JavaScript 04 – DOM - Document Object Model

Introducción

La mayoría de las veces que se programa con Javascript es para que se ejecute en una página web mostrada por el navegador. En este contexto, y para facilitar el desarrollo de la aplicación, tenemos acceso a ciertos objetos que nos permiten interactuar con la página (DOM) y con el navegador (Browser Object Model, BOM).

El **DOM** es una estructura en árbol que representa todos los elementos HTML de la página y sus atributos.

Todo lo que contiene la página se representa como nodos del árbol y mediante el DOM podemos acceder a cada nodo, modificarlo, eliminarlo o añadir nuevos nodos de forma que cambiamos dinámicamente la página mostrada al usuario.

La raíz del árbol DOM es **document** y de este nodo cuelgan el resto de elementos HTML.

Cada elemento constituye su propio nodo y tiene subnodos con sus *atributos*, *estilos* y otros elementos HTML que contiene.

Nota: Un **elemento** HTMl consiste en:

- Una **etiqueta inicial**. Opcionalmente contiene pares “atributo: valor”.
- **Contenido** del elemento (no siempre aparece).
- **Etiqueta final** o de cierre (no siempre aparece).

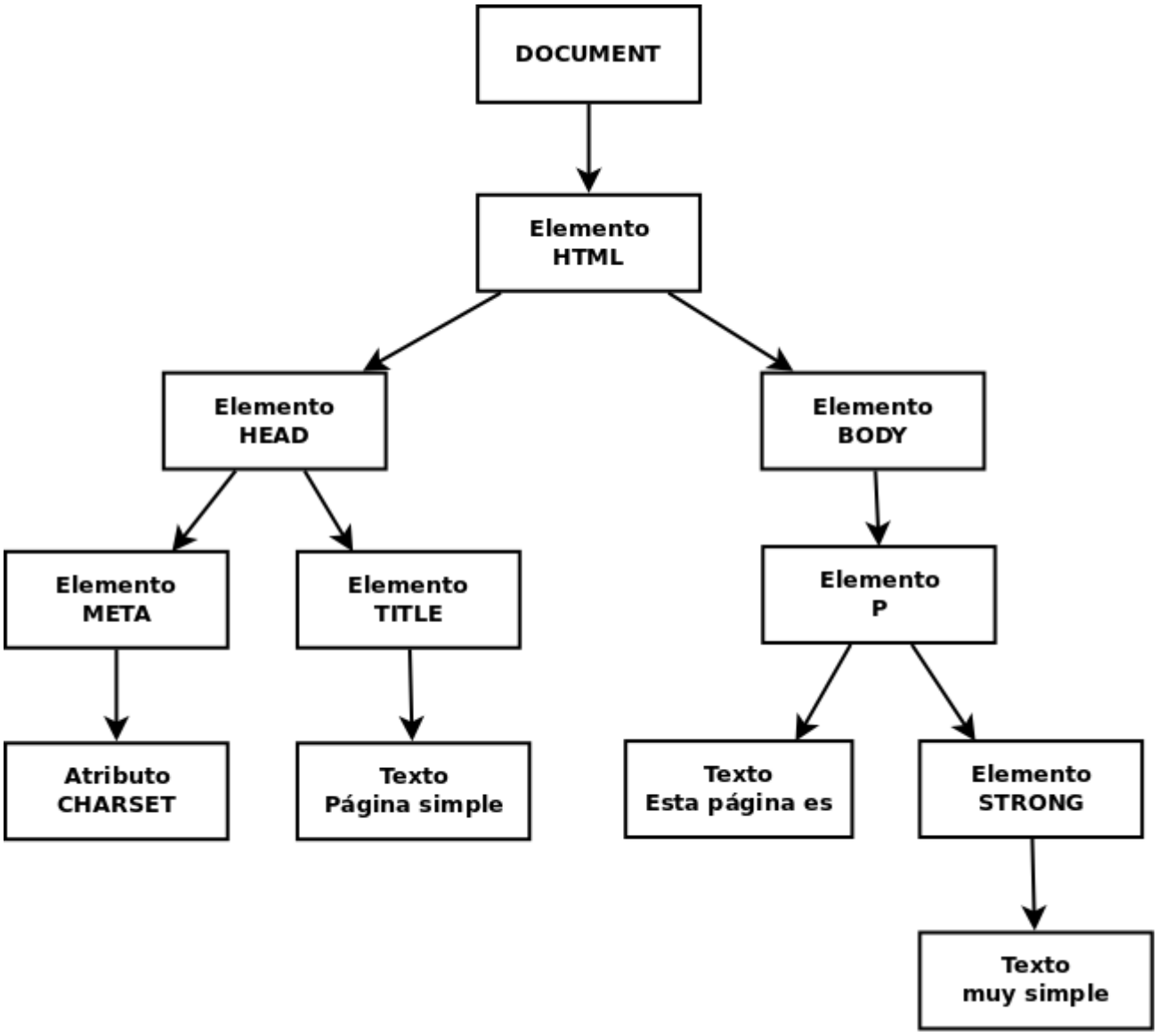
Es frecuente que alguien se refiera a un ‘elemento HTML’ como ‘etiqueta HTML’. Por lo que debes interpretar el significado de **etiqueta** en cada momento según el contexto.

