

# JavaScript en el Navegador

Onclick y DOM

Fundamentos de la Web - Clase 9

# Objetivos de Hoy

## ¿Qué aprenderemos?

- ⚡ Crear y usar funciones en JavaScript
- 🖱 Entender qué son los eventos en la web
- 🎯 Usar el evento `onclick` para hacer páginas interactivas
- 🎨 Manejar el evento `hover` (`onmouseover/onmouseout`)
- 📝 Trabajar con eventos de entrada ( `oninput` , `onchange` )
- ⌚ Usar `setTimeout` para retrasos
- 🔍 Seleccionar elementos del DOM con `querySelector`
- 🎭 Cambiar HTML y CSS desde JavaScript
- 📁 Manipular clases CSS con `classList`




# Funciones en JavaScript

# ¿Qué es una Función?

## Piensa en una función como una receta de cocina

Imagina que tienes una receta para hacer un sándwich. Cada vez que quieres un sándwich, no necesitas recordar todos los pasos desde cero: solo sigues la receta que ya escribiste.

Una función en JavaScript funciona igual:

-  La escribes una vez
-  La usas muchas veces
-  Hace una tarea específica

# ✨ Anatomía de una Función

```
// Declaración de una función
function saludar() {
    console.log("¡Hola, mundo!");
}

// Llamar (ejecutar) la función
saludar(); // Imprime: ¡Hola, mundo!
```

## Componentes clave:

- `function` → Palabra clave para crear una función
- `saludar` → Nombre de la función (puedes elegirlo)
- `()` → Paréntesis (aquí van los parámetros)
- `{ }` → Llaves (aquí va el código que ejecuta)





# Funciones con Parámetros

Los parámetros son como ingredientes variables

```
// Función con parámetro
function saludarPersona(nombre) {
    console.log("¡Hola, " + nombre + "!");
}

// Llamadas con diferentes valores
saludarPersona("Ana");           // Imprime: ¡Hola, Ana!
saludarPersona("Carlos");        // Imprime: ¡Hola, Carlos!
saludarPersona("María");         // Imprime: ¡Hola, María!
```

¿Por qué usar parámetros?

-  Reutilizar la misma función con diferentes valores
-  Hacer tu código más flexible



# Funciones que Devuelven Valores

```
// Función que calcula y devuelve un resultado
function sumar(a, b) {
    return a + b;
}

// Usar el valor devuelto
let resultado = sumar(5, 3);
console.log(resultado); // Imprime: 8

// También puedes usarlo directamente
console.log(sumar(10, 20)); // Imprime: 30
```

La palabra `return` :

-  Devuelve un valor desde la función
-  Termina la ejecución de la función

# ? Pregunta Rápida

¿Qué imprimirá este código?

```
function multiplicar(x, y) {  
    return x * y;  
}  
  
let resultado = multiplicar(4, 5);  
console.log(resultado);
```

Opciones:

- A) 4
- B) 5
- C) 20
- D) 45

Piensa unos segundos... 🤔



# Respuesta

```
function multiplicar(x, y) {  
    return x * y;  
}  
  
let resultado = multiplicar(4, 5);  
console.log(resultado);
```

Respuesta correcta: C) 20

¿Por qué?

- La función `multiplicar` recibe 4 y 5 como parámetros
- Multiplica  $4 \times 5 = 20$
- Devuelve 20 con `return`
- Se guarda en `resultado` y se imprime

¡Excelente si acertaste! 🎉





# Eventos en JavaScript



# ¿Qué son los Eventos?

Los eventos son acciones que ocurren en tu página web

Piensa en un interruptor de luz:

-  Cuando presionas el interruptor (evento)
-  La luz se enciende o apaga (respuesta)

En una página web:

-  Usuario hace clic en un botón (evento)
-  La página responde (ejecuta código JavaScript)

## Tipos Comunes de Eventos

Evento	¿Cuándo ocurre?	Ejemplo de uso
<code>onclick</code>	Al hacer clic	Enviar un formulario
<code>onmouseover</code>	Al pasar el mouse	Mostrar un tooltip
<code>onmouseout</code>	Al sacar el mouse	Ocultar un tooltip
<code>oninput</code>	Al escribir en un input	Búsqueda en tiempo real
<code>onchange</code>	Al cambiar selección	Filtrar opciones

