

LENGUAJE

css

PROGRAMACION II

INTEGRANTES: CADENA JOSEPH,
QUINGALOMBO DANIEL.

DOCENTE: MSC. ALBERTO
ANDOCILLA

SEMESTRE: 7MO A

● INTRODUCCIÓN A CSS

- CSS (Castaing Style Sheets) es un lenguaje que se usa junto con HTML para definir el diseño visual de una página web, como colores, fuentes, tamaños y posiciones. Mientras HTML estructura el contenido, CSS le da forma y estilo, permitiendo que los navegadores muestren páginas web atractivas y adaptables a diferentes dispositivos

Formas de enlazar CSS

En principio, tenemos tres formas diferentes de hacerlo, siendo la primera la más común y la última la menos habitual:

- * Etiqueta `<link rel="stylesheet">`
- * Etiqueta `<style>`
- * Atributo HTML `style="..."`

- Via CSS externo : En el `<head>` del HTML se usa `<link>` para enlazar un archivo CSS externo, así el navegador aplica los estilos rápido. El atributo `type="text/css"` ya no es necesario en HTML5

Estructura de CSS

Los archivos CSS definen estilos que se aplican a documentos HTML mediante reglas formadas por selectores, propiedades y valores. Es importante mantener un código limpio y ordenado usando indentación y una regla por línea para facilitar su lectura y mantenimiento.

CSS MODERNO ANTES VS AHORA

El CSS moderno permite escribir estilos de forma más clara y eficiente usando nuevas funcionalidades como `:is()` para simplificar selectores, nesting para organizar mejor el código, variables personalizadas, y mejoras en centrado, colores y media queries. Estas técnicas reemplazan métodos tradicionales con opciones más limpias y potentes.

¿Por qué se usa CSS?

CSS permite separar el contenido (HTML) del diseño (estilos), unificando todo lo visual en un solo archivo. Esto facilita modificar el diseño en un solo lugar, evita duplicar código, mejora la organización, acelera la carga de páginas y facilita crear versiones para diferentes dispositivos.

Via bloque de estilos:

El bloque de estilos usa la etiqueta `<style>` dentro del HTML para escribir CSS directamente en el documento. Es útil en casos puntuales, pero lo mejor es usar archivos CSS externos para reutilizar y organizar mejor el código.

- Via estilos en línea

Los estilos en línea se aplican directamente en el atributo `style` de una etiqueta HTML. Aunque es útil para casos específicos, no se recomienda usarlos mucho porque no son reutilizables y dificultan el mantenimiento del código.

Propiedad

`width`
Ancho del elemento

`height`
Alto del elemento

`min-width`
Ancho mínimo permitido

`max-width`
Ancho máximo permitido

`min-height`
Altura mínima permitida

`max-height`
Altura máxima permitida

`aspect-ratio`
Relación entre ancho y alto del elemento

`calc()`
Función que permite hacer cálculos (e.g. `width: calc(100% - 50px)`)

• ¿Cómo dibujar con CSS?

Dibujar con CSS, también llamado CSS Art, consiste en crear ilustraciones usando solo código HTML y estilos CSS, sin imágenes externas

Consejos clave:

Usa HTML semántico para estructurar el dibujo.

- Primeros pasos: Estructura HTML para dibujar con CSS

Para empezar a dibujar con CSS, elige un ejemplo sencillo, como un dibujo formado por formas geométricas básicas (por ejemplo, un diskette).

- Las variables CSS (`--variable`) ayudan a organizar y reutilizar valores fácilmente.

COLOR Y FONDOS

Propiedad	Descripción
color	Cambia el color del texto.
background-color	Cambia el color de fondo de un elemento.

FORMAS DE DEFINIR COLORES EN CSS

01.

