

CSS | Estructura y Posicionamiento

Después de esta lección podrás:

1. Posicionar los elementos de tu web en columnas o filas.
2. Dar distintas distancia de espaciado entre elementos HTML.

Estilos aplicados a estructura

La estructura o **layout** de nuestra web nos ayudará a posicionar elementos donde creamos que sean necesarios, de forma que consigamos que la barra de navegación esté siempre arriba y el footer esté siempre abajo.

Atributos Width & Height

Algunos elementos del HTML, como los divs, no serán visibles a pesar de tener color de fondo si no tienen contenido de bloque (textos entre otros). Para hacerlos visibles cuando están vacíos (imagina crear una caja roja en la pantalla) tenemos que darle un ancho y un alto. Eso lo conseguiremos con las propiedades width y height.

Vamos a crear una caja de 300px de ancho y 100px de alto. Para ello, añadiremos un **div** sin contenido con las clases **red-background** y **box** a nuestro HTML:

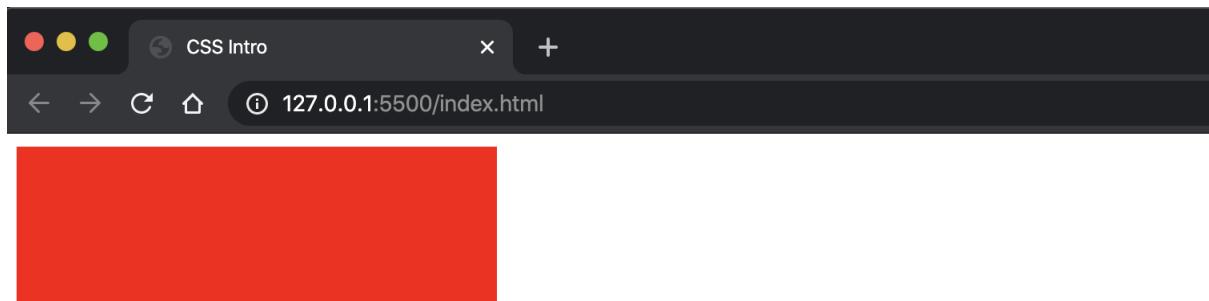
```
<body>
  <div class="box red-background"></div>
</body>
```

Aunque carguemos este HTML en el navegador, no veremos nada hasta que tenga ancho y alto, así que vamos a añadirlos a continuación:

```
.box {
  width: 300px;
  height: 100px;
}

.red-background {
  background-color: red;
}
```

Si lo hemos añadido correctamente e **importado el archivo CSS en nuestro HTML mediante la etiqueta <link>**, veremos esto en el navegador:



Atributos **display: block**, **inline** e **inline-block**

Todos los elementos de nuestra web van a tener un tipo de **display** o forma de mostrarse distinta por defecto. En el caso de los **div** tendremos elementos de tipo **display: block**, y para los **span** tendremos elementos del tipo **display: inline**.

¿Qué significa esto?

Los elementos de tipo **block** ocuparán por defecto el espacio de ancho de pantalla que tengan disponible, no permitiendo a otros elementos estar a su lado.

Los elementos **inline** se pegarán a sus elementos adyacentes intentando ocupar el menor espacio posible.

¡Vamos a verlo con un ejemplo!

En el ejemplo anterior, hicimos un **div** de color rojo, ahora vamos a crear dos **div** más y a hacer uno azul y otro verde:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="styles.css" />
  <title>CSS Intro</title>
</head>
<body>
  <div class="box red-background"></div>
  <div class="box blue-background"></div>
  <div class="box green-background"></div>
</body>
</html>
```

