Unidad 2: CSS

2.1.Introducción a la Creación de un Archivo CSS

Un archivo CSS (Cascading Style Sheets) es una herramienta fundamental para estilizar páginas web, separando el contenido HTML de su apariencia visual. Al crear un archivo CSS, es importante seguir ciertos pasos y buenas prácticas para asegurar que el proceso sea claro, eficiente y escalable.

Sintaxis de CSS (propiedad y valor)

Sintaxis Básica

La sintaxis de CSS se compone de selectores y declaraciones. Un selector define a qué elementos se aplican los estilos, y una declaración incluye una propiedad y un valor. Las declaraciones se agrupan dentro de bloques de declaración.

```
selector {
    propiedad:valor;
}

p {
    color: blue;
}
```

Primeros pasos para crear un archivo CSS:

1. **Creación del archivo:**
    - Usa un editor de texto o un entorno de desarrollo integrado (IDE) como Visual Studio Code o Sublime Text.
    - Guarda el archivo con la extensión `.css` (por ejemplo, `estilos.css`).
2. **Estructura básica de un archivo CSS:**

- El archivo CSS se compone de reglas. Cada regla incluye un **selector** que apunta a los elementos HTML que quieres estilizar, y un bloque de **declaraciones** encerrado entre llaves `{ }`.

- Una declaración se forma con una **propiedad** (lo que deseas cambiar, como color o tamaño) y un **valor** (el ajuste específico que quieres aplicar).**Ejemplo:**

```
h1 {
    color: blue;
    font-size: 24px;
}
```

3. **Conexión al archivo HTML:**

    - Enlaza tu archivo CSS al documento HTML utilizando la etiqueta `<link>` dentro de la sección `<head>`:

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

      Recuerda, como vimos en la clase pasada, **las rutas dependen del lugar en donde te encuentres y del archivo al que quieras dirigirte**.

      Si tu CSS está en una carpeta llamada css y tu HTML en la raíz:

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

      Consejo: pensá en las rutas como un **camino de carpetas**:

      - ./ → para buscar dentro de la carpeta actual.
      - ../ → para salir de la carpeta actual y subir un nivel.

Buenas prácticas iniciales:

- **Organización:** Mantén tu código CSS ordenado y usa comentarios (`/* texto aquí */`) para explicar secciones específicas.
- **Consistencia:** Sigue un formato uniforme, como usar siempre puntos y comas al final de cada declaración y tabulaciones o espacios consistentes para la sangría.
- **Uso de selectores claros:** Evita selectores excesivamente generales o complicados. Empieza con selectores simples como etiquetas (`h1`, `p`) o clases (`.mi-clase`).
- **Manejo de colores y tamaños:** Usa formatos estándar como códigos hexadecimales para colores (`#ff5733`) y unidades relativas como `em` o `rem` para tamaños, ya que ofrecen mayor flexibilidad y adaptabilidad.

2.2.Concepto de Padres e Hijos en CSS

Analogía con las Mamushkas

Imagina que los elementos HTML son como las famosas muñecas rusas conocidas como mamushkas. Cada muñeca puede contener otra más pequeña en su interior, y así sucesivamente. De manera similar, en HTML, los elementos pueden contener otros elementos dentro de ellos, creando una relación de padres e hijos.

Elementos Padres e Hijos

- **Elemento Padre**: Es un elemento que contiene a uno o más elementos dentro de él.
- **Elemento Hijo**: Es un elemento que está contenido dentro de otro elemento.

Ejemplo:

```
<body> <!--Padre del header -->
    <header> <!--Hijo del <body>, Padre del <nav> -->
        <nav> <!--Hijo del <header>, Padre del <ul> -->
            <ul> <!--Hijo del <nav>, Padre del <li> -->
                <li> <!--Hijo del <nav>, Padre del <a> -->
                    <a href=""></a> <!--Hijo del <li>-->
                </li>
            </ul>
        </nav>
    </header>
</body>
```

- <body> es el padre de <header>.
- <header> es hijo de <body> y padre de <nav>.
- <nav> contiene una lista <ul>, que a su vez contiene varios <li>, y dentro de cada <li> hay enlaces <a>.

Este encadenamiento de elementos permite construir la jerarquía lógica de una página web.