## El evento más común: hacer clic

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi primer evento</title>
</head>
<body>
  <button onclick="mostrarMensaje()">Haz clic aquí</button>

  <script>
    function mostrarMensaje() {
      alert("¡Hiciste clic en el botón!");
    }
  </script>
</body>
</html>
```

### ¿Qué pasa aquí?

- 🖱️ Usuario hace clic en el botón
- ⚡ Se ejecuta la función `mostrarMensaje()`



# Ejemplo Práctico: Contador de Clics

```
<!DOCTYPE html>
<html>
<head>
  <title>Contador</title>
</head>
<body>
  <h1>Has hecho <span id="contador">0</span> clics</h1>
  <button onclick="incrementar()">Hacer clic</button>

  <script>
    let clics = 0;

    function incrementar() {
      clics++;
      document.getElementById("contador").innerText = clics;
    }
  </script>
</body>
</html>
```




# La Palabra Mágica: `this`

`this` se refiere al elemento que activó el evento

```
<button onclick="cambiarColor(this)">Cámbiame de color</button>

<script>
  function cambiarColor(elemento) {
    elemento.style.backgroundColor = "lightblue";
    elemento.innerText = "¡Color cambiado!";
  }
</script>
```

¿Por qué usar `this` ?

-  No necesitas saber el ID del elemento
-  Puedes usar la misma función para múltiples elementos
-  El código es más flexible

# ? Pregunta Rápida

¿Qué hace este código?

```
<button onclick="cambiarTexto(this)">Presióname</button>

<script>
  function cambiarTexto(boton) {
    boton.innerText = "¡Ya me presionaste!";
  }
</script>
```

Opciones:

- A) No hace nada
- B) Muestra una alerta
- C) Cambia el texto del botón
- D) Elimina el botón

Piensa unos segundos... 🤔



## Respuesta

```
<button onclick="cambiarTexto(this)">Presióname</button>

<script>
  function cambiarTexto(boton) {
    boton.innerText = "¡Ya me presionaste!";
  }
</script>
```

Respuesta correcta: C) Cambia el texto del botón

¿Por qué?

- `this` pasa el botón como parámetro
- `boton.innerText` cambia el texto del botón
- De "Presióname" a "¡Ya me presionaste!"

## Evento Hover (onmouseover/onmouseout)



# ¿Qué es el Evento Hover?

Hover = pasar el mouse sobre un elemento

En realidad son DOS eventos:

- `onmouseover` → Cuando el mouse entra al elemento
- `onmouseout` → Cuando el mouse sale del elemento

Piensa en una puerta automática:

-  Cuando te acercas → se abre (`onmouseover`)
-  Cuando te alejas → se cierra (`onmouseout`)



# Ejemplo: Botón con Hover

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #miBoton {
      padding: 10px 20px;
      background-color: #3498db;
      color: white;
      border: none;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <button id="miBoton"
    onmouseover="cambiarMensaje('Mouse encima')"
    onmouseout="cambiarMensaje('Mouse afuera')">
    Pasa el mouse
  </button>
  <p id="mensaje">Mouse afuera</p>

  <script>
    function cambiarMensaje(texto) {
      document.getElementById("mensaje").innerText = texto;
    }
  </script>
</body>
</html>
```



# Uso Práctico: Vista Previa de Video



```
<video id="miVideo" src="video.mp4" muted></video>

<script>
  const video = document.getElementById("miVideo");

  // Reproducir al pasar el mouse
  video.addEventListener("mouseover", function() {
    this.play();
  });

  // Pausar al sacar el mouse
  video.addEventListener("mouseout", function() {
    this.pause();
  });
</script>
```

## ¿Dónde se usa esto?

-  Netflix, YouTube (previews)
-  Tiendas online (zoom en productos)

# Hay dos formas de agregar eventos

## Forma 1: En HTML (antigua)

```
<button onclick="miFuncion()">Clic</button>
```

## Forma 2: Con addEventListener (moderna)

```
<button id="miBoton">Clic</button>

<script>
    const boton = document.getElementById("miBoton");

    boton.addEventListener("click", function() {
        alert("¡Hiciste clic!");
    });
</script>
```