```
.box {
  width: 300px;
  height: 100px;
}
```

```

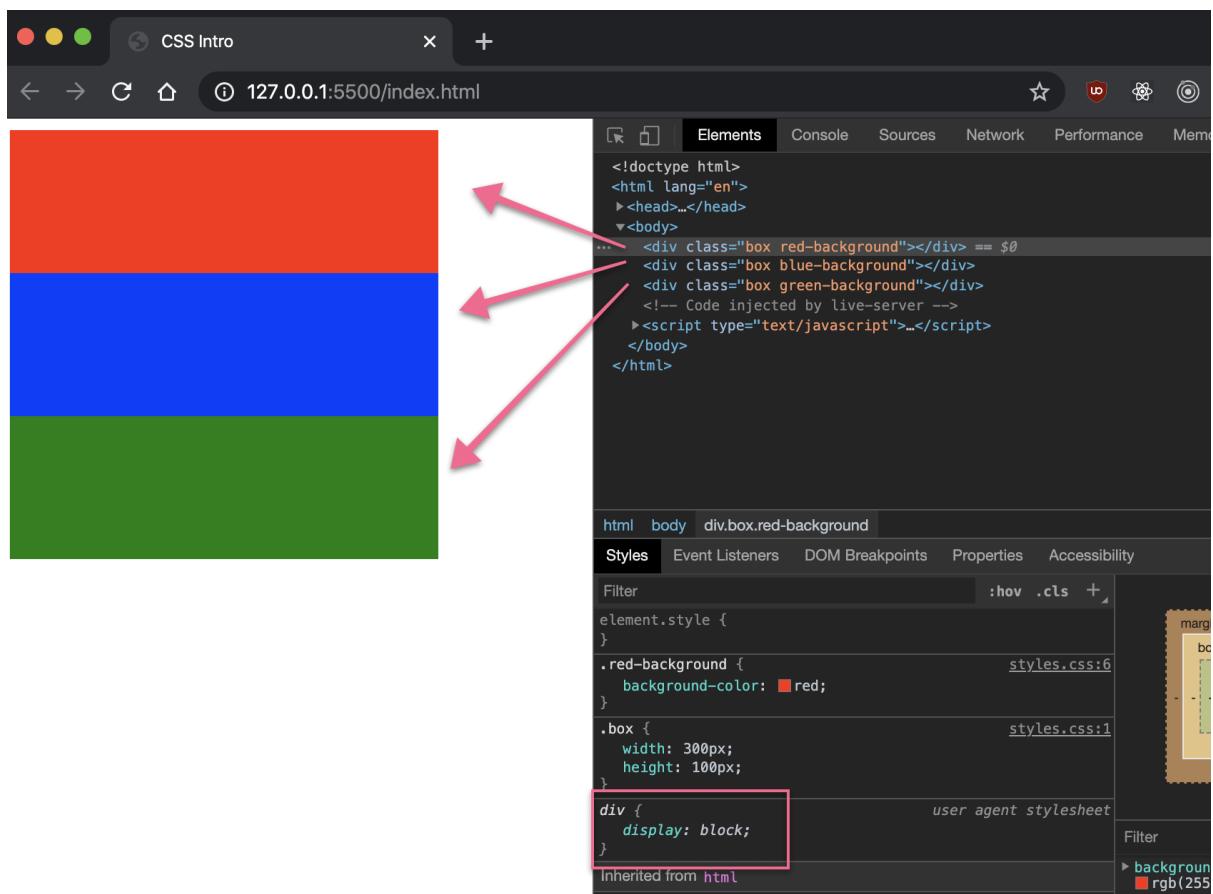
.red-background {
  background-color: red;
}

.blue-background {
  background-color: blue;
}

.green-background {
  background-color: green;
}

```

Si guardamos este código y lo abrimos en nuestro navegador, veremos lo siguiente:



Como podrás observar, cada **div** está representado por una caja de 300px por 100px debido a la clase **box**, y hemos creado una clase con un color de fondo distinto para cada uno.

Lo curioso aquí es que están uno sobre el otro independiente del ancho de pantalla que utilicemos. Esto es debido a que tienen por defecto el atributo **display: block**; y por tanto nunca podrán estar pegados.

Para corregir esto, vamos a darle a la clase **box** el atributo **display: inline** y veremos como queda:

```

.box {
  width: 300px;
  height: 100px;
}

```

```
    display: inline;  
}
```

Pero, si abrimos ahora el navegador nos encontraremos con una **pantalla en blanco** 🤔

¿A qué se debe esto? Un elemento **inline** no puede tener ni alto ni ancho definidos, y debe estar definido por su espaciado o contenido, lo cual puede generar problemas en situaciones como esta.

¿Cómo lo solucionamos? Vamos a darle la propiedad **inline-block** a nuestra clase **box**, que hará que los elementos sigan comportándose como un bloque pero puedan compartir la espacio con otros elementos.

