CSS

Tags apuntes

Meter stylesheet:

```
<!--Externo-->
<head>
    <link rel="stylesheet" href="ruta/archivo.css">
</head>
<!--Interno-->
<head>
    <style>
        body {
            background-color: blue;
            }
    </style>
</head>
<!--En linea-->
<html>
    <body>
        <p style="color:blue">Hola que tal</p>
    </body>
</html>
```

Sintaxis básica css:

```
/* comentario
Multilinia */
p { /* p es el selector */
    color: red; /*color (propiedad) red (valor) */
}
```

Selectores

Selectores simples

Por el nombre de un elemento

```
p, h2{
    color: red;
{
```

Por el nombre de un id

```
/*tiene que ser único -> solo vale para ese elemento*/
#id_elemento {
    text-align: center;
}
```

Por la clase de un elemento

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

Selectores combinación:

Descendiente (div)

Selecciona todos los elementos descendientes del elemento especificado

Sintaxis: `selector_simple1 selector_simple2 {}`

```
div p {
    background-color: yellow;
}
```

```
<div>
    <p>Paragraph 1 in the div.</p>
    <section>
        <p>Paragraph 2 in the div.</p>
    </section>
</div>
<p>Paragraph 3. Not in a div.</p>
```

Hijo (>)

Selecciona todos los elementos que son hijos directos del elemento especificado

Sintaxis: `selector_simple1 > selector_simple2 {}`

```
div > p {
    background-color: yellow;
}
```

```
<div>
    <p>Paragraph 1 in the div.</p>
    <section>
        <p>Paragraph 2 in the div.</p>
    </section>
</div>
<p>Paragraph 3. Not in a div.</p>
```

Hermano adyacente (+)

Selecciona los elementos que son hermanos (tienen el mismo padre) y que esten inmediatamente a continuación. hermano del primero y del tipo del segundo

Sintaxis: `Selector_simple1 + Selector_simple2 {}`

```
div + p {
    background-color: yellow;
}
```

```
<p>Paragraph 5. Not in a div.</p>
<div>
    <p>Paragraph 1 in the div.</p>
    <section>
        <p>Paragraph 2 in the div.</p>
    </section>
</div>
<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>
```

Obligatorio justo debajo → en el siguiente ejemplo nada se subraya:

```
<div>
    <p>Paragraph 1 in the div.</p>
    <section>
        <p>Paragraph 2 in the div.</p>
```

```
    </section>
</div>
<h1>Heading</h1>
<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>
```

Hermano general (~)

Selecciona todos los hermanos que esten debajo

Sintaxis: `selector_simple1 ~ selector_simple2 {}`

```
div ~ p {
    background-color: yellow;
}
```

```
<p>Paragraph 5. Not in a div.</p>
<div>
    <p>Paragraph 1 in the div.</p>
    <section>
        <p>Paragraph 2 in the div.</p>
    </section>
</div>
<h1>Heading</h1>
<p>Paragraph 3. Not in a div.</p>
<h1>Heading2</h1>
<p>Paragraph 4. Not in a div.</p>
```

Selectores pseudoclases

Selecciona elementos enteros cumpliendo algunas condiciones

```
/*Uso*/
p:hover {
    color: blue;
}
/*--------*/
:hover /* Ratón por encima */
:focus /* Elemento que tiene el foco */
:first-child /* Primer hijo */
:last-child /* Último hijo */
```

Selectores pseudoelementos

Selecciona partes de elementos (primera linea de un párrafo), los de pseudoclases solo pueden coger el elemento entero

```
/*Uso*/
p::after {
    color: blue;
}
/*--------*/
::after /*Insertar alago antes*/
::before /*Insertar algo después*/
::first-letter /*Selecciona la primera letra*/
::first-line /*selecciona la primera linea*/
::selection /*La selección del usuario*/
```

Selectores de atributo

Simples

Seleccionar elementos con un atributo concreto

```css
a[target] {
    color: red;
}
```

```html
<a href="http://...">w3schools.com</a>
<a href="http://..." target="_blank">disney.com</a>
<a href="http://..." target="_top">wikipedia.org</a>
```

con valor

```css
a[target="_blank"]{ /*_blank se refería a nueva pestaña*/
    background-color: yellow; /*subrayar de amarillo xd*/
}
```

```html
<a href="http://...">w3schools.com</a>
<a href="http://..." target="_blank">disney.com</a>
<a href="http://..." target="_top">wikipedia.org</a>
```