Por ejemplo, la página HTML:

```
<!DOCTYPE html>
<html>
<head>
```

```
  <meta charset="utf-8">
  <title>Página simple</title>
</head>
<body>
  <p>Esta página es <strong>muy simple</strong></p>
</body>
</html>
```

se convierte en el siguiente árbol DOM:



Cada elemento HTML suele originar 2 nodos:

- **Element**: correspondiente al elemento.
- **Text**: correspondiente a su contenido (lo que hay entre la etiqueta inicial y su par de cierre)

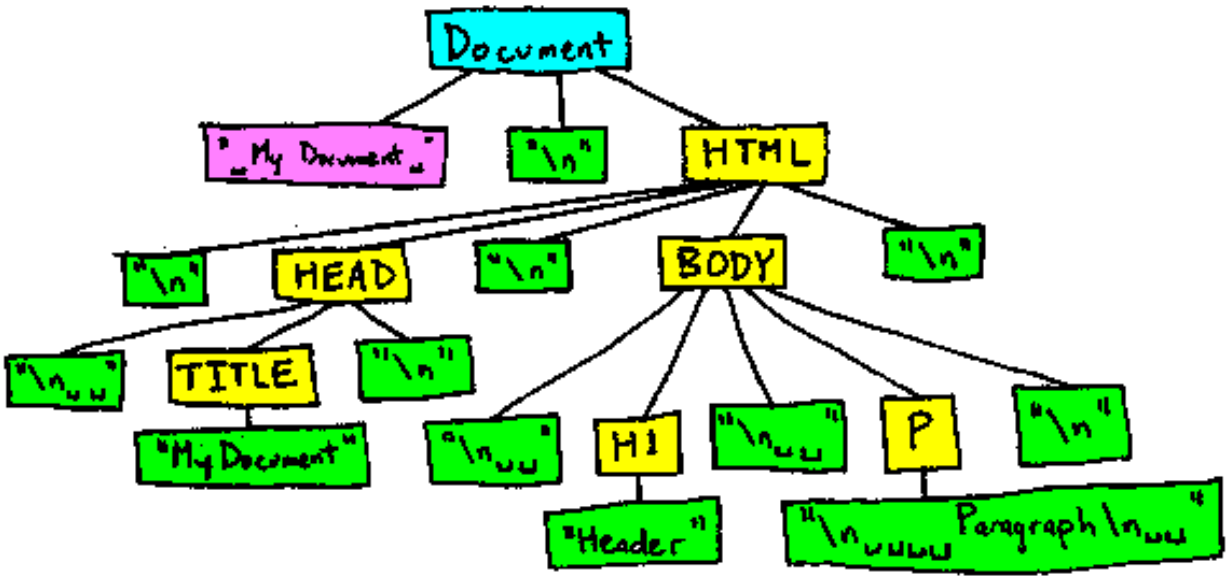
Cada nodo es un objeto con sus propiedades y métodos.

El ejemplo anterior está simplificado porque sólo aparecen los nodos de tipo ***elemento,*** pero en realidad también generan nodos los saltos de línea, tabuladores, espacios, comentarios, etc.

