

{.js}

JavaScript

# INTRODUCCIÓN A JAVASCRIPT: EL LENGUAJE QUE ANIMA LA WEB



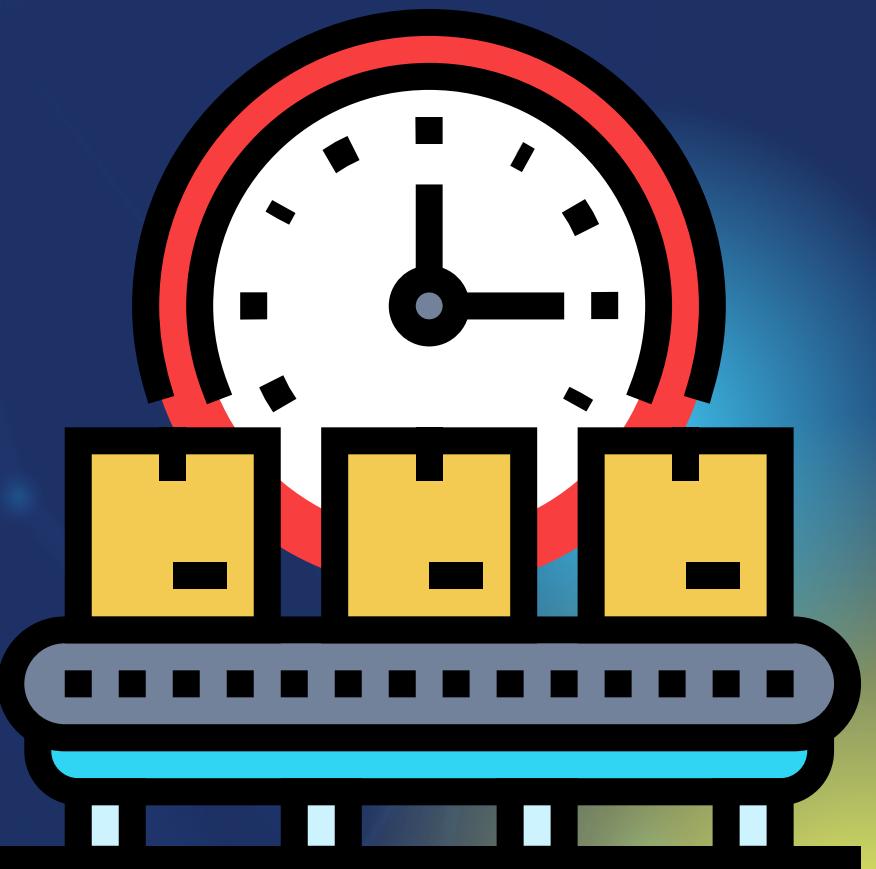
LEONARDO CHAGOYA  
KAREN PÉREZ



# ¿QUÉ ES JAVASCRIPT?

- Es un lenguaje de programación interpretado o Just-in-Time compilado.
- Principalmente utilizado para hacer las páginas web interactivas.
- Corre directamente en el navegador web del usuario (lado del cliente).
- **¡Importante!** No es lo mismo que Java (aunque comparten parte del nombre).
- ¿Para qué sirve? Animar elementos, validar formularios, cargar contenido dinámicamente, interactuar con el usuario, etc.

JS



# ¿POR QUÉ JAVASCRIPT ES IMPORTANTE?

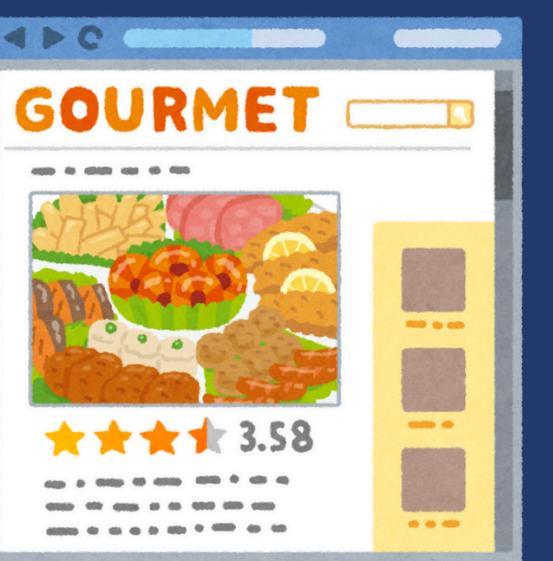
## Interactividad

Permite que las páginas web respondan a las acciones del usuario (deja de ser estatica).



