head>
  <body>
    <header>
      <h1>Zozor</h1>
      <h2>Diarios de viaje</h2>
    </header>
    <nav>
      <ul>
        <li><a href="#">Inicio</a></li>
        <li><a href="#">Blog</a></li>
        <li><a href="#">Continuar</a></li>
      </ul>
    </nav>

    <section>
      <aside>
        <h1>Sobre el autor</h1>
        <p>¡Soy yo, Zozor! Nací el 23 de noviembre de 2005.</p>
      </aside>
      <article>
        <h1>Soy un gran viajero</h1>
        <p>Bla, bla, bla, bla (texto del artículo)</p>
      </article>
    </section>

    <footer>
      <p>Copyright Zozor - Todos los derechos reservados
      <a href="#">¡Escríbeme!</a></p>
    </footer>
  </body>
</html>
```

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Recuerda que sin CSS el diseño de la página quedará como en la siguiente ilustración.



Una página HTML sin CSS

Vamos a intentar situar el menú en la izquierda y el resto del texto en la derecha. Para ello, pondremos el menú en la izquierda con float y dejaremos que el resto del texto se coloque por sí solo a la derecha.

Queremos que el menú tenga 150 píxeles de ancho. También vamos a añadir un borde negro al menú y un borde azul alrededor del cuerpo (utilizando la etiqueta <section>) para resaltarlos:

```
nav {  
  
    float: left;  
    width: 150px;  
    border: 1px solid black;  
}
```

```

section
{
    border: 1px solid blue;
}

```

La siguiente ilustración muestra el resultado. Aún nos queda trabajo.



El menú está posicionado correctamente, pero está pegado al texto

Hay dos fallos (aparte de que aún no es agradable visualmente):

El cuerpo de la página toca el extremo del menú, así que necesita algo de margen.

Y lo que es aún más molesto: ¡el resto del texto continúa debajo del menú!

Aunque queremos situar el pie de página («Copyright Zozor») en la parte inferior, bajo el menú, nos gustaría que la página entera fuera un solo bloque a la derecha.

Para solucionar estos dos problemas de una, hemos de añadir un margen exterior a la derecha de la caja <section>, que además deberá ser mayor que el ancho del menú. Si nuestro menú tiene un ancho de 150px, daremos, por ejemplo, un margen exterior izquierdo de 170px (siguiente ilustración) a nuestra sección de página en la línea 10.

```
nav
{
  display: inline-block;
  width: 150px;
  border: 1px solid black;
}
```

```
section
{
  margin-left: 170px;
  border: 1px solid blue;
}
```



El cuerpo de la página está alineado correctamente a la derecha del menú

Y ahí lo tenés, el contenido de la página está ahora correctamente alineado.

De manera alternativa, también puede darse que quieras ubicar un elemento bajo el menú. En este caso, tendrías que utilizar `clear: both;` como ya hemos visto, para posicionar el resto del texto bajo el elemento que esté a su alrededor.

Ahora vamos a transformar los dos elementos que queremos situar uno al lado del otro usando `display` en elementos `inline-block`: el menú de navegación y la sección central de la página.

```

nav
{
  display: inline-block;
  width: 150px;
  border: 1px solid black;
}

section
{
  display: inline-block;
  border: 1px solid blue;
}

```

El resultado se muestra en la siguiente ilustración.



El menú y el cuerpo están uno al lado del otro, ¡aunque colocados en la parte inferior!

¡Ups!

No es lo que intentábamos. Es normal, en realidad: los elementos inline-block están colocados en la misma línea base, en la parte inferior.

Por suerte, tras haber convertido los elementos en inline-block, podemos usar una nueva propiedad, normalmente reservada para tablas: vertical-align.

Esta propiedad se usa para cambiar la alineación vertical de elementos. Aquí tenemos alguno de los posibles valores que esta propiedad acepta:

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



- baseline: alinea la base del elemento con la base del elemento padre (por defecto);
- top: lo alinea con la parte superior;
- middle: lo centra en vertical;
- bottom: lo alinea al pie;
- (valor en px o como %): lo alinea a una determinada distancia de la línea base.

A continuación, solo hemos de alinear en la parte superior nuestros elementos (líneas 6 y 13), ¡y ya lo tenés!