Otros

```css
/*Selecciona un elemento con un atributo cuyo valor ...*/
a[target~="valor"]{ color: yellow; } /*contiene esa palabra*/
a[target|="valor"]{ color: yellow; } /*empieza por esa palabra*/
a[target^="valor"]{ color: yellow; } /*empieza de forma concreta*/
a[target$="valor"]{ color: yellow; } /*termina de forma concreta*/
a[target*="valor"]{ color: yellow; } /*contiene un valor específico*/
```

Declaraciones:

- Contenido → contenido de la caja como texto o imágenes
- Padding → área vacía alrededor del contenido → Es transparente
- Borde → Borde al rededor del padding y del contenido
- Margen → Area vacía fuera del borde → Es transparente

margin:

Espacio al rededor del elemento por fuera

→ no tiene efecto dentro de las tablas (con razón xd)

```css
p {
    margin-top: 100cm; /*Longitud*/
    margin-right: 100%; /*en relación con el ancho del elemento*/
    margin-bottom: auto;
    margin-left: inherit; /*heredado del pasdre*/
}
```

```css
p{
    margin: 100px 75px 50px 25px; /*top, rigth bottom left*/
}

p{
    margin: 100px 75px 50px; /*top, rigth, bottom y left=right*/
}

p{
    margin: 100px 75px; /*top y bottom, right y left*/ /*Por las esquinas lol*/
}

p{
    margin: 100px; /*top, rigth, bottom y left*/ /*Todos por igual*/
}
```

Se puede colapsar:

```html
<div style="margin-bottom: 30px;">Elemento 1</div>
<div style="margin-top: 20px;">Elemento 2</div>
<!--Se coge el margin más grande, pq los bordes coinciden xd, se separa 30, no suma-->
```

Padding

Igual que margin → pero es el espacio entre el borde y lo de dentro del elemento

También se puede colapsar

Heigth y width

Ancho y alto del contenido.

Luego habrá que sumar el padding, el borde y el margen

Mismos valores (% → con respecto al ancho del bloque, px, cm) que margin

none → no hay ni maximo ni mínimo

```css
img {
    max-heith: ;
    min-width: ;
}
```

Text

```css
p {
    color: ; /*Color del texto*/
    background-color: ;
    text-align: ;/*Alineación del texto (center, justify, right, left)*/
    direction: ; /* direccional del texto, ltr (left to right) o rtl*/
    vertical-align: ; /*baseline, top, midddle o bottom, alineacion vertical
    de un elemento en un texto o un elemento de una celda*/
    text-decoration: ; /*none, underline, overline, line-through*/
    text-transform: ; /*uppercase, lowercase, capitalize*/
    text-indent: ; /*Sangrado de la primera linea -> indicar cuanto (20px)*/
    letter-spacing: ; /* espaciado de los caracteres (px, cm, etc)*/
    line-height: ; /*altura de cada linea de texto*/
    word-spacing: ; /*espacio entre palabras*/
    white-space: ; /*Como se manejan los espacios (normal, nowrap (no permite saltos
    de linea), pre(preserva espacios y saltos de linea), pre-wrap(preserva
    espacios y permite saltos de linea autos), pre-line(preserva saltos de linea
    pero colapsa espacios))*/
    text-shadow: ; /*sombra -> x-offset, y-offset, blur-radius, color*/
}
p {
```

```
    text-shadow: 2px 2px 5px gray; /* Sombra con desplazamiento y difuminado */
}
```

Font

```
p{
    font-family: "Times New Roman", "Times", "serif";
    font-style: normal/italic/oblique;
    font-weight: normal/bold;
    font-variant: normal/small-caps;
    font-size
    font: font-style font-variant font-weight font-size/line-
height font-family;
}
```

Position

Tipo de posicionamiento usado para el elemento

```
div {
    position: static;/*Predeterminado xd -> colocado segun el flujo normal de la pag*/
    position: relative; /*Relativo a su posicion por defecto*/
    position: fixed; /*no se mueve al hacer scroll*/
    position: absolute; /*relativo al ancestro más cercano*/
    position: sticky; /*se mueve al hacer scroll, se queda pegado*/
}
/*USar tambien para posicionar*/
div{
    position: top;
    position: right;
    position: bottom;
    position: left;
    z-index: 2; /*para superponer cosas xd*/
}
```