## Experiencia de Usuario (UX)

Mejora la usabilidad y el atractivo de un sitio web, haciéndolo más dinámico y responsivo.



## Pilar del Desarrollo Web

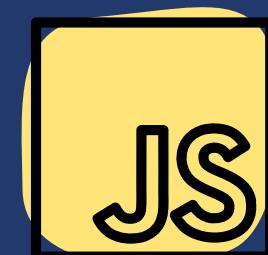
HTML



CSS

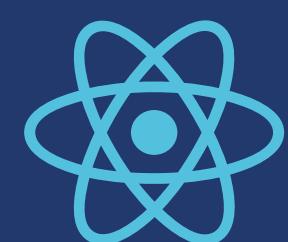


Javascript



## Versatilidad

Frontend.  
Servidores (Node.js).  
Aplicaciones móviles (React Native).  
Aplicaciones de escritorio (Electron).



# ¿DÓNDE COLOCAMOS EL CÓDIGO JAVASCRIPT?

El código JavaScript se enlaza a un archivo HTML usando la etiqueta `<script>`.

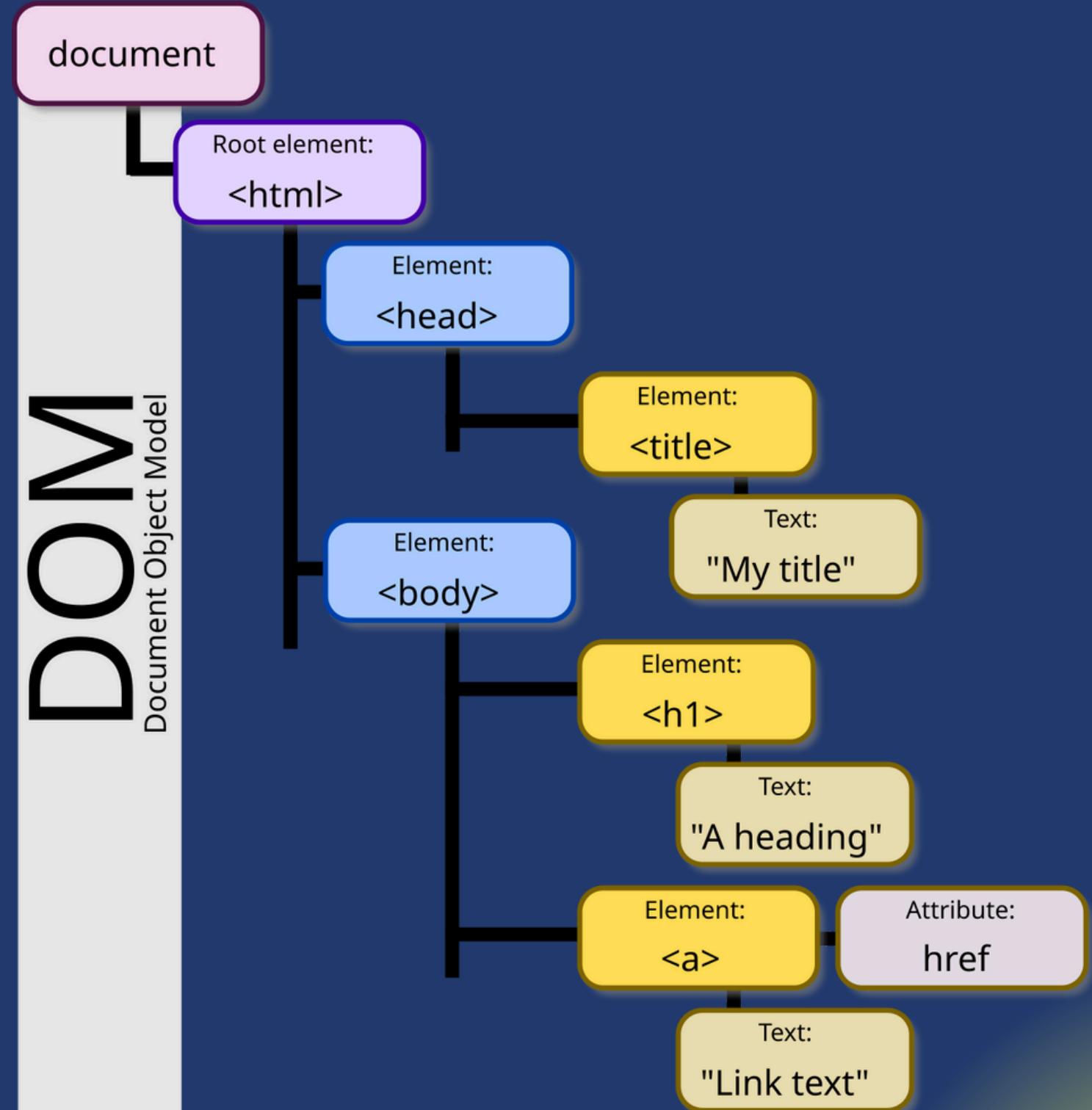
- Generalmente se coloca justo antes del cierre de la etiqueta `</body>`.
  - ¿Por qué ahí? Para asegurar que todo el HTML se cargue completamente antes de que el script intente interactuar con él. Si el script intenta manipular un elemento que aún no existe, causará un error.
- Ejemplo:

```
<body>
    <!-- Mucho contenido HTML aquí -->

    <script src="../javascript/main.js"></script>
    <script src="https://kit.fontawesome.com/8ceb30feb6.js"
        crossorigin="anonymous">
    </script>
</body>
```