En el siguiente ejemplo podemos ver TODOS los nodos que realmente se generan. La página:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Document</title>
</head>
<body>
  <h1>Header</h1>
  <p>
    Paragraph
  </p>
</body>
</html>
```

se convierte en el siguiente árbol DOM:



Acceso a los nodos

Los principales métodos para acceder a los diferentes nodos son:

getElementById(id)

- **.getElementById(id)**: devuelve el nodo con la *id* pasada. Ej.:

```
let nodo = document.getElementById('main');    // nodo contendrá el nodo cuya id es _main_
```

Cuidado de no confundir con el método **.getElementsByName(*nombre*)**, que devuelve los elementos con el atributo *name* especificado.

getElementsByClassName(clase)

- **.getElementsByClassName(clase)**: devuelve una colección (similar a un array) con todos los nodos de la *clase* indicada. Ej.:

```
let nodos = document.getElementsByClassName('error');   // nodos contendrá todos los nodos cuya clase es _error_
```

NOTA: las colecciones son similares a arrays (se accede a sus elementos con *[indice]*) pero no se les pueden aplicar los métodos *filter*, *map*, … a menos que se conviertan a arrays con *Array.from()*

getElementsByTagName(elemento)

- **.getElementsByTagName(elemento)**: devuelve una colección con todos los nodos del tipo *elemento* HTML indicado. Ej.:

```
let nodos = document.getElementsByTagName('p');   // nodos contendrá todos los nodos de tipo  _<p>_
```

querySelector(selector)

- **.querySelector(selector)**: devuelve el primer nodo seleccionado por el *selector* CSS indicado. Ej.:

```
let nodo = document.querySelector('p.error');   // nodo contendrá el primer párrafo de clase _error_
```

querySelectorAll(selector)

- **.querySelectorAll(selector)**: devuelve una NodeList con todos los nodos seleccionados por el *selector* CSS indicado. Ej.:

```
let nodos = document.querySelectorAll('p.error');   // nodos contendrá todos los párrafos de clase _error_

let nodo = document.querySelectorAll('text');   // nodo contendrá el elemento con ID=text
```

`Más información en: https://developer.mozilla.org/es/docs/Web/API/Document/querySelectorAll`

NOTA: Los objetos **`NodeList`** son colecciones de nodos como los devueltos por propiedades como `Node.childNodes` y el método `document.querySelectorAll()`..

NOTA: al aplicar estos métodos sobre *document* se seleccionarán sobre la página (objeto document). Pero podrían también aplicarse a cualquier nodo, en ese caso la búsqueda se realizaría sólo entre los descendientes de dicho nodo.

También tenemos 'atajos' para obtener algunos elementos comunes:

- `document.documentElement`: devuelve el nodo del elemento <*html*>
- `document.head`: devuelve el nodo del elemento <*head*>
- `document.body`: devuelve el nodo del elemento <*body*>
- `document.title`: devuelve el nodo del elemento <*title*>
- `document.link`: devuelve una colección con todos los hiperenlaces del documento
- `document.anchor`: devuelve una colección con todas las anclas del documento
- `document.forms`: devuelve una colección con todos los formularios del documento

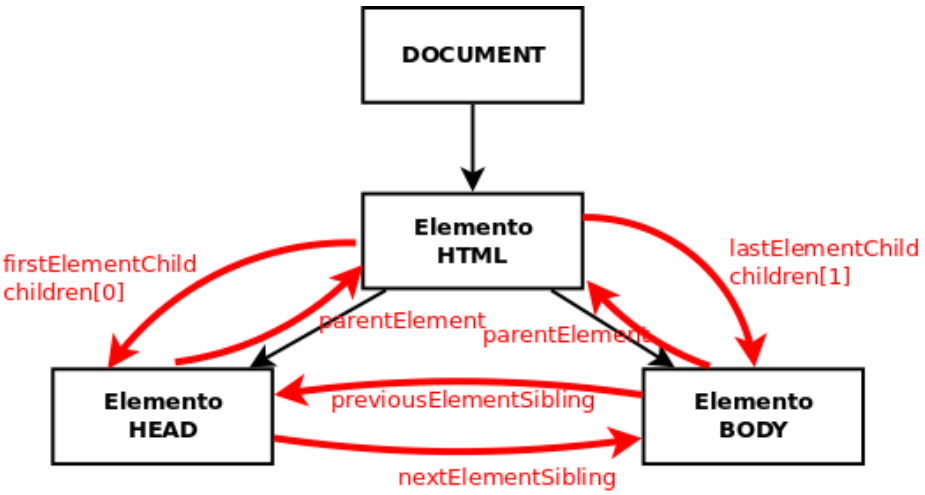
- `document.images`: devuelve una colección con todas las imágenes del documento
- `document.scripts`: devuelve una colección con todos los scripts del documento

EJERCICIO: Para hacer los ejercicios de este tema descárgate esta página de ejemplo y ábrela en tu navegador. Obtén por consola, al menos de 2 formas diferentes lo que se pide:

1. El elemento con id 'input2'
2. La colección de párrafos
3. Lo mismo pero sólo de los párrafos que hay dentro del div 'lipsum'
4. El formulario (ojo, no la colección con el formulario sino sólo el formulario)
5. Todos los inputs
6. Sólo los inputs con nombre 'sexo'
7. Los items de lista de la clase 'important' (sólo los LI)

Acceso a nodos a partir de otros

En muchas ocasiones queremos acceder a cierto nodo a partir de uno dado. Para ello tenemos los siguientes métodos que se aplican sobre un elemento del árbol DOM:



- `elemento.parentElement`: devuelve el elemento padre de *elemento*
- `elemento.children`: devuelve la colección con todos los elementos hijo de *elemento* (sólo elementos HTML, no comentarios ni nodos de tipo texto)
- `elemento.childNodes`: devuelve la colección con todos los hijos de *elemento*, incluyendo comentarios y nodos de tipo texto por lo que no suele utilizarse
- `elemento.firstElementChild`: devuelve el elemento HTML que es el primer hijo de *elemento*
- `elemento.firstChild`: devuelve el nodo que es el primer hijo de *elemento* (incluyendo nodos de tipo texto o comentarios)
- `elemento.lastElementChild`, `elemento.lastChild`: igual pero con el último hijo
- `elemento.nextElementSibling`: devuelve el elemento HTML que es el siguiente hermano de *elemento*
- `elemento.nextSibling`: devuelve el nodo que es el siguiente hermano de *elemento* (incluyendo nodos de tipo texto o comentarios)
- `elemento.previousElementSibling`, `elemento.previousSibling`: igual pero con el hermano anterior
- `elemento.hasChildNodes`: indica si *elemento* tiene o no nodos hijos
- `elemento.childElementCount`: devuelve el nº de nodos hijo de *elemento*

IMPORTANTE: a menos que interesen comentarios, saltos de página, etc …, **siempre** hay que usar los métodos que sólo devuelven elementos HTML, no todos los nodos.

EJERCICIO: Siguiendo con la página de ejemplo obtén desde la consola, al menos de 2 formas diferentes:

1. El primér párrafo que hay dentro del div 'lipsum'
2. El segundo párrafo de 'lipsum'
3. El último item de la lista
4. El elemento *label* de 'Escoge sexo'

Propiedades de un nodo

Las principales propiedades de un nodo son:

- `elemento.innerHTML`: todo lo que hay entre la etiqueta que abre *elemento* y la que lo cierra, incluyendo otras etiquetas HTML. Por ejemplo, si *elemento* es el nodo:

  `<p>Esta página es <strong>muy simple</strong></p>`

