PROGRAMACION 3

julio 2025 - **apuntes clase 2**

opazo maria luz

Atajos Generales en CMD y PowerShell

atajos:

Navegación y Edición

- **Ctrl + C** → Interrumpir (matar) un proceso en ejecución.
- **Ctrl + V** → Pegar texto (en versiones modernas de CMD/PowerShell).
- **Ctrl + A** → Seleccionar todo el texto en la línea actual.
- **Ctrl + ← / →** → Moverse entre palabras.
- **Ctrl + M** → Entrar en modo "Mark" para seleccionar texto (similar a Shift + Flechas).
- **Esc** → Borrar la línea actual.

Historial de Comandos

- **F3** → Repetir el último comando.
- **F7** → Mostrar historial de comandos (en CMD).
- **↑ / ↓** → Navegar por el historial de comandos.
- **Ctrl + R** → Buscar en el historial (en PowerShell).

Gestión de Ventana

- **Alt + Enter** → Entrar/Salir de pantalla completa (en CMD clásico).
- **Ctrl + Shift + C / V** → Copiar/Pegar en **Windows Terminal**.

Atajos Específicos de PowerShell

- **Tab** → Autocompletar comandos y nombres de archivos.
- **Ctrl + D** → Cerrar la sesión de PowerShell (similar a `exit`).
- **Ctrl + L** → Limpiar la pantalla (`Clear-Host`).
- **Ctrl + Shift + ← / →** → Seleccionar palabras completas.

Atajos en Windows Terminal (Moderno)

- **Ctrl + Shift + T** → Abrir nueva pestaña.
- **Ctrl + Shift + W** → Cerrar pestaña actual.
- **Ctrl + Tab** → Cambiar entre pestañas.
- **Alt + ↑ / ↓** → Moverse entre paneles (si hay splits).
- **Ctrl + Shift + + / -** → Aumentar/Disminuir tamaño de fuente.

Diferencias clave entre CMD y PowerShell:

- **CMD** es más limitado en atajos y funcionalidad.
- **PowerShell** tiene más características, como autocompletado con **Tab** y búsqueda en historial (**Ctrl + R**).
- **Windows Terminal** (recomendado) soporta pestañas, splits y personalización.

Repaso:

mkdir + nombre

crea una carpeta con su nombre

ni index.html

Crear archivo (nombre “index” extensión “html”)

code .

abre el visual studio

preguntas de clase:

ejemplo:

- cuando escribimos un libro primero lo tenemos que mandar el manuscrito a un editor antes de mandar a la imprenta para que lo lea
en HTML, nosotros seriamos los escritores, y nuestro editor quien sería?
en html el que nos lee es el navegador
- importante: cuando trabajemos con html NO PONEMOS ESTILO a nada, dejamos simplemente la estructura SOLA por el momento
- si yo escribo texto dentro del body SIN colocar alguna etiqueta, o en cualquier parte de mi html, cuando corra ese archivo va a mostrar la palabra IGUAL porque html, es un archivo de texto plano
- el JSON conecta el backend con el frontend

Estructura / composición basica de html

index.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <meta http-equiv="X-UA-Compatible" content="ie=edge">
7       <title>Document</title>
8   </head>
9   <body>
10
11  </body>
12  </html>
13
```

el idioma en el que esta creada, si usamos español entonces “ES”

el tipo de carácter que acepta en el documento (hay un utf - 6)

define que se adapta a la pantalla

titulo

Estructura basica de una web:

- Header
- Main
- footer

Cómo renderiza los elementos el navegador ?
De arriba para abajo de izquierda a derecha

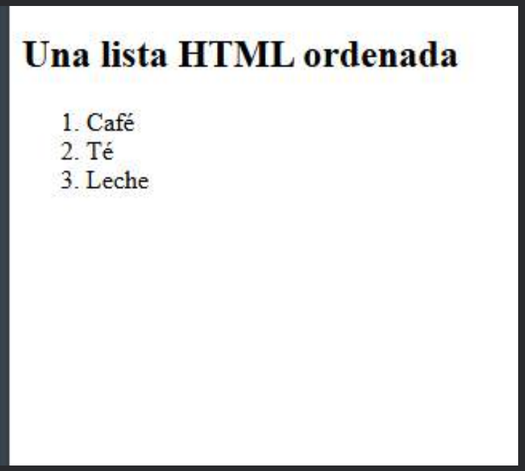
listas: ordenadas y desordenadas