```
.box {  
    width: 300px;  
    height: 100px;  
    display: inline-block;  
}
```

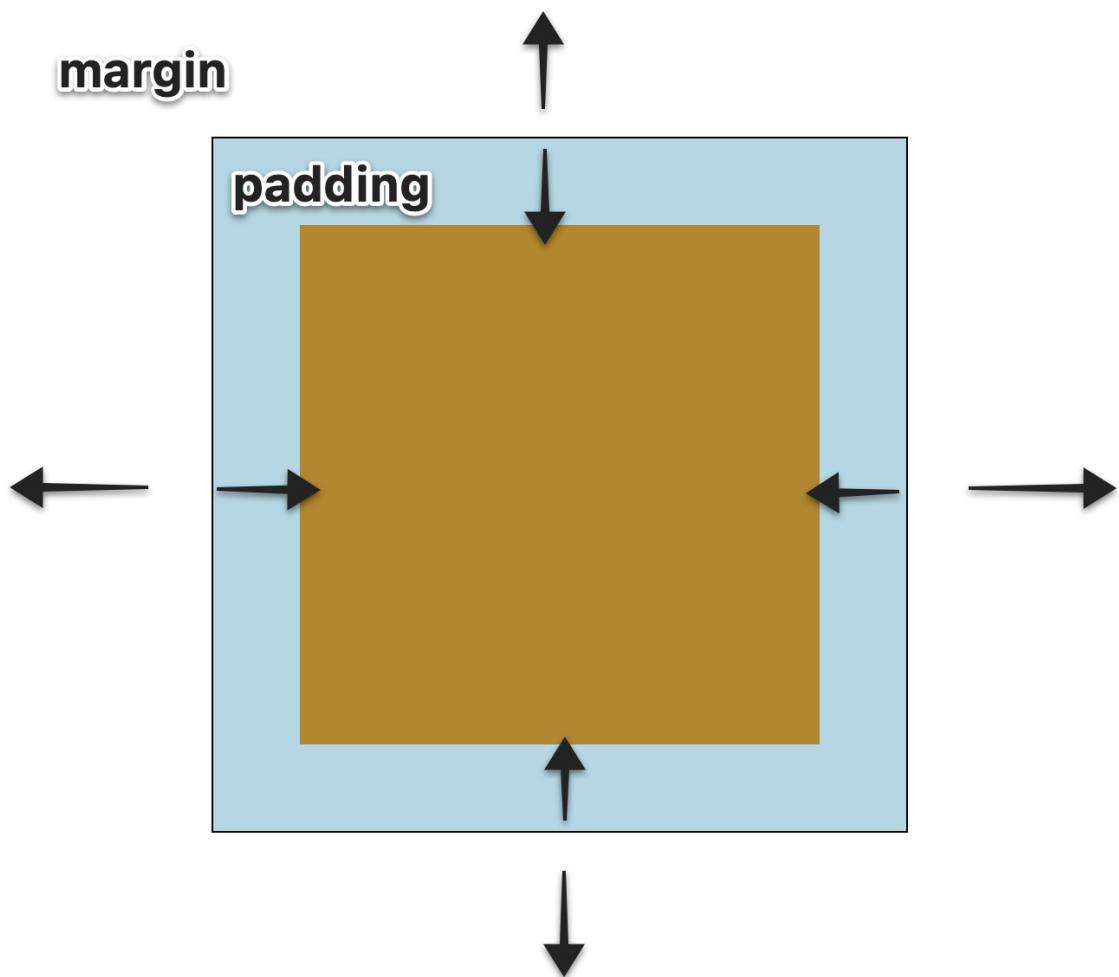
Y conseguiremos lo siguiente (observa como se adaptan al ancho de pantalla disponible):



[Aquí](#) puedes ver la lista de los elementos tipo block e inline organizados y diferenciados.

Propiedades de espaciado: padding y margin

Si consideramos cada elemento de la web como una caja, ¿cómo veríamos un <div> dentro de otro <div>? Vamos a verlo:



Suponemos un `<div>` de **background-color: lightblue** que contiene a otro `<div>` de menor tamaño y con **background-color: darkgoldenrod**. Hemos centrado al div interior, pero no usando ninguno de los método anteriores, es la propiedad **padding** la que decide como se ve el contenido de dentro. Y si tuviésemos elementos cercanos al div superior, estarían empujados por la propiedad **margin**, razón por la cual está centrado en nuestro `<body>`.