Importante → unidades

pt → 1pto = 1/72 inch y 1inch = 2,54 cm

em → relativo al tamaño fuente → 2em es dos veces el tamaño de fuente actual

rem → relativo al tamaño de fuente del elemento raíz, suele equivaler a 16 px

ch → relativo al tamaño del 0

% relativo al tamaño del padre

Colores:

- Nombres: Lista de los nombres de colores
- RGB(Red, Green, Blue): rgb(255, 99, 71)
- HEX: #ff6347
- HSL (Hue, Saturation, Lightness): hsl(9, 100%, 54%)
- RGBA (RGB + Alpha): rgba(255, 99, 71, 0.5)
- HSLA (HSL + Alpha): hsla(9, 100%, 64%, 0.5)

Specifity:

Cuanto más especifico un elemento, mayor propiedad → id va por delante de class

id > class > element

En caso de igualdad, el que aparezca después en el archivo

!important

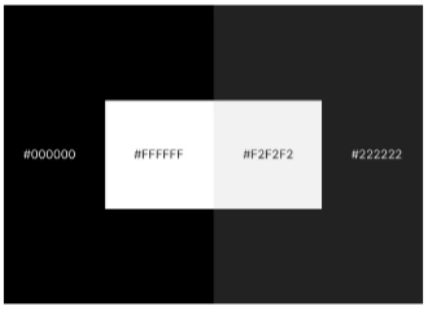
Si añadimos !important → gana a todo

```css
#myid {
background-color: blue;
}
.myclass {
background-color: gray;
}
p {
background-color: yellow !important;
}
```

```html
<p id="myid" class="myclass">
    Texto de prueba
</p>
```

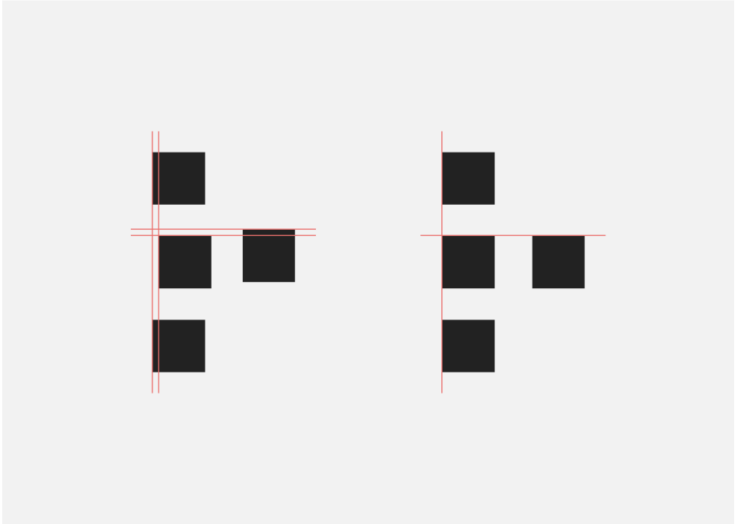
Recomendaciones de diseño

no usar negro ni blanco → casi

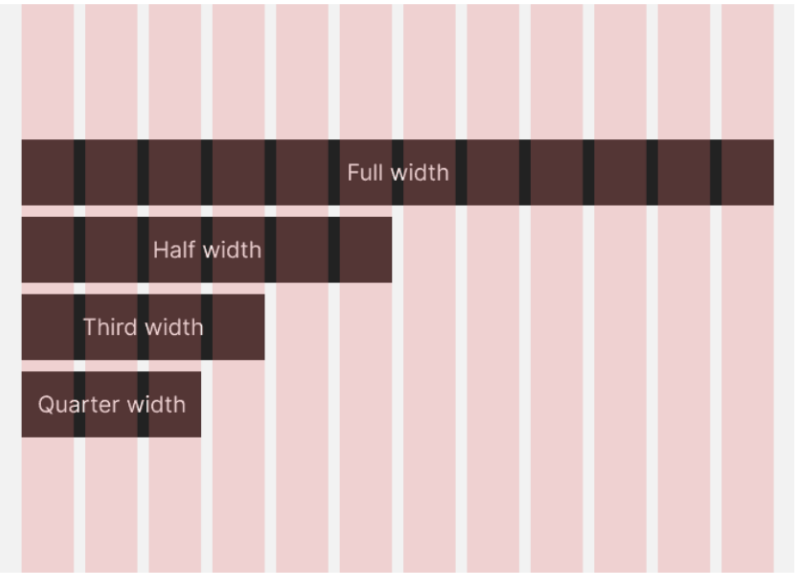


Alto contraste para elementos importantes

Elementos deben estar alineados



Usar 12 columnas (columnas de 1, 2,3, 4 y 6)



