FORMACIÓN PROFESIONAL
MONTECASTELO

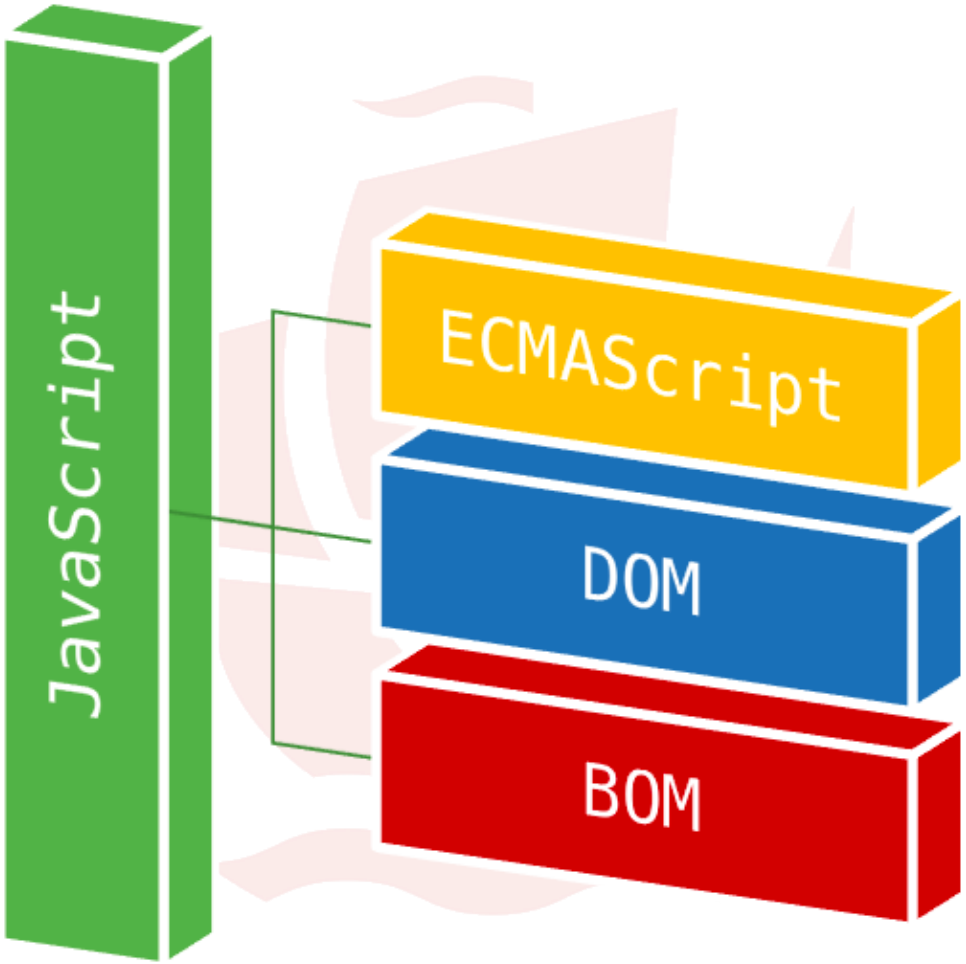
BOM y DOM

Desarrollo Web en Entorno Cliente

Ciclo Superior de Desarrollo de Aplicaciones Web

2020/2021

BOM y DOM



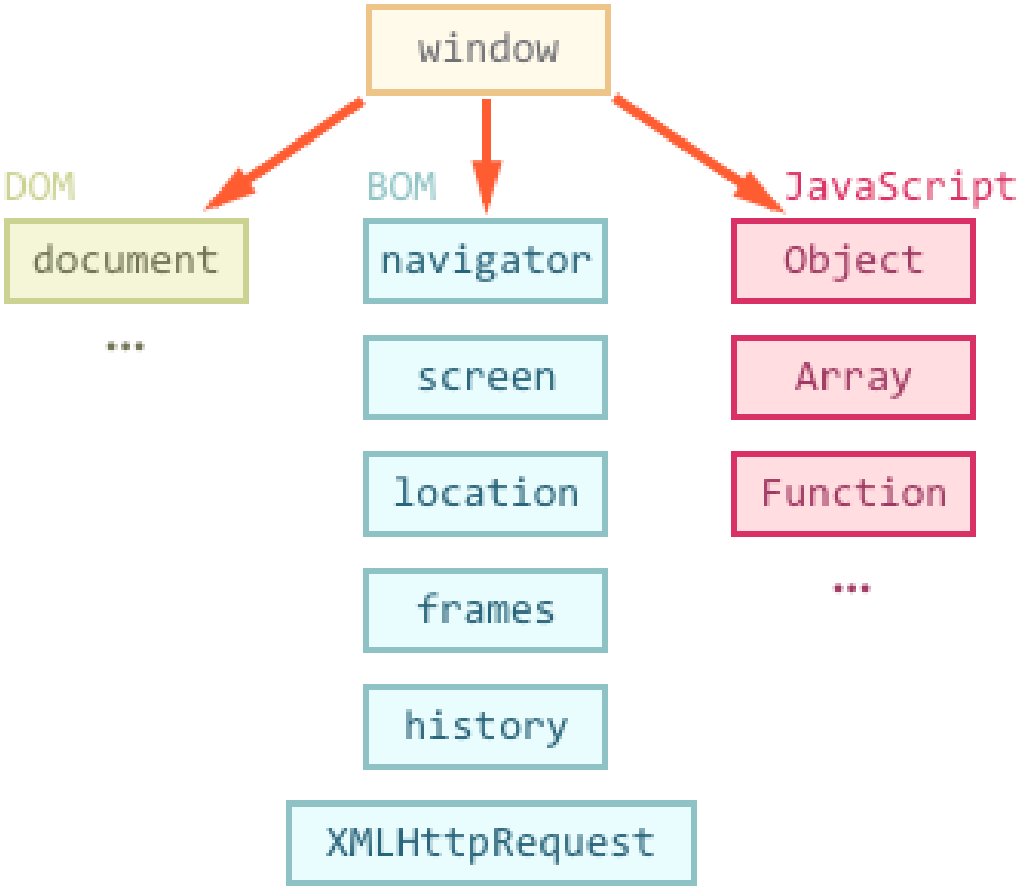
Objeto `Window`

- La programación de JavaScript involucra tres bloques que interactúan entre sí.
- ECMAScript define la interfaz de programación para *scripts*.
- DOM define la interfaz de programación para acceder al contenido del documento web.
- BOM define la interfaz de programación para acceder al navegador web.

Objeto `Window`

- En JavaScript, el objeto `Window` es el elemento raíz, de tal modo que todos los demás objetos están conectados a él de modo directo o indirecto.
- Por ese motivo, no es necesario referenciarlo explícitamente.
- Todas las variables, funciones y objetos de JavaScript son miembros de `Window`.
- Las funciones globales son sus métodos.
- Las variables globales y el DOM son propiedades.

BOM y DOM


window
DOM
document
...
BOM
navigator
screen
location
frames
history
XMLHttpRequest
JavaScript
Object
Array
Function
...

Objeto `Window`

- El objeto `Window` representa una ventana abierta en el navegador.
- Cada vez que se abre una nueva ventana del navegador se crea un objeto `Window`.
- En un navegador basado en pestañas cada pestaña contiene su propio objeto `Window`.
- Es decir, `Window` no se comparte entre pestañas.

BOM