```
<!DOCTYPE html>
<html>
< cuerpo >

< h2 > Una lista HTML ordenada </ h2 >

< ol >
  < li > Café </ li >
  < li > Té </ li >
  < li > Leche </ li >
</ ol >

</ cuerpo >
</html>
```

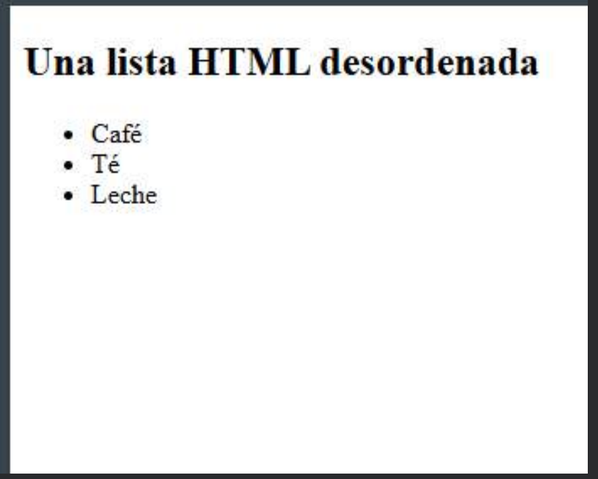



```
<!DOCTYPE html>
<html>
< cuerpo >

< h2 > Una lista HTML desordenada </ h2 >

<ul>
  < li > Café </ li >
  < li > Té </ li >
  < li > Leche </ li >
</ul>

</ cuerpo >
</html>
```



¿Qué son los elementos semánticos?

Un elemento semántico describe claramente su significado tanto para el navegador como para el desarrollador.

Ejemplos de elementos **no semánticos** : `<div>` y `<span>` - No dice nada sobre su contenido.

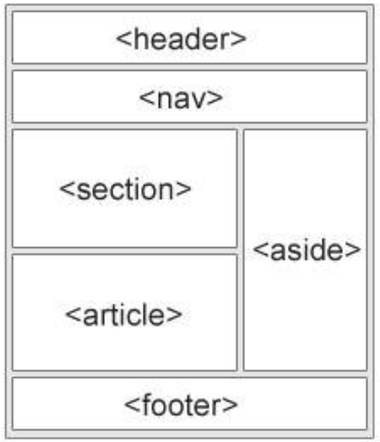
Ejemplos de elementos **semánticos** : `<img>`, `<table>`, y `<article>` - Define claramente su contenido.

Elementos semánticos en HTML

Muchos sitios web contienen código HTML como: para indicar navegación, encabezado y pie de página.

En HTML hay varios elementos semánticos que se pueden utilizar para definir diferentes partes de una página web:

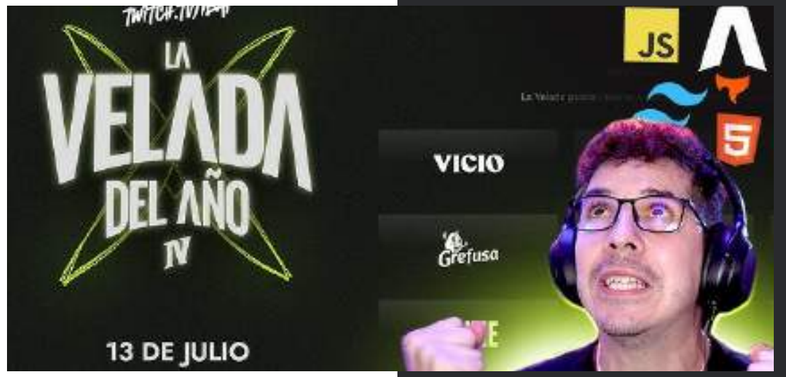
- <artículo>
- <aparte>
- <detalles>
- <figcaption>
- <figura>
- <pie de página>
- <encabezado>
- <principal>
- <marca>
- <navegación>
- <sección>
- <resumen>
- <tiempo>



la semantica como la coherencia de nuestra web y html la va a tener en cuenta el CEO de google.

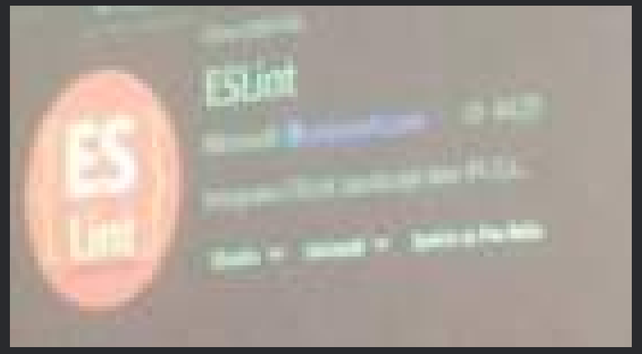
el CEO no lo vamos a ver en clases

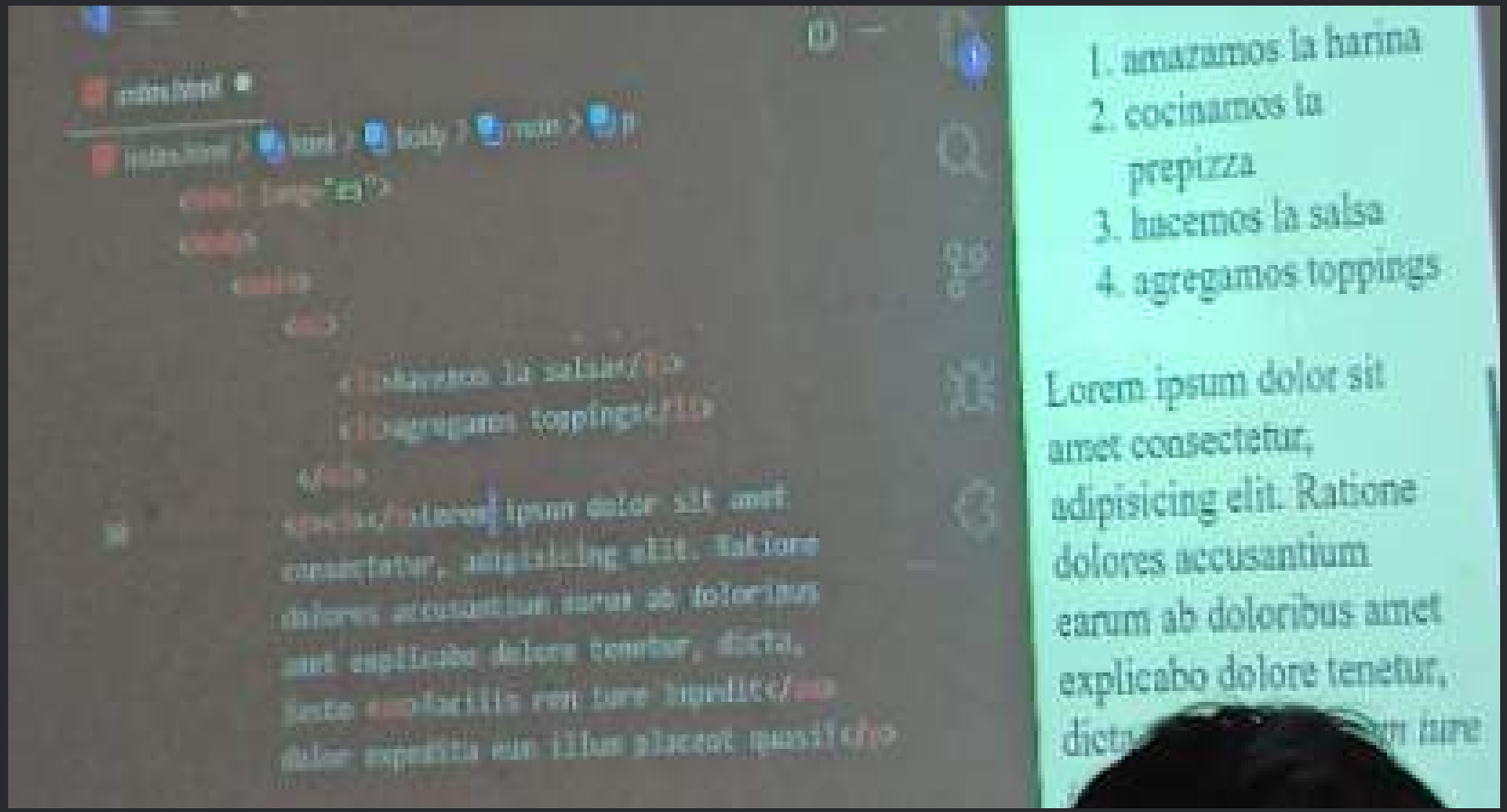
pero es útil para entender que mientras mejor semántica, usabilidad y estructura que tenga la pagina, mejor posicionada va a estar nuestra web



Ejemplo de Midu dev y wen de la velada

extensiones nuevas




1. amazamos la harina
2. cocinamos la prepizza
3. hacemos la salsa
4. agregamos toppings
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Ratione dolores accusantium earum ab doloribus amet explicabo dolore tenetur,

🧠 Introducción a UX/UI en la web

- **UI (User Interface)** se refiere a la **interfaz visual**: colores, botones, tipografía, layout. Es lo que el usuario **ve**.
- **UX (User Experience)** es la **experiencia de uso**: cómo navega, qué tan fácil es entender la web, si encuentra rápido lo que busca. Es lo que el usuario **siente**.

📌 Principios básicos para diseñar una web con buen UX/UI

1. **Claridad y simplicidad:**
   Menos es más. El usuario debe entender qué hacer sin pensar demasiado.
2. **Jerarquía visual:**
   Usa tamaños, colores y espacios para guiar la mirada del usuario. Lo más importante primero.
3. **Consistencia:**
   Mantener mismos colores, estilos y navegación en todas las páginas.
4. **Feedback inmediato:**
   Cuando el usuario hace algo (click, enviar formulario), necesita una respuesta (mensaje, animación, cambio visual).
5. **Carga rápida y rendimiento:**
   Nadie quiere esperar. Optimizar imágenes y código mejora la experiencia.
6. **Responsive design:**
   Que funcione bien en **PC, tablet y celular**.

♿ Accesibilidad web – Reglas básicas (WCAG)

La accesibilidad busca que **todas las personas** (incluyendo con discapacidades) puedan navegar la web.

Principios clave (POUR):

1. **Perceptible:**
   - Texto alternativo en imágenes (`alt`).
   - Contraste alto entre texto y fondo.
   - Subtítulos en videos.
2. **Operable:**
   - Navegación por teclado (sin mouse).
   - Evitar elementos que parpadean rápido (pueden provocar ataques epilépticos).
   - Botones grandes y clicables.
3. **Comprensible:**
   - Lenguaje claro y simple.
   - Instrucciones visibles.
   - Formularios con etiquetas explicativas.
4. **Robusto:**
   - Compatible con lectores de pantalla y tecnologías de asistencia.
   - Uso correcto de etiquetas HTML semánticas (`<nav>` , `<main>` , `<footer>` , etc.).

web util para ver un mal uso del ux ui

userinyerface.com