Nesting en HTML: la jerarquía visible

Esta relación de **padres e hijos** se refleja en la forma en que escribimos el código. Al aplicar **indentación** (sangría), cada elemento hijo se escribe un nivel hacia adentro respecto de su padre.

Esto se logra fácilmente con la tecla **Tab**, que desplaza la línea hacia adentro, mostrando visualmente la jerarquía. De este modo, la jerarquía queda **visible y organizada** en el archivo: es más fácil leerlo, entender cómo se relacionan los elementos y detectar errores si falta cerrar una etiqueta.

Podemos pensar que la indentación en HTML es como **abrir una mamushka tras otra**: cada nivel hacia adentro nos muestra la estructura que está contenida en el anterior.

Ejemplo:

```
<ul> <!--Padre->
    <li></li> <!--Hijo->
</ul>
```

Herencia y cascada en CSS

En este video, explicaremos desde la práctica cómo funciona la **herencia en CSS** y cómo los estilos se aplican en **cascada**. Veremos más ejemplos de cómo los **elementos hijos heredan propiedades** de sus padres y cómo los navegadores aplican las reglas CSS en orden descendente.Esperamos que este video te ayudade a entender cómo funcionan la **herencia y la cascada en CSS**.

En el video trabajaste ejemplos prácticos de **herencia y cascada en CSS**, viendo cómo los estilos se transmiten de padres a hijos y cómo se aplican en orden de prioridad. A continuación encontrarás un desarrollo teórico que refuerza esas ideas, con explicaciones y ejemplos adicionales para que tengas una referencia clara mientras practicas en tus propios proyectos.

Herencia

En **CSS**, los estilos aplicados a un **elemento padre** pueden transmitirse automáticamente a sus **elementos hijos**. Esto se llama **herencia**. No todas las propiedades CSS son heredables, pero aquellas que afectan el texto, como `color` y `font-family`, sí lo son.

Por ejemplo, si aplicamos un color y una tipografía al `<section>`, todos los textos dentro de él heredarán esos estilos, a menos que se indique lo contrario.

Ejemplo:

HTML

```
<section>
    <article>
        <h2>Título</h2>
        <p>Lorem ipsum dolor sit amet...</p>
    </article>
```

```
</section>
```

CSS

```
section {
  color: blue;
  font-family: Arial, sans-serif;
}
```

En este caso:

- El **<h2>** y el **<p>** **heredan** el color azul y la fuente Arial del <section>.
- No hicimos nada extra: la herencia ocurre de forma automática.

Cascada en CSS

La cascada es el principio que determina qué estilos se aplican a un elemento cuando hay conflictos. CSS asigna un peso a cada regla según su origen (inline, interno, externo), especificidad y orden de aparición.

Especificidad

La especificidad se calcula con base en el tipo de selectores usados. Los estilos inline tienen mayor especificidad que los selectores de ID, clases y elementos.

Orden de Aparición

Si dos reglas tienen la misma especificidad, la última que aparece en el código tendrá prioridad.

Ejemplo:

HTML

```
<p id="texto" class="parrafo">Hola mundo</p>
```

CSS

```
/* Selector de elemento (menos específico) */
p {
  color: blue;
}
/* Selector de clase */
.parrafo {
  color: green;
}
/* Selector de ID (más específico) */
#texto {
  color: red;
```

```
}
```

Resultado: El texto se mostrará en **rojo**, porque el selector de **ID** tiene mayor especificidad que el de clase o elemento.

Práctica estos conceptos para mejorar tus habilidades y asegurar que tus estilos se apliquen de manera efectiva en tus páginas web.

Sobrescribir estilos en los elementos hijos

A veces queremos que un **hijo** se vea distinto al resto de los elementos contenidos en su **padre**. Para lograrlo usamos **selectores más específicos**, que tienen mayor peso que la herencia y permiten sobrescribir los estilos generales.