- El Modelo de Objetos del Navegador (Browser Object Model, BOM) permite que JavaScript pueda interaccionar con el navegador acerca de cuestiones no relacionadas con el contenido de la página.
- Para las cuestiones relacionadas con el contenido de la página se accede al DOM.
- A diferencia del DOM, no hay un estándar público para el BOM, pero la mayoría de los navegadores han implementado casi las mismas funcionalidades para favorecer la interoperabilidad de los scripts.

Objetos del BOM

- `Navigator`: gestionar la información sobre el navegador.
- `Screen`: guarda información gráfica sobre la ventana.
- `History`: información sobre las URLs visitadas por el usuario dentro de la ventana actual.
- `Location`: información de la URL actual.

DOM

- Cuando se solicita una página en el navegador, éste recupera el HTML correspondiente y lo analiza.
- El navegador construye un modelo de la estructura del documento y emplea dicho modelo para presentar la página en la pantalla.
- Dicho modelo se denomina *Document Object Model*, o DOM.

DOM

- El DOM es una estructura de datos que el programador puede leer y modificar.
- Cada vez que se modifica el DOM, la página mostrada en pantalla se actualiza para reflejar los cambios introducidos.
- De ese modo, la modificación del DOM permite la generación dinámica de páginas web.

DOM

- Por cada etiqueta del documento HTML existe en el DOM un objeto que lo representa.
- Este objeto contiene información como el tipo de etiqueta que representa, atributos y contenido el mismo.
- Los elementos del DOM pertenecen a la clase `Node`.

