

Clase 3

¿Qué es CSS?

Vincular CSS en HTML

Selectores: etiqueta, clase, ID

Jerarquías - propiedades generales

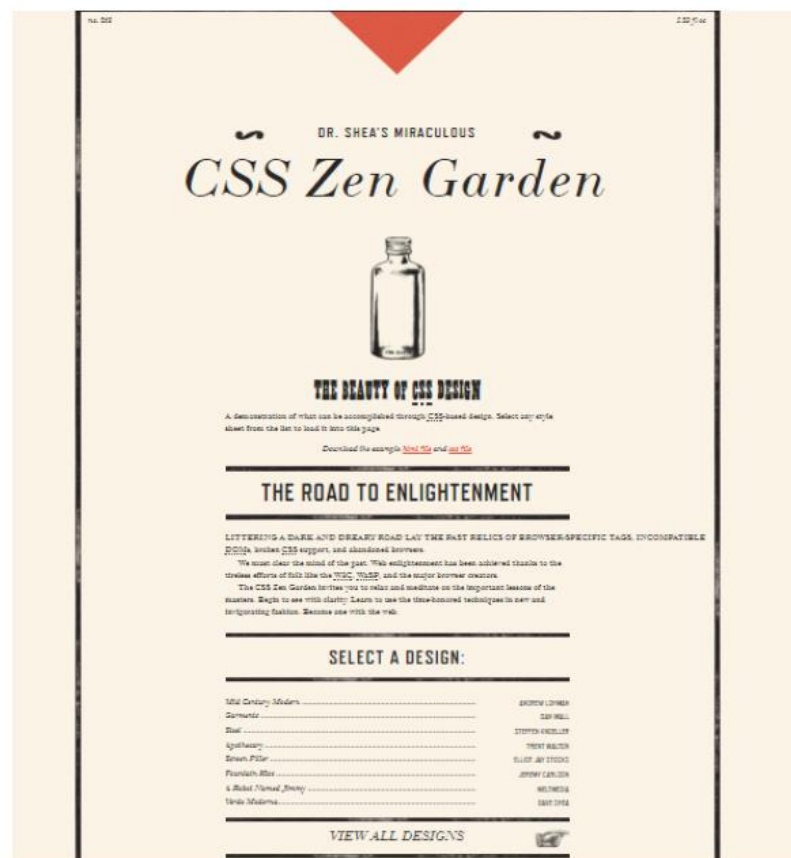
Font, centrados, background

Color, box shadow, degradados



Bordes

¿Qué es CSS?

CSS (Cascading Style Sheets) es un lenguaje web para aplicar formato visual al HTML. Con CSS se puede cambiar por completo el aspecto de cualquier etiqueta HTML, sin modificarla.



Sintaxis

```
/* Ejemplo de estilos por etiquetas */  
body {  
|   background-color:  lightblue;  
|   border: 5px dotted  black;  
}  
  
/* Ejemplo de estilos por clases (recordar llamarlo en el archivo html) */  
.textoCentrado {  
|   text-align: center;  
}
```

Padres e Hijos

Cuando hay una etiqueta “dentro” de otra, aplicamos el concepto de padres e hijos. En este caso, `<section>` es padre de `<article>` y `<article>` es padre de `<h1>` y `<p>`.

```
<body>
  <section>
    <article>
      <h1>Primer título</h1>
      <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. </p>
    </article>
  </section>
</body>
```

Esto nos habilita a agregar atributos específicos a etiquetas “hijas” sin alterar los del “padre”.