# EL DOM (DOCUMENT OBJECT MODEL)

- El DOM es una representación estructurada del documento HTML como un árbol de objetos. Piensa en cada etiqueta HTML (como `<div>`, `<p>`, `<img>`) como un "nodo" en este árbol.
- JavaScript usa el DOM para acceder, modificar y manipular el contenido, la estructura y el estilo de una página web.
- HTML es el esqueleto, CSS es la ropa, y JavaScript es el cerebro que mueve el cuerpo. El DOM es el sistema nervioso que conecta el cerebro con el cuerpo.



# SELECCIONANDO ELEMENTOS DEL DOM

- Para poder interactuar con un elemento HTML (cambiar su texto, ocultarlo, etc.), primero debemos "encontrarlo" o "seleccionarlo" en el DOM.
- Métodos comunes para seleccionar elementos:

```
document.getElementById('idDelElemento');
```

```
document.querySelector('selectorCSS')
```

```
document.querySelectorAll('selectorCSS')
```

- Los IDs deben ser únicos en una página
- Selecciona el primer elemento que coincide con el selector CSS especificado .
- Selecciona TODOS los elementos que coincidan con el selectorCSS. Devuelve una NodeList (que es como un array de elementos).

```
socialIcons.forEach(icon => {
```

# Eventos en JavaScript



- Son acciones que ocurren en el navegador o que el usuario realiza (ej. clic de ratón, pulsación de tecla, carga de la página, envío de un formulario, etc.).
- 'Event Listeners': JavaScript nos permite "escuchar" estos eventos y ejecutar una función específica cuando ocurren.
- Método **addEventListener()**: Es la forma estándar de registrar un escuchador de eventos.

```
elemento.addEventListener('tipoDeEvento', funcionAConsultar);
```

## Tipos de eventos comunes (y en nuestro proyecto):

```
btnPlay.addEventListener('click', () => {
```

"**click**": Cuando se hace clic en un elemento  
(ejemplo:

```
btnPlay.addEventListener('click', ...)).
```

'**DOMContentLoaded**': Cuando el HTML está completamente cargado  
y parseado (ejemplo:

```
document.addEventListener('DOMContentLoaded', ...)).
```

Otros: 'mouseover', 'keydown', 'submit', 'load', etc.