¿Qué es y para qué sirve la propiedad margin de un elemento HTML?

Si tuviésemos que definir la propiedad **margin**, diríamos que es el espacio que es empujado por un elemento desde su borde hacia afuera, empujando otros elementos circundantes. Es decir, dado un `<div>` con margin, los `<div>` que lo rodeen estarían separados de este primero tanto como margin hubiese.

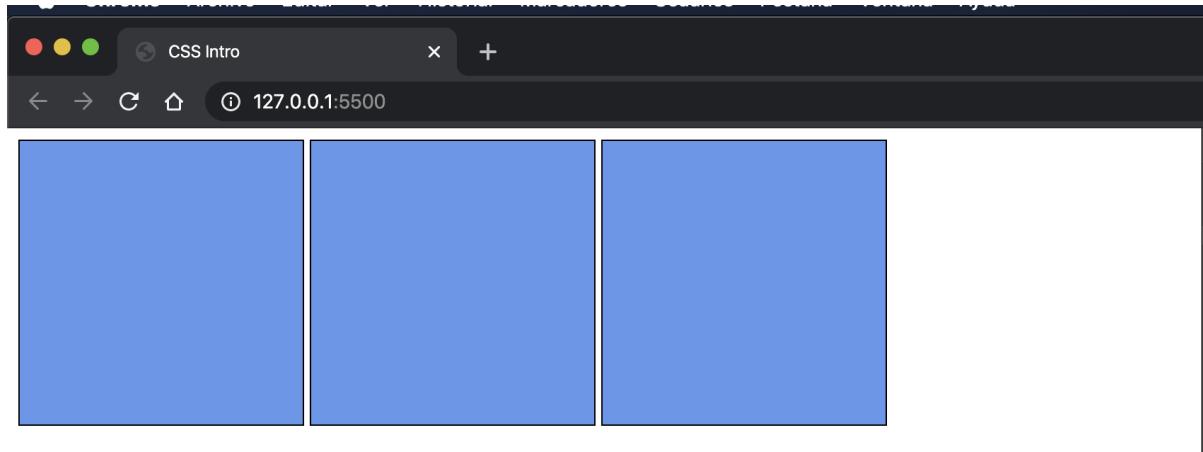
Vamos a crear tres `<div>` y a verlo con un ejemplo:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>CSS Intro</title>
  </head>
  <body>
    <div class="box"></div>
    <div class="box has-margin"></div>
```

```
<div class="box"></div>
</body>
</html>
```

```
.box {
  width: 200px;
  height: 200px;
  background-color: cornflowerblue;
  border: 1px solid black;
  display: inline-block;
}
```