```
nav
{
  display: inline-block;
  width: 150px;
  border: 1px solid black;
  vertical-align: top;
}

section
{
  display: inline-block;
  border: 1px solid blue;
  vertical-align: top;
}
```

El menú y el cuerpo se alinean en la parte superior uno al lado del otro

Fijáte en que el cuerpo <section> no ocupa todo el ancho. ¡Claro, si no es un bloque! La sección solamente ocupa el espacio que necesita. Si no es lo que querés en tu diseño, cambia el tamaño de la sección con width.

¡Y no hay más! Ni hay que preocuparse de los márgenes, ni existe riesgo de que el texto sobresalga del menú... En resumen, ¡que es perfecto!

Posicionamiento absoluto, fijo y relativo

Existen algunas otras técnicas bastante especiales que se emplean para posicionar elementos con precisión en la página:



Posicionamiento absoluto: nos permite posicionar un elemento en cualquier parte de la página (superior izquierda, inferior derecha, centro, etc.).

Posicionamiento fijo: es lo mismo que el posicionamiento absoluto, pero aquí el elemento permanece visible incluso si te desplazas hacia la parte inferior de la página.

Es parecido, si hacemos memoria, al principio `background-attachment: fixed`;

Posicionamiento relativo: permite al elemento estar fuera de su posición normal.

Del mismo modo que los objetos flotantes, los posicionamientos absolutos, fijos y relativos también funcionan en etiquetas en línea. Sin embargo, verás que es más frecuente usarlos en etiquetas de bloque que en etiquetas en línea.

En primer lugar, tendrás que elegir una de las tres opciones de posicionamiento. Para ello, se utiliza la propiedad CSS `position` y se le da uno de estos valores:

- **absolute:** posicionamiento absoluto;
- **fixed:** posicionamiento fijo;
- **relative:** posicionamiento relativo.

Pasaremos a examinar cada una de estas opciones de posicionamiento.

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



Posicionamiento absoluto

El posicionamiento absoluto permite situar un elemento en cualquier lugar de la página. Para situar un elemento de manera absoluta has de introducir:

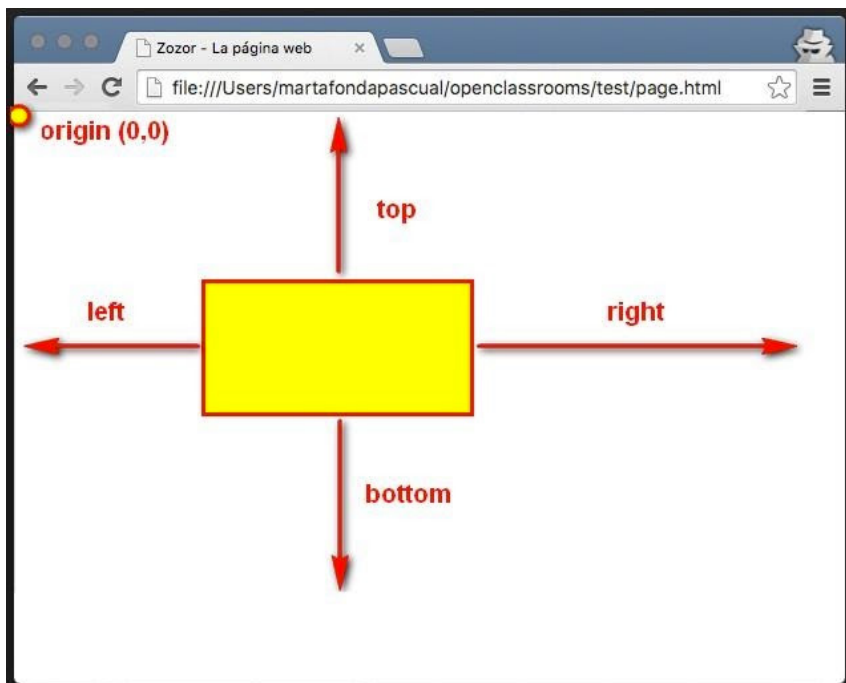
```
element
{
    position: absolute;
}
```

¡Pero no basta con esto! Aunque hemos especificado que el posicionamiento ha de ser absoluto, hemos de indicar en qué parte de la página queremos que se sitúe el bloque. Para ello, vamos a utilizar cuatro propiedades CSS:

- **left**: posición relativa en la izquierda de la página;
- **right**: posición relativa en la derecha de la página;
- **top**: posición relativa en la parte superior de la página;
- **bottom**: posición relativa en el pie de la página;

Podés darles un valor en píxeles, como 14px, o un porcentaje, como 50%.

Si todavía no ha quedado claro, la siguiente ilustración debería ayudarte a entenderlo.



Posicionamiento de elementos absolutos

Mediante esta propiedad deberías ser capaz de situar tu bloque correctamente.

Así pues, tendrás que usar la propiedad `position` correctamente y una de las cuatro propiedades anteriores como mínimo (`top`, `left`, `right` o `bottom`). Si por ejemplo introducís:

```
element
{
  position: absolute;
  right: 0px;
  bottom: 0px;
}
```

..significará que el bloque se situará a la derecha, en la parte inferior derecha de la página (0 píxeles tomando como referencia la derecha de la página, 0 tomando la parte inferior de la página).

Si intentamos poner nuestro bloque `<nav>` en la parte inferior derecha de la página, obtendremos el resultado mostrado en la siguiente ilustración.



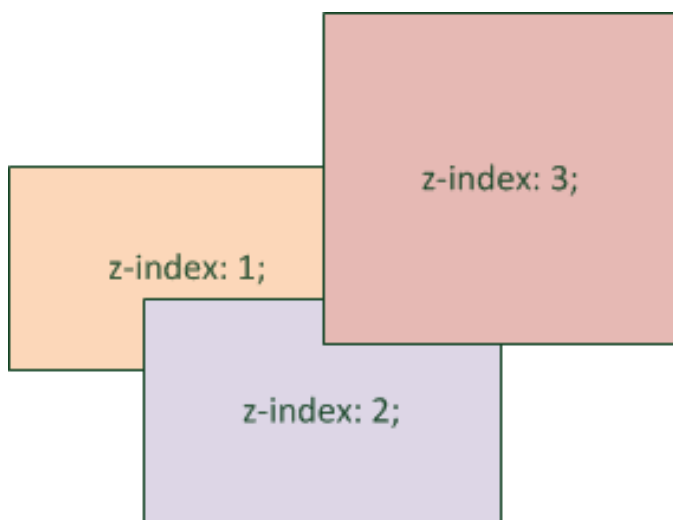
El menú se sitúa en la esquina inferior derecha de la pantalla

Puedes, por supuesto, añadir un margen interior (padding) al menú para separarlo ligeramente del borde.

Los elementos posicionados de manera absoluta están sobre el resto de elementos de la página. Además, si situas dos elementos de manera absoluta en el mismo lugar es posible que se solapen. En estos casos, utiliza la propiedad `z-index` para especificar qué elemento debería aparecer encima de los otros elementos.

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
    z-index: 1;
}
element 2
{
    position: absolute;
    right: 30px;
    bottom: 30px;
    z-index: 2;
}
```

El elemento con el valor `z-index` más alto se situará por encima de los otros, tal y como se muestra en la siguiente ilustración.