Selector HIJO
↓
Selector PADRE → `section article {`
 `background-color: #cccccc;`
 `width: 500px;`
 `height: 500px;`
}

Padres e Hijos

Cuando queremos seleccionar una etiqueta, debemos incluir las etiquetas padre para ser más específicos al aplicar estilos.



```
section {  
  padding: 50px 30px 20px 60px;  
  margin-left: 40px;  
}  
  
section article {  
  background-color: #cccccc;  
  width: 500px;  
  height: 500px;  
}  
  
section article p {  
  line-height: 4;  
}
```

```
article {  
  background-color: aqua;  
}
```



Insertar CSS en HTML

- ❑ **Forma EXTERNA:** dentro de <head> llamamos al archivo CSS (recordar rutas relativas y absolutas).

```
<link rel="stylesheet" href="./pruebas.css"/>
```

- ❑ **Forma INTERNA 1:** recomendable utilizarla dentro de <head> (se puede usar en <body> pero es más desprolijo).

```
<style>  
|    /*Código CSS*/  
</style>
```

- ❑ **Forma INTERNA 2:** no es recomendable ya que es difícil de mantener. Se utiliza para parches o pruebas.

```
<section style="background-color: ■ aqua;">
```

Clases

Se utilizan para darles estilos a partes específicas del código.

El nombre de las clases es libre, pero debería relacionado con su contenido.

Además hay que tener en cuenta que deben comenzar con un “.”

```
.textoCentrado{  
|   text-align: center;  
}
```

Una vez creada, en el archivo HTML se debe usar el atributo “**class**” para llamarla, y en el valor se debe colocar su nombre. Todos los elementos HTML admiten este atributo.

```
<h1 class="textoCentrado">Primer título</h1>
```

Si queremos aplicar más de una clase, quedaría así:

```
<h1 class="textoCentrado textoRojo">Primer título</h1>
```

Atributo ID

- ✓ Se usa para nombrar porciones de código y sectores.
- ✓ Es posible ponerle un ID a cualquier elemento HTML para darle un “nombre”.
- ✓ Al querer aplicar un estilo a distintos lugares de la página, y no poder hacerlo con las etiquetas, podemos utilizar este ID.
- ✓ Los nombres siguen las mismas reglas que los de las clases, pero en el CSS comienzan con un “#”.

```
#nombreid {  
|   text-align: right;  
}
```

En el archivo HTML utilizamos el atributo “id” y en su valor irá el nombre.

```
<section id="nombreid">
```


Class vs ID

	¿Se puede reutilizar su nombre en HTML?	¿Se puede usar varias veces en un atributo en el HTML?	¿Cuándo lo uso?
ID	NO	NO	Nombrar secciones, divisiones de código
CLASS	SI	SI	Especificar diseño aparte del código
Ejemplo ID	id="productos" id="productos2"		<section id="productos">
Ejemplo CLASS	class ="bordes" class ="bordes"	class ="bordes destacado"	<p class ="destacado">

Selección de HTML mediante CSS

Por etiquetas

Por clase (“.”)

Por ID (“#”)

```
section article p {  
  |   line-height: 4;  
  |  
  }  
  
.textoCentrado{  
  |   text-align: center;  
  |  
  }  
  
#nombreid {  
  |   text-align: right;  
  |  
  }
```

Nombres de Clases e ID

No se pueden crear nombres separados por espacios, por lo que se recomienda utilizar “camelCase”.

camelCase nos permite leer de forma más simple palabras compuestas.

```
.textoCentrado{
```

Herencia y Cascada

- ❑ Se heredan de un elemento padre algunos estilos, y el elemento hijo podría tener otros estilos aplicados, por su parte, por lo que tendremos que saber cuál será el estilo dominante en ese caso.

```
div {  
  color: ■ red;  
}
```

```
<div>  
  <p>  
    Este párrafo quedará rojo por herencia.  
  </p>  
</div>
```

- ❑ Para saber cuáles son los estilos dominantes utilizaremos el concepto de cascada. El navegador leerá de arriba hacia abajo, de ahí viene el nombre de cascada.

```
div {  
  color: ■ red;  
}  
  
div {  
  color: ■ green;  
}
```

[Más info](#)

Precedencia de Declaraciones

Cuando hay reglas distintas que apuntan hacia el mismo objeto:

- Si son propiedades distintas, se suman (se combinan).
- Si tienen alguna propiedad repetida, se reemplazan.