Ejemplo de Estilos Específicos

HTML

```html
<section>
    <article>
        <h2>Título</h2>
        <p>Lorem ipsum dolor sit amet...</p>
    </article>
</section>
```

CSS

```css
/* Estilo aplicado al elemento padre <section> */
section {
color: blue;
font-family: Arial, sans-serif;
}
/* Estilo específico para el elemento hijo <h2> */
section h2 {
color: red;
font-size: 24px;
}
/* Estilo específico para el elemento hijo <p> */
section p {
font-size: 16px;
}
```

En este caso:

- El `<h2>` es **hijo de `<section>`**, hereda la fuente Arial, pero **su color pasa a rojo** porque el selector `section h2` tiene mayor especificidad que el estilo general.
- El `<p>` también es hijo de `<section>`: mantiene el color azul heredado, pero recibe un **nuevo tamaño de fuente** definido solo para él.

Así, la **relación padre-hijo en HTML** se refleja también en CSS: el padre transmite estilos generales, y el hijo puede sobrescribirlos si usamos reglas más específicas.

-

Conclusión

Entender la relación de padres e hijos en CSS es esencial para aplicar estilos de manera eficiente y organizada en un documento HTML. La herencia de estilos permite mantener la consistencia visual, mientras que los selectores específicos para elementos hijos ofrecen flexibilidad para personalizar la apariencia de cada parte del contenido. Al igual que las mamushkas, cada elemento HTML puede contener otros elementos, y CSS te permite controlar cómo se ven y se comportan estos elementos anidados.

2.3.Diferentes formas de insertar CSS en un documento HTML

CSS (Cascading Style Sheets) se puede aplicar a un documento HTML de varias formas: mediante una vinculación externa, una inserción interna con la etiqueta `<style>`, y el uso de estilos en línea. A continuación, se describen estos métodos, sus ventajas y desventajas, y se muestran ejemplos de cómo aplicarlos en diferentes contextos.

1. Vinculación externa

La vinculación externa consiste en enlazar un archivo CSS separado al documento HTML mediante la etiqueta `<link>` dentro del `<head>`.

Ejemplo:

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

Ventajas:

- **Separación de Contenidos**: Mantiene el HTML limpio y separado del CSS.
- **Reutilización**: Un solo archivo CSS puede ser reutilizado en múltiples páginas web.
- **Mantenimiento**: Facilita la actualización de estilos, ya que los cambios en el archivo CSS afectan a todas las páginas que lo utilizan.

Desventajas:

- **Carga Adicional**: Requiere una solicitud HTTP adicional para cargar el archivo CSS, lo que puede afectar el tiempo de carga de la página.

2. Inserción Interna con la Etiqueta `<style>`

La inserción interna coloca el CSS directamente dentro del documento HTML utilizando la etiqueta **`<style>`** dentro del **`<head>`**.

Ejemplo:

```
<head>
    <style>
        body {
        background-color: lightblue;
        }
        h1 {
        color: navy;
        margin-left: 20px;
        }
    </style>
</head>
```

Ventajas:

- **Menos Solicitudes HTTP**: No requiere una solicitud adicional para cargar los estilos.
- **Conveniencia**: Útil para estilos específicos de una sola página.

Desventajas:

- **No Reutilizable**: Los estilos definidos de esta manera no se pueden reutilizar en otras páginas.
- **Mantención Compleja**: Si se utilizan muchas reglas de estilo, el documento HTML puede volverse difícil de manejar y mantener.

3. Estilos en Línea

Los estilos en línea se aplican directamente a los elementos HTML mediante el atributo **`style`**.

Ejemplo:

```
<body>
    <h1 style="color: navy; margin-left: 20px;">Título</h1>
    <p style="color: green;">Este es un párrafo.</p>
</body>
```

Ventajas:

- **Específico y Rápido**: Ideal para aplicar estilos rápidos y específicos a un solo elemento.
- **No Requiere `<head>`**: Puede ser usado directamente en el contenido del documento.

Desventajas:

- **No Reutilizable**: Los estilos deben ser escritos repetidamente para cada elemento.
- **Difícil de Mantener**: Los estilos en línea dispersos pueden hacer que el HTML sea difícil de leer y mantener.
- **Especificidad Alta**: Los estilos en línea tienen una especificidad alta, lo que puede complicar el uso de CSS externo o interno.

Comparación de Métodos:

Vinculación Externa:

- **Pros**: Reutilización, mantenimiento sencillo, separación de contenido.
- **Contras**: Requiere solicitudes HTTP adicionales.

Inserción Interna:

- **Pros**: Menos solicitudes HTTP, conveniente para estilos específicos de una página.
- **Contras**: No reutilizable, difícil de mantener en grandes cantidades.

Estilos en Línea:

- **Pros**: Aplicación rápida y específica, no requiere `<head>`.
- **Contras**: No reutilizable, difícil de mantener, alta especificidad.

Conclusión

Elegir la forma adecuada de insertar CSS en un documento HTML depende del contexto y las necesidades del proyecto. La vinculación externa es ideal para proyectos grandes con múltiples páginas, la inserción interna es útil para estilos específicos de una sola página, y los estilos en línea son mejores para ajustes rápidos y específicos. Comprender las ventajas y desventajas de cada método permite aplicar CSS de manera eficiente y organizada.

2.4.Selectores y propiedades

Selectores

Bienvenidos al video sobre **"Selectores de CSS"**. En este video, detallaremos los diferentes tipos de **selectores en CSS**, incluyendo **selectores por ID, clase y etiqueta**. Explicaremos cómo utilizar cada tipo de selector y mostraremos ejemplos prácticos de su

aplicación en documentos HTML. Esperamos que este video ayude a entender mejor cómo utilizar los diferentes tipos de **selectores en CSS**

Ahora que en el video viste una introducción a los **selectores en CSS** y cómo aplicarlos en ejemplos prácticos, continuamos con una explicación escrita que refuerza esos conceptos. A continuación encontrarás un repaso de los diferentes tipos de selectores —por etiqueta, clase, ID y atributos— junto con ejemplos de código que te permitirán profundizar en lo aprendido y tenerlos como referencia mientras practicas.

Tipos de Selectores

- **Selector de Elemento**: Selecciona todos los elementos de un tipo específico.
```
p {
    color: blue;
}
```
- **Selector de Clase**: Selecciona elementos con una clase específica.
```
.clase {
    color: green;
}
```
- **Selector de ID**: Selecciona un elemento con un ID específico.
```
#id {
    color: red;
}
```

- **Selectores de Atributo**: Selecciona elementos basados en un atributo. Por ejemplo, para estilizar todos los campos de entrada de texto:
```
input[type="text"] {
    background-color: yellow;
    border: 1px solid #ccc;
}
```

Práctica estos conceptos para mejorar tus habilidades y asegurar que tus estilos se apliquen de manera efectiva en tus páginas web.

Propiedades

En este video, exploraremos diversas **propiedades de CSS**, enfocándonos en el **color**, los **estilos de texto** y las **propiedades de fondo**. Veremos ejemplos prácticos de cómo cambiar el color del texto, aplicar diferentes estilos a las palabras y utilizar imágenes de fondo en CSS. También abordaremos las **unidades de medida absolutas y relativas**, y la

importancia de los **resets CSS**. Esperamos que este video te ayudade a entender cómo utilizar diversas **propiedades de CSS** para estilizar tus páginas web.

En el video conociste cómo aplicar distintas **propiedades de CSS** para dar estilo a tus páginas: color, tipografía y fondos, junto con ejemplos prácticos. Ahora, reforzaremos esos conceptos con un repaso teórico que te servirá como guía. Verás cómo funcionan las propiedades más usadas, la diferencia entre unidades absolutas y relativas, y por qué los **resets CSS** son importantes para mantener coherencia en el diseño entre diferentes navegadores.

Propiedades en CSS

En CSS, las **propiedades** son las instrucciones que definen cómo se verá un elemento HTML. Cada propiedad se escribe dentro de una declaración, acompañada de un **valor** que le indica al navegador el estilo específico a aplicar.

Propiedades de color

- **color:** define el color del texto.
```
p {
  color: blue;
}
```
- **background-color:** establece el color de fondo del elemento.

```
div {
  background-color: lightgray;
}
```

Los colores pueden escribirse en distintos formatos:

- Hexadecimal: `#ff0000` (rojo).
- RGB: `rgb(255, 0, 0)`.
- HSL: `hsl(0, 100%, 50%)`.

Propiedades de texto

- **font-family:** define la tipografía.

```
h1 {
  font-family: Arial, sans-serif;
}
```

- **font-size:** establece el tamaño de la fuente.

```
p {
  font-size: 16px;
```

```
}
```

- **font-weight**: controla el grosor del texto (`normal`, `bold`, `lighter`, o valores numéricos como `400`, `700`).
- **text-align**: alinea el texto (`left`, `right`, `center`, `justify`).
- **text-decoration**: agrega o elimina decoraciones como subrayado.
- **list-style:** define el estilo de las viñetas en listas (`disc`, `circle`, `square`, `none`).

Propiedades de fondo

- **background-image:** permite colocar una imagen de fondo.

```
body {
  background-image: url("img/fondo.jpg");
}
```

- **background-repeat**: controla si la imagen se repite (`repeat`, `no-repeat`).
- **background-size**: ajusta el tamaño de la imagen (`cover`, `contain`, valores en px o %).

Unidades de medida

- **Absolutas**: siempre tienen el mismo valor sin importar el contexto. Ejemplo:
  - `px` (píxeles).
  - `cm`, `mm`, `in` (más raras en web).

- **Relativas**: dependen del contexto o del tamaño del elemento padre. Ejemplo:
  - `em`: relativo al tamaño de fuente del elemento.
  - `rem`: relativo al tamaño de fuente de la raíz (`html`).
  - `%`: relativo al tamaño del elemento padre.
  - `vw`, `vh`: relativo al ancho o alto de la ventana gráfica (viewport).

Consejo: en diseño web moderno se recomienda usar **unidades relativas** (`rem`, `em`, `%`, `vw`) para lograr diseños más flexibles y adaptables a distintos dispositivos.

Resets CSS

Cada navegador aplica estilos por defecto (márgenes, tamaños de texto, etc.), lo que puede generar inconsistencias entre Chrome, Firefox, Edge, etc.

Un **reset CSS** elimina esos estilos predeterminados, asegurando que el diseño parta desde una base uniforme.

Ejemplo simple de reset:

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

El símbolo * en CSS se conoce como **selector universal**.

- **Significa "todos los elementos"**: aplica las reglas que escribas a **absolutamente todos los elementos HTML del documento** (párrafos, encabezados, divs, imágenes, formularios, etc.).

- Es muy útil cuando querés establecer un punto de partida común para todo el sitio, como en el resets.

¡Importante! No es necesario memorizar todas las propiedades; puedes buscarlas en la web a medida que avances y necesites nuevas, mientras desarrollas tu experiencia.

Conclusión

Las propiedades en CSS te permiten controlar desde lo más básico (colores y texto) hasta aspectos más complejos (fondos, tamaños y adaptabilidad). Combinadas con buenas prácticas como el uso de unidades relativas y resets CSS, son la clave para construir sitios web consistentes, atractivos y funcionales.

2.5.¿Cómo utilizar fuentes externas?

En este video, explicaremos cómo aplicar diferentes **estilos tipográficos en CSS**, incluyendo la descarga y uso de **fuentes personalizadas**. Veremos ejemplos de cómo **vincular fuentes externas** y aplicar estilos tipográficos específicos a los elementos HTML.

Esperamos que este video te haya ayudado a entender mejor cómo aplicar estilos tipográficos en CSS y cómo utilizar **fuentes personalizadas** en tus proyectos web.

Practica estos conceptos para mejorar la tipografía y el diseño de tus páginas web.

2.6.Recomendaciones y Buenas Prácticas

1. **Mantén el Código CSS Organizado**

    - Usa comentarios para explicar secciones del CSS.
    - Utiliza nombres de clases e IDs descriptivos y coherentes con el contenido o función de los elementos.

2. **Usa una Hoja de Estilo Externa**

Esto facilita el mantenimiento y la reutilización de los estilos en múltiples páginas.

3. **Minimiza el Uso de Estilos en Línea**

   Prefiere usar hojas de estilo externas o internas para mantener el HTML limpio y estructurado.

4. **Evita la Sobrespecificidad**

   No hagas los selectores más específicos de lo necesario, ya que puede dificultar la sobrescritura de estilos y el mantenimiento del código.

Conclusión

Dominar la sintaxis básica de CSS, comprender la herencia y la cascada, y saber utilizar selectores de manera eficiente son habilidades esenciales para cualquier desarrollador web. Incorporar buenas prácticas y aprovechar selectores como los de atributo mejorará la organización, claridad y efectividad del código CSS, optimizando el mantenimiento y la escalabilidad de tus proyectos.