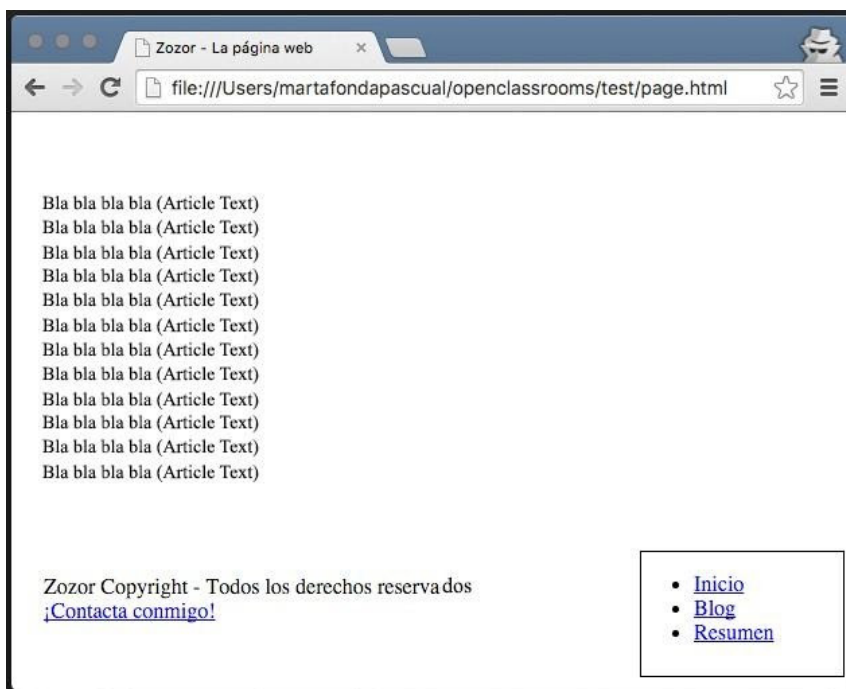


Posicionamiento Fijo

El principio es exactamente el mismo que el del posicionamiento absoluto, salvo que esta vez el bloque queda fijo en su posición incluso si te desplazas hacia el pie de la página

```
element
{
  position: fixed;
  right: 0px;
  bottom: 0px;
}
```

Inténtalo y comprueba los resultados. Verás que en este caso el menú sigue mostrándose en la parte inferior derecha incluso si desplazas la página hacia abajo (ilustración siguiente).



Posicionamiento relativo


El posicionamiento relativo es algo más complicado y puede dar con facilidad algunas complicaciones. Este tipo de posicionamiento te permite realizar «ajustes»: el elemento está fuera de su posición normal.

Tomemos como ejemplo un texto importante situado entre dos etiquetas . Para empezar, lo pondré sobre un fondo rojo para poder identificarlo mejor.

```
strong
{
  background-color: red;
  color: yellow;
}
```

Esta vez, el procedimiento que he mostrado antes de posicionamiento absoluto y fijo no sirve. ¿Por qué? Porque el origen ha cambiado: el punto de coordenadas (0, 0) ya no se encuentra en la parte superior izquierda de la venta como en el caso anterior. No, esta vez el origen está situado en la parte superior izquierda... de la posición actual de tu elemento.

¿A que es un poco enrevesado? Es el principio de la posición relativa. El diagrama de la siguiente ilustración debería ayudarte a entender dónde se sitúa el origen de los puntos.

No hay duda  **este texto es importante** si quieres entender

Posicionamiento Relativo

Así pues, si introducís `position: relative;` y aplicás cualquiera de las propiedades `top`, `left`, `right` o `bottom`, el texto sobre fondo rojo se moverá tomando como referencia su posición actual.

Utilicemos un ejemplo: quiero desplazar mi texto 55 píxeles a la derecha y 10 píxeles hacia abajo; así, lo que le voy a indicar es que quiero que se sitúe a 55 píxeles del borde izquierdo y a 10 píxeles del borde superior (líneas 6 a 8):

```
strong
{
  background-color: red;
  color: yellow;
  position: relative;
  left: 55px;
  top: 10px;
}
```

El texto estará fuera de su posición inicial, tal como se indica en la siguiente ilustración.

No hay duda  **este texto es importante** si quieres entender

Este texto está fuera de su posición inicial

Contenido desarrollado por Arnoldo Néstor Mauro, Coordinador del Punto Digital Ituzaingó, Corrientes.



En resumen

CSS se utiliza para el diseño de página de un sitio web. Tenemos a nuestra disposición diferentes técnicas.

El posicionamiento **flotante** (utilizando la propiedad *float*) es una de las técnicas más utilizadas actualmente. Te permite, por ejemplo, situar un menú a la izquierda o a la derecha de la página. Esta propiedad, sin embargo, no se diseñó en un principio con este fin, y, de sernos posible, será mejor evitar esta técnica.

El posicionamiento **inline-block** consiste en asignar el tipo *inline-block* a nuestros elementos mediante la propiedad *display*. Se comportarán como elementos en línea (posicionamiento izquierdo a derecha), pero podrán ser redimensionados como bloques (mediante *width* y *height*).

- El posicionamiento **absoluto** te permite situar un elemento en cualquier lugar de la página, indicando la distancia en píxeles.
- El posicionamiento **fijo** es parecido al posicionamiento absoluto, pero el elemento permanece visible incluso si te desplazas hacia el pie de la página.
- El posicionamiento **relativo** permite que un bloque esté fuera de su posición normal.

Esta técnica es preferible a la de posicionamiento flotante.

Un elemento A, posicionado de manera absoluta dentro de otro elemento B (a su vez posicionado de modo absoluto, fijo o relativo), se posicionará tomando como referencia este elemento B y no la esquina superior izquierda de la página.



Jefatura de
Gabinete de Ministros
Argentina

Secretaría de
Innovación Pública