Estructura del DOM

- Del mismo modo que el HTML se estructura en base a una jerarquía de etiquetas, el DOM tiene una estructura en forma de árbol de nodos, la cual refleja la estructura del documento HTML.
- En el DOM, la variable `document` proporciona acceso al nodo raíz de dicho árbol.

Estructura en árbol del DOM

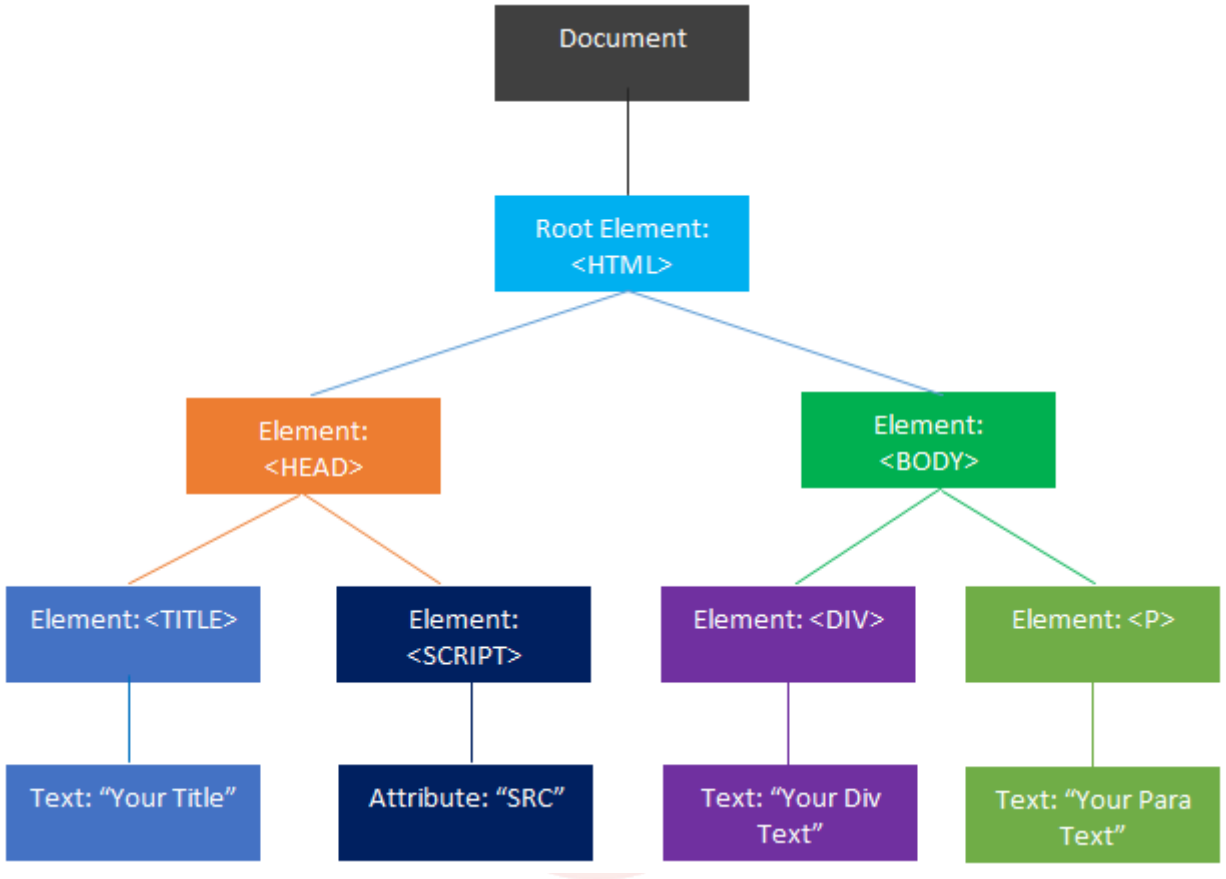
- Reflejando la estructura de un documento HTML, la variable `document` cuenta con las siguientes propiedades:

  `.documentElement` → representa <html>

  `.head` → representa <head>

  `.body` → representa <body>

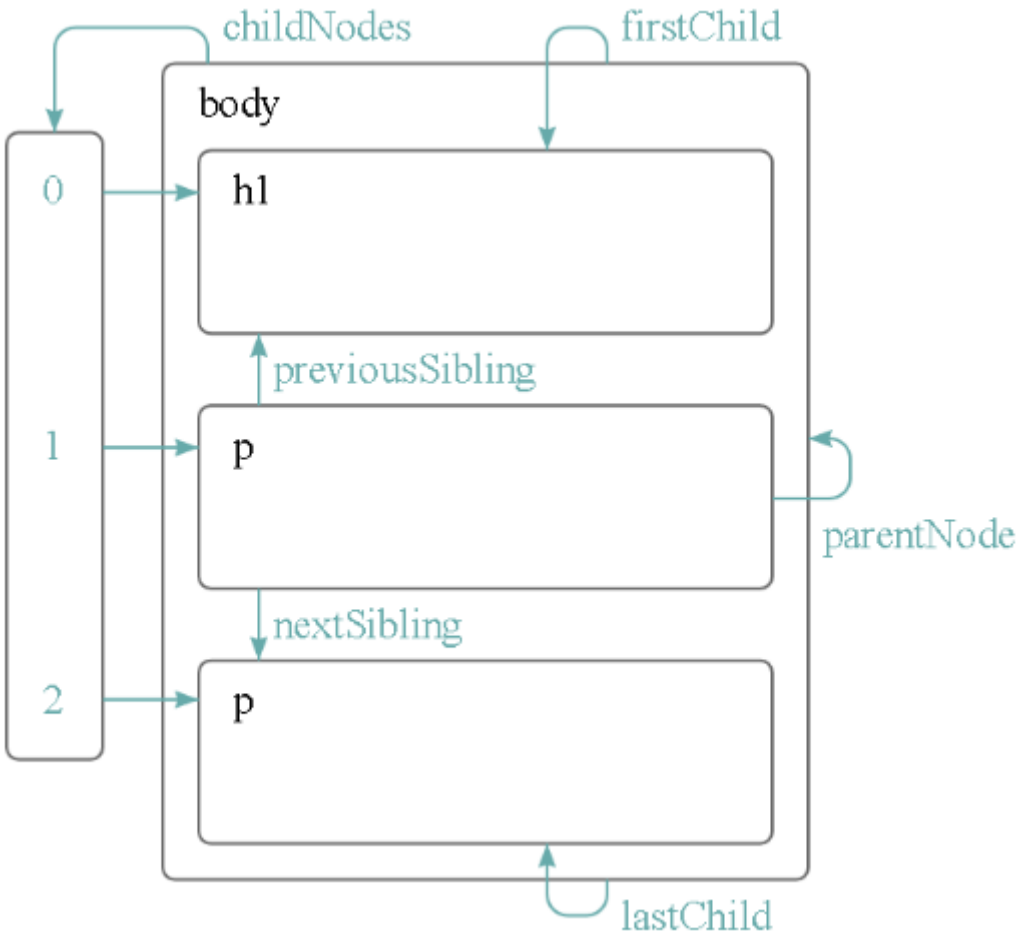
Estructura en árbol del DOM



Recorrer el DOM

- Cada nodo del árbol del DOM puede contener enlaces a otros elementos del árbol.
- También cuenta con propiedades para acceder a su contenido y atributos.
- `.innerHTML` accede al código HTML del nodo.
- `.textContent` accede al valor de texto que contenga la etiqueta del nodo.
- `.value` devuelve el valor introducido por el usuario en el nodo (para elementos de entrada de información, como <input>)

Recorrer el DOM



Recorrer el DOM

- `.childNodes` devuelve un objeto NodeList con todos los nodos del árbol DOM que son descendientes del nodo en cuestión.
- `.children` tiene un coportamiento parecido a `.childNodes`, pero solamente devuelve nodos que corresponden a elementos, es decir, etiquetas HTML, excluyendo texto, comentarios, etc.

Localizar elementos del DOM

- Los siguientes son los principales métodos para recuperar elementos concretos del DOM

`.getElementsByTagName("unaEtiqueta")`

→ en base al nombre de la etiqueta

`.getElementsById("unId")`

→ en base al valor del atributo `id`

`.getElementsByClassName("unaClase")`

→ en base al valor del atributo `class`

Localizar elementos del DOM

- Los siguientes métodos soportan selectores de CSS para recuperar listas de elementos:

  `.querySelector("selector")`

  → recupera el primer elemento

  `.querySelectorAll("selector")`

  → recupera todos los elementos

- Devuelve una lista estática (no viva)

Listas “vivas” (*live*)

- Una lista viva se actualiza dinámicamente a la vez que se actualiza el documento subyacente.
- Los métodos anteriores devuelven un objeto `NodeList`
- `NodeList` es una lista viva de los elementos encontrados en el orden en que aparecen en el árbol del documento.

NodeList

- Gran parte de los métodos del DOM que trabajan con un conjunto de nodos devuelven un objeto `NodeList`
- `NodeList` cuenta con una propiedad `.length` y propiedades accesibles mediante índices para cada uno de los componetes de la lista, pero no se trata de un *array*.