Veremos esto en nuestra pantalla:



¿Y si queremos que el <div> de en medio empuje al resto? ¡Vamos a añadirle la propiedad **margin** y a cambiarle el color para que destaque más!

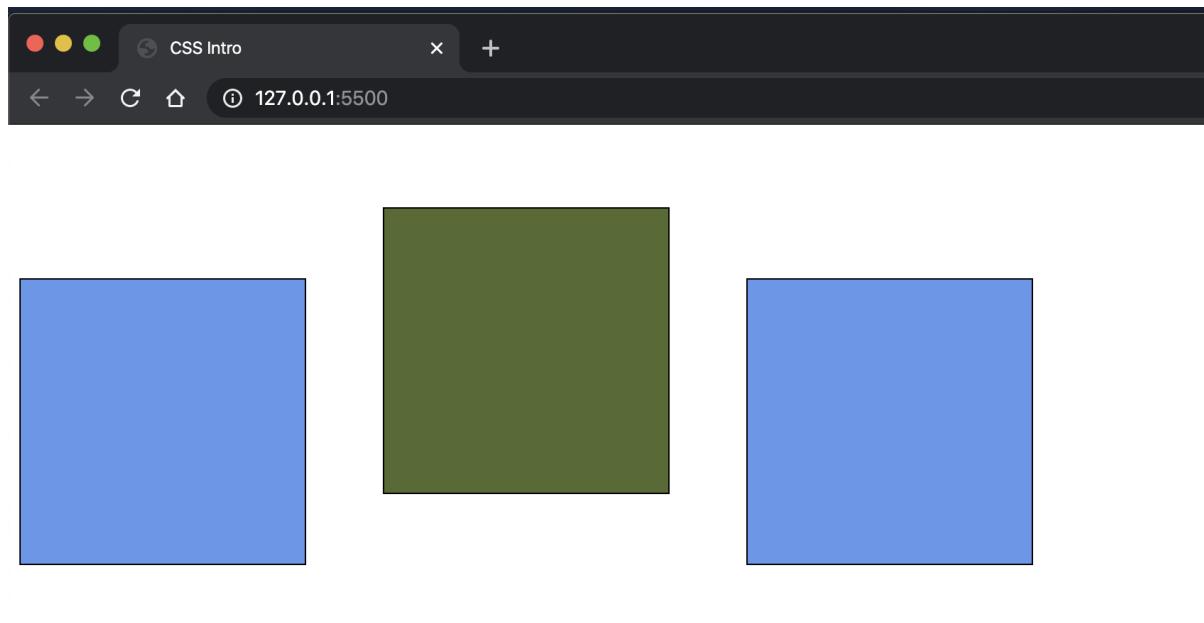
- Recuerda que margin también puede ser definido por lados, con margin-top, margin-right, margin-bottom y margin-left, todos definidos con píxeles.

```
<!-- Añadimos la clase has-margin al div central -->
<body>
  <div class="box"></div>
  <div class="box has-margin"></div>
  <div class="box"></div>
</body>
```

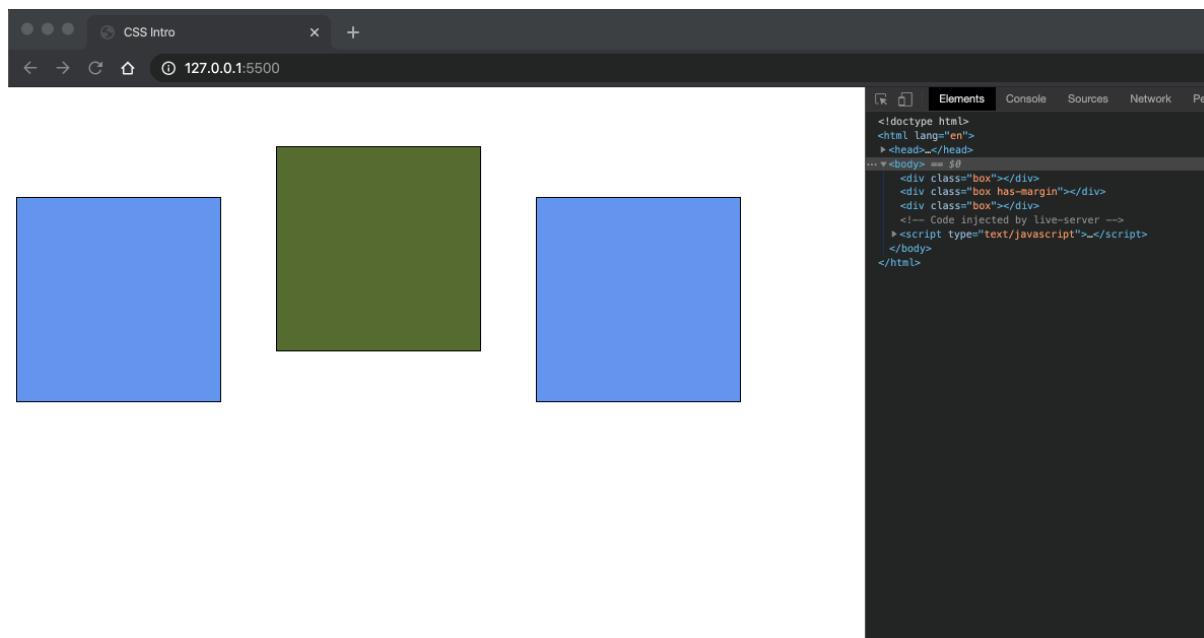
Añadimos la nueva clase que acabamos de definir en el HTML:

```
.has-margin {
  background-color: darkolivegreen;
  margin: 50px;
}
```

