«Talento Tech»

Front-End JS

Clase 06

Buenos Aires Ciudad

Clase 06 | CSS3 - Modelo de caja, posicionamiento y Flexbox

Temario:

1. **Modelado de caja y propiedades**
    - Concepto del Modelo de Caja
    - Contenido, Relleno, Borde y Margen
    - Propiedades Clave: width, height, padding, border, margin
2. **Posicionamiento y Visualización**
3. **Selectores Avanzados**
4. **¿Qué es Flexbox?**
    - Conceptos Básicos de Flexbox
    - Contenedor Flex y Ejes
5. **Propiedades para el Contenedor Flex y los Flex Items**
    - Propiedades para el Contenedor
    - Propiedades para los Flex Items

Objetivo de la clase:

En esta clase vamos a profundizar en el **modelo de caja de CSS**, entendiendo cómo se construyen los elementos con contenido, padding, border y margin. Aprenderemos a usar propiedades esenciales como `width`, `height`, `padding`, `border` y `margin` para controlar el tamaño y el espaciado.

También vamos a ver cómo posicionar elementos en la pantalla usando distintos tipos de **posicionamiento** (`static`, `relative`, `absolute`, `fixed`, `sticky`) y cómo influye la propiedad `display` en su comportamiento.

A continuación, conoceremos **selectores avanzados** para escribir reglas más específicas, incluyendo descendientes, hijos directos y hermanos.

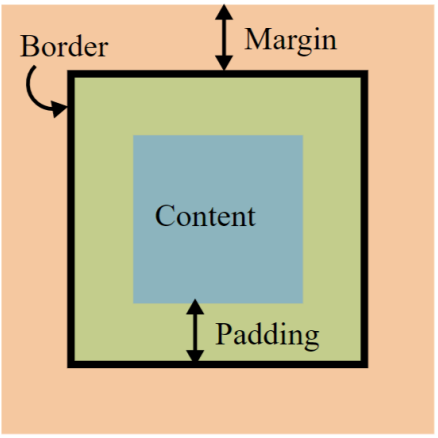
Finalmente, nos introduciremos en **Flexbox**, una herramienta clave para distribuir y alinear elementos de forma flexible y ordenada. Veremos cómo funcionan sus ejes y sus propiedades principales, tanto para el contenedor como para los elementos hijos.

Con estos conceptos, vas a poder tener un control más preciso sobre la estructura y diseño de tus páginas.

Modelo de caja y propiedades

El **modelo de caja** es una de las características más importantes de CSS, ya que determina cómo se estructuran y se muestran los elementos en una página web. Cada elemento de HTML que hemos aprendido es representado por una caja que incluye:

- **Contenido (content):** El área donde se muestra el contenido, como texto, imágenes, etc.
- **Relleno (padding):** Espacio entre el contenido y el borde. Es transparente y permite que el fondo del contenido se muestre.
- **Borde (border):** El límite que rodea el relleno y el contenido. Puede ser estilizado con diferentes colores, grosores y estilos.
- **Margen (margin):** Espacio exterior al borde. También es transparente y permite separar un elemento de otro.



Ejemplo:

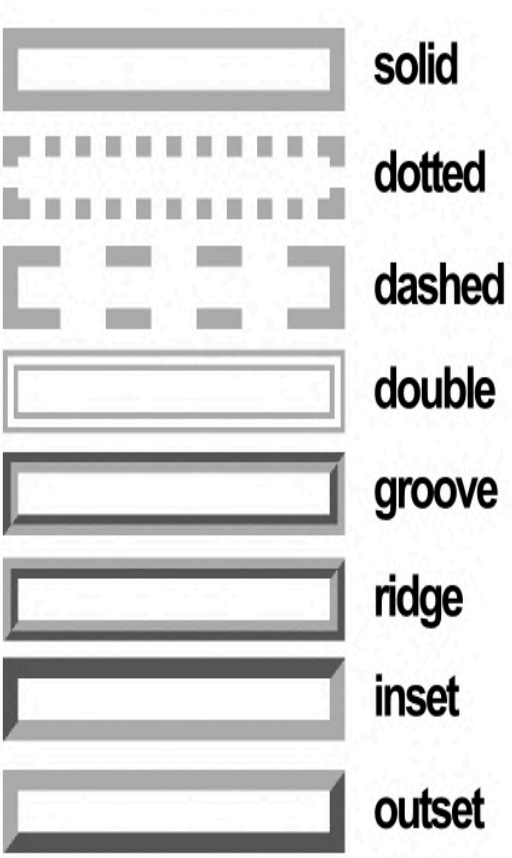
```
.box {
   width: 300px;
   height: 150px;
   padding: 10px;
   border: 2px solid #333;
   margin: 20px;
}
```

En este ejemplo, `.box` es un contenedor con un área de contenido de 300x150 píxeles. Tiene un relleno de 10 píxeles, un borde sólido de 2 píxeles y un margen de 20 píxeles alrededor.