Listas sólidas

- Una lista sólida se pueda modificar sin afectar al documento, al contrario de las listas vivas.
- Se puede emplear `Array.from()` para convertir un `NodeList` en un `Array`.

```
let fields = document.querySelectorAll(“p");
for (let field of Array.from(fields)) {
    console.log(field.textContent);
}
```

Creación de nodos del DOM

- Los siguientes son los principales métodos para creación de elementos del DOM:

  `document.createElement("etiqueta")`

  → crea un nodo para una etiqueta

  `document.createTextNode("texto")`

  → crea un nodo de texto

Creación de nodos del DOM

- Es importante tener presente que crear un nodo no lo inserta en el árbol DOM.
- A tal efecto, deben emplearse los métodos para modificar el DOM.
- Ejemplo:

```
var imagen = document.createElement("img");
var cont = document.getElementById("div1");
cont.appendChild(imagen);
```

Modificar el DOM

- Los siguientes son los principales métodos para modificar elementos del DOM:

  `.appendChild(nuevoHijo)`

  → añade como último descendiente

  `.insertBefore(nuevoHijo, nodo)`

  → añade `nuevoHijo` justo antes que `unNodo`

  `.replaceChild(nuevoHijo, nodo)`

  → reemplaza `unNodo` por `nuevoHijo`

  `.remove()`

  → elimina el nodo sobre el que se invoca de su padre `.removeChild(unHijo)`

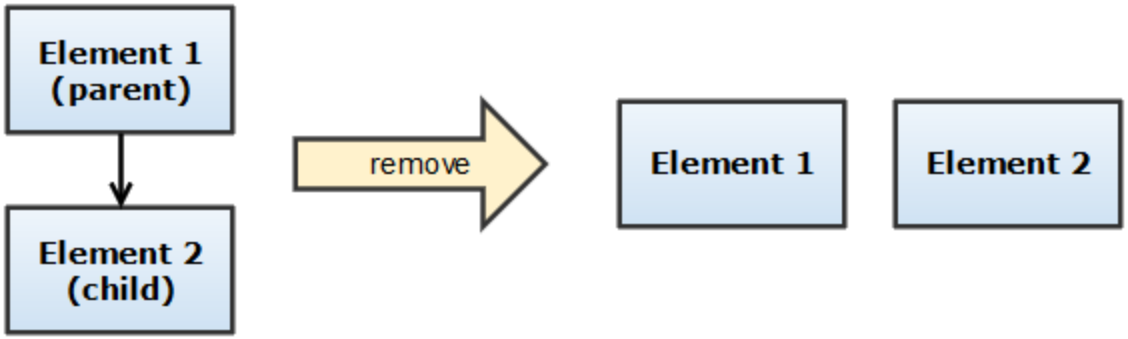
  → elmina `unHijo` del nodo sobre el que se invoca

Modificar el DOM

- Es importante tener presente que un elemento solo puede estar insertado en un lugar del árbol DOM.
- Todas las operaciones que insertan un nodo en algún lugar del árbol, lo eliminan antes de su posición anterior (en el caso de que ya estuviese situado).

Modificar el DOM

- Es importante tener presente que eliminar significa romper la relación jerárquica entre hijos y padres, pero no elimina los nodos en cuestión.



Gestión de atributos del DOM

- Los atributos más comunes del estándar pueden accederse mediante propiedades con el mismo nombre.
- Ejemplo:

```
let imagen = document.createElement("img");
imagen.src = ruta;
imagen.alt = “texto alternativo”;
```

Gestión de atributos del DOM

- Para el resto de los atributos se emplean los siguientes métodos:

  `.setAttribute("nombre", "valor")`

  → establece un atributo del elemento

  `.getAttribute("nombre")`

  → recupera el valor de un atributo

  `.className`

  → recupera el valor del atributo `class`

Modificación de estilos

- JavaScript puede modificar directamente los estilos de cualquier elemento a través de la propiedad `.style`
- Esta propiedad contiene un objeto que a su vez posee propiedades para los posibles estilos.

Modificación de estilos

- En CSS, algunas propiedades contienen guión. En nombre de dichas propiedades fue adaptado en JavaScript.

```
x.style.fontFamily = “Verdana”
// equivalente:
x.style[“font-family”] = “Verdana”
```

Referencias

- JavaScript en w3schools
- JavaScript en Mozilla Developer Network

FORMACIÓN PROFESIONAL

MONTECASTELO