Obtendremos esto cuando recarguemos el navegador:



¡Todo se ha movido! 🎉 Esto se debe a que ahora el <div> central empuja con su margin en todas las direcciones 50px, y por tanto ha modificado la posición de todo a su alrededor. Vamos a verlo con un GIF y usaremos el inspector de nuestro navegador:



Si queremos que solo haya "empuje" con el margin en los laterales, vamos a cambiar el CSS para que afecte solamente a **margin-right** y **margin-left**. Esto podemos hacerlo de dos formas, ya que la propiedad margin puede definirse con varios valores:

```
margin: 100px; -> Empuja en todas las direcciones 100 pixeles  
margin: 100px 50px; -> Empuja 100px verticalmente y 50px horizontalmente  
margin: 20px 30px 40px 50px; -> Empuja 20px arriba, 30px derecha, 40px abajo y 50px izquierda, como si fuese un reloj empezando desde
```

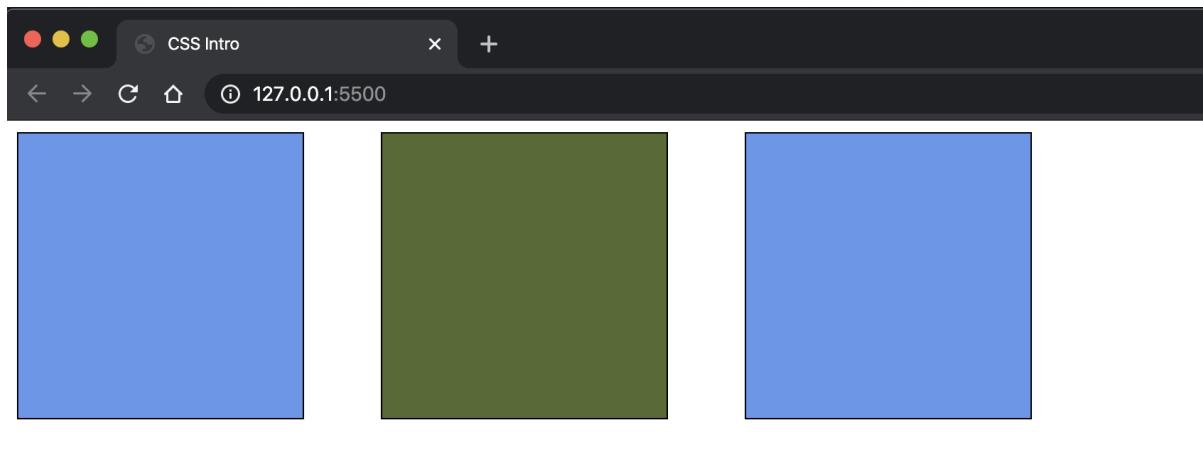
O también usando los márgenes de forma más específica:

```
margin-top: 10px; -> Empuja 10px hacia arriba usando el margin  
margin-right: 20px; -> Empuja 20px hacia la derecha usando el margin  
margin-bottom: 30px; -> Empuja 30px hacia abajo usando el margin  
margin-left: 40px; -> Empuja 40px hacia la izquierda usando el margin
```

Para este caso, usaremos la propiedad margin con dos valores, ¡prueba el que más te guste!

```
.has-margin {  
    background-color: darkolivegreen;  
    margin: 0 50px;  
}
```

Y ahora endremos esto en el navegador:



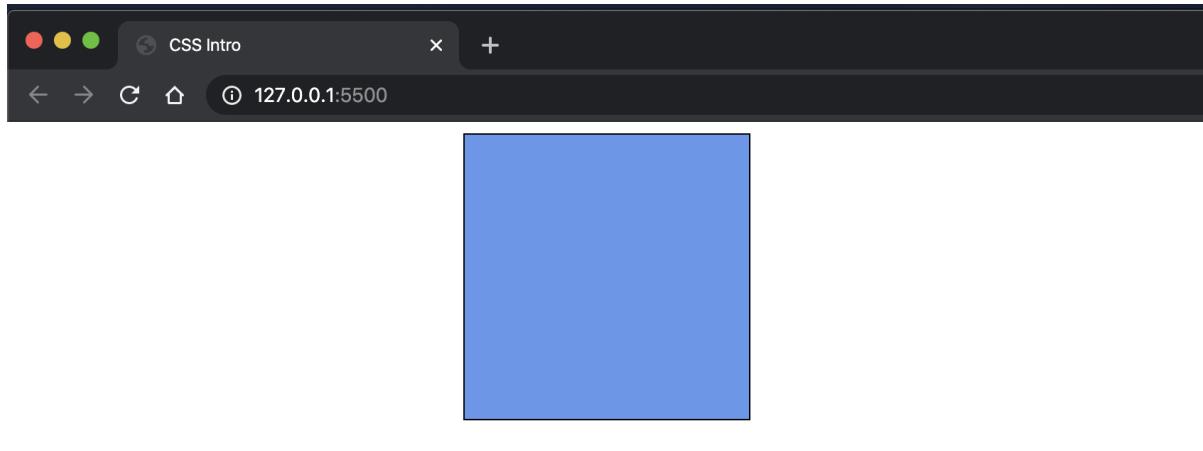
Ahora están empujados lateralmente, ¡tal y como queríamos! Ahora que has visto margin, sabes hacer que los elementos del HTML estén espaciados desde su borde hacia el exterior.

¿Sabías que... podemos centrar un elemento dentro de su contenedor usando margin? Usaremos la propiedad margin con los valores **0 auto**; para conseguirlo, ¡vamos a centrar un nuevo div!

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <link rel="stylesheet" href="styles.css" />
```

```
<title>CSS Intro</title>
</head>
<body>
  <div class="box-centered"></div>
</body>
</html>
```

```
.box-centered {
  width: 200px;
  height: 200px;
  background-color: cornflowerblue;
  border: 1px solid black;
  margin: 0 auto; /* Propiedad margin con los valores 0 vertical y auto horizontal */
}
```



¡Y ya lo tenemos centrado! Esto lo usaremos a menudo, así que pruébalo tú mism@ si puedes 🙌

¿Qué es y para qué sirve la propiedad padding de un elemento HTML?

Acabamos de ver como empujar hacia el exterior desde el borde de un elemento, pero, ¿y si lo hacemos desde el borde hacia el interior? Para eso utilizamos la propiedad **padding**, y se utiliza prácticamente igual que el margin.

Imagina que tenemos una web de artículos, y queremos que el texto tenga una sangría a la izquierda de 50px. Para conseguirlo, podemos hacer que el elemento que contenga el texto tenga un padding con ese valor, ¡vamos a ello!

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="styles.css" />
    <title>CSS Intro</title>
  </head>
  <body>
    <article class="text-container">
      <p>Este es el texto que usaremos para recordar que Iron Man nos salvó a todos cuando usó las gemas del infinito en la batalla co
    </article>
  </body>
</html>
```

Añadimos la propiedad **padding-left** al contenedor:

```
.text-container {  
    padding-left: 50px;  
    border: 1px solid black;  
}
```

Y obtenemos lo siguiente (veremos en el GIF como usamos el inspector de HTML para ver el padding):



Puedes ver como el **<article>** que contiene el texto **<p>** tiene una sangría izquierda de 50px, ¡ya sabemos usar el padding!



- Prueba a crear varios elementos **<div>** y añadir contenido a su alrededor y dentro de estos, y añade todas las propiedades que hemos ido viendo para posicionar el contenido. ¡No te olvides de repasar todo antes de comenzar con los ejercicios de CSS!