Nota: Los márgenes y rellenos pueden colapsar bajo ciertas circunstancias, especialmente en elementos adyacentes, lo que puede afectar la disposición visual.

Border:

- **hidden**: Oculto. Idéntico a none, salvo para conflictos con tablas.
- **dotted**: Borde basado en puntos.
- **dashed**: Borde basado en rayas (línea discontinua).
- **solid**: Borde sólido (línea continua).
- **double**: Borde doble (dos líneas continuas).
- **groove**: Borde biselado con luz desde arriba.
- **ridge**: Borde biselado con luz desde abajo. Opuesto a groove.
- **inset**: Borde con profundidad «hacia dentro».
- **outset**: Borde con profundidad «hacia fuera». Opuesto a inset.



Posicionamiento y visualización

El **Posicionamiento** en CSS es una herramienta poderosa para controlar la ubicación de los elementos en la página. Existen diferentes modos de posicionamiento:

- **`static`:** Valor por defecto. Los elementos se colocan en su posición natural dentro del flujo del documento.
- **`relative`:** Permite desplazar un elemento respecto a su posición original sin alterar el flujo del documento.
- **`absolute`:** Posiciona el elemento relativo a su contenedor posicionado más cercano o al viewport si no hay un contenedor posicionado.
- **`fixed`:** El elemento se fija en una posición relativa al viewport y no se mueve al hacer scroll.
- **`sticky`:** Combina características de `relative` y `fixed`. El elemento se comporta como `relative` hasta que alcanza un umbral, momento en que se "pega" y se comporta como `fixed`.

Selectores avanzados

Los **selectores avanzados** en CSS permiten aplicar estilos a elementos específicos con mayor precisión. Algunos de los selectores avanzados más comunes incluyen:

Selector de descendientes ( ): Aplica estilos a elementos que están dentro de otro elemento.

```
div p {
    color: blue;
}
```

- Aplica el color azul a todos los párrafos dentro de cualquier `div`.

Selector de hijos directos (>): Aplica estilos solo a los elementos que son hijos directos de un contenedor.

```
div > p {
    color: green;
}
```

- Aplica el color verde a los párrafos que son hijos directos de un `div`.

Selector de hermano adyacente (+): Aplica estilos al elemento que sigue inmediatamente a otro.

```
h1 + p {
    font-size: 1.2em;
}
```

- Aplica un tamaño de fuente diferente al párrafo que sigue inmediatamente después de un `h1`.

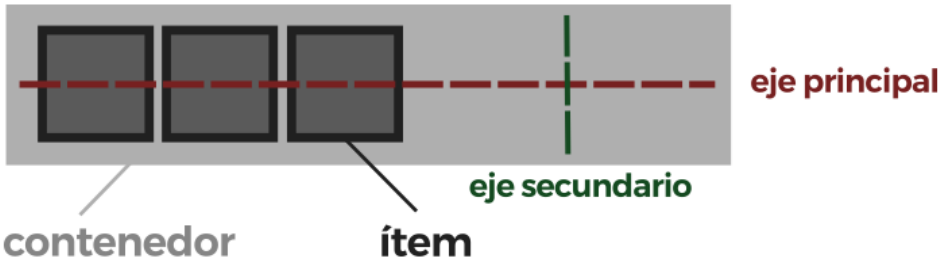
Selector general de hermanos (~): Aplica estilos a todos los elementos que son hermanos de un elemento especificado.

```
h2 ~ p {
    color: red;
}
```

- Aplica el color rojo a todos los párrafos que son hermanos de un `h2`.

Flexbox

Flexbox es un modelo de diseño que facilita la creación de layouts dinámicos y flexibles, especialmente cuando se trata de ajustar elementos dentro de un contenedor. Flexbox es ideal para crear diseños de una sola dimensión, ya sea en un eje horizontal (filas) o vertical (columnas).