## Ventajas de addEventListener:

-  Separa HTML de JavaScript



# Ejercicio: Cambio de Color con Hover

## ¡Hora de practicar!

Crea un archivo `hover-color.html` con:

1. Un `<div>` con texto "Pasa el mouse aquí"
2. CSS: fondo gris, padding 20px, texto centrado
3. Con JavaScript:
  - Al pasar el mouse → fondo verde
  - Al sacar el mouse → fondo gris

## Pistas:

- Usa `onmouseover` y `onmouseout`
- Cambia `style.backgroundColor`



**Tiempo:** 10 minutos

## DOM: Query Selector





# ¿Qué es el DOM? Document Object Model

El DOM es como un árbol genealógico de tu página HTML

```
document
├── html
│   ├── head
│   │   └── title
│   └── body
│       ├── h1
│       ├── p
│       └── button
```

Con JavaScript podemos:

- 🔍 Buscar elementos
- ✎ Modificarlos
- + Agregar nuevos
- 🗑 Eliminarlos

# querySelector: Tu Selector Universal

Ya conoces selectores CSS, ¿verdad?

```
/* En CSS */  
#titulo { color: blue; }  
.clase { font-size: 20px; }  
div p { text-align: center; }
```

¡Puedes usar LOS MISMOS selectores en JavaScript!

```
// En JavaScript  
const titulo = document.querySelector("#titulo");  
const elemento = document.querySelector(".clase");  
const parrafo = document.querySelector("div p");
```

## Ejemplos de querySelector

```
<h1 id="titulo">Hola Mundo</h1>
<p class="texto">Párrafo 1</p>
<p class="texto">Párrafo 2</p>
<div class="caja">
  
</div>
```

```
// Por ID
const titulo = document.querySelector("#titulo");
// Por clase (devuelve el PRIMERO)
const primerTexto = document.querySelector(".texto");
// Por etiqueta
const h1 = document.querySelector("h1");
// Selectores combinados
const imagen = document.querySelector(".caja img");
```

## Importante: querySelector vs querySelectorAll

```
// querySelector → Devuelve el PRIMER elemento que coincida
const primerParrafo = document.querySelector(".texto");
console.log(primerParrafo); // <p class="texto">Párrafo 1</p>

// querySelectorAll → Devuelve TODOS los elementos
const todosLosParrafos = document.querySelectorAll(".texto");
console.log(todosLosParrafos); // NodeList [p.texto, p.texto]

// Puedes recorrer todos con un bucle
todosLosParrafos.forEach(function(parrafo) {
    console.log(parrafo.innerText);
});
```

# getElementById vs querySelector

## Dos formas de seleccionar por ID

```
// Forma antigua (solo para IDs)
const elemento1 = document.getElementById("miId");

// Forma moderna (para cualquier selector)
const elemento2 = document.querySelector("#miId");
```

### ¿Cuál usar?

- `getElementById` → Solo IDs, ligeramente más rápido
- `querySelector` → Más versátil, recomendado

💡 **Recomendación:** Usa `querySelector` para ser consistente

# ? Pregunta Rápida

¿Qué selecciona este código?

```
const elemento = document.querySelector(".caja button");
```

Opciones:

- A) Todos los botones dentro de elementos con clase "caja"
- B) El primer botón dentro de un elemento con clase "caja"
- C) Todos los elementos con clase "caja" que sean botones
- D) Causa un error

Piensa unos segundos... 🤔

## ✓ Respuesta

```
const elemento = document.querySelector(".caja button");
```

Respuesta correcta: B) El primer botón dentro de un elemento con clase "caja"

¿Por qué?

