UD 3
El DOM

DESARROLLO WEB EN ENTORNO CLIENTE

Técnico de Grado Superior Desarrollo de Aplicaciones Web

2024-25

J. Mario Rodríguez
jrodper183e@g.educaand.es

Contenidos

- El modelo de objetos del documento (DOM).
- Árbol de nodos.
- Tipos de nodos.
- Acceso directo a los nodos. Acceso al documento desde código.
- Creación y eliminación de nodos.
- Acceso directo a los atributos y propiedades de los nodos. Modificación de nodos.
- Programación de eventos.
- Desarrollo de aplicaciones Web en capas.

El modelo de objetos del documento (DOM)

El DOM es una representación jerárquica de un documento HTML o XML.

Permite al lenguaje de programación en el cliente acceder y modificar…

- el contenido
- la estructura
- el estilo

… del documento mientras se está ejecutando en el navegador.

El DOM convierte un documento en un conjunto de nodos, que pueden ser manipulados de manera dinámica.

El modelo de objetos del documento (DOM)

- Documento:

  El archivo HTML o XML que se está visualizando en el navegador.

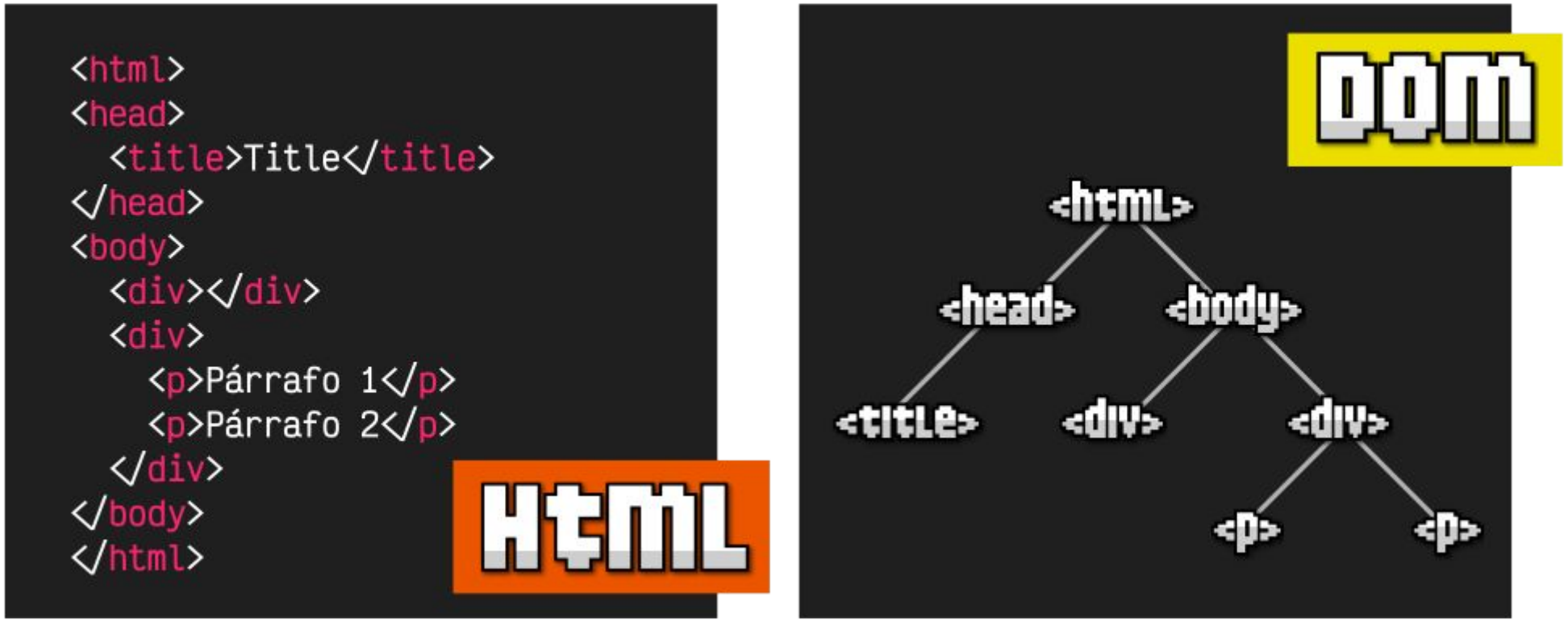
- Modelo:

  Representa la estructura jerárquica en forma de árbol.

- Objetos:

  Cada parte del documento se trata como un objeto que puede ser manipulado (etiquetas, texto, atributos, etc.).