# Manipulación de Estilos y Clases

JavaScript puede **cambiar dinámicamente los estilos CSS de los elementos HTML.**

- 1. Propiedad **style** (Estilos inline):
  - Acceder y modificar propiedades CSS directamente en la propiedad **style** del elemento.
  - Ejemplo:

```
modal.style.display = 'block';
```

- 2: Propiedad **classList** (¡Recomendado!):
  - Mantiene tus estilos definidos en tu CSS y tu JavaScript solo gestiona la lógica.
  - Ejemplo:

```
elemento.classList.add('nombreClase'); /*Añade una clase al elemento.*/
elemento.classList.remove('nombreClase'); /*Quita una clase del elemento.*/
elemento.classList.toggle('nombreClase'); /*Si la clase está presente, la quita; si no lo está, la añade.*/
```

## En nuestro proyecto

```
icon.classList.add('clicked');
```

```
icon.classList.remove('clicked');
```

**classList.add('clicked')** agrega la clase **clicked** al elemento representado por la variable icon. Si el ícono (en este caso, el enlace de la red social) no tiene la clase **clicked**, esta clase se añadirá

**classList.remove('clicked')** elimina la clase **clicked** del elemento representado por la variable icon. Esto es útil si deseas revertir el cambio visual aplicado cuando se añade la clase

# Controlando Elementos de Video

Los elementos `video` en HTML tienen propiedades y métodos propios que JavaScript puede controlar para gestionar la reproducción multimedia.

Propiedades útiles:

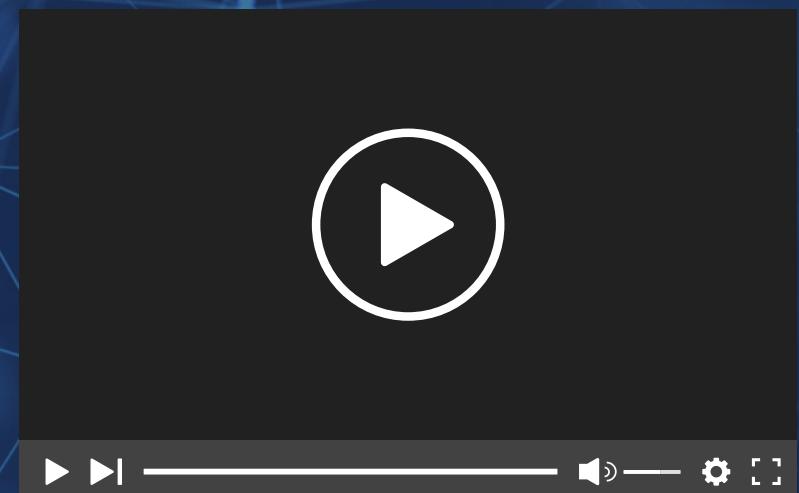
`video.currentTime` Permite obtener o establecer el tiempo actual de reproducción del video (en segundos). Útil para reiniciar el video.

`video.pause();` Una propiedad de solo lectura que devuelve true si el video está en pausa, false si está reproduciéndose.

Métodos importantes:

`video.play();` Inicia o reanuda la reproducción del video.

`video.pause();` Pausa la reproducción del video.



# Temporizadores (setTimeout)

- A veces necesitamos que una parte de nuestro código JavaScript se ejecute después de un cierto período de tiempo, no inmediatamente.
- `setTimeout(funcion, retrasoEnMilisegundos);`
- Ejecuta una función (o un bloque de código) solo una vez, después del retraso especificado en milisegundos.

Utilizado para animaciones, mostrar mensajes temporales, o como en nuestro caso, para coordinar acciones con animaciones CSS.



JS

# Resumen de Conceptos Clave

**JavaScript:** El lenguaje que da vida a nuestras páginas web.

**DOM:** El "mapa" de nuestro HTML que JavaScript puede leer y modificar.

**Selección de Elementos:** Métodos como getElementById, querySelector, querySelectorAll para encontrar elementos.

**Eventos:** Acciones del usuario o del navegador a las que podemos responder.

addEventListener(): La herramienta clave para escuchar y reaccionar a los eventos.

**Manipulación de Estilos:** Usar style para cambios directos o classList para gestionar clases CSS.

**Control de Video:** Propiedades (currentTime) y métodos (play(), pause()) para interactuar con <video>.

setTimeout(): Para ejecutar código con un retraso específico.