Lineas de unos 70 caracteres para mejorar legibilidad

Usar paletas de colores accesibles (+ 4% de la población es daltonica)

https://htmlcheatsheet.com/css/

Ejemplos → mirar diapositivas

Columnas

- De igual tamaño

```
<div class="column">
  <h2>Título</h2>
  <p>Lorem ipsum (…)</p>
</div>
<div class="column">
  <h2>Título</h2>
  <p>Lorem ipsum (…)</p>
</div>
```

```
.column {
  float: left;
  width: 50%;
}
```

Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Título

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

ojo que si añadimos padding o margin se nos hacen dos párrafos, a tomar por saco las columnas

Responsive web design

¿Qué es?

- Es una técnica para que una página web **se vea bien en todos los dispositivos**, desde ordenadores hasta smartphones.
- Utiliza exclusivamente **HTML** y **CSS** para lograr adaptabilidad.
- Evita el uso de **scroll horizontal**, mejorando la experiencia de usuario.

Elementos clave del Responsive Web Design

1. Viewport

- La etiqueta <meta> para el viewport asegura que el contenido se adapta correctamente al tamaño de la pantalla del dispositivo.
- Ejemplo de implementación:

Usar etiqueta en el heading:

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

Media Queries

- Permiten aplicar estilos diferentes según características del dispositivo, como:
- **Ancho y alto** del dispositivo.
- **Orientación** (horizontal o vertical).
- **Resolución**.
- Se definen con la regla @media y se colocan normalmente al final del archivo CSS.

Sintaxis básica:

```
@media mediatype and (mediafeature) {
    /* Estilos específicos */
}
```

- **mediatype**: Generalmente screen para dispositivos visuales.
- **mediafeature**: Por ejemplo, max-width, min-width.

Ejemplo práctico:

```
.column {
    float: left;
    width: 25%;
    padding: 20px;
}
@media screen and (max-width: 992px) {
    .column {
        width: 50%; /* Cambia a 2 columnas */
    }
}
@media screen and (max-width: 600px) {
    .column {
        width: 100%; /* Cambia a 1 columna */
    }
}
```

3. Grillas y sistemas de diseño fluidos

- Uso de propiedades CSS como grid o flexbox para organizar elementos de manera dinámica.
- Los anchos y alturas se definen en porcentajes para garantizar flexibilidad.

Grid

¿Qué es?

- Sistema de diseño en dos dimensiones (filas y columnas) recomendado para dividir páginas en regiones.
- Evolución de las tablas (<table>), que **no deberían usarse para el diseño de layouts**.

Características:

- Declarar un contenedor con display: grid.
- Versátil y flexible para estructurar layouts complejos.
- Permite especificar posiciones exactas para los elementos en filas y columnas.

No soportado por internet explorer

Ejemplo básico:

```
.container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr; /* Tres columnas iguales */
    gap: 10px; /* Espacio entre columnas y filas */
}
.item {
```

```
    background-color: lightblue;
}
```

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
</div>
```

Flexbox

¿Qué es?

- Sistema de diseño en una sola dimensión: organiza elementos en **filas** o **columnas**.
- Ideal para layouts más simples o alineación y distribución de elementos dentro de un contenedor.

Características:

- Declarar un contenedor con display: flex.
- Soporte para **alineación** y **distribución dinámica** de elementos.
- Los elementos pueden **flexionar** (crecer o reducirse) según el espacio disponible.

Ejemplo básico:

```
.container {
    display: flex;
    justify-content: space-between; /* Distribuir elementos con espacio entre ellos */
    align-items: center; /* Alinearlos verticalmente al centro */
}
.item {
    background-color: lightcoral;
    padding: 10px;
}
```

```
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

No soportado por Internet Explorer.

Sass

¿Qué es?

- **Syntactically Awesome Stylesheets**: un preprocesador CSS diseñado para proyectos grandes.
- Basado en **Ruby**.
- **Características clave:**
- Permite usar:
- **Variables**: Definir colores, tamaños o valores reutilizables.
- **Nesting**: Anidar reglas para representar jerarquías en el CSS.
- **Módulos**: Dividir estilos en varios archivos y combinarlos.
- **Mixins**: Fragmentos reutilizables de código CSS.

Less

¿Qué es?

- **Leaner Style Sheets**: un preprocesador CSS inspirado en Sass, pero basado en **JavaScript**.
- **Características clave:**
- Similar a Sass, permite usar:
- **Variables**.
- **Nesting**.
- **Módulos**.
- Tiene un enfoque minimalista.

CSS Frameworks

Características:

- Librerías
- Facilitan el desarrollo web
- Hacen uso de CSS
- Pueden tener JS
- Resetean la hoja de estilos
- Layout en forma de grid (responsive)
- Mobile first
- Tipografía
- Fuentes
- Iconos

Desventajas

- Aprender a trabajar con el framework
- Mucho código extra que no usaras
- Similitud de las webs que lo usen
- Dependencia del framework

Ejemplos:

1. **Bootstrap**

- El más popular.
- Proporciona grillas, componentes (botones, formularios, alertas), y diseño responsive.
- Fácil prototipado.

2. **Tailwind CSS**

- Framework de **utilidades primero**.
- Ligero y escalable.
- Permite crear interfaces personalizadas rápidamente.

3. **Bulma**

- Framework modular y responsivo.
- Facilita el desarrollo con estilos limpios y componentes sencillos.

4. **Foundation**

- Orientado a **mobile-first**.
- Altamente personalizable con herramientas de diseño responsivo.

5. **Materialize**

- Basado en **Material Design** de Google.
- Enfoque en la experiencia de usuario (UX).
- **No actualizado desde 2018.**

Semantic UI

- Diseñado con un lenguaje natural, orientado a la **semántica** del diseño.

7. **UIKit**

- Ligero y minimalista.
- Ideal para diseños simples y modulares.

Bootstrap

framework de CSS

```
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"crossorigin="anonymous">
</head>
```

Contenido:

- Archivos .min
  - Versión reducida del archivo original
  - Se eliminan saltos de línea y espacios innecesarios
  - Se usa en producción
  - bootstrap.css (200KB) → bootstrap.min.css (159KB)
- Archivos .map
  - Source maps
  - Para trabajar con herramientas de desarrollo
  - Conversión que se ha usado para generar el archivo minimizado (.min)
  - Más información
- Archivos rtl (Right to Left): soporte para escritura de derecha a izquierda
  CS

Elementos:

- Containers
- Grid
- Alerts
- Carrousel → no usar, nadie lo usa
- Forms
- NavBars
- Popovers, Barras de progreso, Botones, iconos ....

Elemento display:

1. block
   - Hace que el elemento se comporte como un **bloque**.
   - Ocupa todo el ancho disponible (por defecto).
   - Empieza en una nueva línea.

   Ejemplo:

   ```
   div {
       display: block;
   }
   ```

   Ejemplo visual:

   ```
   [Elemento 1]
   [Elemento 2]
   ```

2. inline
   - Hace que el elemento se comporte como un **elemento en línea**.
   - No empieza en una nueva línea.
   - Solo ocupa el ancho necesario.

   Ejemplo:

   ```
   span {
       display: inline;
   }
   ```

   Ejemplo visual:

   ```
   [Elemento 1][Elemento 2]
   ```

3. inline-block
   - Combina características de block e inline:

     Se comporta como un elemento **en línea**, pero permite aplicar propiedades de tamaño como width y height.

   Ejemplo:

   ```
   div {
       display: inline-block;
       width: 100px;
       height: 50px;
   }
   ```

   Ejemplo visual:

   ```
   [Elemento 1][Elemento 2]
   ```

4. none
   - Oculta el elemento por completo, **eliminándolo del flujo del documento**.
   - No ocupa espacio en la página.

   Ejemplo:

   ```
   div {
       display: none;
       width: 100px;
       height: 50px;
   }
   ```

   Ejemplo visual:

   ```
   
   ```

5. flex

   Activa el Flexbox (mirar arriba)

6. grid

   Activa GRID (mirar arriba)