Árbol de nodos



https://lenguajejs.com/dom/introduccion/que-es/

Árbol de nodos

El DOM organiza el documento como un árbol de nodos. Cada elemento del documento (etiquetas, texto, comentarios, etc.) es un nodo. Los nodos tienen relaciones entre sí: el nodo raíz es el documento, y cada etiqueta dentro de él es un nodo hijo. Cada nodo puede tener múltiples nodos hijos.

Tipos de **relaciones entre nodos**:

- **Padre:** Un nodo que contiene a otros nodos.
- **Hijo:** Un nodo que es contenido por otro nodo.
- **Hermanos:** Nodos que están al mismo nivel.

Tipos de nodos

Desde el navegador, la forma de acceder al DOM es a través de un objeto Javascript llamado document, que representa el árbol DOM de la página de la pestaña del navegador donde nos encontramos.

En su interior pueden existir varios tipos de elementos, principalmente serán objetos de tipo:

- element es la representación genérica de una etiqueta: *HTMLElement*.
- node es una unidad más básica, la cuál puede ser element o un nodo de texto.

https://lenguajejs.com/dom/introduccion/que-es/

Tipos de nodos

Todos los elementos HTML, dependiendo del elemento que sean, tendrán un tipo de dato específico asociado, por ejemplo:

Tipo de dato específico	Etiqueta	Descripción
ELEMENT HTMLDivElement		Etiqueta divisoria (en bloque).
ELEMENT HTMLSpanElement		Etiqueta divisoria (en línea).
ELEMENT HTMLImageElement	<img>	Imagen.
ELEMENT HTMLAudioElement	<audio>	Contenedor de audio.

Existen muchos tipos de datos específicos, prácticamente uno por cada etiqueta HTML.

https://lenguajejs.com/dom/introduccion/que-es/

Tipos de nodos

Además, podemos considerar al contenido y atributos y comentarios como nodos del DOM:

Tendríamos entonces:

- Nodo de Elemento:
  Representa etiquetas HTML
- Nodo de Texto:
  Representa el contenido textual dentro de un elemento.
- Nodo de Atributo:
  Representa un atributo dentro de una etiqueta HTML.
- Nodo de Comentario:
  Representa los comentarios HTML.

```
<!DOCTYPE html>
<html>
<body>
    <!-- Este es un comentario -->
    <h1 id="titulo">Hola Mundo</h1>
</body>
</html>
```

Tipos de nodos

Reto: Identificar nodos

Parte de un código html básico e identifica los nodos con JavaScript*

Por ejemplo:

```html
<!DOCTYPE html>
<html>
<body>
    <!-- Este es un comentario -->
    <h1 id="titulo">Hola Mundo</h1>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="es">
<body>
    <h1 id="titulo">Hola Mundo</h1>
    <p>Este es un párrafo</p>

    <script>
        let h1 = document.getElementById("titulo");
        console.log(h1.nodeType); // 1: Nodo de elemento
        console.log(h1.firstChild.nodeType); // 3: Nodo de texto
    </script>
</body>
</html>
```

* usa console.log()

Acceso directo a los nodos

Acceso a los nodos del DOM directamente utilizando métodos proporcionados por JavaScript:

- `document.getElementById()`

  Selecciona un elemento por su id.
- `document.getElementsByClassName()`

  Selecciona todos los elementos de una clase.
- `document.getElementsByTagName()`

  Selecciona todos los elementos de un tipo de etiqueta.

Acceso al documento desde código

```
let titulo = document.getElementById('titulo');
console.log(titulo.innerHTML); // Imprime: Hola Mundo
```

```
<!DOCTYPE html>
<html>
<body>
    <p id="texto">Texto original</p>

    <script>
        let parrafo = document.getElementById("texto");
        parrafo.innerHTML = "Texto modificado desde JavaScript";
    </script>
</body>
</html>
```

Creación y eliminación de nodos

Crear y eliminar nodos dinámicamente:

- Crear nodos:
  `document.createElement()` crea un nuevo elemento.
- Eliminar nodos:
  `parentNode.removeChild()` elimina un nodo.

```
let nuevoParrafo = document.createElement('p');
nuevoParrafo.textContent = 'Este es un nuevo párrafo.';
document.body.appendChild(nuevoParrafo);
```

Creación y eliminación de nodos

Reto: Crear y eliminar nodos

- Añade un nuevo elemento <li> a una lista no ordenada
- Elimínalo después de 5 segundos.

```
<!DOCTYPE html>
<html>
<body>
    <ul id="miLista">
        <li>Elemento 1</li>
        <li>Elemento 2</li>
    </ul>

    <script>
        // Crear nuevo elemento
        let nuevoElemento = document.createElement("li");
        nuevoElemento.textContent = "Elemento 3";
        document.getElementById("miLista").appendChild(nuevoElemento);

        // Eliminar después de 5 segundos
        setTimeout(() => {
            document.getElementById("miLista").removeChild(nuevoElemento);
        }, 5000);
    </script>
</body>
</html>
```

Acceso directo a los atributos y propiedades de los nodos

Podemos modificar los atributos y propiedades de los nodos:

- Atributos:
  - `element.setAttribute(attr, value)`
  - `element.id` accediendo directamente
- Propiedades:

  Como innerHTML, textContent, className, etc.

```
let img = document.createElement('img');
img.setAttribute('src', 'imagen.jpg');
img.alt = "Descripción de la imagen";
document.body.appendChild(img);
```

Modificación de nodos

Reto: cambiar atributos

Crea un botón que, al hacer clic, cambie el src de una imagen.

```html
<!DOCTYPE html>
<html>
<body>
    <img id="miImagen" src="imagen1.jpg" alt="Imagen 1">
    <button onclick="cambiarImagen()">Cambiar imagen</button>

    <script>
        function cambiarImagen() {
            document.getElementById("miImagen").src = "imagen2.jpg";
        }
    </script>
</body>
</html>
```

Programación de eventos

Podemos manejar eventos en los nodos del DOM, como clics, movimiento del ratón, etc., usando `addEventListener()`.

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

Programación de eventos

Reto: Programar un evento de doble clic

Crea un botón que al hacer doble clic cambie su color.

```
<!DOCTYPE html>
<html>
<body>
    <button id="botonColor">Doble clic para cambiar color</button>

    <script>
        document.getElementById("botonColor").addEventListener('dblclick', function() {
            this.style.backgroundColor = "red";
        });
    </script>
</body>
</html>
```

Programación de eventos

También podemos crear eventos

```
let event = new Event(type[, options]);
```

y lanzar eventos

```
<button id="elem" onclick="alert('Clic!');">Click automático</button>

<script>
  let event = new Event("click");
  elem.dispatchEvent(event);
</script>
```

https://es.javascript.info/dispatch-events

Programación de eventos

Reto: Simulación de click con el teclado

Crea un botón en la pantalla que realice una acción (por ejemplo, mostrar un mensaje).

Además de poder hacer clic con el ratón, debes permitir que al pulsar la tecla *Enter* se simule un clic en el botón. Utiliza el método dispatchEvent para disparar el evento de clic manualmente cuando el usuario presione la tecla *Enter*.

Programación de eventos

Reto: Simulación de click con el teclado

Crea un botón en la pantalla que realice una acción (por ejemplo, mostrar un mensaje).

Además de poder hacer clic con el ratón, debes permitir que al pulsar la tecla *Enter* se simule un clic en el botón. Utiliza el método dispatchEvent para disparar el evento de clic manualmente cuando el usuario presione la tecla *Enter*.

Reto: Simulación de click con el teclado

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Simulación de clic</title>
</head>
<body>
    <h1>Simulación de clic</h1>
    <button id="miBoton">Haz clic o pulsa Enter</button>
    <p id="mensaje"></p>

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

        boton.addEventListener("click", function() {
            mensaje.textContent = "¡Clic!";
        });

        document.addEventListener("keydown", function(event) {
            if (event.key === "Enter") {
                let clickEvent = new Event('click');
                boton.dispatchEvent(clickEvent);
            }
        });
    </script>
</body>
</html>
```

Desarrollo de aplicaciones Web en capas

Las capas separan las distintas responsabilidades de una aplicación web.

Las capas típicas son:

- Capa de presentación (UI)
  Interfaz que interactúa con el usuario (HTML, CSS).
- Capa de lógica de negocio (JS)
  Donde ocurre la lógica que gestiona la interacción del usuario.
- Capa de acceso a datos (API, *backend*)
  Donde se manejan los datos que la lógica necesita.

Desarrollo de aplicaciones Web en capas

Ejemplo de una página que recibe un input, lo procesa y actualiza la interfaz:

```html
<!DOCTYPE html>
<html>
<body>
    <input type="text" id="nombre" placeholder="Escribe tu nombre">
    <button onclick="mostrarSaludo()">Saludar</button>
    <p id="saludo"></p>

    <script>
        function mostrarSaludo() {
            let nombre = document.getElementById("nombre").value;
            document.getElementById("saludo").textContent = "Hola, " + nombre;
        }
    </script>
</body>
</html>
```