- `.caja button` es un selector de descendiente
- Busca un `<button>` dentro de un elemento con clase `caja`
- `querySelector` siempre devuelve EL PRIMERO que encuentra
- Si quisieras todos, usarías `querySelectorAll`

¡Excelente! 🎉



## Cambiar HTML y CSS desde JavaScript







# Cambiar el Texto de un Elemento

## Dos propiedades importantes:

```
const titulo = document.querySelector("h1");  
  
// innerText → Cambia solo el texto  
titulo.innerText = "Nuevo título";  
  
// innerHTML → Puede incluir etiquetas HTML  
titulo.innerHTML = "Título con <strong>negrita</strong>";
```

## ¿Cuál usar?

-  `innerText` → Para texto simple (más seguro)
-  `innerHTML` → Para HTML (úsalo con cuidado)



# Cambiar Estilos CSS

Accede a los estilos con `.style`

```
const parrafo = document.querySelector("p");  
  
parrafo.style.color = "blue"; // Cambiar color del texto  
  
parrafo.style.backgroundColor = "yellow"; // Cambiar fondo  
  
parrafo.style.fontSize = "24px"; // Cambiar tamaño de fuente  
  
parrafo.style.padding = "10px"; // Cambiar múltiples propiedades  
parrafo.style.borderRadius = "5px";
```

⚠ **Nota:** En CSS es `background-color`, en JS es `backgroundColor` (camelCase)

# Trabajar con Clases CSS, la mejor práctica

```
<style>
  .destacado {
    background-color: yellow;
    padding: 10px;
    font-weight: bold;
  }
</style>

<p id="texto">Este es un texto normal</p>

<script>
  const texto = document.querySelector("#texto");
  texto.classList.add("destacado"); // Agregar una clase




  texto.classList.remove("destacado"); // Quitar una clase
  // Alternar (toggle): si tiene la clase, la quita; si no, la agrega
  texto.classList.toggle("destacado");
</script>
```



# Ejemplo Completo: Modo Oscuro

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      transition: all 0.3s ease;
    }
    .dark-mode {
      background-color: #333;
      color: white;
    }
  </style>
</head>
<body>
  <button onclick="toggleModo()">Cambiar Modo</button>
  <script>
    function toggleModo() {
      document.body.classList.toggle("dark-mode");
    }
  </script>
</body>
</html>
```

## ¿Por qué usar clases?

-  Agrupa todos los estilos en un solo lugar
-  Fácil de activar/desactivar
-  Más limpio y mantenible



# Cambiar Atributos HTML

```
const imagen = document.querySelector("img");  
imagen.src = "nueva-imagen.jpg"; // Cambiar la fuente de la imagen  
imagen.alt = "Nueva imagen descriptiva"; // Cambiar el texto alternativo  
  
const enlace = document.querySelector("a"); // Cambiar cualquier atributo  
enlace.href = "https://nuevo-sitio.com";  
enlace.target = "_blank"; // Abrir en nueva pestaña
```

## Atributos comunes:

- `src` → Fuente de imágenes, videos
- `href` → Destino de enlaces
- `value` → Valor de inputs
- `disabled` → Desactivar elementos



# Ejercicio: Cambiar Imagen al Hacer Clic

¡Hora de practicar!

Crea un archivo `cambiar-imagen.html` con:

1. Una imagen (usa <https://picsum.photos/300>)
2. Un botón que diga "Cambiar Imagen"
3. Al hacer clic, cambia la imagen a otra URL diferente

Pistas:

- Usa `onclick` en el botón
- Usa `querySelector` para seleccionar la imagen
- Cambia el atributo `src`



## Eventos de Entrada



## oninput: Detectar al Escribir

```
<!DOCTYPE html>
<html>
<body>
  <input type="text" id="nombre" placeholder="Escribe tu nombre">
  <h2>Hola, <span id="saludo">visitante</span></h2>

  <script>
    const inputNombre = document.querySelector("#nombre");
    const spanSaludo = document.querySelector("#saludo");

    inputNombre.addEventListener("input", function() {
      spanSaludo.innerText = this.value || "visitante";
    });
  </script>
</body>
</html>
```

¿Qué hace `this.value` ?

- `this` → El input que disparó el evento
- `value` → El texto que contiene

# onchange: Detectar Cambios

Se activa cuando terminas de cambiar el valor

```
<label>¿Qué prefieres comer?</label>
<select id="comida">
  <option value="">Selecciona...</option>
  <option value="Pizza">Pizza</option>
  <option value="Sushi">Sushi</option>
  <option value="Tacos">Tacos</option>
</select>

<script>
  const select = document.querySelector("#comida");

  select.addEventListener("change", function() {
    alert("¡Elegiste " + this.value + "!");
  });
</script>
```

## vs oninput vs onchange

Característica	oninput	onchange
¿Cuándo se activa?	Cada vez que cambia	Al terminar de editar
Uso en <code>&lt;input&gt;</code>	Cada letra	Al salir del campo
Uso en <code>&lt;select&gt;</code>	No aplica	Al cambiar opción
Mejor para	Búsqueda en tiempo real	Validación de formularios

```
// oninput → Búsqueda mientras escribes
inputBusqueda.addEventListener("input", buscarEnTiempoReal);

// onchange → Validar cuando termina
inputEmail.addEventListener("change", validarEmail);
```



# Ejemplo: Contador de Caracteres

```
<!DOCTYPE html>
<html>
<body>
  <textarea id="mensaje" rows="4" cols="50"
    placeholder="Escribe tu mensaje..."></textarea>
  <p>Caracteres: <span id="contador">0</span></p>

  <script>
    const textarea = document.querySelector("#mensaje");
    const contador = document.querySelector("#contador");

    textarea.addEventListener("input", function() {
      const cantidad = this.value.length;
      contador.innerText = cantidad;

      // Cambiar color si excede 100 caracteres
      if (cantidad > 100) {
        contador.style.color = "red";
      } else {
        contador.style.color = "black";
      }
    });
  </script>
</body>
</html>
```



## Ejercicio: Selector de Temperatura

Crea un archivo `temperatura.html` con:

1. Un `<select>` con opciones: "Buenos Aires", "Madrid", "Nueva York"
2. Un `<p>` que muestre la temperatura
3. Al cambiar la ciudad, muestra una temperatura diferente: Buenos Aires: 25°C, Madrid: 18°C, Nueva York: 12°C

Pistas:




- Usa `onchange`
- Usa `querySelector` para actualizar el texto
- Puedes usar un `if/else` o un objeto

# **setTimeout: Retrasos en JavaScript**

# ¿Qué es setTimeout?

## Ejecuta código después de un tiempo

Piensa en un temporizador de cocina:

-  Pones 5 minutos
-  Mientras tanto, haces otras cosas
-  Cuando termina, suena la alarma

```
// Ejecutar después de 3 segundos (3000 milisegundos)
setTimeout(function() {
    alert("¡Han pasado 3 segundos!");
}, 3000);

console.log("Este mensaje aparece inmediatamente");
```



## 12 Sintaxis de setTimeout

```
setTimeout(funcion, tiempoEnMilisegundos);
```

Dos formas de escribirlo:

```
// Forma 1: Función anónima
setTimeout(function() {
    console.log("¡Hola!");
}, 2000);

// Forma 2: Función con nombre
function mostrarMensaje() {
    console.log("¡Hola!");
}
setTimeout(mostrarMensaje, 2000); // SIN paréntesis ()
```

⚠ **Importante:** NO pongas paréntesis `()` cuando pases la función



# Orden de Ejecución

## JavaScript no espera a setTimeout

```
console.log("1. Inicio");

setTimeout(function() {
  console.log("2. Después de 2 segundos");
}, 2000);

console.log("3. Final");
```

¿En qué orden se imprime?

1. Inicio
2. Final
3. Después de 2 segundos

## ¿Por qué?

- JavaScript continúa ejecutando sin esperar
- `setTimeout` "programa" la función para después
- El código sigue corriendo normalmente



## Ejemplo: Notificación con Retraso

```
<!DOCTYPE html>
<html>
<body>
  <button onclick="mostrarNotificacion()">Mostrar Notificación</button>
  <p id="notificacion" style="display: none; color: green;">
    ✓ ¡Operación exitosa!
  </p>
  <script>
    function mostrarNotificacion() {
      const notif = document.querySelector("#notificacion");





      // Mostrar la notificación
      notif.style.display = "block";

      // Ocultarla después de 3 segundos
      setTimeout(function() {
        notif.style.display = "none";
      }, 3000);
    }
  </script>
</body>
</html>
```

# Ejemplo: Carga de Página

```
// Mostrar mensaje de carga al cargar la página
window.addEventListener("load", function() {
    alert("Cargando datos...");

    // Simular carga de 2 segundos
    setTimeout(function() {
        alert("¡Datos cargados!");
    }, 2000);
});
```

-  Simuladores de carga
-  Notificaciones temporales
-  Animaciones con delay
-  Timeouts de sesión

## ✗ Cancelar un setTimeout