Con respecto a la precedencia para reemplazarse:

- ✓ **ID** pisa cualquier otra regla.
- ✓ **Class** sobrescribe las reglas de etiqueta, pero no las de ID.
- ✓ Las **etiquetas** tienen la menor precedencia.
- ✓ Los estilos inline (atributo style) sobrescribirán cualquier estilo externo de CSS. Si bien tienen mayor especificidad, no es recomendable utilizarlos.

!IMPORTANT

Si tenemos 3 reglas CSS, es poco probable que se pisen, pero si tenemos un CSS extenso puede pasar. Para estos casos se utiliza la declaración “**!important**”, que corta la precedencia. Se escribe después del valor de la propiedad CSS “más importante”. Se utiliza uno por cada valor.

OJO!! No deberías necesitar más de 5 “!important”.

Propiedades: Color

Desde Google, podemos buscar “[color picker](#)”.

Desde VSC, podemos posicionar el mouse en el cuadradito de color y elegir.

Tenemos varios tipos de valores, aunque nos centraremos en los siguientes 3:

- Por nombre del color (Ej. red)
- Hexadecimal (Ej. #FFFFFF)
- RGB (Ej. 50, 212, 227). Si agregamos un valor más, determinaremos la opacidad. Cada color admitirá hasta 256.



Propiedades: Reseteo CSS

Los **reset** contienen en su código fuente definiciones para propiedades problemáticas, que los diseñadores necesitan unificar desde un principio.

Por ejemplo, para subsanar la diferencia entre los márgenes por defecto del sitio y la ventana, los diseñadores de sitios webs suelen declarar lo siguiente al comienzo de sus hojas de estilo:

“*”: todos los elementos contenidos en el HTML.

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

De esa manera, el diseñador o la diseñadora se verán obligados a declarar luego los márgenes necesarios en el diseño de su página web, en cada uno de los lugares donde se requiera, sin tener que dejar ese aspecto a decisión de ningún navegador, y minimizando las diferencias visuales entre los mismos.

Propiedades: listas y texto

- ❑ Aplicando la propiedad `list-style-type` eliminamos las viñetas de las listas.

```
ol, ul {  
  list-style-type: none;  
}
```

- ❑ Con `font-style` decidiremos entre normal e itálica.

```
.fontStyle{  
  font-style: normal;  
  font-style: italic;  
}
```

- ❑ Con `font-weight` decidiremos entre normal e itálica.

```
.fontWeight{  
  font-weight: bold;  
  font-weight: normal;  
}
```

Propiedades: Fuentes

- ❑ Aplicando la propiedad **font-size** definimos el tamaño de la letra.

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

```
.textoRelativo {  
  font-size: 200%;  
}
```

- ❑ Con **font-family** definiremos la familia de fuentes de la página.

```
'Courier New', Courier, monospace  
'Franklin Gothic Medium', 'Arial Narrow', Arial, ...  
'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet ...  
'Lucida Sans', 'Lucida Sans Regular', 'Lucida Gra...  
'Segoe UI', Tahoma, Geneva, Verdana, sans-serif  
'Times New Roman', Times, serif  
'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Gr...  
Arial, Helvetica, sans-serif  
Cambria, Cochin, Georgia, Times, 'Times New Roman...  
Georgia, 'Times New Roman', Times, serif  
Impact, Haettenschweiler, 'Arial Narrow Bold', sa...  
Verdana, Geneva, Tahoma, sans-serif
```

[Link](#) para ver cómo funciona cada fuente en distintos sistemas operativos.

Tipografías



Las tipografías Serif llevan detalles adicionales en los bordes de las letras. Estos detalles transmiten clasicismo, formalidad, precisión, tradición, delicadeza y/o refinamiento.

Las tipografías Sans Serif no tienen estos detalles. Se utilizan en entornos digitales porque estos detalles son difíciles de plasmar en píxeles. Transmiten fuerza, modernidad, elegancia y actualidad.

