UT3.2. Introducción a JavaScript en HTML

Lenguajes de Marcas - 1ºDAW

Virginia Zornoza

1. ¿Qué es JavaScript?

- Lenguaje de programación interpretado
- Funciona del lado del cliente (aunque puede usarse en el servidor con Node.js)
- Permite la interacción dinámica con las páginas web.

2. Cómo incluir JavaScript en HTML

- **En línea**: usando el atributo onclick, onmouseover, etc.

```
<button onclick="alert('¡Hola!')">Haz clic aquí</button>
```

- **Interno**: Entre etiquetas <script> en el mismo archivo HTML

```
<script>
  console.log('Hola desde JavaScript interno');
</script>
```

- **Externo**: Referenciando un archivo .js.

```
<script src="script.js"></script>
```

3. Sintaxis básica

- Variables: var, let, const

```
let mensaje = 'Hola';
const PI = 3.14;
var nombre = 'Juan';
```

- Tipos de datos: number, string, boolean, object, array, null, undefined
- Operadores básicos: aritméticos, comparación, lógicos

```
let suma = 5 + 3;  // 8
let mayor = 10 > 5; // true
```

4. Interacción con el DOM

- Seleccionar elementos:

```
document.getElementById('id');
document.querySelector('.clase');
```

-

Método	Selector	Devuelve	En Vivo	Uso Frecuente
`getElementById`	ID (`id`)	Elemento único	No	Seleccionar un único elemento.
`getElementsByClassName`	Clase (`class`)	HTMLCollection	Sí	Seleccionar múltiples elementos por clase.
`getElementsByTagName`	Etiqueta (`tag`)	HTMLCollection	Sí	Seleccionar múltiples elementos por etiqueta.
`querySelector`	Selector CSS	Primer elemento	No	Seleccionar un elemento con flexibilidad CSS.
`querySelectorAll`	Selector CSS	NodeList estáti	No	Seleccionar múltiples elementos con flexibilidad CSS.

4. Interacción con el DOM

- Seleccionar elementos:
  - document.getElementsByClassName()
  - Qué hace: selecciona todos los elementos que tienen una clase específica.
  - Resultado: Devuelve una colección en vivo (HTMLCollection) de todos los elementos con la clase indicada.
  - Nota: Para acceder a un elemento específico, usa índices: elementos[0].

```
let elementos = document.getElementsByClassName('mi-clase');
console.log(elementos); // Colección de elementos con la clase 'mi-clase'
```

-

4. Interacción con el DOM

- Seleccionar elementos:
  - document.getElementsByTagName()
  - Qué hace: Selecciona todos los elementos con una etiqueta específica.
  - Resultado: Devuelve una colección en vivo de todos los elementos con esa etiqueta.

```
let parrafos = document.getElementsByTagName('p');
console.log(parrafos); // Colección de elementos <p>
```

4. Interacción con el DOM

- Seleccionar elementos:
  - document.querySelector()
  - Qué hace: Selecciona el primer elemento que coincide con un selector CSS.
  - Resultado: Devuelve el elemento encontrado o null si no encuentra ninguno.

```
let elemento = document.querySelector('.mi-clase');
console.log(elemento); // Primer elemento con la clase 'mi-clase'
```

-

4. Interacción con el DOM

- Seleccionar elementos:
  - document.querySelectorAll()
  - Qué hace: selecciona **todos** los elementos que coinciden con un selector CSS.
  - Resultado: devuelve una **NodeList** estática (no en vivo) de los elementos.

```
let elementos = document.querySelectorAll('div.mi-clase');
console.log(elementos); // Todos los <div> con la clase 'mi-clase'
```

Nota: Puedes recorrer la NodeList con un bucle forEach

```
elementos.forEach(elemento => console.log(elemento));
```

4. Interacción con el DOM

¿Qué método usar?

- Usa querySelector o querySelectorAll si necesitas selecciones complejas basadas en CSS.
- Usa getElementById para selecciones rápidas por ID.
- Evita usar getElementsByClassName y getElementsByTagName si necesitas manipular una lista estática; prefiere querySelectorAll.

4. Interacción con el DOM

html

```html
<div id="uno" class="mi-clase">Div 1</div>
<div id="dos" class="mi-clase">Div 2</div>
<p>Parágrafo 1</p>
<p class="mi-clase">Parágrafo 2</p>
```

javascript

```javascript
// Seleccionar por ID
let idUno = document.getElementById('uno');
console.log(idUno.innerText); // "Div 1"

// Seleccionar por clase
let claseElementos = document.getElementsByClassName('mi-clase');
console.log(claseElementos[1].innerText); // "Div 2"

// Seleccionar por etiqueta
let parrafos = document.getElementsByTagName('p');
console.log(parrafos.length); // 2

// Seleccionar con querySelector
let primerDiv = document.querySelector('div.mi-clase');
console.log(primerDiv.innerText); // "Div 1"

// Seleccionar con querySelectorAll
let todosLosParrafos = document.querySelectorAll('p');
console.log(todosLosParrafos.length); // 2
```