```
// Guardar el timeout en una variable
const miTimeout = setTimeout(function() {
    alert("Este mensaje NO aparecerá");
}, 5000);

// Cancelar el timeout antes de que se ejecute
clearTimeout(miTimeout);
```

## Ejemplo práctico:

```
let timeout;

function reiniciarTemporizador() {
  // Cancelar el anterior si existe
  clearTimeout(timeout);
  // Crear uno nuevo
  timeout = setTimeout(function() {
    alert("¡Se acabó el tiempo!");
  }, 10000);
}
```

## Ejercicio: Mensaje Temporal

¡Hora de practicar!

Crea un archivo `mensaje-temporal.html` con:

1. Un botón que diga "Mostrar Mensaje"
2. Un `<div>` oculto con un mensaje
3. Al hacer clic:
  - Muestra el mensaje
  - Después de 5 segundos, ocúltalo automáticamente



## Pistas:

- Usa `onclick` en el botón
- Usa `setTimeout` para ocultar
- Cambia `style.display` entre "block" y "none"

 **Tiempo:** 15 minutos

# Ejercicios Prácticos



# Ejercicio 1: Botón Modo Oscuro/Claro

Crea un archivo `modo-tema.html` con:

1. Un botón que diga "Modo Oscuro"
2. Al hacer clic, debe:
  - Cambiar el fondo del body a negro y texto a blanco
  - Cambiar el texto del botón a "Modo Claro"
3. Al volver a hacer clic, debe revertir los cambios

**Conceptos practicados:**

- Eventos onclick
- `classList.toggle()`
- Cambiar `innerText`



## Ejercicio 2: Card con Botón que se Oculta

Crea un archivo `card-ocultar.html` con:

1. Una tarjeta (div) con:
  - Un título "Suscríbete al Newsletter"
  - Un párrafo "Recibe las últimas noticias"
  - Un botón "Cerrar"
2. Al hacer clic en "Cerrar", el botón debe desaparecer

**Conceptos practicados:**

- Evento onclick
- `querySelector`
- `style.display = "none"`



## Ejercicio 3: Contador con Mensaje

1. Un título que muestre "Clicks: 0"
2. Un botón "Incrementar"
3. Al hacer clic:
  - Incrementa el número
  - Si llega a 5, muestra un mensaje "¡Bien hecho!"
  - Si llega a 10, muestra "¡Eres un experto!"

### Conceptos practicados:

- Variables
- Condicionales
- innerText
- Eventos



## Ejercicio 4: Cambiar Texto de Párrafo

Crea un archivo `cambiar-parrafo.html` con:

1. Un párrafo con texto "Este es el texto original"
2. Dos botones: "Texto 1" y "Texto 2"
3. Al hacer clic en cada botón, cambia el texto del párrafo

**Conceptos practicados:**

- `querySelector`
- Múltiples funciones
- `innerText`
- Pasar parámetros



## Recursos Adicionales

# Documentación y Tutoriales

## Recursos oficiales

-  [MDN Web Docs - JavaScript](#)
-  [MDN - Eventos](#)
-  [MDN - DOM](#)




## Tutoriales interactivos

-  [JavaScript.info](#)
-  [FreeCodeCamp - JavaScript](#)
-  [Codecademy - JavaScript](#)



# Herramientas Útiles

## Para practicar en línea








-  [CodePen](#) - Editor online
-  [JSFiddle](#) - Prueba código rápidamente
-  [JS Bin](#) - Colaboración en tiempo real

## Extensiones de VSCode

- ✨ Live Server - Ver cambios en tiempo real
- ✨ JavaScript (ES6) code snippets
- ✨ Prettier - Formatear código

# Desafíos para Practicar

## Proyectos pequeños para mejorar

1.  Selector de temas - Múltiples temas de color
  2.  Lista de tareas - Agregar y eliminar items
  3.  Juego de dados - Números aleatorios
  4.  Galería de imágenes - Navegación con botones
  5.  Cronómetro - Cuenta regresiva
  6.  Calculadora simple - Operaciones básicas
-  **Consejo:** Empieza con proyectos pequeños y ve aumentando la complejidad



## Resumen de Conceptos Clave

## 1. Funciones