```
let contenido = elemento.innerHTML;   // contenido='Esta página es <strong>muy simple</strong>'
```

- `elemento.textContent`: todo lo que hay entre la etiqueta que abre *elemento* y la que lo cierra, pero ignorando otras etiquetas HTML. Siguiendo con el ejemplo anterior:

```
let contenido = elemento.textContent;   // contenido='Esta página es muy simple'
```

- `elemento.value`: devuelve la propiedad 'value' de un <input> (en el caso de un <input> de tipo text devuelve lo que hay escrito en él).
  Como los <inputs> no tienen etiqueta de cierre (</input>) no podemos usar *.innerHTML* ni *.textContent*.
  Por ejemplo: si *elem1* es el nodo `<input name="nombre">` y *elem2* es el nodo `<input type="radio" value="H"> Hombre`

```
let cont1 = elem1.value;   // cont1 valdría lo que haya escrito en el <input> en ese momento
let cont2 = elem2.value;   // cont2="H"
```

Otras propiedades:

- `elemento.innerText`: igual que *textContent*
- `elemento.focus`: da el foco a *elemento* (para inputs, etc). Para quitarle el foco `elemento.blur`
- `elemento.clientHeight` / `elemento.clientWidth`: devuelve el alto / ancho visible del *elemento*
- `elemento.offsetHeight` / `elemento.offsetWidth`: devuelve el alto / ancho total del *elemento*
- `elemento.clientLeft` / `elemento.clientTop`: devuelve la distancia de *elemento* al borde izquierdo / borde superior
- `elemento.offsetLeft` / `elemento.offsetTop`: devuelve los *píxels* que hemos desplazado *elemento* a la izquierda / abajo

EJERCICIO: Obtén desde la consola, al menos de 2 formas:

1. El innerHTML de la etiqueta de 'Escoge sexo'
2. El textContent de esa etiqueta

3. El valor del primer input de sexo
4. El valor del sexo que esté seleccionado (difícil, búscalo por Internet)