Unidad 1: HTML

Introducción al Diseño Web

La importancia de tener un bosquejo antes de iniciar un proyecto web

Cuando comenzamos un proyecto web, es crucial tener una planificación clara y detallada. Un bosquejo nos permite visualizar la estructura y funcionalidad del sitio antes de adentrarnos en el desarrollo. Por ejemplo: construir una casa sin planos es algo posible pero con un gran riesgo de tener muchos errores, lo cual retrasaría su construcción ya que seria improvisar sobre la marcha. En un proyecto web es igual, podemos desarrollar una web sin un bosquejo pero entorpecería el desarrollo ya que sería improvisar sobre la marcha, y muchas veces puede demorar más en darse por finalizado un proyecto.

En entornos de desarrollo profesional, rara vez trabajamos de manera aislada. Generalmente formamos parte de un equipo multidisciplinario donde el área de diseño se encarga de elaborar los bosquejos iniciales —ya sea en forma de sketches, wireframes o prototipos— que luego sirven de guía para el equipo de desarrollo. Esto no solo asegura coherencia visual y consistencia en la experiencia de usuario, sino que también facilita la comunicación entre diseñadores y programadores, reduciendo el margen de error y optimizando los tiempos del proyecto.

A continuación, se explican los conceptos clave que conforman esta planificación: Sketch, Wireframe, Mockup, Prototipo Interactivo y Diseño Final.

Tipos de Prototipos

1. Sketch

¿Qué es?

Un sketch es un dibujo rápido y básico que representa las ideas iniciales del diseño de una página web. Suele realizarse a mano alzada en papel o utilizando herramientas digitales simples.

Importancia

- **Generación de Ideas**: Permite plasmar ideas de forma rápida y libre.
- **Comunicación**: Facilita la comunicación inicial entre los miembros del equipo y con el cliente.
- **Iteración Rápida**: Es fácil de modificar, permitiendo iteraciones rápidas y frecuentes.

2. Wireframe

¿Qué es?

Un wireframe es una representación más detallada de la estructura de la página web, destacando la disposición de los elementos principales sin enfocarse en detalles estéticos. Generalmente, se realiza en blanco y negro.

Importancia

- **Claridad Estructural**: Ayuda a definir claramente la jerarquía y disposición de los elementos.
- **Enfoque en la Usabilidad**: Permite centrarse en la funcionalidad y la experiencia del usuario.
- **Base para el Desarrollo**: Sirve como guía para los diseñadores y desarrolladores en las etapas posteriores.

3. Mockup

¿Qué es?

Un mockup es una representación visual detallada del diseño de la página, incorporando colores, tipografías e imágenes. A diferencia del wireframe, el mockup se enfoca en el aspecto estético del sitio.

Importancia

- **Visualización del Diseño Final**: Ofrece una vista previa detallada del aspecto final del sitio web.

- **Feedback Visual**: Facilita la obtención de retroalimentación específica sobre el diseño.
- **Guía Estética**: Proporciona una referencia clara para los desarrolladores en términos de estilo y apariencia.

4. Prototipo Interactivo

¿Qué es?

Un prototipo interactivo es una versión funcional del mockup que permite la interacción del usuario, simulando la navegación y funcionalidad del sitio web. No es un producto final, pero se asemeja mucho a cómo se comportará el sitio real.

Importancia

- **Prueba de Usabilidad**: Permite probar y mejorar la experiencia del usuario antes del desarrollo final.
- **Detección de Problemas**: Ayuda a identificar y resolver problemas de navegación y funcionalidad.
- **Validación con Clientes**: Proporciona una herramienta poderosa para obtener feedback de los clientes de forma temprana.

5. Diseño Final

¿Qué es?

El diseño final es la versión completamente desarrollada y funcional del sitio web, listo para ser lanzado. Incluye todos los elementos visuales y de interacción definidos en los pasos anteriores.

Importancia

- **Entrega Completa**: Representa el producto terminado, listo para su publicación y uso.
- **Cumplimiento de Objetivos**: Asegura que se han cumplido todos los requisitos de diseño, funcionalidad y usabilidad.
- **Satisfacción del Cliente**: Garantiza que el resultado final cumple con las expectativas y necesidades del cliente.

Conclusión

Tener un bosquejo claro antes de iniciar un proyecto web es fundamental para asegurar un proceso de desarrollo organizado y eficiente. Desde el sketch inicial hasta el diseño final, cada etapa del bosquejo contribuye a la creación de un sitio web funcional, estético y centrado en el usuario.

1.2.Experiencia del Usuario (UX)

Hasta ahora trabajamos en la importancia de **tener un bosquejo inicial** y cómo avanzar paso a paso desde un **sketch rápido** hasta un **wireframe estructurado**, incorporando la retroalimentación de los compañeros para mejorar el diseño. Ese recorrido nos permitió entender cómo se organizan los elementos principales de una página y cómo se validan las primeras ideas de forma simple y ágil.

En este punto, damos un paso más allá y empezamos a pensar no solo en la **estructura visual del sitio**, sino también en la **experiencia de las personas que lo van a usar**. Aquí aparece el concepto de **Experiencia del Usuario (UX)**, que nos invita a mirar el diseño con ojos de usuario:

- ¿La navegación es clara y fácil de entender?
- ¿El contenido está organizado de manera lógica y accesible?
- ¿El sitio resulta agradable y usable para distintas personas, incluyendo aquellas con alguna discapacidad?

¿Qué es UX?

La Experiencia del Usuario (UX) se refiere a cómo se siente un usuario al interactuar con un producto o servicio. En desarrollo web, se enfoca en crear sitios y aplicaciones que sean fáciles de usar, intuitivos y agradables.

Importancia de UX

- **Satisfacción del Usuario**: Mejora la satisfacción y retención de usuarios.
- **Accesibilidad**: Asegura que el sitio sea accesible y usable para una amplia audiencia, incluyendo personas con discapacidades.
- **Competitividad**: Un buen diseño de UX puede diferenciar un producto de la competencia y atraer más usuarios.

Ejemplo práctico guiado:

De la misma forma en que pedimos feedback sobre nuestros sketches y wireframes, ahora vamos a **evaluar la UX de un sitio real**, entrenando la mirada crítica para detectar oportunidades de mejora.

1. **Objetivo**: Evaluar la UX de una página web existente.
2. **Instrucciones**:
    - Elige una página web que uses regularmente.
    - Analiza:
        - ¿Es fácil encontrar información clave?
        - ¿Es intuitivo navegar entre secciones?
        - ¿Qué elementos mejorarías para optimizar la experiencia del usuario?
3. **Tarea Práctica**: Rediseña una sección de la página según los problemas detectados.

Producto Mínimo Viable (MVP)

Al igual que un sketch o un wireframe nos permiten probar y ajustar antes de invertir mucho tiempo en el desarrollo, un MVP es la versión mínima de un producto digital que se lanza al mercado para obtener feedback real. El objetivo no es entregar algo perfecto, sino contar con una primera versión que nos permita aprender de los usuarios y mejorar de forma iterativa.

Entonces… ¿Qué es un MVP?

Un **Producto Mínimo Viable (MVP)** no es simplemente una versión reducida de un producto, sino una estrategia de validación. Se trata de construir la versión más simple posible que permita a los usuarios experimentar la propuesta de valor central, sin invertir aún en todas las funcionalidades ni en el diseño final. Su finalidad no es solo “existir” en el mercado, sino aprender de la interacción real de los usuarios, validar hipótesis y orientar las siguientes fases de desarrollo con datos concretos.

Beneficios de un MVP

- **Validación de Ideas**: Permite validar rápidamente si hay demanda para el producto sin necesidad de un desarrollo completo.
- **Ahorro de Recursos**: Minimiza los costos y esfuerzos iniciales al enfocarse solo en las características esenciales.
- **Iteración Rápida**: Facilita realizar mejoras y añadir nuevas funcionalidades basadas en la retroalimentación de los usuarios reales.

Conclusión

De esta manera, lo que empezamos como simples bosquejos y prototipos se vincula con una estrategia más amplia de desarrollo: **diseñar pensando en la experiencia del usuario y validar las ideas a través de versiones mínimas que puedan crecer con el tiempo**.

Utilizar diferentes tipos de prototipos, enfocarse en la UX y desarrollar un MVP son prácticas esenciales en el desarrollo web. Estas estrategias aseguran que los productos sean funcionales, intuitivos y bien recibidos por los usuarios, reduciendo riesgos y optimizando recursos.

1.3.Introducción a HTML

HTML (HyperText Markup Language) es el lenguaje de marcado estándar para crear documentos destinados a ser visualizados en navegadores web. A continuación, se explican los elementos estructurales de HTML, la diferencia entre HTML y CSS, las tecnologías complementarias como CSS y JavaScript, las reglas para nombrar archivos y la importancia del archivo `index.html`.

Podemos imaginar una página web como si fuera el **cuerpo humano**. En este paralelismo:

HTML

Es el **esqueleto** del cuerpo humano: los huesos que sostienen y organizan cada parte. Es el lenguaje de marcado que define la **estructura de la página web**. Permite organizar el contenido en encabezados, párrafos, listas, imágenes o formularios. Sin HTML, no habría una base sobre la cual construir el sitio.

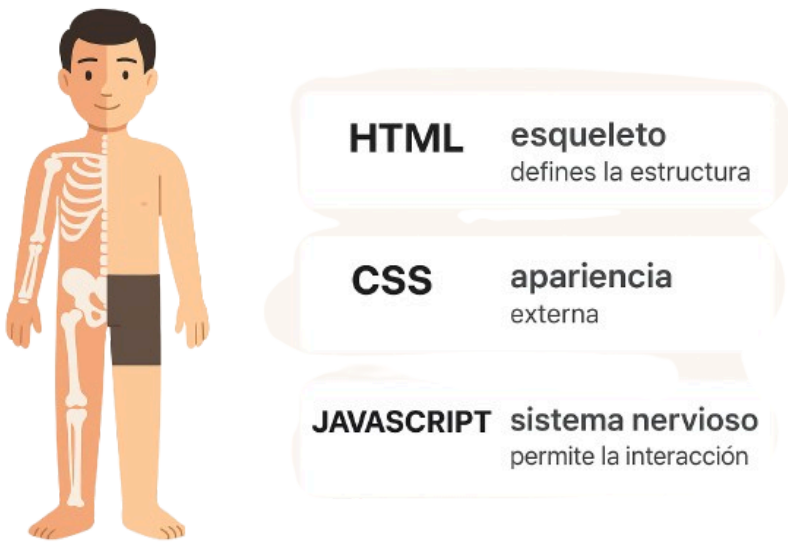
CSS

Es la **apariencia externa**: la piel, la ropa, los colores de ojos y cabello que hacen visible la identidad de cada persona. Es el lenguaje de estilos que controla la **presentación visual**. Permite aplicar tipografías, colores, tamaños, márgenes, disposición de elementos, adaptabilidad (responsive design) y hasta animaciones simples.

JavaScript

Es el **sistema nervioso**: lo que permite reaccionar, moverse, interactuar y responder al entorno. Es el lenguaje de programación que añade **interactividad y dinamismo**. Permite validar formularios en tiempo real, manipular el DOM, responder a eventos (clics, scroll), generar animaciones y comunicarse con servidores para actualizar datos sin recargar la página.

De esta forma, el desarrollo web combina estas tres capas: la **estructura (HTML)**, la **apariencia (CSS)** y la **interactividad (JavaScript)**, tal como un cuerpo humano combina huesos, apariencia y sistema nervioso para funcionar en armonía.



Reglas para nombrar archivos

Cuando desarrollamos un sitio web, no solo importa escribir un buen código: sino también es clave mantener una **organización clara y consistente de los archivos**. Nombrar correctamente cada documento facilita el trabajo en equipo, evita errores en los servidores y permite que el proyecto sea fácil de mantener y escalar. En entornos profesionales, donde varias personas trabajan sobre el mismo repositorio, una mala convención de nombres puede generar confusión, duplicados o incluso fallas al momento de publicar el sitio.

Por eso, existen una serie de **buenas prácticas para nombrar archivos** que garantizan orden, compatibilidad y accesibilidad. Entre ellas, una de las más importantes es el uso del archivo **index.html**, que funciona como la puerta de entrada a cualquier página web.

Buenas prácticas

- **Nombres descriptivos**: Los nombres de archivo deben ser descriptivos y reflejar el contenido del archivo.
- **Sin espacios**: Utiliza guiones (``) o guiones bajos (`_`) en lugar de espacios.
- **Minúsculas**: Preferiblemente, utiliza minúsculas para evitar problemas de compatibilidad en servidores sensibles a mayúsculas y minúsculas.
- Evita **tildes**, `ñ` y **caracteres** no ASCII. Usa *kebab-case* (ej.: `sobre-nosotros.html`)
- **Extensiones correctas**: Asegúrate de utilizar las extensiones correctas (`.html`, `.css`, `.js`).

Ejemplos

- `index.html`
- `estilos.css`
- `script.js`

Importancia del Archivo `index.html`

¿Qué es?

El archivo `index.html` es el archivo principal de una página web. Cuando un navegador solicita una URL sin especificar un archivo, el servidor web busca el archivo `index.html` en la carpeta raíz del sitio.

Importancia

- **Punto de Entrada**: Sirve como punto de entrada principal para los visitantes del sitio web.
- **Estándar de la Industria**: Es una convención ampliamente aceptada y reconocida por los servidores web.

- **Navegabilidad**: Facilita la navegación del usuario al proporcionar una página de inicio predeterminada.

Organización de carpetas

Además de nombrar bien los archivos, es fundamental mantener una **estructura de carpetas organizada** dentro del proyecto. Esto permite localizar rápidamente los recursos, compartir el trabajo con otros desarrolladores y evitar duplicados o confusiones cuando el proyecto escala.

Una estructura básica recomendada para proyectos web puede ser:

```
/mi-sitio
│
├── index.html          (archivo principal de la web)
├── /pages              (páginas adicionales del sitio)
│     ├── about.html
│     ├── contact.html
│     └── services.html
├── /css                (hojas de estilo)
│     └── estilos.css
├── /js                 (scripts de JavaScript)
│     └── script.js
├── /img                (imágenes del sitio)
│     ├── logo.png
│     └── banner.jpg
└── /assets             (otros recursos: íconos, fuentes, etc.)
      └── fuente.woff2
```

Buenas prácticas en la organización de carpetas:

- **Separación por tipo de archivo:** agrupa CSS, JS, imágenes y otros recursos en carpetas distintas.
- **Carpeta de páginas:** usa `pages` para organizar todas las páginas adicionales que no son el `index.html`. Esto mantiene la raíz del proyecto limpia y clara.
- **Estructura clara y predecible:** todos los miembros del equipo deben entender dónde buscar cada recurso.
- **Uso de nombres consistentes:** aplica las mismas reglas de nombrado (minúsculas, guiones, descripciones claras) también en las carpetas.
- **Evitar sobrecargar la raíz:** deja en la raíz solo lo esencial (ej. `index.html`) y coloca los demás archivos en subcarpetas.

Conclusión

Comprender cómo funcionan HTML, CSS y JavaScript es el primer paso para construir sitios web sólidos: HTML aporta la estructura, CSS define la apariencia y JavaScript añade la interactividad. Juntas, estas tecnologías permiten dar forma a páginas web funcionales, accesibles y atractivas, del mismo modo que en un cuerpo humano el esqueleto, la apariencia externa y el sistema nervioso trabajan en conjunto para hacerlo completo y operativo.

Sin embargo, programar no es solo escribir código. La organización del proyecto también es parte del trabajo profesional. Nombrar archivos de manera clara y coherente, utilizar el archivo `index.html` como puerta de entrada y mantener una estructura de carpetas ordenada con secciones como `/pages`, `/css`, `/js` o `/img` garantiza que el sitio sea mantenible, escalable y comprensible para todo el equipo.

En síntesis, un buen desarrollador no solo domina el código, sino que también cuida el orden y las buenas prácticas que hacen posible que un proyecto web crezca de manera eficiente y colaborativa.

1.4.Elementos Estructurales de HTML

¿Qué son?

Los elementos estructurales de HTML son las etiquetas que definen la estructura y contenido de una página web. Cada elemento HTML está representado por una etiqueta, y los más comunes incluyen:

- `<html>`: La raíz del documento HTML.
- `<head>:` Contiene metadatos sobre el documento, como el título y enlaces a hojas de estilo.
- `<title>:` Define el título del documento que se muestra en la pestaña del navegador.
- `<body>:` Contiene todo el contenido visible de la página web.
- `<header>`: Define un encabezado para el documento o una sección.
- `<nav>`: Contiene enlaces de navegación.
- `<main>`: Representa el contenido principal del documento. (Por ejemplo: Si armamos una web de venta de autos, el contenido principal serían los autos que se encuentran en venta)
- `<section>`: Define una sección en el documento.
- `<article>`: Representa contenido autónomo.
- `<footer>`: Define un pie de página para el documento o una sección.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Venta de Autos</title>
</head>
<body>
    <header>
        <h1>Venta de Autos</h1>
```

Etiquetas y Sintaxis de HTML

Para poder construir páginas web necesitamos un lenguaje que el navegador pueda **interpretar y traducir en contenido visible**. Ese lenguaje es **HTML**, y su base está en las **etiquetas**.

¿Qué son las etiquetas de HTML?

Las etiquetas de HTML son comandos que indican al navegador cómo debe mostrar el contenido en la página web. Cada etiqueta tiene un propósito específico y se escribe entre corchetes angulares (`< >`).

Sintaxis de HTML

La sintaxis básica de una etiqueta HTML incluye una etiqueta de apertura, contenido y una etiqueta de cierre:

```
<etiqueta>Contenido</etiqueta>
```

Por ejemplo:

```
<p>Este es un párrafo.</p>
```

Tipos de etiquetas

Etiquetas Abiertas

La mayoría de las etiquetas en HTML funcionan bajo una estructura de **apertura y cierre**. Esto significa que delimitan un bloque de contenido que va entre ambas etiquetas.

Sintaxis básica:

```
<etiqueta>Contenido</etiqueta>
```

- La **etiqueta de apertura** indica el inicio del contenido (por ejemplo `<p>`).
- La **etiqueta de cierre** marca el final del contenido, y se distingue porque lleva una barra inclinada (`/`) antes del nombre de la etiqueta (por ejemplo `</p>`).
- En el medio se coloca el **contenido** que será mostrado en la página (texto, enlaces, otros elementos HTML).

Ejemplo:

```
<p>Este es un párrafo.</p>
```

`<p>` abre un párrafo y `</p>` lo cierra.

Etiquetas auto-cerradas

En HTML existen algunas etiquetas que **no necesitan un contenido dentro** porque cumplen una función puntual y directa. A diferencia de las etiquetas normales, que se abren y se cierran (por ejemplo ** ** Contenido ** **), estas etiquetas se **auto-cierran** en una sola línea.

Ejemplos comunes son:

- `<img />` → se utiliza para insertar imágenes. La información necesaria (la ruta del archivo y el texto alternativo) va en forma de atributos dentro de la misma etiqueta.
- `<br />` → inserta un salto de línea, es decir, corta el texto y lo continúa en la siguiente línea.
  `<hr />` → dibuja una línea horizontal que sirve para separar secciones de contenido.
- `<meta />` → usada en el `<head>` para indicar información adicional como la codificación de caracteres o la descripción de la página.

1.5.Estructura básica de un documento HTML

Un documento HTML típico tiene la siguiente estructura básica:

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Título de la Página</title>
</head>
<body>
    <h1>Encabezado Principal</h1>
    <p>Este es un párrafo de ejemplo.</p>
</body>
</html>
```

Descripción de la estructura:

- **<!DOCTYPE html>**: Declara el tipo de documento y la versión de HTML.
- **<html>**: La raíz del documento HTML.
- **<head>**: Contiene metadatos y enlaces a recursos externos.
- **<body>**: Contiene el contenido visible de la página web.

Elementos esenciales de HTML

<head>

¿Qué es?

El elemento **<head>** contiene metadatos sobre el documento HTML, como el título, enlaces a hojas de estilo, scripts y metaetiquetas.

Contenido común:

<title>:

Define el título de la página que aparece en la pestaña del navegador.

Importancia

- **SEO**: Ayuda a los motores de búsqueda a entender el contenido de la página.
- **Usabilidad**: Facilita la identificación de la página por parte del usuario.

<meta>:

Proporciona metadatos como la codificación de caracteres y la configuración de la vista.

Atributos Comunes:

- **charset:** Define la codificación de caracteres del documento
  ```
  <meta charset="UTF-8">
  ```
- **name y content:** Proporcionan información como la descripción de la página y las palabras clave.
  ```
  <meta name="description" content="Descripción de la página web">
  <meta name="keywords" content="HTML, CSS, JavaScript">
  ```

<body>

¿Qué es?

El elemento **<body>** contiene todo el contenido visible de la página web, como textos, imágenes, videos, enlaces y formularios.

Importancia

- **Contenido Principal**: Todo lo que se muestra al usuario en la página web se encuentra dentro de **<body>**.
- **Interactividad**: Elementos interactivos como botones y formularios se colocan en esta sección.

Conclusión

Comprender los conceptos fundamentales de HTML es esencial para el desarrollo web. Las etiquetas y la sintaxis de HTML proporcionan la estructura básica del documento, mientras que los elementos esenciales como **<head>**, **<title>**, **<meta>** y **<body>** definen la organización y el contenido de la página web. Estos conocimientos son la base para crear sitios web bien estructurados y accesibles.

1.6.Elementos de Bloque y de Línea en HTML

En HTML, los elementos se clasifican en dos tipos principales: elementos de bloque y elementos en línea. Esta clasificación determina cómo se comportan los elementos en la página web y cómo se organizan dentro del documento.

Elementos de Bloque

¿Qué son?

Los elementos de bloque ocupan todo el ancho disponible de su contenedor y siempre comienzan en una nueva línea. Estos elementos se utilizan para estructurar y dividir el contenido en secciones lógicas.

Ejemplos Comunes

- **<p>**: Representa un párrafo de texto.
- **<h1>** **a** **<h6>**: Representan los diferentes niveles de encabezados.
- ** **: Un contenedor genérico para agrupar otros elementos.
- **<section>**: Define una sección en el documento.
- **<article>**: Representa contenido autónomo.
- **<header>**: Define un encabezado para un documento o sección.
- **<footer>**: Define un pie de página para un documento o sección.

Uso Dentro del <body>

Los elementos de bloque se utilizan para organizar y estructurar el contenido de manera clara y lógica. Por ejemplo:

```
<body>
    <header>
        <h1>Título Principal</h1>
    </header>
    <section>
        <h2>Subtítulo</h2>
        <p>Este es un párrafo dentro de una sección.</p>
```

```
    </section>
    <footer>
        <p>Pie de página del documento.</p>
    </footer>
</body>
```

Elementos en Línea

¿Qué son?

Los elementos en línea (inline) no inician una nueva línea y solo ocupan el espacio necesario para el contenido. Se utilizan principalmente para estilizar partes específicas del contenido dentro de un bloque.

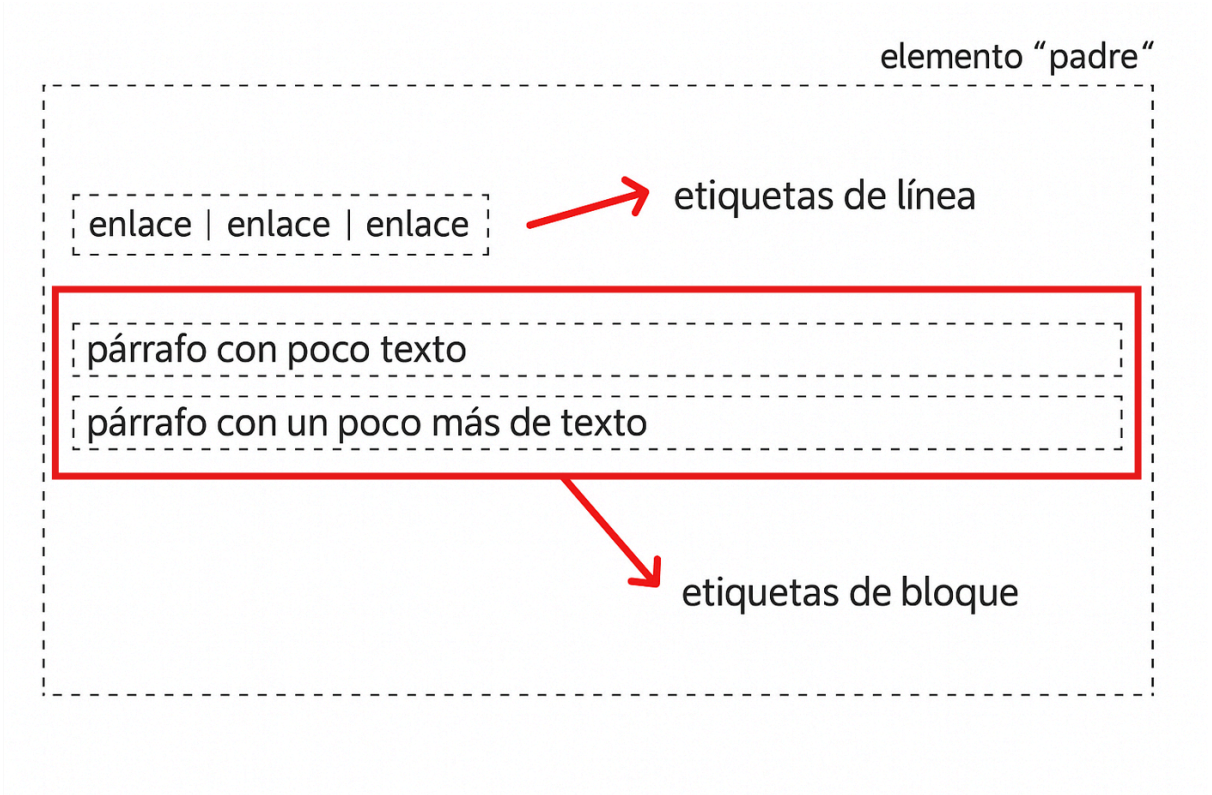
Ejemplos Comunes

- **`<a>`**: Representa un hipervínculo.
- **`<span>`**: Un contenedor genérico para estilizar partes del texto.
- **`<strong>`**: Representa texto de importancia fuerte, generalmente mostrado en negrita.
- **`<em>`**: Representa texto enfatizado, generalmente mostrado en cursiva.
- **`<img>`**: Representa una imagen.
- **`<br>`**: Inserta un salto de línea.

Uso Dentro del `<body>`

Los elementos en línea se utilizan dentro de los elementos de bloque para aplicar estilos específicos o insertar contenido inline. Por ejemplo:

```
<body>
<p>Este es un párrafo con un <a href="#">enlace</a> y algo de <strong>texto en negrita</strong>.</p>
<p>Otro párrafo con una <em>énfasis</em> en una palabra.</p>
<p>Imagen en línea: <img src="imagen.jpg" alt="Descripción de la imagen"></p>
</body>
```



Jerarquía de Encabezados

¿Qué son los Encabezados?

Los encabezados (**<h1>** a **<h6>**) se utilizan para definir títulos y subtítulos en el contenido. Representan diferentes niveles de importancia, siendo **<h1>** el nivel más alto y **<h6>** el más bajo.

Una regla que tiene el `<h1>` es que se debe tener uno solo por página, ya que es una buena práctica por un tema del SEO y de accesibilidad.

Importancia de la Jerarquía

- **Accesibilidad**: Ayuda a los usuarios con lectores de pantalla a navegar por el contenido de manera efectiva.
- **SEO**: Los motores de búsqueda utilizan la jerarquía de los encabezados para indexar y entender la estructura del contenido.
- **Organización**: Facilita la lectura y comprensión del contenido al proporcionar una estructura lógica.

Ejemplo de Uso Correcto

Mantener una jerarquía lógica de encabezados es crucial:

```
<body>

<h1>Título Principal</h1>

<h2>Subtítulo de la Sección</h2>

<p>Contenido de la sección...</p>

<h3>Subsección</h3>

<p>Contenido de la subsección...</p>

<h2>Otro Subtítulo de Sección</h2>

<p>Más contenido...</p>

</body>
```

Conociendo a fondo HTML: Enlaces y Multimedia

En este video, aprenderemos cómo crear enlaces en HTML, incluyendo **enlaces absolutos, relativos e internos**. También cubriremos cómo insertar **imágenes, videos y audios** en un documento HTML utilizando las etiquetas correspondientes. Mostraremos ejemplos en pantalla para cada tipo de enlace y contenido multimedia.

Esperamos que este video te haya ayudado a entender mejor cómo crear enlaces y agregar contenido multimedia en HTML. **Practica estos conceptos** para mejorar tus habilidades y enriquecer tus páginas web.

Target _blank

El atributo `target="_blank"` hace que un enlace se abra en una **nueva pestaña o ventana** cuando se hace clic en él. Esto puede ser útil para mantener la página principal abierta mientras se consulta otro recurso o se envía un correo electrónico.

Las **rutas relativas** en HTML cambian según la ubicación del archivo en el que estés trabajando. No siempre vas a escribir el mismo camino: depende de si el archivo está en la carpeta raíz o dentro de otra carpeta (como `/pages`). Por eso es importante comprender cómo moverse entre carpetas: entrar a una (`./`), salir de ella (`../`) o enlazar a un archivo que se encuentra en el mismo nivel.

- Si estás en `index.html` (en la raíz) y querés ir a una página dentro de `/pages`:
  `<a href="pages/about.html">Sobre mí</a>`
- Si estás en `pages/about.html` y querés volver al `index.html` de la raíz:
  `<a href="../index.html">Inicio</a>`
- Si querés enlazar a otra página dentro de la misma carpeta `/pages`:
  `<a href="contact.html">Contacto</a>`

Recomendaciones y buenas prácticas para el desarrollo de sitios web con HTML

El desarrollo de sitios web utilizando HTML implica más que solo conocer la sintaxis y los elementos. Adoptar buenas prácticas es esencial para crear sitios web eficientes, mantenibles y accesibles. A continuación, se presentan algunas recomendaciones clave.

Creación de una Estructura Lógica

Organización del Contenido

- **Usa Elementos Semánticos**: Utiliza etiquetas HTML5 semánticas como `<header>`, `<nav>`, `<article>`, `<section>`, `<footer>` para estructurar tu contenido de manera lógica y significativa.
- **Jerarquía de Encabezados**: Mantén una jerarquía clara de encabezados (`<h1>` a `<h6>`) para organizar el contenido y facilitar la navegación tanto para los usuarios como para los motores de búsqueda.

Ejemplo

```
<body>
    <header>
        <h1>Título Principal</h1>
        <nav>
            <ul>
                <li><a href="#">Inicio</a></li>
                <li><a href="#">Sobre Nosotros</a></li>
                <li><a href="#">Contacto</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section>
            <h2>Sección Principal</h2>
            <p>Contenido de la sección principal.</p>
        </section>
        <article>
```

```
            <h2>Artículo Relevante</h2>
            <p>Contenido del artículo.</p>
        </article>
    </main>
    <footer>
        <p>© 2024 Mi Sitio Web</p>
    </footer>
</body>
```

Uso de Comentarios

¿Por qué usar comentarios?

- **Documentación**: Facilitan la comprensión del código, tanto para ti como para otros desarrolladores que puedan trabajar en el proyecto en el futuro.
- **Depuración**: Ayudan a identificar secciones del código durante el proceso de depuración.

Sintaxis de comentarios

```
<!-- Este es un comentario en HTML -->
<p>Contenido visible en la página web.</p>
<!-- Fin de la sección principal -->
```

Buenas Prácticas

- **Comentarios Claros y Concisos**: Escribe comentarios que sean claros y directos al punto.
- **No Comentar en Exceso**: Evita agregar comentarios innecesarios que puedan sobrecargar el código.

1.7.Listas y tablas

En este video, aprenderemos cómo crear **listas y tablas en HTML**. Cubriremos las **listas ordenadas y no ordenadas**, cómo **anidar listas**, y cómo crear **tablas para organizar datos** de manera efectiva. Mostraremos ejemplos en pantalla para ilustrar cada concepto.

Esperamos que este video te haya ayudado a entender mejor cómo crear listas y tablas en HTML. **Practica estos conceptos** para mejorar tus habilidades y organizar tu contenido de manera efectiva en tus páginas web.

Formularios

En este video, aprenderemos cómo crear **formularios en HTML** utilizando las etiquetas `<form>`, `<input>`, `<textarea>` y `<select>`. También explicaremos los **atributos necesarios** para los formularios y mostraremos ejemplos de diferentes tipos de controles, como **cajas de texto, botones de radio, casillas de verificación y menús desplegables**.

Esperamos que este video te haya ayudado a entender mejor cómo crear formularios en HTML. **Practica estos conceptos** para mejorar tus habilidades y crear formularios efectivos y funcionales en tus páginas web.

1.8.La Revolución del Desarrollo con HTML y IA

El uso de HTML es el pilar del desarrollo web. Durante años, los desarrolladores han tenido que escribir cada línea de código manualmente, lo cual, si bien es efectivo, es un proceso que puede consumir mucho tiempo, ser propenso a errores humanos y ralentizar la productividad. Hoy en día, gracias a herramientas impulsadas por inteligencia artificial como **MyMap.AI**, es posible generar automáticamente páginas web a partir de descripciones textuales, cambiando completamente el panorama del desarrollo web.

La IA no solo facilita la creación rápida de código HTML, sino que también permite a los desarrolladores enfocarse más en el diseño, la funcionalidad y las necesidades del cliente, en lugar de preocuparse por los detalles del código base.

1. ¿Qué es MyMap.AI?

MyMap.AI es una herramienta basada en inteligencia artificial diseñada específicamente para generar código HTML de manera automática a partir de simples descripciones textuales. Esta tecnología es capaz de transformar ideas o esquemas iniciales en código completamente funcional, desde la estructura más básica hasta componentes web más avanzados. Esto significa que cualquier persona, sin importar su nivel de experiencia en programación, puede aprovechar **MyMap.AI** para crear rápidamente prototipos o páginas web funcionales.

¿Cómo funciona?

- **Descripción a código**: El usuario describe, en lenguaje natural, la estructura o funcionalidad deseada. Por ejemplo, puedes decir: "Genera una página con un menú superior, tres secciones de contenido y un pie de página". A partir de esta descripción, **MyMap.AI** genera automáticamente el código HTML correspondiente.
- **Modificación en tiempo real**: Si necesitas ajustar algún detalle o agregar nuevos elementos, puedes simplemente describir los cambios, y la IA generará el nuevo

código de forma automática. Esto permite iteraciones rápidas sin necesidad de modificar manualmente el código.

Importancia

MyMap.AI no solo automatiza el proceso de escritura de HTML, sino que también aporta una serie de ventajas clave:

- **Reducción de tiempo**: Gracias a la IA, los desarrolladores pueden ahorrar horas de codificación manual, enfocándose en otros aspectos más críticos del proyecto.
- **Facilidad para principiantes**: No se requieren conocimientos avanzados de HTML para generar una estructura básica o incluso compleja de un sitio web. Esto democratiza el desarrollo web, permitiendo que cualquier persona con una idea pueda llevarla a cabo.
- **Prototipado rápido**: Ideal para crear prototipos de sitios web en minutos, los cuales pueden ser compartidos, revisados y ajustados antes de entrar en una etapa de desarrollo más profunda.

Prompt de ejemplo

"Genera una página HTML con un menú de navegación superior, un cuerpo de tres secciones y un pie de página con enlaces a redes sociales."

Este tipo de prompt permite a **MyMap.AI** crear una página completamente funcional basada en descripciones simples. Los resultados pueden ser modificados en tiempo real, según las necesidades del proyecto.

Casos de uso

- **Desarrolladores web**: Utilizan **MyMap.AI** para crear prototipos rápidos de sitios web, permitiendo iteraciones rápidas y mejoras en tiempo real.
- **Equipos de diseño**: La herramienta facilita la comunicación entre los diseñadores y los desarrolladores al transformar bocetos o wireframes en código HTML real que puede ser visualizado y ajustado de inmediato.
- **Marketing digital**: Para crear landing pages (páginas de aterrizaje) optimizadas en cuestión de minutos, mejorando la conversión sin necesidad de contar con un desarrollador especializado.

Ventajas adicionales

- **Menos errores humanos**: Al automatizar la creación de código, se reduce significativamente el margen de error, especialmente en proyectos grandes donde el trabajo manual podría llevar a inconsistencias.
- **Escalabilidad**: Puedes comenzar con una estructura básica y, a medida que crece el proyecto, simplemente añadir nuevas secciones o funcionalidades a través de nuevas descripciones.

Limitaciones

A pesar de sus ventajas, **MyMap.AI** aún tiene ciertas limitaciones. Aunque es muy eficiente para generar estructuras HTML, puede requerir ajustes manuales si se desea un nivel más avanzado de personalización o si necesitas integrarlo con CSS o JavaScript específicos para lograr un diseño o funcionalidad avanzada.

Enlace de referencia

MyMap.AI - Generador HTML

MyMap

Checklists rápidas (para actividades)

Sketch → Wireframe

- Header/nav visibles y entendibles
- Jerarquía clara (H1→H2→H3)
- Flujo principal en 3 clics o menos
- Footer con info útil/legales
- Versión mobile en una sola columna

Revisión HTML (IA o manual)

- `<!DOCTYPE html>` + `<html lang="es">`
- `<meta charset>` + `<meta name="viewport">`
- **Un** `<h1>` por página
- Semántica: `header/nav/main/section/article/aside/footer`
- Menú con `aria-current="page"`
- Imágenes con `alt` descriptivo
- Links externos: `target="_blank" rel="noopener noreferrer"`

- Rutas relativas correctas (desde `pages/*` usar `../`)
- Sin etiquetas sin cerrar

Prompt de auditoría (copy-paste)

> Actúa como revisor de HTML semántico y accesible. Evalúa este archivo:
>
> 1. Jerarquía de encabezados (un solo `<h1>`; orden lógico `<h2>`, `<h3>`…)
> 2. Uso de etiquetas semánticas (`header/nav/main/section/article/aside/footer`)
> 3. Accesibilidad mínima: `lang`, meta viewport, `alt`, `label/for`, `aria-current`
> 4. Rutas relativas del menú
> 5. Etiquetas sin cerrar
>
> Devuélveme una lista de problemas concretos y propuestas de corrección sin reescribir todo el archivo.

Mini-rúbrica sugerida (para evaluar entregas)

- **Estructura y semántica (30%)**: etiquetas correctas, jerarquía de títulos, `index.html`.
- **Navegación (20%)**: menú consistente en 5 páginas, rutas relativas OK, `aria-current`.
- **Accesibilidad básica (20%)**: `lang`, viewport, `alt`, foco visible, skip link.
- **Organización del proyecto (20%)**: carpetas y naming (minúsculas, `-`, sin tildes/ñ).
- **Formulario (10%)**: `label/for`, tipos correctos, `required`, `autocomplete`.

Bonus (pequeños aportes de valor)

- **SEO básico**: `title` único por página + `meta description` (~155–160 caracteres).
- **Open Graph** (cuando vean redes): `og:title`, `og:description`, `og:image`.
- **Buenas prácticas**: evita *divitis*, usa `section` cuando hay encabezado; `article` si el bloque es autónomo (ej. una entrada de blog).