```
function nombreFuncion(parametro) {  
    return valor;  
}
```

## 2. Eventos

```
elemento.addEventListener("click", function() {  
    // código  
});
```

## 3. Seleccionar elementos

```
const elemento = document.querySelector("#id");
```

## 4. Modificar elementos

```
elemento.innerText = "nuevo texto";
```

## Conceptos Esenciales

Concepto	Descripción	Ejemplo
<b>Función</b>	Bloque de código reutilizable	<code>function saludar() {}</code>
<b>onclick</b>	Evento de clic	<code>onclick="miFuncion()"</code>
<b>this</b>	Elemento que dispara el evento	<code>function(this) {}</code>
<b>querySelector</b>	Seleccionar elementos	<code>querySelector("#id")</code>
<b>innerText</b>	Cambiar texto	<code>elemento.innerText = "Hola"</code>
<b>classList</b>	Manipular clases	<code>elemento.classList.add()</code>
<b>setTimeout</b>	Retraso de ejecución	<code>setTimeout(fn, 1000)</code>



# Trucos y Mejores Prácticas

## 1. Siempre usa nombres descriptivos

```
// ❌ Mal  
function f1() {}  
  
// ✅ Bien  
function mostrarMensaje() {}
```

## 2. Prefiere addEventListener sobre onclick en HTML









```
// ✅ Mejor práctica  
boton.addEventListener("click", miFuncion);
```

## 3. Usa classList en lugar de style directo

```
// ✅ Mejor práctica  
elemento.classList.toggle("activo");
```

# Checklist de la Clase

## Asegúrate de saber:

-  Crear y llamar funciones con parámetros
-  Usar eventos onclick, onmouseover, onmouseout
-  Seleccionar elementos con querySelector
-  Cambiar texto con innerText/innerHTML
-  Modificar estilos con classList
-  Trabajar con eventos de entrada (oninput, onchange)
-  Usar setTimeout para retrasos
-  Entender qué es `this` en eventos

Si dominas estos conceptos, ¡estás listo para seguir! 







## Próximos Pasos





# ¿Qué Viene Después?






En las siguientes clases veremos:

-  Bucles con el DOM - `querySelectorAll` y `forEach`
-  Arrays y objetos avanzados - Manipular datos complejos
-  Animaciones con JavaScript - Transiciones y efectos
-  Peticiones a APIs - Obtener datos externos
-  `localStorage` - Guardar datos en el navegador
-  Proyectos interactivos - Mini juegos y apps

¡El JavaScript recién comienza! 

## Proyecto sugerido: Dashboard Personal

Crea una página con:







-  Botón de modo oscuro/claro
-  Reloj que muestre la hora actual
-  Contador de tareas completadas
-  Selector de color de tema
-  Mensaje de bienvenida personalizado

## Recursos que necesitarás:

- Eventos onclick
- `querySelector`
- `setTimeout`
- `classList`

# Consejos Finales

## Para dominar JavaScript:

1.  Practica todos los días - Aunque sea 15 minutos
2.  Aprende a depurar - Usa console.log()
3.  Lee código de otros - GitHub, CodePen
4.  Construye proyectos - La mejor forma de aprender
5.  Haz preguntas - No hay preguntas tontas
6.  Documenta tu código - Tu yo del futuro te lo agradecerá

¡El camino del desarrollador es de práctica constante! 

## 1. Olvidar los paréntesis al llamar funciones

```
// ❌ Mal
setTimeout(miFuncion(), 1000); // Se ejecuta inmediatamente

// ✅ Bien
setTimeout(miFuncion, 1000); // Se ejecuta después del delay
```

## 2. No usar == en comparaciones

```
// ❌ Puede dar resultados inesperados
if (valor == "5") {}

// ✅ Mejor
if (valor === "5") {}
```

## 3. Olvidar que querySelector solo devuelve el primero

```
// Si quieres todos, usa querySelectorAll
```

 ¡Felicidades!

Ya dominas Onclick y el DOM

Has aprendido a hacer páginas web interactivas

## ? Preguntas

¿Alguna duda sobre Onclick o el DOM?

 ¡Excelente Trabajo!

¡Nos vemos en la siguiente clase con más JavaScript!

No olvides completar todos los ejercicios 