

CSS



CSS



¿Qué es CSS?

CSS es un lenguaje de estilos utilizado para describir la presentación de un documento HTML. A través de CSS, puedes controlar el diseño visual de una página web, incluyendo aspectos como colores, fuentes, márgenes, espaciado, alineación, y más.

CSS permite separar el contenido (HTML) de la presentación, lo que facilita la actualización y mantenimiento de las páginas web.

Ventajas de CSS

- **Separación de contenido y presentación:** Permite mantener el HTML limpio y enfocado en la estructura, mientras que el CSS maneja la apariencia.
- **Reutilización:** Un solo archivo CSS puede ser utilizado por múltiples páginas web, permitiendo cambios globales rápidos.
- **Accesibilidad:** Mejora la accesibilidad al permitir adaptaciones específicas para diferentes dispositivos y necesidades de usuarios.

Sintaxis de CSS

La sintaxis de CSS consiste en reglas que especifican cómo aplicar estilos a los elementos HTML. Cada regla está compuesta de un selector y un bloque de declaración.

- **Selector:** Identifica el elemento HTML al que se aplicará el estilo.
- **Bloque de declaración:** Contiene propiedades y valores que definen el estilo.

```
p {  
  color: blue;  
  font-size: 14px;  
}
```

Selectores básicos: Selector de tipo

Selecciona todos los elementos de un tipo específico, como **p**, **h1**, **div**.

```
p {  
  color: blue;  
  font-size: 14px;  
}
```

Selectores básicos: Selector de clase

Selecciona elementos que tienen una clase específica. Se define con un punto `.` seguido del nombre de la clase.

```
.destacado {  
    background-color: #ffeb3b;  
    font-weight: bold;  
}
```

Selectores básicos: Selector de ID

Selecciona un elemento con un ID específico. Se define con un # seguido del nombre del ID.

```
#especial {  
    color: #007bff;  
    text-align: center;  
    font-style: italic;  
}
```

Selectores básicos: Selector universal

Selecciona todos los elementos en la página. (*)

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


Enlazado y embebido de CSS

Existen diferentes formas de agregar CSS a un documento HTML:

Enlazado (External CSS): Se utiliza un archivo CSS externo que se enlaza al documento HTML mediante la etiqueta `<link>` en la sección `<head>`. Este método es ideal para mantener un diseño uniforme en múltiples páginas.

```
<link rel="stylesheet" href="styles.css">
```

Enlazado y embebido de CSS

Embebido (Internal CSS): Se define el CSS dentro de la sección `<head>` del documento HTML usando la etiqueta `<style>`. Este método es útil para estilos que sólo afectan una página específica.

```
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
  }
</style>
```

Enlazado y embebido de CSS

En línea (Inline CSS): Se aplica CSS directamente en un elemento HTML mediante el atributo `style`. Este método se debe usar con moderación, ya que puede hacer el código menos limpio y difícil de mantener.

```
<p style="color: green;">Este es un texto verde.</p>
```

Enlazado y embebido de CSS
















En línea (Inline CSS): Se aplica CSS directamente en un elemento HTML mediante el atributo `style`. Este método se debe usar con moderación, ya que puede hacer el código menos limpio y difícil de mantener.

```
<p style="color: green;">Este es un texto verde.</p>
```

Colores

Los navegadores modernos soportan 140 colores predefinidos. Puedes consultarlos aquí:

<https://htmlcolorcodes.com/color-names/>

	IndianRed		Lavender		GreenYellow
	LightCoral		Thistle		Chartreuse
	Salmon		Plum		LawnGreen
	DarkSalmon		Violet		Lime
	LightSalmon		Orchid		LimeGreen

Colores

Para los colores también se puede usar RGB o HEX.

Para la transparencia se utiliza el Alpha (0 es totalmente transparente, 1.0 nada transparente)

```
color: #ff6347;  
color: rgba(255, 99, 71, 0.5);  
color: rgb(255, 99, 71, 0.5);
```



#	DAFF33
R	218
G	255
B	51

Propiedades de texto

color: Define el color del texto.

font-size: Establece el tamaño de la fuente.

font-family: Especifica la familia de fuentes para el texto.

font-weight: Controla el grosor del texto (normal, bold).

line-height: Define el espacio entre las líneas de texto.

text-align: Alinea el texto dentro de su contenedor (left, center, right).

text-decoration: Añade decoraciones al texto, como subrayado o tachado.

text-transform: Controla la capitalización del texto (uppercase, lowercase, capitalize).

letter-spacing: Ajusta el espacio entre caracteres.