Conceptos básicos de Flexbox:

- **Contenedor Flex (`display: flex`):** El elemento padre que contiene los ítems flexibles.
- **Eje principal:** Por defecto, es el eje horizontal donde los ítems se alinean.
- **Eje secundario:** Es el eje perpendicular al principal, utilizado para la alineación secundaria.

Ejemplo de contenedor flex:

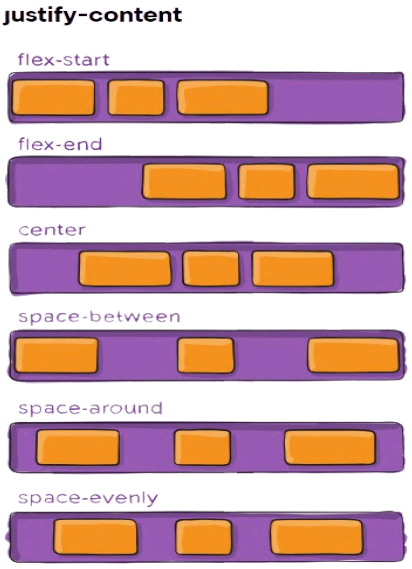
```
.container {
   display: flex;
   justify-content: center;
   align-items: center;
   height: 300px;
   background-color: #eaeaea;
}

.item {
   background-color: #4caf50;
   color: white;
   padding: 20px;
   margin: 10px;
}
```

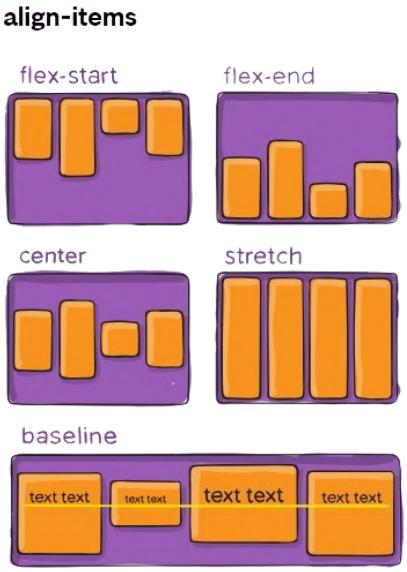
Este código crea un contenedor flex que centra los elementos `.item` tanto horizontal como verticalmente.

Propiedades para el contenedor flex y los flex items

En Flexbox, hay varias propiedades que permiten controlar la disposición de los ítems flexibles dentro del contenedor:

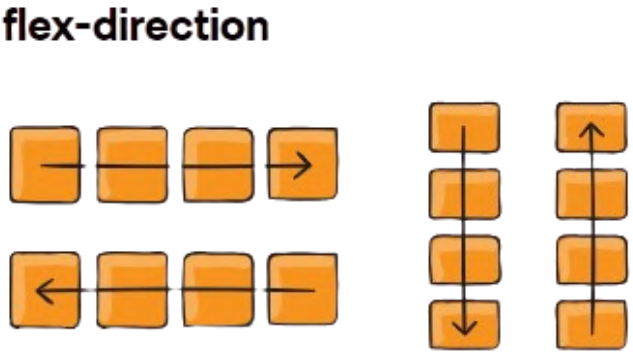


`justify-content`: Alinea los ítems a lo largo del eje principal.
Ejemplos: `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`.



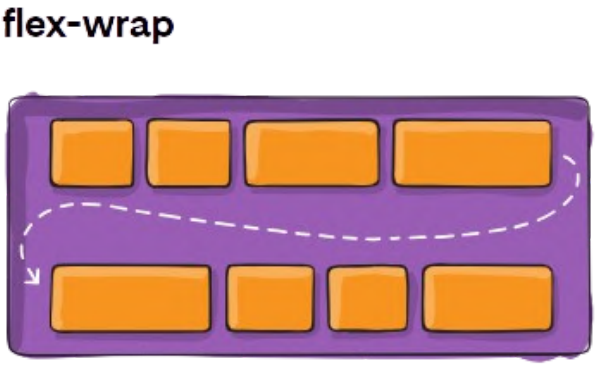
`align-items`: Alinea los ítems a lo largo del eje secundario.

Ejemplos: `stretch` (por defecto), `flex-start`, `flex-end`, `center`, `baseline`.



`flex-direction`: Establece la dirección del eje principal. Valores comunes incluyen `row` (por defecto), `column`, `row-reverse`, y `column-reverse`.

