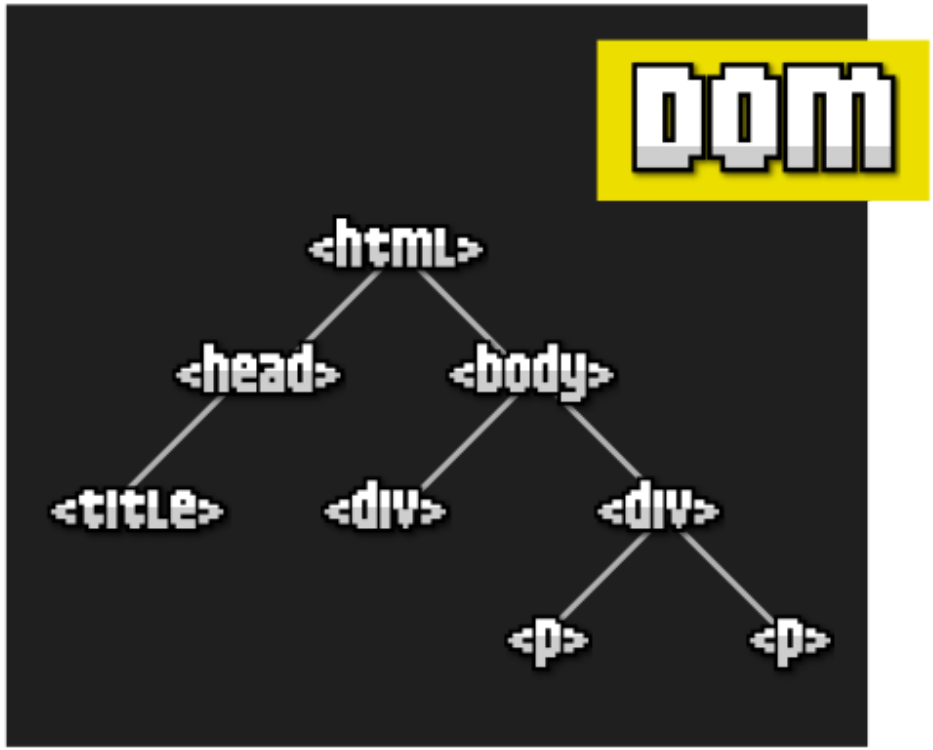
Manipulacion del DOM (03-10-23)

DOM

Las siglas DOM significan Document Object Model, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM.



Como seleccionar objetos del DOM

getElementById()

Dado un contenedor:

```
<div id="containerId" class="containerClass" name="containerName">
    ...
</div>
```

Se puede seleccionar con JS mediante el atributo id del elemento HTML:

```
const containerById = document.getElementById("containerId")
```

getElementByClassName()

El anterior elemento se puede seleccionar con la clase del elemento:

```
const containerByClassName = document.getElementByClassName("containerClass")
```

getElementByName()

O con el atributo name del elemento HTML:

```
const containerByName = document.getElementByName("containerName")
```

getElementByTagName()

Este metodo permite seleccionar un arreglo de elementos que coincidan con el nombre de la etiqueta dada como argumento.

Por ejemplo:

```
const arrElementsByTagName = document.getElementByTagName("div")
```

O bien, suponiendo una estructura mas compleja, puede seleccionarse un elemento especifico de este arreglo mediante:

```
const firstElementByTagName = document.getElementByTagName("div")[0]
```

querySelector()

Selecciona un elemento por:

1. Id:

```
const containerId = document.querySelector("#containerId")
```

2. Clase:

```
const containerClass = document.querySelector(".containerId")
```

3. Etiqueta:

```
const containerTag = document.querySelector("div")
```

querySelectorAll()

Selecciona un arreglo de elementos con los metodos de querySelector(): id, clase o etiqueta.

```
const arrElements = document.querySelectorAll(".container")
```

Gestionar clases del DOM

className()

`className` obtiene y establece el valor del atributo `class` del elemento especificado.

Sintaxis:

```
var cName = elementNodeReference.className;
elementNodeReference.className = cName;
```

> cName es una variable de cadena representando la clase o la lista de clases separada por espacios, del elemento en cuestión.

Ejemplo:

```
let elm = document.getElementById("item");

if (elm.className === "active") {
  elm.className = "inactive";
} else {
  elm.className = "active";
}
```

classList()

La propiedad de sólo lectura `Element.classList` devuelve una colección activa de `DOMTokenList (en-US)` de los atributos de clase del elemento.

Usar `classList` es una forma práctica de acceder a la lista de clases de un elemento como una cadena de texto delimitada por espacios a través de `element.className`.

Sintaxis:

```
var elementClasses = elementNodeReference.classList;
```

> `elementClasses` es un `DOMTokenList (en-US)` que representa el atributo clase de `elementNodeReference`. Si el atributo clase no está definido o está vacío, `elementClasses.length` devuelve 0. `element.classList` por sí mismo es de sólo lectura, aunque puede ser modificado usando los métodos `add()` y `remove()`.

Metodos:

1. `add()`**:** Añade las clases indicadas. Si estas clases existieran en el atributo del elemento serán ignoradas.
`div.classList.add("anotherclass");`

2. `remove()`**:** Elimina las clases indicadas. Nota: Eliminar una clase que no existe NO produce un error.
`div.classList.remove("foo");`

3. `toggle()`**:** Cuando sólo hay un argumento presente: Alterna el valor de la clase; ej., si la clase existe la elimina y devuelve false, si no, la añade y devuelve true. Cuando el segundo argumento está presente: Si el segundo argumento se evalúa como true, se añade la clase indicada, y si se evalúa como false, la elimina. `div.classList.toggle("visible");`

4. `contains()`**:** Comprueba si la clase indicada existe en el atributo de clase del elemento, por lo que devuelve `bool`.
`alert(div.classList.contains("foo"));`

5. `replace(oldClass, newClass)`**:** Reemplaza una clase existente por una nueva. `div.classList.replace("foo", "bar");`