También podemos utilizar las fuentes alojadas en “Google Fonts”.

```
<head>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
  <title>Document</title>
  <link rel="stylesheet" href="pruebas.css"/>
</head>
```

Propiedades: Texto

- ❑ Aplicando la propiedad **text-align** definimos hacia dónde se alinea el texto. Se puede elegir entre “center, right, left o justify”.

```
.textoCentrado{  
|   text-align: center;  
}
```

- ❑ Con **line-height** definimos el interlineado. Podemos elegir entre “none, números, longitud o porcentaje”.

```
.interlineado {  
|   line-height: 1.6;  
}
```

- ❑ Con **text-decoration** agregamos características al texto. Podemos elegir entre...

```
❏ dashed  
❏ dotted  
❏ double  
❏ line-through
```

```
❏ none  
❏ overline  
❏ solid  
❏ underline  
❏ wavy
```

Propiedades: Fondos

- ❑ **background-color**: color de fondo.
- ❑ **background-image**: agregar una imagen de fondo.
- ❑ **background-repeat**: repetición de las imágenes de fondo.
- ❑ **background-position**: posiciono las imágenes de fondo.
- ❑ **background-size**: tamaño del fondo.

```
.ejemplo {  
    background-image: url(./images/foto.webp);  
    background-repeat: no-repeat;  
    background-size: cover;  
}
```

Unidades de Medidas

Tenemos unidades absolutas y relativas.

Absolutas:

❑ **Px (pixels):** es la unidad que usan las pantallas.

Relativas:

❑ **Rem:** relativa a la configuración de tamaño de la raíz (etiqueta HTML).

❑ **Porcentaje:** tomando en cuenta que 16px es 100%.

❑ **Viewport:** se utilizan para layouts responsivos.

Las más convenientes para los textos son las unidades relativas, ya que al cambiar el tamaño de la ventana, cambiarán los tamaños de las letras.

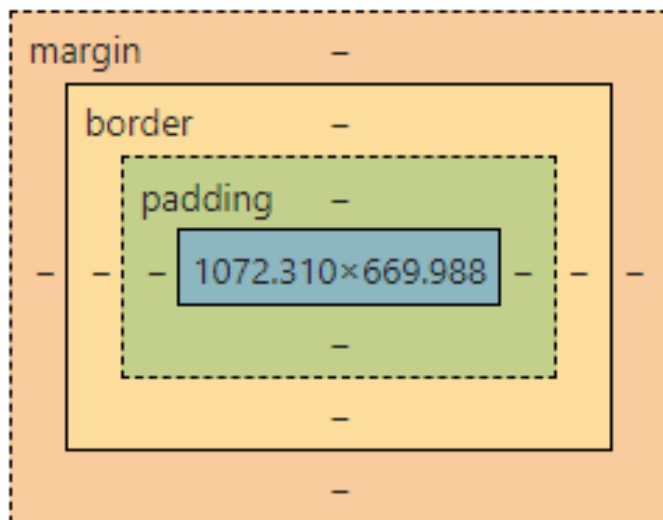
Box Modeling

Todos los elementos del HTML son cajas rectangulares.

Recordemos que tenemos elementos...

❑ De línea: se verán uno al lado del otro solo ocupan el espacio necesario para mostrar sus contenidos.

❑ De bloque: se verán uno debajo del otro y ocupan todo el espacio disponible hasta el final de la línea.



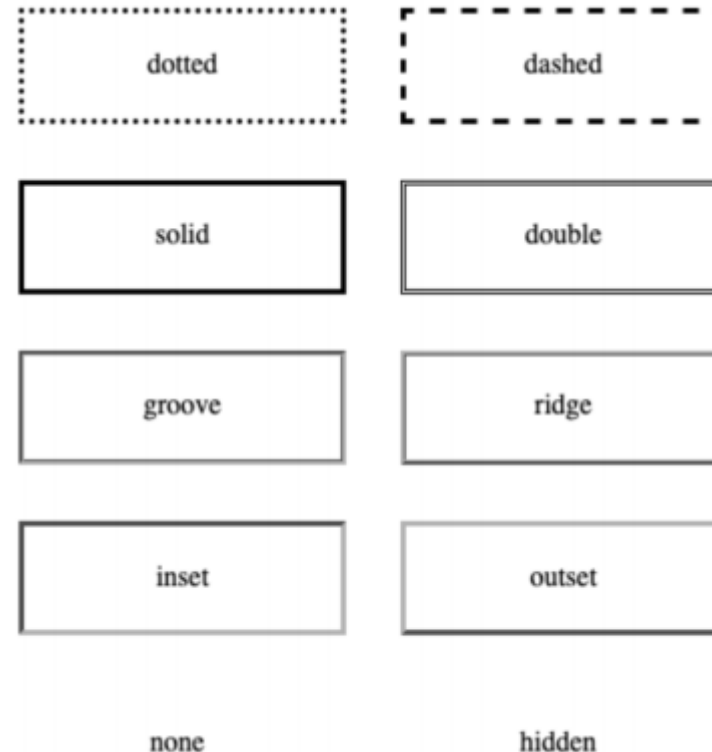
Box model: Todas las etiquetas (cajas) tienen propiedades en común.

- **Content:** espacio para el texto o imágenes.
- **Border:** límite entre el elemento y el espacio externo.
- **Padding:** separación entre el borde y el contenido de la caja, espacio interior.
- **Margin:** separación entre el borde y el exterior de la caja, espacio exterior.

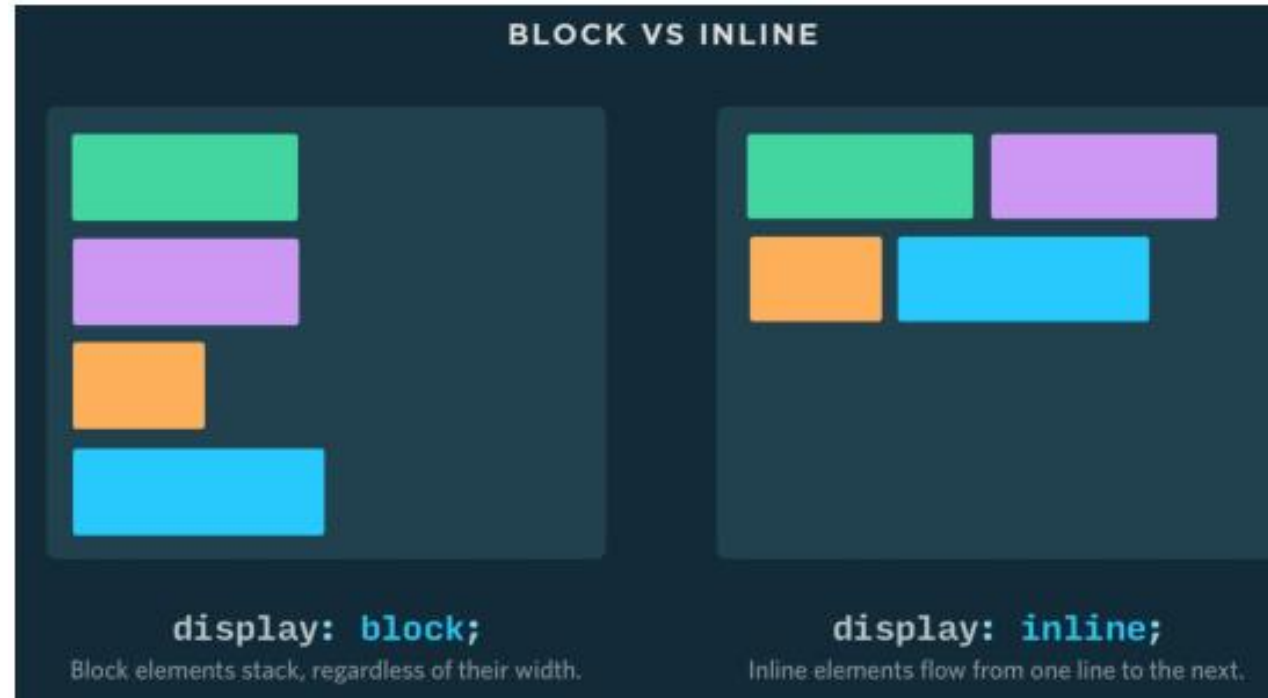
Box Modeling

- ❖ Margin: tenemos “margin, margin-top, right, bottom y left”. Siempre en este orden.
- ❖ Padding: tenemos “padding, padding-top, right, bottom y left”. En este orden.
- ❖ Border: también tenemos border, top, right, bottom y left, pero sí o sí vamos a tener que darle tres valores:

- Tipo de borde (border-style)
- Grosor (border-width)
- Color (border-color)



Block vs Inline



Se usan para marcar texto, imágenes y formularios.

Listado

Se usan para marcar estructura, es decir, una división de información o código.

Listado

Display

Se encarga de definir cómo se ve un elemento HTML. Los dos comportamientos más importantes son:

- Pasar un elemento de bloque a uno de línea → **“inline”**
- Pasar un elemento de línea a uno de bloque → **“block”**

“inline-block”: permite tomar lo mejor de ambos grupos, tenemos padding y margin hacia arriba y abajo.

	Width	Height	Padding	Margin
Bloque	SI	SI	SI	SI
En línea	NO	NO	Solo costados	Solo costados
En línea y bloque	SI	SI	SI	SI

“display: none” oculta y quita un elemento del layout, es decir, no ocupa su lugar.

Para profundizar: **“position”**

Ancho y Alto

- ❑ Ancho (**width**): propiedad CSS que controla la anchura de la caja de los elementos.
- ❑ Alto (**height**): propiedad CSS que controla la altura de la caja de los elementos.

Ninguna admite valores negativos, y los que estén en porcentaje se calculan a partir de la propiedad de su elemento padre.

```
div {  
  background-color:   aqua;  
  width: 50%;  
  height: 120px;  
}
```



Si algún elemento tiene ancho o alto fijos, cualquier contenido que exceda la caja será visible, y al agregar más contenido podría llegar a superponerse.

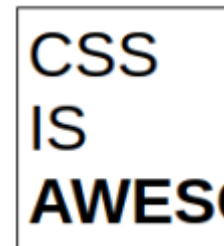
Overflow

Con esta propiedad solucionaremos el problema anterior. Tiene 4 valores posibles:

- ❖ **visible**: valor por defecto (el excedente se ve).
- ❖ **hidden**: el excedente no se muestra, lo corta (Recomendado).
- ❖ **scroll**: genera una barra de scroll en los dos ejes (x,y) del elemento, aunque no se necesite.
- ❖ **auto**: genera el scroll solamente en el eje necesario.

```
div {  
  /* propiedades decorativas */  
  border: solid 1px □ black;  
  padding: 5px;  
  display: inline-block;  
  font-size: 32px;  
  font-family: Arial;  
  /* propiedades que hacen el "problema" */  
  width: 100px;  
  height: 110px;  
  /* solucion */  
  overflow: hidden;  
}
```

```
<div>  
  CSS IS <strong>AWESOME</strong>  
</div>
```

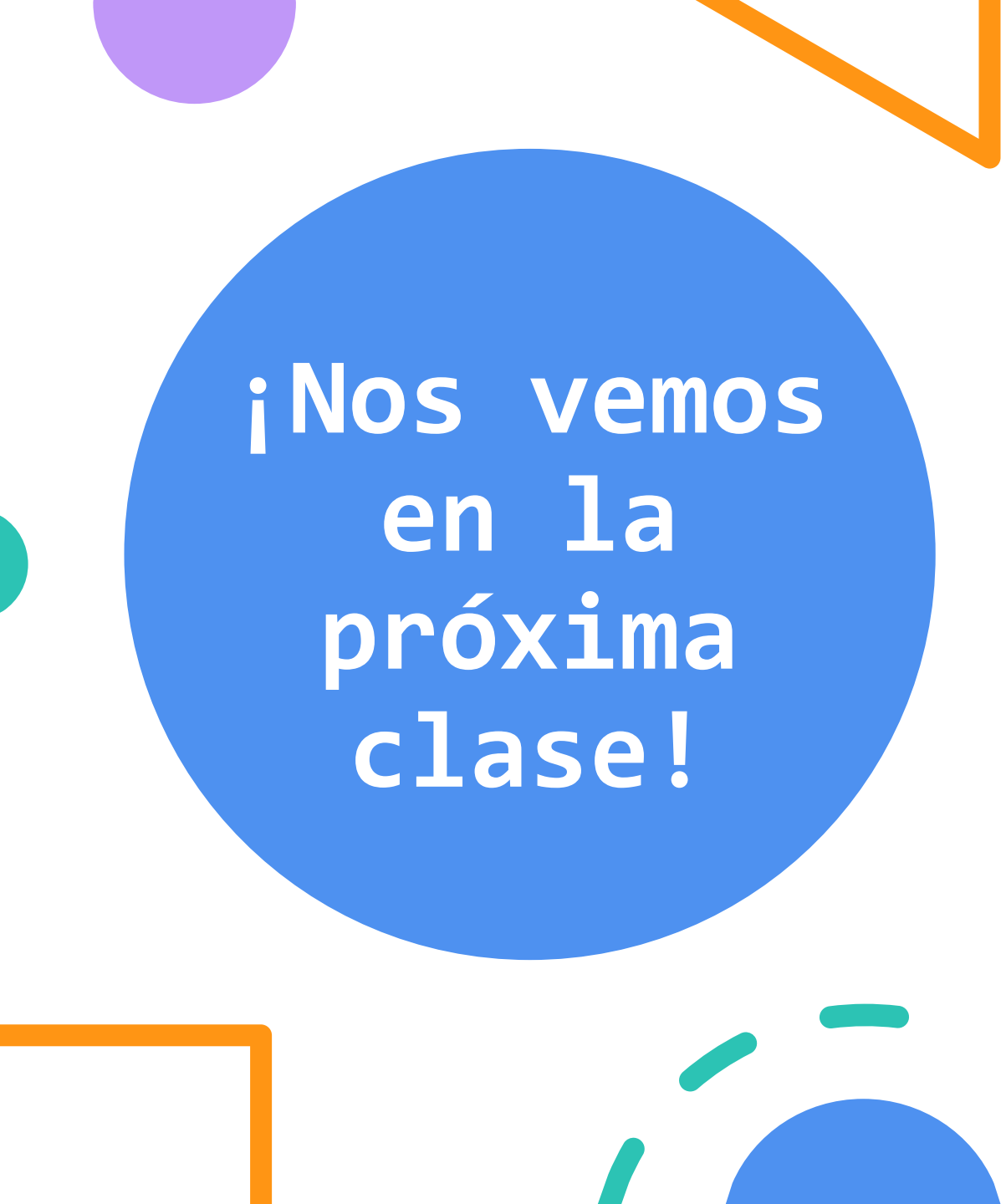


¡A practicar!

Ejercicio 3: Crear un archivo CSS y aplicar todos los estilos que quieran y necesiten.

Ejercicio 4: Agregar al CSS de tu proyecto márgenes (**margin**), rellenos (**padding**), bordes (**border**) y un menú (**position**).

¡No se olviden de implementar lo aprendido en la clase de Git y GitHub al finalizar!



**¡Nos vemos
en la
próxima
clase!**

Rocío Decurgez

rociodecurgez.bam@gmail.com

1150968244