Propiedades de fondo

background-color: Establece el color de fondo de un elemento.

background-image: Aplica una imagen de fondo a un elemento.

background-repeat: Controla si la imagen de fondo se repite o no, y en qué dirección lo hace. (repeat, no-repeat, repeat-x, repeat-y)

background-position: Determina la posición inicial de la imagen de fondo dentro del elemento. Los valores pueden ser palabras clave (e.g., center, top, bottom, left, right), o valores específicos en píxeles o porcentajes.

background-size: Ajusta el tamaño de la imagen de fondo.

background-attachment: Controla si la imagen de fondo se desplaza junto con el contenido o permanece fija en su posición cuando el usuario hace scroll.

Propiedades de borde

border-width: Define el ancho del borde. Se puede especificar para todos los bordes o individualmente para cada lado.

border-style: Define el estilo del borde. Los valores pueden ser `none`, `solid`, `dashed`, `dotted`, `double`, `groove`, `ridge`, `inset`, `outset`.

border-color: Define el color del borde. Puede ser un solo color o diferentes colores para cada lado.

border-radius: Redondea las esquinas del borde. Puedes especificar un solo valor para todas las esquinas o valores individuales para cada esquina.

Propiedades de dimensión

width: Define el ancho de un elemento.

height: Define la altura de un elemento.

max-width: Establece el ancho máximo que un elemento puede tener.

max-height: Establece la altura máxima de un elemento.

min-width: Establece el ancho mínimo que un elemento debe tener.

min-height: Establece la altura mínima de un elemento.

Cajas en CSS (display)

block: Las cajas de bloque ocupan todo el ancho disponible de su contenedor padre.

inline: Ocupan solo el espacio necesario para el contenido y no interrumpen el flujo del texto. No permiten propiedades de dimensión.

inline-block: Se comportan como un elemento en línea en términos de disposición (es decir, no interrumpen el flujo del texto), pero pueden tener dimensiones (width, height) como un bloque.



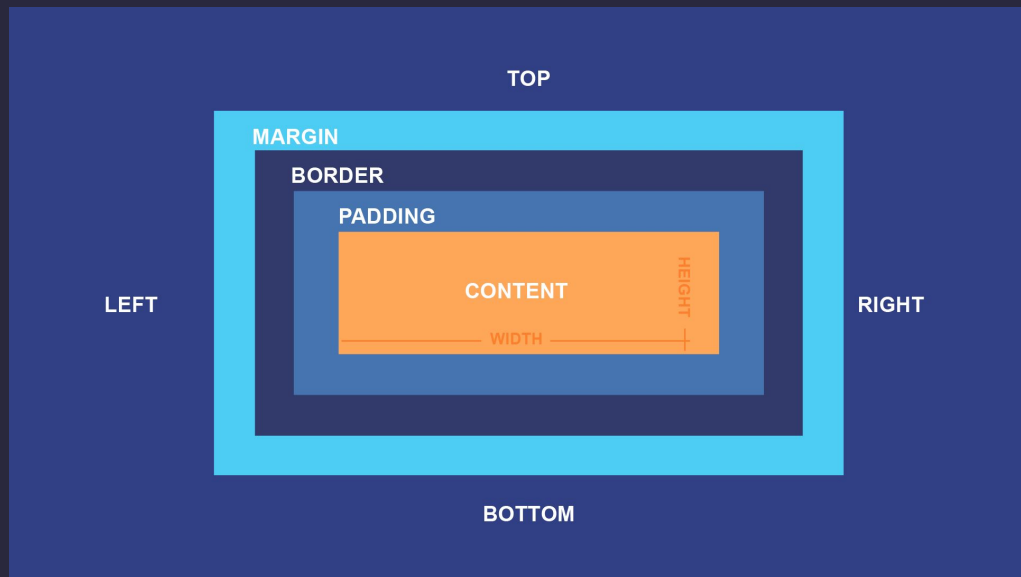
inline

inline

block

inline-block

Box Model



Box shadow

La propiedad box shadow da una sombra a las cajas.

El orden de los valores es:

`box-shadow: x y desenfoque borde color`

Text shadow

La propiedad box shadow da una sombra al texto.

El orden de los valores es:

`text-shadow: x y desenfoque color`

Position

La propiedad `position` en CSS se utiliza para definir cómo se posiciona un elemento dentro de su contenedor o en la página.

Dependiendo del valor que se le asigne, se controla cómo se comporta el elemento respecto a sus hermanos y su contenedor. Los valores principales que puede tomar son: **`static` (por defecto), `relative`, `absolute`, `fixed` y `sticky`.**

position: relative

Un elemento con **position: relative** se posiciona de acuerdo al flujo normal, pero puedes ajustar su posición con **top**, **left**, **right**, y **bottom** sin afectar a otros elementos. El espacio original que ocupa en el documento permanece reservado.

Top y left tienen preferencia sobre right y bottom.

z-index

Controla el apilamiento de elementos en el eje Z, que es el eje perpendicular a la pantalla (profundidad).

El z-index solo tiene efecto en elementos que tienen una posición diferente de static.

Acepta valores enteros (positivos y negativos). Un elemento con un z-index mayor se apilará sobre uno con un z-index menor.

Se recomienda utilizar entre 30 y 50 de diferencia entre cada uno.

position: absolute

Un elemento con **position: absolute** se posiciona en relación al contenedor más cercano que tenga una posición distinta de static (generalmente el body si ningún contenedor tiene otra posición).

Si no se especifica una dimensión para el elemento, el tamaño se ajustará al contenido.

Sale del flujo normal del documento, por lo que no afecta ni es afectado por otros elementos.

position: fixed

Un elemento con **position: fixed** se posiciona con respecto a la ventana del navegador, es decir, su posición no cambiará aunque el usuario haga scroll.

Como en absolute, sale del flujo normal del documento.

position: sticky

Un elemento con **position: sticky** actúa como relative hasta que se cumple una **condición de desplazamiento (scroll)**.

Cuando llega a una posición definida por top, left, etc., se comporta como fixed, quedándose fijo en la pantalla.

Pseudoelementos

Permiten aplicar estilos a partes específicas de un elemento sin necesidad de modificar el HTML. Son útiles para estilizar partes del contenido que no tienen su propio elemento HTML.

Nota:

- No funcionan con elementos `display: inline`.
- No funcionan con elementos que se cierran automáticamente, es decir, que no tienen etiqueta de cierre, como `` e `<input>`

Pseudoelementos

- `::first-letter` Estiliza la primera letra de un bloque de texto.
- `::first-line` Estiliza la primera línea de un bloque de texto.
- `::selection` Aplica estilos al texto que se selecciona con el cursor.
- `::before` Inserta contenido antes del contenido real de un elemento.
- `::after` Inserta contenido después del contenido real de un elemento.

Pseudoclases

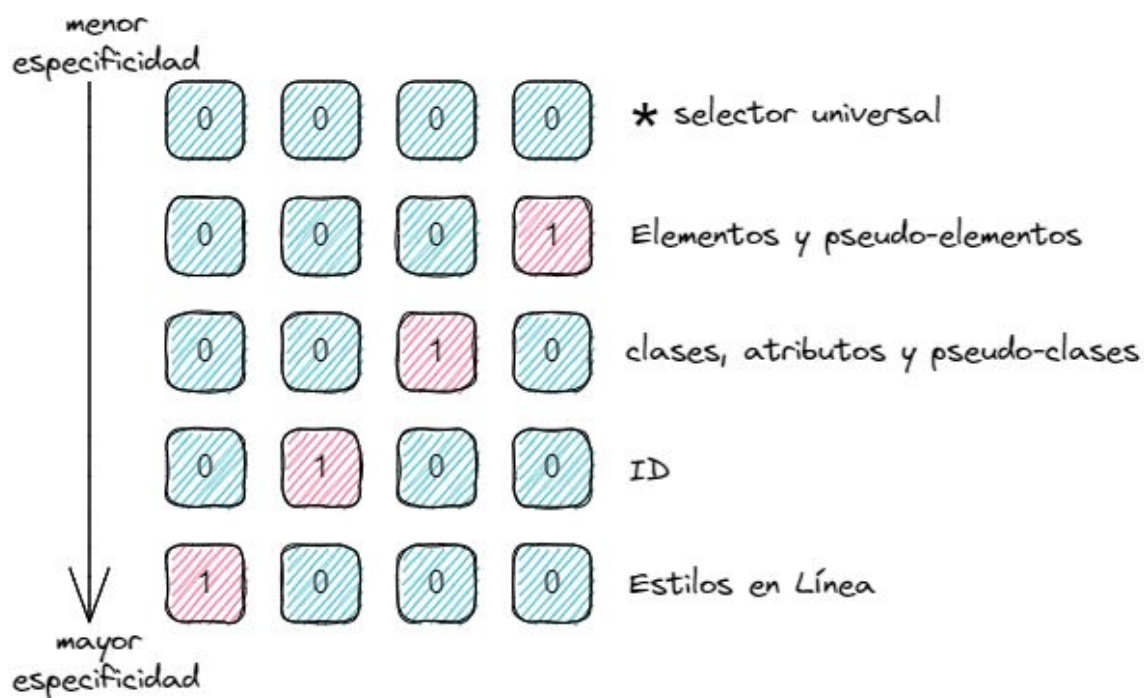
- **:link** Se aplica a cualquier enlace (<a>) con el atributo href que todavía no ha sido visitado por el usuario.
- **:visited** Selecciona los enlaces que ya han sido visitados por el usuario
- **:hover** Se aplica cuando el usuario coloca el cursor sobre un elemento.
- **:focus** Se aplica cuando un elemento, como un campo de texto, gana el foco (al hacer clic o al tabular hacia él).
- **:active** Se activa cuando un elemento está siendo presionado.

Orden recomendado: link, visited, hover, active.

Pseudoclases

- `:enabled` y `:disabled` Selecciona los elementos de formulario que están habilitados y deshabilitados.
- `:checked` Selecciona los elementos de formulario que están marcados o seleccionados (como los checkboxes o radios).
- `:first-of-type` y `:last-of-type` Seleccionan el primer o último hijo de un tipo específico dentro de su padre.
- `:first-child` y `:last-child` Selecciona el primer y último hijo de su elemento padre.
- `:nth-child` Selecciona el enésimo hijo de su padre. `n` puede ser un número, una fórmula o una palabra clave.
- `:nth-of-type(n)` Selecciona el enésimo hijo del mismo tipo de elemento

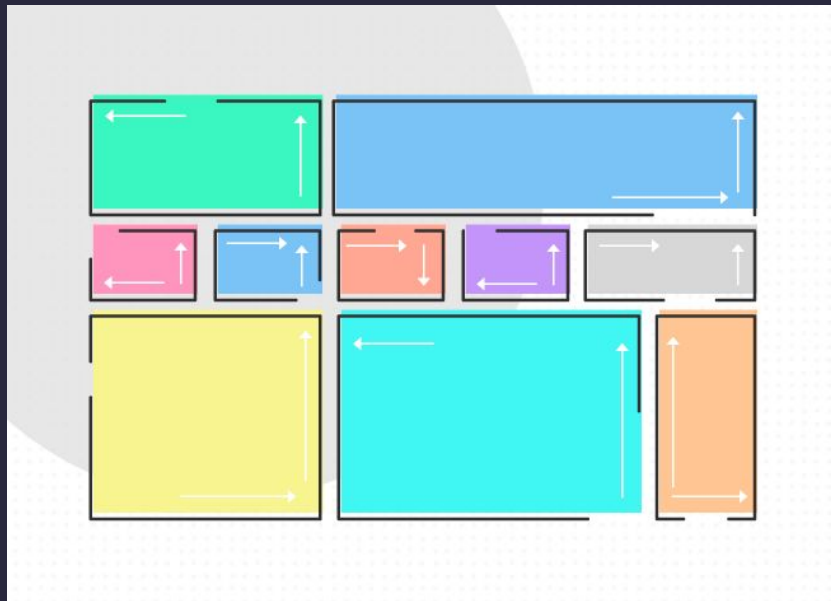
Especificidad



!important gana a todos

Flexbox

Permite crear diseños complejos de manera sencilla y eficiente. Con flexbox, los elementos hijos de un contenedor se distribuyen automáticamente en función del espacio disponible, lo que permite un diseño más flexible y adaptable a diferentes tamaños de pantalla.

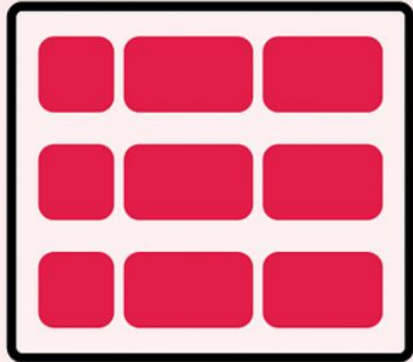


Flexbox: propiedades del contenedor (padre)

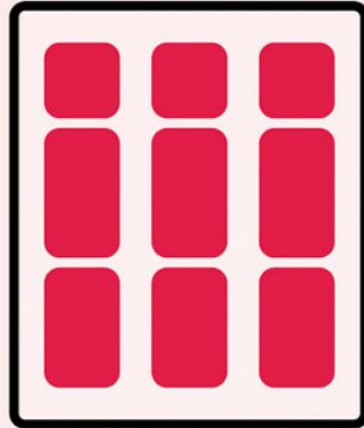
- **display: flex** Activa flexbox en el contenedor.
- **flex-direction** Define la dirección de los elementos dentro del contenedor (row, column, row-reverse, column-reverse).
- **justify-content** Controla la alineación horizontal (flex-start, center, flex-end, space-between, space-around, space-evenly).
- **align-items** Controla la alineación vertical (flex-start, center, flex-end, stretch).
- **flex-wrap** Permite que los elementos se distribuyan en varias líneas si es necesario (nowrap, wrap, wrap-reverse).

Define la dirección de los elementos dentro del contenedor

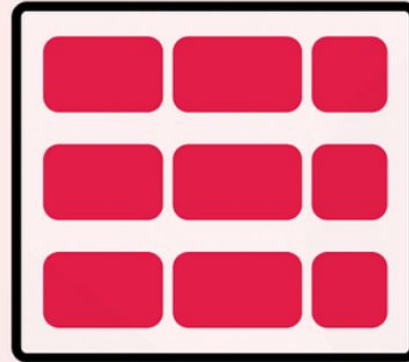
flex-direction



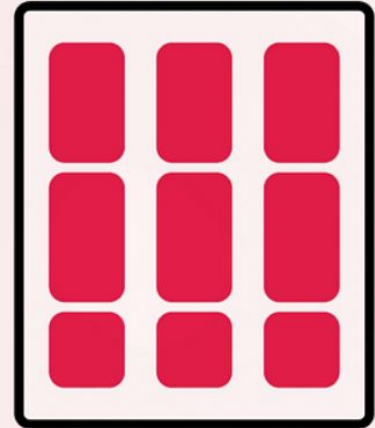
row



column



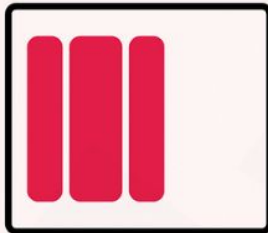
row-reverse



column-reverse

Controla la alineación horizontal

justify-content



flex-start



center



flex-end



space-between



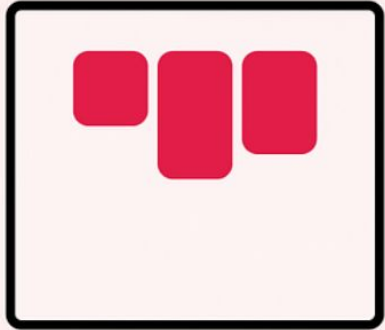
space-around



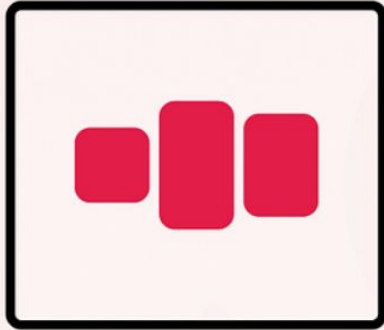
space-evenly

Controla la alineación vertical

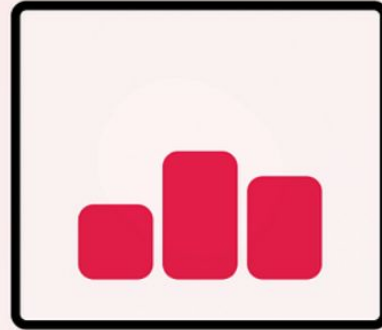
align-items



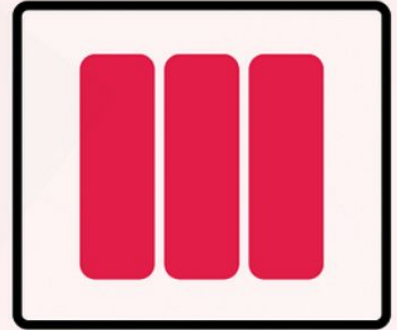
flex-start



center

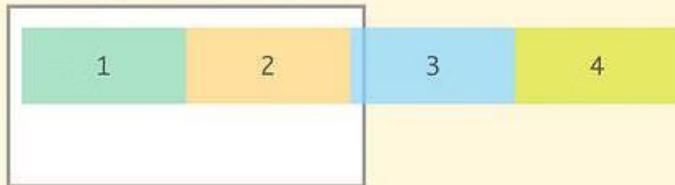


flex-end

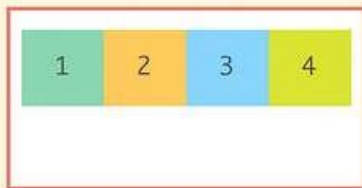


stretch

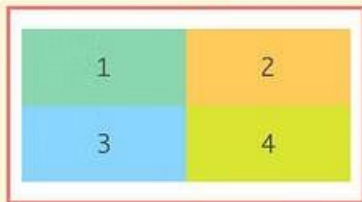
original size



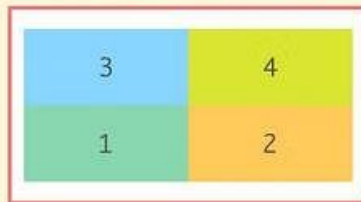
nowrap



wrap



wrap-reverse



flex-wrap

Permite que los elementos se distribuyan en varias líneas si es necesario

Flexbox: propiedades de los elementos hijos

- **flex-grow:** Define cuánto debe crecer un elemento en relación con los demás.
- **flex-shrink:** Define cuánto debe encogerse un elemento en relación con los demás.
- **flex-basis:** Define el tamaño inicial de un elemento antes de que se aplique el flex-grow o flex-shrink.
- **align-self:** Controla la alineación vertical de un elemento individual dentro del contenedor (auto, flex-start, center, flex-end, stretch).
- **order** Controla el orden en el que se muestran los elementos dentro del contenedor. Por defecto, todos tienen order: 0.
- **flex:** shorthand (flex-grow flex-shrink flex-basis)

Funciones en CSS

Las funciones en CSS son herramientas que permiten manipular valores de manera dinámica, lo que ofrece flexibilidad al diseñar estilos.

La función **calc()** permite realizar cálculos matemáticos para valores numéricos en CSS. Es útil para establecer medidas relativas o combinar unidades diferentes.

```
.container {  
  width: calc(100% - 50px);  
}
```


Funciones en CSS





La función **var()** se utiliza para llamar a una variable CSS previamente definida. Esto te permite reutilizar valores en diferentes reglas CSS. Ver: CSS Variables.

Las funciones **rgb()** **rgba()** permiten definir colores con mayor precisión, utilizando valores numéricos para los componentes de color o transparencia.

La función **url()** se utiliza para especificar la ubicación de un recurso, comúnmente imágenes, fuentes o archivos multimedia.

Funciones en CSS

Las funciones **linear-gradient()** y **radial-gradient()** permiten crear degradados de color.

```
div {  
  background: linear-gradient(to right,  red,  yellow);  
}  
  
div {  
  background: radial-gradient(circle,  red,  yellow);  
}
```

Funciones en CSS

Estas funciones son útiles para establecer valores que dependen de ciertas condiciones.

min(): Devuelve el menor de los valores especificados.

max(): Devuelve el mayor de los valores especificados.

clamp(): Establece un valor dentro de un rango, con un mínimo y máximo definidos.



CSS Variables

Las variables CSS, también conocidas como Custom Properties, son una característica poderosa que permite a los desarrolladores definir valores reutilizables para ser utilizados en las hojas de estilo. Estas variables facilitan la gestión y mantenimiento del CSS, especialmente en proyectos grandes o complejos.

CSS Variables

Las variables CSS se definen con un nombre precedido por dos guiones (--) y pueden ser usadas en cualquier lugar de la hoja de estilos.

Una vez definidas, las variables CSS se pueden utilizar en otras propiedades CSS usando la función `var()`.

```
:root {  
  --primary-color:  #3498db;  
  --secondary-color:  #2ecc71;  
  --font-size-large: 24px;  
}  
  
body {  
  color: var(--primary-color);  
  font-size: var(--font-size-large);  
}  
  
h1 {  
  background-color: var(--secondary-color);  
}
```

CSS Variables: Valores por defecto

CSS permite definir un valor por defecto para una variable en caso de que no esté definida o sea inválida.

En este ejemplo, si la variable `--primary-color` no existe, se utilizará el color `black`.

```
p {  
  color: var(--primary-color, black);  
}
```

CSS Grid

Es un sistema de diseño avanzado que permite crear estructuras de diseño de manera eficiente y flexible. Está diseñado específicamente para manejar tanto la disposición de elementos en filas como en columnas, ofreciendo un control bidimensional.

El **contenedor** se convierte en un grid cuando se le aplica la propiedad **display: grid;**

Los **elementos hijos** del contenedor se llaman **grid items** y son los elementos que serán organizados dentro del grid.

CSS Grid: Filas y columnas

Para crear la estructura de la cuadrícula, se utilizan las propiedades **grid-template-columns** y **grid-template-rows**. Estas permiten definir el número de columnas y filas, y sus tamaños.

Las unidades como **fr** (fracción) permiten asignar partes proporcionales del espacio disponible.

Se puede usar también como abreviación **grid-template: filas / columnas**

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 2fr; /*Segunda columna doble de la primera */  
  grid-template-rows: 100px 300px; /*2 filas */  
}
```


CSS Grid: Colocar elementos

Los elementos pueden posicionarse explícitamente usando las propiedades **grid-column** y **grid-row**. Estas permiten que un elemento se extienda a través de varias columnas o filas.

Se colocan dos valores: (Ejemplo: **grid-column**: inicio / fin)

- **inicio**: Número de la columna donde el elemento comienza.
- **fin**: Número de la columna donde el elemento termina (exclusivo).

```
.item {  
  grid-column: 1 / 3; /* Empieza en la columna 1 y termina antes de la columna 3 */  
}
```

CSS Grid: Colocar elementos

- **Valores negativos:** Tanto en `grid-column` como en `grid-row`, se pueden usar valores negativos. Un valor negativo cuenta desde el final del grid hacia el inicio. Por ejemplo, `grid-column: -1 / -2;` coloca el elemento en la penúltima y última columna.
- **span:** Se puede usar la palabra clave `span` para que un elemento abarque varias columnas o filas sin necesidad de especificar exactamente el número de inicio o fin.

```
.item {  
  grid-column: span 2; /* Ocupa dos columnas */  
  grid-row: span 3;    /* Ocupa tres filas */  
}
```

CSS Grid: grid-template-areas

Permite organizar y nombrar áreas específicas dentro de la cuadrícula, lo que facilita la colocación de elementos en un diseño.

Cada área de la cuadrícula se denomina con un nombre y, usando el valor de **grid-area**, se puede asignar un elemento HTML a esa área.

```
.grid-container {  
  display: grid;  
  grid-template-areas:  
    "header header header"  
    "sidebar content content"  
    "footer footer footer";  
}
```

CSS Grid: grid-template-areas

Cada área de la cuadrícula se asocia con un elemento utilizando la propiedad `grid-area`.

- El div con la clase `header` se asigna al área llamada "header".
- El div con la clase `sidebar` se asigna al área llamada "sidebar".
- El div con la clase `content` se asigna al área llamada "content".
- El div con la clase `footer` se asigna al área llamada "footer".

```
.header {  
  grid-area: header;  
}  
  
.sidebar {  
  grid-area: sidebar;  
}  
  
.content {  
  grid-area: content;  
}  
  
.footer {  
  grid-area: footer;  
}
```

¿Flexbox o Grid?

Grid

Es ideal cuando necesitas organizar el diseño tanto en filas como en columnas, es decir, cuando estás creando una estructura compleja en dos dimensiones.

- Diseño de páginas completas: Creando un layout
- Distribuir contenido en filas y columnas
- Control explícito del espacio
- Diseños complejos que cambian de posición

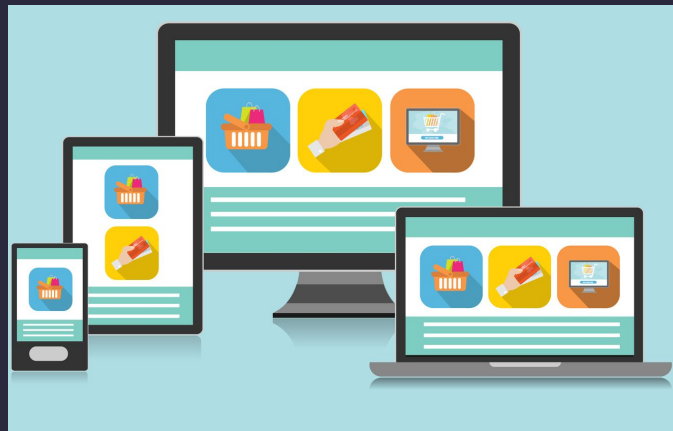
Flex

Es más adecuado cuando solo necesitas alinear y distribuir elementos en una dimensión, es decir, ya sea en una fila o una columna.

- Alinear elementos en una fila o columna
- Distribuir el espacio entre elementos
- Centrado de elementos
- Composiciones flexibles

Sitios web responsivos

Un sitio web responsivo es aquel que se adapta automáticamente al tamaño de pantalla y al dispositivo que está siendo utilizado (computadoras, tablets, smartphones, etc.), ofreciendo una experiencia de usuario óptima sin importar las dimensiones del dispositivo.



Viewports

El viewport es el área visible de una página web en el dispositivo que el usuario está utilizando. En el desarrollo web responsivo, es crucial configurar correctamente el viewport para garantizar que el contenido se adapte adecuadamente a diferentes tamaños de pantalla.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

width=device-width Ajusta el ancho del viewport al ancho del dispositivo.

initial-scale=1.0 Establece el nivel de zoom inicial cuando la página se carga.

Media Queries

Las media queries permiten aplicar estilos CSS en función de las características del dispositivo, como el ancho de pantalla. Se definen puntos de corte (breakpoints) para aplicar diferentes estilos en función del tamaño del dispositivo.

```
@media (max-width: 480px) {  
  body{  
    background: blue;  
    color: white;  
  }  
}
```


Breakpoints

Los breakpoints son los puntos de corte en CSS donde se aplican distintos estilos para adaptarse a diferentes tamaños de pantalla. Aunque los breakpoints pueden ajustarse según las necesidades del diseño, existen algunos valores estándar que se utilizan comúnmente para diseñar sitios web responsivos, adaptándose a los tipos de dispositivos más comunes.

Breakpoints más comunes

320px – 480px: Dispositivos móviles

481px – 768px: iPads, Tablet

769px – 1024px: Pantallas pequeñas, laptops

1025px – 1200px: Escritorio, pantallas grandes

1201px y más— Pantallas extra grandes, TV

Operador lógico “and”

Permite especificar múltiples reglas que deben cumplirse simultáneamente para que los estilos se apliquen.

Por ejemplo, se pueden combinar condiciones como el ancho mínimo y máximo de la pantalla, o agregar otras características como la orientación del dispositivo (horizontal o vertical).

```
@media (min-width: 768px) and (max-width: 1024px) {  
    /* Estilos para pantallas entre 768px y 1024px */  
}
```

```
@media (max-width: 768px) and (orientation: Landscape) {  
    /* Estilos para pantallas de hasta 768px en modo horizontal */  
}
```

Frameworks

Un framework de CSS es una colección predefinida de reglas, estilos y componentes CSS que facilitan y aceleran el desarrollo de interfaces de usuario (UI) en sitios web y aplicaciones. Los frameworks CSS proporcionan un conjunto de clases y estilos listos para usar, que permiten a los desarrolladores diseñar rápidamente páginas web sin tener que escribir CSS desde cero.

Características:

- Sistema de cuadrícula (grid)
- Estilos predeterminados
- Componentes de UI
- Compatibilidad con navegadores
- Diseño responsivo

Frameworks: Ventajas

- **Ahorro de tiempo:** Proporcionan una estructura y estilos predeterminados, acelerando el diseño de interfaces web.
- **Consistencia:** Aseguran una apariencia uniforme en toda la aplicación.
- **Facilidad de uso:** Los desarrolladores pueden crear diseños profesionales sin escribir mucho código.
- **Responsividad:** Facilitan la creación de sitios web que se adaptan automáticamente a diferentes tamaños de pantalla.
- **Comunidad y soporte:** Los frameworks populares cuentan con una gran comunidad y documentación, lo que facilita la resolución de problemas.

Frameworks: Bootstrap

Bootstrap es un framework front-end desarrollado para facilitar la creación de sitios web responsive y con un diseño consistente. Es conocido por sus componentes predefinidos y su sistema de diseño basado en una cuadrícula.

Link al sitio oficial:

<https://getbootstrap.com/>



Frameworks: Bootstrap

Ventajas	Desventajas
Rápido de implementar con componentes y estilos listos para usar.	Menos flexible para personalización profunda.
Documentación extensa y gran comunidad.	Puede ser "pesado" si se incluyen todos los componentes, afectando el rendimiento.
Compatible con todos los navegadores modernos.	

Frameworks: Tailwind CSS

Tailwind CSS es un framework de utilidades que permite crear diseños personalizados directamente en el HTML, utilizando clases utilitarias. En lugar de ofrecer componentes predefinidos, Tailwind se enfoca en proporcionar clases de bajo nivel que permiten un control total sobre el diseño.

Link al sitio oficial:

<https://tailwindcss.com/>



Frameworks: Tailwind

Ventajas	Desventajas
Gran flexibilidad y personalización sin restricciones de componentes predefinidos.	El código HTML puede volverse extenso y difícil de leer por las muchas clases utilitarias.
Ligero, con opción de eliminar clases no utilizadas para optimizar rendimiento.	Mayor curva de aprendizaje para quienes vienen de frameworks como Bootstrap con componentes preconstruidos.
Facilita la creación de diseños a medida sin escribir mucho CSS.	