4. Interacción con el DOM

- Seleccionar elementos
- Modificar contenido:
  - innerText: cambia contenido de texto del elemento, respetando el formato visual. Ideal para modificar solo texto visible.

```
document.getElementById('miElemento').innerText = 'Nuevo texto';
```

  - innerHTML: cambia el contenido HTML completo del elemento, incluyendo etiquetas HTML.

```
document.getElementById('miElemento').innerHTML = '<strong>Texto en negrita</strong>';
```

4. Interacción con el DOM

- Seleccionar elementos
- Modificar contenido
- Modificar estilos:

```
document.getElementById('miElemento').style.color = 'red';
document.getElementById('miElemento').style.backgroundColor = 'blue';
document.getElementById('miElemento').style.fontSize = '20px';
```

Nota: las propiedades CSS con guiones (-) se escriben en camelCase en JavaScript (por ejemplo, background-color es backgroundColor)

5. Eventos básicos

- Asociar eventos:

```
document.getElementById('miBoton').addEventListener('click', function() {
  alert('¡Botón clickeado!');
});
```

1. `document.getElementById('miBoton')`

- **Qué hace**: Selecciona el elemento del DOM con el ID `miBoton`.
- **Cómo funciona**: Si tienes un elemento en tu HTML como este:

  ```
  <button id="miBoton">Haz clic aquí</button>
  ```
- La función `document.getElementById('miBoton')` buscará y devolverá este botón

5. Eventos básicos

- Asociar eventos:

```
document.getElementById('miBoton').addEventListener('click', function() {
  alert('¡Botón clickeado!');
});
```

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

- **Qué hace**: Añade un "escuchador de eventos" al elemento seleccionado.
- **Detalles**:
  - El primer argumento (`'click'`) es el tipo de evento que queremos escuchar. En este caso, el evento ocurre cuando el usuario hace clic en el botón.
  - El segundo argumento es una función que se ejecutará **cuando ocurra ese evento**.

5. Eventos básicos

- Asociar eventos:

```
document.getElementById('miBoton').addEventListener('click', function() {
  alert('¡Botón clickeado!');
});
```

3. `function() { alert('¡Botón clickeado!'); }`

- **Qué hace**: Define una **función anónima** que se ejecutará cuando ocurra el evento (cuando el botón sea clickeado).
- **Acción específica**:
  - En este caso, la función muestra un cuadro de alerta con el mensaje **"¡Botón clickeado!"**.

```
document.getElementById('miBoton').addEventListener('click', function(
  alert('¡Botón clickeado!');
});
```

5. Eventos básicos

Ventajas de usar `addEventListener`

1. **Múltiples eventos**: Puedes añadir varios escuchadores al mismo elemento sin sobrescribir.
   javascript

```
document.getElementById('miBoton').addEventListener('click', function() {
  console.log('Evento 1 ejecutado');
});
document.getElementById('miBoton').addEventListener('click', function() {
  console.log('Evento 2 ejecutado');
});
```

   Ambos eventos se ejecutarán al hacer clic.

2. **Separación del HTML y JavaScript**: Es mejor práctica mantener el código JavaScript separado del HTML, en lugar de usar `onclick` directamente en el atributo del HTML.
3. **Control de eventos avanzados**: Puedes usar otras características como evitar el comportamiento por defecto de un evento, propagación, etc.

6. Ejemplos básicos

Ejercicio 1: Hola, mundo

1. Crea un archivo HTML con un botón que muestre "¡Hola, mundo!" en un alert al hacer clic.

6. Ejemplos básicos

Ejercicio 2: Cambiar contenido

- Crea una página con un párrafo y un botón.
- Al hacer clic en el botón, el texto del párrafo debe cambiar a "Texto modificado".

6. Ejemplos básicos

Ejercicio 3: Estilo dinámico

- Crea un archivo HTML con un botón y un div.
- Al hacer clic en el botón, el fondo del div debe cambiar de color.

6. Ejemplos básicos

Ejercicio 1: Hola, mundo

Solución:

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Hola Mundo</title>
</head>
<body>
  <button onclick="saludar()">Haz clic aquí</button>
  <script>
    function saludar() {
      alert('¡Hola, mundo!');
    }
  </script>
</body>
</html>
```

6. Ejemplos básicos

Ejercicio 2. Cambiar contenid

Solución:

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Cambiar contenido</title>
</head>
<body>
  <p id="miParrafo">Texto original</p>
  <button onclick="cambiarTexto()">Cambiar texto</button>
  <script>
    function cambiarTexto() {
      document.getElementById('miParrafo').innerText = 'Texto modificado';
    }
  </script>
</body>
</html>
```

6. Ejemplos básico

Ejercicio 3. Estilo dinámico

Solución:

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Estilo dinámico</title>
  <style>
    #miDiv {
      width: 200px;
      height: 100px;
      background-color: lightgray;
    }
  </style>
</head>
<body>
  <div id="miDiv"></div>
  <button onclick="cambiarColor()">Cambiar color</button>
  <script>
    function cambiarColor() {
      document.getElementById('miDiv').style.backgroundColor = 'lightblue';
    }
  </script>
</body>
</html>
```