Método	Ejemplo	Comentario breve
Palabras clave	blue, red, orange	Fácil de usar, pero limitado en variedad y precisión.
Hexadecimal	#ff0000, #00ffff	Muy común; define el color en formato RGB en base 16.
rgb()	rgb(255, 0, 0)	Define rojo, verde y azul como valores 0–255.
rgba()	rgba(255, 0, 0, 0.5)	Igual que <code>rgb()</code> pero con canal alfa (transparencia).
hsl()	hsl(120, 100%, 50%)	Color por tono, saturación y luminosidad.
hsla()	hsla(120, 100%, 50%, 0.3)	Como <code>hsl()</code> pero con transparencia.
hwb()	hwb(200 0% 0%)	Alternativa a <code>hsl</code> , más intuitiva para algunos casos.
lab() / oklab()	lab(50% 20 30)	Basado en percepción humana del color.

PALABRAS CLAVES DEL SISTEMA

- canvas, canvastext
- linktext, visitedtext, activetext
- buttonface, buttontext, buttonborder
- field, fieldtext
- highlight, highlighttext
- selecteditem, selecteditemtext
- mark, marktext
- graytext

¿Qué es RGB?



3

1

RGB es un modelo de color que combina Rojo (Red), Verde (Green) y Azul (Blue) en distintas intensidades para crear casi cualquier color.

1

- `rgb(255, 0, 0)` → Rojo puro
- `rgb(255, 255, 0)` → Amarillo (Rojo + Verde)
- `rgb(255, 255, 255)` → Blanco (todos al máximo)
- `rgb(0, 0, 0)` → Negro (todos al mínimo)

sintaxis moderna

Sintaxis moderna (recomendada)

css

CopiarEditar
.element {
background-color: `rgb(125 80 10);` /* Números
0-255 */
background-color: `rgb(55% 25% 75%);` /*
Porcentajes 0%-100% */
background-color: `rgb(100% 50% 25% / 0.5);` /* Con
transparencia (0-1 o %) */
}

FORMATO HEXADECIMAL

i QUE ES

Es una forma abreviada del modelo RGB que representa los colores con un número precedido por #, usando el sistema numérico hexadecimal

Que es RGB

RGB es un modelo de color que combina Rojo (Red), Verde (Green) y Azul (Blue) en distintas intensidades para crear casi cualquier color

- `rgb(255, 0, 0)` → Rojo puro
- `rgb(255, 255, 0)` → Amarillo (Rojo + Verde)
- `rgb(255, 255, 255)` → Blanco (todos al máximo)
- `rgb(0, 0, 0)` → Negro (todos al mínimo)

HSL significa Hue (matiz), Saturation (saturación) y Lightness (luminosidad). Es un modelo de color intuitivo que facilita crear y ajustar colores visualmente.

- Hue (Matiz): el color base (de 0° a 360°).
 - o 0° = rojo, 120° = verde, 240° = azul

SINTAXIS MODERNA

- Saturation: intensidad del color (0% apagado → 100% vibrante).
- Lightness: brillo del color (0% negro → 100% blanco).

¿Qué es HSL?

SINTAXIS MODERNA

```
.element {  
background-color: hsl(120 50% 50%); /* Verde medio */  
background-color: hsl(240 100% 75% / 50%); /* Azul claro con 50% opacidad */  
}
```

SINTAXIS ANTIGUA

```
.element {  
background-color: hsl(120, 50%, 50%);  
background-color: hsl(240, 100%, 75%, 0.5); /* También existe hsla() */  
}
```



color-mix() permite mezclar dos colores en un espacio de color determinado (como srgb, hsl, lab, oklch, etc.), generando automáticamente un color intermedio.

CON PORCENTAJE EXPLÍCITO

```
background-color: color-mix(in srgb, red 30%, blue 70%);
```

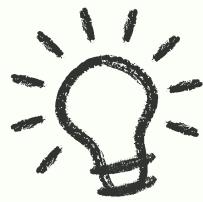
LA FUNCIÓN COLOR-MIX

USO CON ESPACIOS DE COLOR POLARES + INTERPOLACIÓN

```
background-color: color-mix(in oklch longer hue, red 40%, blue 60%);
```

1 USO PRÁCTICO: ACLARAR U OSCURECER UN COLOR

```
:root {  
  --main-color: indigo;  
  --dark: color-mix(in srgb, var(--main-color), black 25%);  
  --light: color-mix(in srgb, var(--main-color), white 25%);  
}
```



Un espacio de color define el rango de colores que se pueden representar en un sistema (pantalla, impresora, web, etc.). El más común en la web es sRGB

En CSS, puedes usar distintos espacios de color como display-p3, prophoto-rgb o xyz para obtener colores más precisos y ricos en dispositivos compatibles.

ESPACIOS DE COLOR

Usa la función color() para especificar colores en un espacio concreto:

css
CopiarEditar
background: color(display-p3 50%
25% 75%);



3.5 ESPACIOS DE COLOR



¡QUE ES UN ESPACIO DE COLOR?

Un espacio de color define el rango de colores que se pueden representar en un sistema (pantalla, impresora, web, etc.). El más común en la web es sRGB.

Usa la función color() para especificar colores en un espacio concreto:

css

Copiar Editar

```
background: color(display-p3 50% 25% 75%);
```

1

2

MODELOS DE CAJAS

El modelo de cajas es la forma en que CSS representa visualmente un elemento HTML. Cada elemento se compone de:

- Contenido (azul): lo que ves, como texto o imágenes.
- Padding (verde): espacio interno entre el contenido y el borde.
- Borde (negro): el contorno del elemento.
- Margen (naranja): espacio externo entre elementos



DIMENSIONES EN CSS

3

Para establecer el ancho (width) y el alto (height) de un elemento HTML usamos estas propiedades.

Por defecto, su valor es auto, lo que significa que el tamaño se adapta al contenido.



4

Cuando el contenido de un elemento es más grande que su tamaño definido, se produce un desbordamiento.

Para controlarlo, usamos la propiedad overflow.



EJEMPLOS EN OVERFLOW

- visible (por defecto): el contenido sobresale.
- hidden: el contenido sobrante se oculta.
- clip: igual que hidden, pero sin desplazamiento.
- scroll: siempre muestra barras de desplazamiento.
- auto: muestra barras solo si es necesario.



DESVORDAMIENTO EN OVERFLOW

LOGICA CSS



Navegadores web

- Los navegadores (clientes) son programas para acceder y ver páginas web, como Chrome, Firefox, Edge, Safari u Opera.
- Idealmente, las webs se verían igual en todos, pero cada navegador tiene diferencias por su motor y prioridades.
- Motores principales:
 - o Chrome, Edge, Opera usan Blink (basado en Chromium).
 - o Firefox usa Quantum.
 - o Safari usa Webkit.
- W3C define estándares, pero cada navegador implementa a su ritmo.
- Historia: Internet Explorer fue dominante, pero se estancó y perdió terreno frente a Chrome y otros.
- Navegadores populares hoy:
 - o Chrome (Google)
 - o Firefox (Mozilla)
 - o Edge (Microsoft)
 - o Opera (Opera)
 - o Safari (Apple)

CALCULOS CSS

¿Qué es una función CSS?

- Son herramientas que permiten realizar cálculos o transformaciones directamente en CSS.
- Ayudan a evitar repetir valores y facilitan cambios globales sin errores.
- Se usan junto a propiedades como width, height, etc.

• TIPOS DE FUNCIONES CSS MATEMÁTICAS

Función	Tipo	Descripción breve
<code>calc()</code>	Cálculo	Suma/resta/multiplica/divide valores (ej: 100% - 50px)
<code>min() / max()</code>	Comparación	Devuelven el valor mínimo o máximo entre varios
<code>clamp()</code>	Comparación	Limita un valor entre un mínimo y un máximo
<code>abs() / sign()</code>	Signo	Valor absoluto o signo (+/-)

Función	Tipo	Descripción breve
<code>round() / mod() / rem()</code>	Escalonadas	Redondeo, módulo y resto
<code>sin() / cos() / tan()</code>	Trigonométricas	Funciones trigonométricas básicas
<code>asin() / acos() / atan() / atan2()</code>	Trigonométricas	Funciones trigonométricas inversas
<code>pow() / sqrt() / hypot()</code>	Exponenciales	Potencias, raíces cuadradas, hipotenusa
<code>log() / exp()</code>	Exponenciales	Logaritmos y potencias de e

COMBINADORES

¿Qué es un Combinador CSS?

Un combinador CSS es un símbolo que une dos o más selectores para formar un selector más complejo y preciso. Así puedes seleccionar elementos HTML basándote en su relación con otros elementos (como descendencia, hermanos, etc.).

Tipos de Combinadores CSS

Nombre	Símbolo	Ejemplo	Significado
Descendiente	(espacio)	#page div	Selecciona todos los elementos div dentro de #page (a cualquier nivel).
Hijo	>	#page > div	Selecciona solo los hijos directos div de #page (primer nivel).
Hermano adyacente	+	div + div	Selecciona el div que sigue inmediatamente después de otro div (mismo nivel).
Hermano general	~	div ~ div	Selecciona todos los div hermanos que aparecen después de otro div (mismo nivel).
Universal	*	#page *	Selecciona todos los elementos dentro de #page.

● 7 Layouts



Diseño o estructura visual de una página web.

·Display=> Cambia el tipo o forma de representación del elemento.

ØDisplay: inline=> Se coloca a continuación del otro (en horizontal). Ignora dimensiones.

ØDisplay: block=> Se coloca encima de otro (en vertical).

ØDisplay: inline-block=> Híbrido en línea-bloque. Actúa como un elemento en línea, pero obedece dimensiones.

Alinear y centrar:

·start=> Alinea el elemento al inicio respecto a su contenedor padre.

·center => Centra el elemento respecto a su contenedor padre.

·end=> Alinea el elemento al final respecto a su contenedor padre.

La propiedad display multi-keyword:

·inline => Elemento en línea Ocupa sólo el espacio necesario. Ignora tamaños. Se alinea uno tras otro. ()

•

Guía de Introducción a Grid:

·inline-grid => Establece una cuadrícula con ítems en línea, de forma equivalente a inline-block.

·grid => Establece una cuadrícula con ítems en bloque, de forma equivalente a block.

Alinear y centrar con Grid CSS:

·justify-items (start; end; center; stretch)

·align-items (start; end; center; stretch)

Grid por áreas:

·grid-template-areas => Indica la disposición de las áreas en el grid. Cada texto entre comillas simboliza una fila.

·grid-area => Indica el nombre del área. Se usa sobre ítems hijos del grid.

La propiedad position:

·static => Posicionamiento estático. Utiliza el orden natural de los elementos HTML.

·relative => Posicionamiento relativo. Los elementos se mueven ligeramente en base a su posición estática.

·absolute => Posicionamiento absoluto. Los elementos se colocan en base a un contenedor padre.

·fixed => Posicionamiento fijo. Idem al absoluto, pero aunque hagamos scroll no se mueve.

·sticky => Posicionamiento fijo con desplazamiento («pegajoso»). Muy usado para pegar elementos a una zona.

CSS Anchor Position:

·anchor-name => Declara un elemento como ancla de referencia que permitirá posicionar otros elementos.

·position-anchor => Posicionamos el elemento mediante el ancla de referencia indicado.

Øwidth => Ancho del elemento de referencia.

Øheight => Alto del elemento de referencia.

Øblock => Versión lógica del modo de escritura del contenedor del elemento de referencia.

Øinline => Versión lógica del modo de escritura del contenedor del elemento de referencia.

Øself-block => Versión lógica del modo de escritura del elemento de referencia.

Øself-inline => Versión lógica del modo de escritura del elemento de referencia.

Colocación automática en Grid:

·row => Los elementos se colocan en filas, completando la fila antes de pasar a la siguiente.

·column => Los elementos se colocan en columnas, completando la columna antes de pasar a la siguiente.

Celdas irregulares en Grid:

·grid-column-start => Indica en qué columna empezará el ítem de la cuadrícula.

·grid-column-end => Indica en qué columna terminará el ítem de la cuadrícula.

·grid-row-start => Indica en qué fila empezará el ítem de la cuadrícula.

·grid-row-end => Indica en qué fila terminará el ítem de la cuadrícula.

Líneas con nombre en Grid:

·grid-template-columns [línea1] col1 [línea2] col2 ... [últimalínea] Las líneas se definen entre corchetes.

·grid-template-rows [línea1] fila1 [línea2] fila2 ... [últimalínea] Las líneas se definen entre corchetes.

● 8 Interfaz de Usuario



- Línea externa que resalta elementos visualmente.
- `.outline-width (medium; thin; thick) => Tamaño de los bordes. Tamaño predefinido o específico.`
- `.outline-style => Estilo de la línea divisoria.`
- `.outline-color (invert) => Invierte el color de esta.`
- `.outline-offset (size) => Desplazamiento del contorno.`

- Cursor del ratón:
- `cursor (palabra clave de cursor) => Muestra un cursor de ratón específico.`
- `.default => Muestra el cursor del ratón por defecto del sistema. Usualmente, una flecha/cursor.`
- `.crosshair => Muestra una cruceta. Útil para tareas en las que requieres precisión.`
- `.help => Muestra un cursor de ayuda. Generalmente, una interrogación o un puntero con interrogación.`
- `.move => Muestra un cursor para mover elementos. Se suele representar con flechas hacia todos lados.`
- `.pointer => Muestra un cursor para hacer click. Usualmente, una mano o algún tipo de apuntador.`

- Barras de desplazamiento:
- `.scrollbar-color => Le da color a la barra de desplazamiento. Primer parámetro a la barra y segundo al fondo.`
- `.scrollbar-width => Le da un tamaño a la barra de desplazamiento. Los valores soportados son auto, thin y none.`
- `.scrollbar-gutter => Reserva espacio (evita desplazamientos inesperados al mostrar/ocultar la barra).`
- `auto => La barra de desplazamiento ocupa un margen cuando overflow está en auto o scroll y el contenido se desborda.`

- Presentación de Datos:
- Tablas CSS:
- `<table> => Etiqueta que contiene todo el contenido de la tabla.`
- `<tr> => Representa una fila de una tabla.`
- `<th> => Representa la cabecera o título de una columna.`
- `<td> => Representa una celda de una columna.`
- Modificar características de una tabla HTML:
- `.border-collapse (separate; collapse) => Aplicado sobre la tabla, elimina el espacio de relleno entre celdas.`
- `.border-spacing => Amplia el espacio de relleno entre tabla y celdas.`

- Listas CSS:
- `.list-style-image (none; url(image.png)) => Indica una imagen para usar como viñeta de ítem.`
- `.list-style-position (inside; outside) => Establece o elimina indentación de ítems sobre la lista.`
- `.list-style-type (disc; decimal; type; none => Establece el tipo de viñeta que se va a utilizar.`
- Contadores CSS:
- `.counter-reset (id valor inicial; none) => Resetea el contador id al valor inicial (0 por defecto).`
- `.counter-increment => Incrementa un contador, indicándole su id de referencia.`

- Modificar características de una tabla HTML:
- `.border-collapse (separate; collapse) => Aplicado sobre la tabla, elimina el espacio de relleno entre celdas.`
- `.border-spacing => Amplia el espacio de relleno entre tabla y celdas.`
- `.caption-side (top; bottom) => Mueve el elemento <caption> del interior de una tabla.`
- `.empty-cells (show; hide) => Hace desaparecer visualmente una celda vacía (sin contenido).`
- `.table-layout (auto; fixed) => Indica si las celdas deben ajustarse o tener un tamaño fijo.`

- CSS Snap Scroll: Desplazamiento que se detiene en secciones.
- `.scroll-snap-type => Establece el tipo de desplazamiento de scroll y su dirección.`
- `.scroll-padding (auto ; size) => Establece el relleno que tendrá el scroll.`
- `.scroll-padding-top => Indica un padding de separación en la zona superior al detener el scroll.`
- `.scroll-padding-right => Idem a la derecha.`
- `.scroll-padding-bottom => Idem en la zona inferior.`
- `.scroll-padding-left => Idem a la izquierda.`
- `.scroll-padding => Propiedad de atajo. 1, 2, 3 ó 4`
- Parámetros (funciona igual que una propiedad padding).
- `.scroll-margin:`
- `.scroll-margin-top => Indica un margin de separación en la zona superior al detener el scroll.`
- `.scroll-margin-right => Idem en la zona derecha.`
- `.scroll-margin-bottom => Idem en la zona inferior.`
- `.scroll-margin-left => Idem en la zona izquierda.`
- `.scroll-margin => Propiedad de atajo. 1, 2, 3 ó 4 parámetros (funciona igual que una propiedad margin).`

- Líneas con nombre en Grid:
- `.grid-template-columns [linea1] col1 [linea2] col2 ... [últimalinea] Las líneas se definen entre corchetes.`
- `.grid-template-rows [linea1] fila1 [linea2] fila2 ... [últimalinea] Las líneas se definen entre corchetes.`
- Colocación automática en Grid:
- `row => Los elementos se colocan en filas, completando la fila antes de pasar a la siguiente.`
- `column => Los elementos se colocan en columnas, completando la columna antes de pasar a la siguiente.`
- `.row dense => Idem a row, pero llenando los huecos si los hay.`

● 9 Efectos



Sombras CSS en textos:

- `.text-shadow (none)` => No aplica ninguna sombra en el texto (o la quita si existía previamente).
- `.text-shadow` => Aplica sombra color negro, desplazándola(horizontal) y (vertical).
- `.text-shadow` => Idem a la anterior, pero establece un desenfoque a la sombra (0 sin desenfoque).
- `.text-shadow` => indicando un personalizado para la sombra.

Sombras CSS en cajas:

- `.box-shadow (none)` => Elimina (o simplemente no establece) sombra en un elemento.
- `.box-shadow` => Crea sombra color negro desplazándola en horizontal y/o vertical.
- `.box-shadow` => Idem a anterior, pero difuminando la sombra.
- `.box-shadow` => Idem a anterior, pero aplica un factor de crecimiento a la sombra.
- `.box-shadow` => Idem a la anterior, cambiando el color de la sombra.
- `.box-shadow => inset` Idem a la anterior, pero aplicando sombra interna en lugar de externa.

Efectos (Filtros CSS):

- Modos de fusión CSS:
- `.mix-blend-mode (normal)` => Aplica un modo de fusión específico a un elemento.
- `.background-blend-mode (normal)` => Aplica un modo de fusión específico a un fondo.
- `.isolation (auto; isolate)` => Establece si un elemento debe aislarse del resto.
- `.polygon` => Crea una forma poligonal siguiendo las coordenadas indicadas.
- `.path` => Crea una forma basada en un trayecto SVG.

Máscaras y Recortes:

- Formas básicas:
- `.rect` => Crea una forma rectangular. Posición: x, y. Tamaños: w y h.
- `.inset` => Crea una forma rectangular "hacia dentro".
- `.inset(round)` => Idem a la anterior, pero con bordes redondeados tamaño radius.
- `.xywh` => Crea una forma rectangular definiendo su desplazamiento y tamaño.
- `.xywh (round)` => Idem a la anterior, pero con bordes redondeados tamaño radius.

Sombras idénticas CSS:

- `.drop-shadow =>` Crea una sombra idéntica utilizando la propiedad filter.
- `.drop-shadow =>` Idem a la anterior, con un desplazamiento y además un desenfoque.
- `.drop-shadow =>` Idem a la anterior, pero indicando un color específico.

Ø `mask-image+ (url(image) | url(svgfile#id))`

Ø `mask-mode (match-source; Alpha; luminance)`

Ø `mask-repeat (repeat; repeat-x; repeat-y; no-repeat space; round)`

vadd => Superpone o suma la máscara actual sobre el resto de máscaras inferiores (source over)

La propiedad float:

- `.float (none; left; right)` => Cambia el flujo para que el elemento flote a la izquierda o derecha.
- `.clear (none; left; right; both)` => Impide que los elementos puedan flotar en la orientación indicada.

La propiedad clip-path:

- `.clip-path (none)` => Deja el elemento sin recortar o elimina el recorte.
- `.clip-path` => Crea un recorte con una forma geométrica básica.
- `.clip-path url("file.svg#name")` => Crea un recorte con forma del elemento con id "name" de una imagen externa SVG.

La propiedad shape-outside:

- `.shape-outside (none)` => Indica una forma para adaptarse o la deshabilita.
- `.shape-outside` => Indica una forma a partir de una imagen o gradiente.
- `.shape-margin` => Margen (positivo) para aplicar a la forma de la imagen.
- `.shape-image-threshold` => Indica la sensibilidad de la transparencia para el umbral de forma.

Máscaras en CSS:

- `mask-image` => Indica una imagen, gradiente o forma SVG para usar como máscara.
- `mask-mode` => Indica si usar canales alfa o de luz como máscara.
- `mask-repeat` => Indica cómo se repetirá una máscara con tamaño y posición.
- `mask-position` => Indica la posición donde debe empezar la máscara.
- `mask-clip` => Indica el área al que afectará la máscara.
- `mask-origin` => Indica como posicionar el área que afectará a la máscara.
- `mask-size` => Permite darle un tamaño específico a la máscara.
- `mask-composition` => Si hay múltiples máscaras, define como aplicas composición.
- `mask` => Propiedad de atajo para hacer todo lo anterior.

● 10 Responsive



- El Responsive Web Design (RWD) es una técnica que permite que una web se adapte automáticamente a cualquier dispositivo: móviles, tablets, PCs, etc
 - Responsive se ajusta de forma continua al tamaño de pantalla.
 - Adaptive se adapta solo en ciertos puntos fijos.

- Los porcentajes en CSS se basan en el tamaño del contenedor padre, no del navegador.
- Ejemplo: `.menu { width: 30% }, .content { width: 70% }.`
- Usa min-width y max-width para definir límites de tamaño adaptables.
Ejemplo: `img { max-width: 100%; height: auto; }`
- El viewport es la parte visible de la web.
Ejemplo: `<meta name="viewport" content="initial-scale=1, width=device-width">`
- Usar Flexbox o Grid en lugar de solo porcentajes.

las preferencias de usuario

Son reglas especiales en CSS que permiten adaptar la web según las configuraciones del sistema operativo del usuario.

@media query

Valores

¿Qué detecta?

- prefers-color-scheme
 - light, dark
 - Tema claro u oscuro del sistema
- prefers-reduced-motion
- reduce, no-preference
 - Preferencia por eliminar animaciones

Medios impresos con CSS (@media print)

Las media queries no solo sirven para adaptar contenido a pantallas, también se pueden usar para dar formato especial a las páginas cuando se imprimen.

Código básico:

```
@media print {  
/* Ocultar elementos que no sirven en papel */  
.navigation, .banner, .menu {  
display: none;  
}}
```

Medios paginados con CSS (@page, orphans, break-*)

CSS también permite controlar el diseño de documentos paginados, como impresiones o archivos PDF, mediante reglas y propiedades especiales.

La regla @page:

Define el tamaño y márgenes de cada página.

```
@page {  
size: A4 portrait;  
margin: 1in;  
}
```

Control de saltos de página:

```
@media print {  
table, img, pre, code {  
break-inside: avoid;  
}}
```

Propiedad

- margin
 - Márgenes de impresión
 - orphans, widows
- Evita párrafos cortados al final/inicio de páginas

Media Queries

Son reglas especiales de CSS que permiten aplicar estilos condicionalmente, según características del dispositivo (ancho de pantalla, tipo).

Sintaxis Basico:

```
@media (condición) {  
/* Estilos si se cumple la condición */  
}@media not (condición) {  
/* Estilos si NO se cumple la condición */  
}
```

Media Features

Permiten aplicar estilos específicos según las capacidades del dispositivo del usuario, haciendo posible un diseño responsive.

Categorías clave

1. Tamaño del dispositivo

```
@media (width <= 600px) { ... }
```

```
@media (aspect-ratio: 3/2) { ... }
```

2. Desbordamiento del contenido

```
@media (overflow-block: scroll) { ... }
```

Container Queries

permiten aplicar estilos CSS basados en el tamaño de un contenedor, en lugar del tamaño de la pantalla.

Definir el contenedor

```
.container {  
container-name: parent;  
container-type: inline-size;  
}
```

Usar la regla @container

```
@container parent (max-width: 550px) {  
.item {  
background: blue;  
}  
}
```

● 11 Transformaciones y dibujos



Las transformaciones en CSS permiten modificar visualmente elementos HTML aplicando efectos como moverlos, rotarlos, escalar o deformar, en 2D y 3D

- Funciones básicas 2D:
 - `translate(x, y)`: mueve el elemento en los ejes X e Y.
 - `scale(factor)`: aumenta o reduce el tamaño.
 - `rotate(grados)`: gira el elemento.
 - `skew(x, y)`: inclina el elemento.

Translaciones 2D en CSS:

- Las translaciones mueven un elemento de un lugar a otro usando la propiedad `transform` y las funciones:
 - `translateX(x)`: mueve horizontalmente.
 - `translateY(y)`: mueve verticalmente.
 - `translate(x, y)`: mueve en ambos ejes, combinación de las dos anteriores.

Nueva propiedad `translate`

- `.element {`
- `translate: 50px; /* Mueve 50px en X */`
- `translate: 50px 150px; /* Mueve 50px en X y 150px en Y */`

Rotaciones 2D en CSS:

Las rotaciones giran un elemento alrededor de un eje usando la propiedad `transform` y funciones específicas:

- `rotate(z)`: rota el elemento sobre sí mismo (eje Z).
- `rotateX(x)`: rota el elemento alrededor del eje horizontal X.
- `rotateY(y)`: rota el elemento alrededor del eje vertical Y.

Ejemplos:

```
.element {  
    transform: rotateX(30deg) rotateY(20deg);  
}.element {  
}
```

Escalado 2D en CSS:

El escalado cambia el tamaño de un elemento usando la propiedad `transform` y las funciones:

- `scaleX(fx)`: escala horizontalmente por un factor fx.
- `scaleY(fy)`: escala verticalmente por un factor fy.
- `scale(fx, fy)`: escala simultánea en X e Y (atajo).
- `scale(fx)`: escala igual en ambos ejes (equivale a `scale(fx, fx)`).

Ejemplo:

```
.element {  
    transform: scale(2, 0.5); /* Doble ancho, mitad alto  
    */  
}
```

- ¿Cómo dibujar con CSS?

Dibujar con CSS, también llamado CSS Art, consiste en crear ilustraciones usando solo código HTML y estilos CSS, sin imágenes externas

Consejos clave:

Usa HTML semántico para estructurar el dibujo.

- Primeros pasos: Estructura HTML para dibujar con CSS

Para empezar a dibujar con CSS, elige un ejemplo sencillo, como un dibujo formado por formas geométricas básicas (por ejemplo, un diskette).

- Las variables CSS (`--variable`) ayudan a organizar y reutilizar valores fácilmente.

Deformaciones 2D en CSS

Las funciones `skewX()` y `skewY()` inclinan un elemento en 2D aplicando un ángulo sobre los ejes X o Y.

- `skewX(xdeg)`: inclina el elemento en el eje X.
- `skewY(ydeg)`: inclina el elemento en el eje Y.

Ejemplo:

```
.element {  
    transform: skewX(-5deg);  
    transform: skewY(25deg);  
    transform: skewX(5deg) skewY(15deg);  
}
```

Transformaciones 3D en CSS

CSS permite aplicar transformaciones 3D añadiendo el eje Z (profundidad) a las transformaciones 2D, usando la propiedad `transform` junto con funciones específicas.

Funciones principales:

Translación 3D:

- `translateZ(z)`: mueve el elemento en el eje Z.

Rotación 3D:

- `rotateZ(zdeg)`: rota sobre el eje Z (profundidad).
- `perspective(n)`: define la perspectiva para el efecto 3D.

Perspectivas en CSS

la perspectiva es clave para dar profundidad y realismo visual, simulando un punto de fuga.

backface-visibility

- Controla la visibilidad de la cara trasera del elemento 3D.

Ejemplo básico con animación 3D

```
.container {  
    width: 180px;  
    height: 180px;  
    position: relative;  
    transform-style: preserve-3d;  
    animation: spin 2s linear infinite;  
}  
.container img {  
    width: 100%;  
    height: 100%;  
    position: absolute;  
}
```

- Dos imágenes superpuestas giran 360°.
- `backface-visibility: hidden` en la imagen trasera evita que se muestre cuando está "de espaldas", simulando mejor el efecto 3D real.