`flex-wrap`: Permite a los ítems flexibles pasar a la siguiente línea si no caben en el contenedor. Valores comunes incluyen `nowrap` (por defecto), `wrap`, y `wrap-reverse`.



Propiedades para los ítems flex:

- **`flex-grow`:** Permite que un ítem crezca para ocupar el espacio disponible. Un valor de `1` significa que el ítem puede crecer para llenar el espacio.
- **`flex-shrink`:** Permite que un ítem se encoja si es necesario para evitar desbordamiento. Un valor de `0` significa que el ítem no se encogerá.
- **`flex-basis`:** Establece el tamaño inicial del ítem antes de que el espacio se distribuya. Puede ser un valor fijo (como `px`, `em`, `%`) o `auto`.

Ejemplo completo de Flexbox:

```
.container {
   display: flex;
   flex-direction: row;
   flex-wrap: wrap;
   justify-content: space-between;
   align-items: center;
}

.item {
   flex: 1 1 200px;
   margin: 10px;
   background-color: #2196F3;
   color: white;
   padding: 20px;
}
```

En este ejemplo, los ítems `.item` se distribuyen en filas, se envuelven cuando es necesario, y se alinean al centro del eje secundario. Cada ítem ocupa un mínimo de 200px y puede crecer si hay espacio disponible.

💭**Storytelling**

Espacios inteligentes y distribución con Flexbox

Tu sitio ya tiene estilo… ahora vamos a ordenarlo como se merece

Lo que viene ahora...

Ya personalizaste tu web con fuentes y fondos únicos. Pero todavía falta algo:
ordenar correctamente el contenido y distribuirlo con lógica visual. Es momento de dominar el **modelo de caja (box model)** y utilizar **Flexbox**, una herramienta moderna para acomodar elementos de forma dinámica y adaptable.

Lucía – Product Owner

“Quiero que el sitio de Talento Lab se vea limpio, que los espacios respiren, que el contenido esté bien separado. El diseño no es solo lo que se ve: es cómo se organiza.”

Tomás – Desarrollador Senior

“Entender el modelo de caja es clave para posicionar bien cada parte del sitio. Y con Flexbox vas a lograr que la sección de productos se vea perfecta en computadoras, tablets y celulares.”

Ejercicio práctico #1

Crear Tarjetas de Producto con Estilos Completos

Tomás (Desarrollador Senior)

“Ahora que dominan medidas, colores, fuentes y el modelo de caja, ¡es momento de crear algo más real!
Vamos a construir **tarjetas de producto** usando todas las propiedades que fueron aprendiendo.”

¿Qué tenés que hacer?

Dentro de tu page servicios/productos en una sección dentro del `<main>` (`<section class="productos">`), creá al menos **3 tarjetas de producto** pueden utilizar etiquetas como `<article>` .

Cada tarjeta debe tener:

- Imagen del producto
- Nombre del producto (con fuente externa)
- Descripción breve
- Precio destacado
- Un botón de “Agregar al carrito” con ícono de Font Awesome o Flaticon.

Ejercicio práctico #2

Tarjetas de producto

Lucía – Product Owner

“Ya aprendiste a diseñar las tarjetas de producto. Ahora vamos a enfocarnos en cómo **distribuirlas** correctamente en pantalla. Y no solo eso: también te voy a mostrar cómo se usa Flexbox en **otras partes clave** de la web, como la barra de navegación.”

¿Qué tenés que hacer?

- **Una distribución de productos con Flexbox**

1. Dentro de tu HTML, modificaremos la sección creada con la clase `"productos"`.
2. Dentro de esa sección, ordenaremos a los 3 `<article>` que representan a los productos diseñados en el ejercicio anterior.
3. En tu archivo CSS, aplicá `display: flex` al padre para distribuir a los hijos de forma horizontal.

Asegurate de:

- Usar `gap` para separar las tarjetas
- Usar `flex-wrap` para que se adapten en pantallas más chicas
- Usar unidades relativas o anchos fijos para los productos según el diseño

Barra de navegación

¿Qué tenés que hacer?

- Distribuir los ítems de la barra de navegación

Revisá o creá una barra de navegación `<nav>` que contenga una lista `<ul>` con enlaces (`<a>`).

En tu CSS, usá `display: flex` en la lista para distribuir horizontalmente los ítems.

Usá propiedades como:

- `justify-content: space-around` o `space-between`
- `text-decoration: none` para quitar el subrayado
- `padding`, `color`, `background-color` para darle estilo

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad