DOM

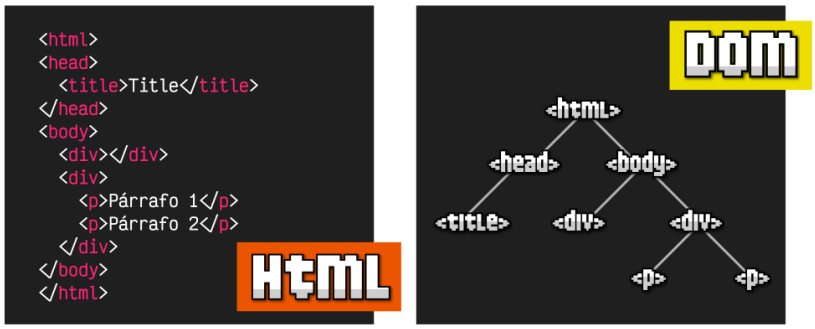
Document Object Model

Document Object Model

Les sigles **DOM** signifiquen **Document Object Model**, o el que és el mateix, l'estructura del document HTML.

El DOM és una API de W3C que permet que els programes i els scripts d'una pàgina Web actualitzar i accedir tant als continguts els estils i l'estructura d'un document HTML o XML. Per tant, el DOM ens permetrà accedir dinàmicament a les pàgines web, interactuar i respondre a les accions dels usuaris (com prémer un botó, moure el ratolí, fer clic en una part del document, escriure un text, etc.), sense haver de recarregar la pàgina.

Una pàgina **HTML** està formada per múltiples etiquetes **HTML**, anidades una dins de l'altra, formant un **arbre d'etiquete**s relacionades entre sí, que es denomina **arbre DOM** (o simplement DOM).



Quan ens referim al **DOM** ens referim a **aquesta estructura d'arbre**, que ens permet accedir i modificar els elements de **l'HTML** des de **Javascript**, afegint noves etiquetes, modificant o eliminant altres, canviant els seus atributs HTML, afegint classes, canviant el contingut de text, etc.

En Javascript, la forma d'accedir al DOM és a través d'un objecte anomenat **`document`**, que representa **l'arbre DOM de la pàgina**, en el seu interior poden existir diversos tipus d'elements, però principalment seran **`ELEMENT`** o **`NODE`**.

En resum els principals conceptes del DOM són:

- **`Document`**: és l'objecte principal del DOM, que representa tot el document **HTML** o **XML** carregat al navegador. És el punt d'entrada principal per accedir als altres elements del DOM.

```
// Accedim a l'objecte document
const doc = document;
// Podem utilitzar-lo per accedir a altres elements
const titol = doc.getElementById('titol');
```

A l'exemple, accedim a l'objecte **document**, que representa el document HTML actual. Després, utilitzem el mètode **getElementById** per obtenir una referència al títol de la pàgina utilitzant el seu identificador "titol"

- **Node**: En el DOM, cada element, atribut o text del document és considerat un **node**. Això inclou no només els elements HTML, sinó també els nodes de text i els atributs associats amb els elements.

```
// Accedim a un node de text
const textNode = document.createTextNode('Este és un text');

// Creem un element
const element = document.createElement('div');

// Afegim el node de text com a fill de l'element
element.appendChild(textNode);
```

Al codi anterior, creem un node de text utilitzant el mètode **createTextNode** amb el text "Este és un text". Després, creem un element **div** utilitzant **createElement**. Finalment, afegim el node de text com a fill de l'element utilitzant el mètode **appendChild**.

- **Element**: Els elements són un tipus especial de node que representa les etiquetes **HTML com , <p>, **, etc. Cada element pot tenir atributs i contingut de text associat.

```
// Creem un nou element div
const divElement = document.createElement('div');

// Afegim un atribut a l'element
divElement.setAttribute('id', 'nouDiv');

// Afegim contingut de text a l'element
divElement.textContent = 'Aquest és un nou div';

// Afegim l'element al document
document.body.appendChild(divElement);
```

A l'exemple, creem un nou element **div** utilitzant el mètode **createElement**. Després, afegim l'atribut **id** a l'element utilitzant **setAttribute**. Assignem contingut de text a l'element utilitzant la propietat **textContent**. Finalment, afegim l'element al cos del document utilitzant el mètode **appendChild**.

- **Atribut**: Un atribut és una característica específica d'un element HTML, com l'atribut **id** o **class**. Els atributs proporcionen informació addicional sobre **l'element** i poden ser utilitzats per identificar o estil·litzar l'element.

```
// Accedim a un element pel seu id
const element = document.getElementById('meuElement');

// Canviem un atribut de l'element
element.setAttribute('alt', 'Nova descripció');
```

Accedim a un element utilitzant el seu **id** amb **getElementById**. A continuació, utilitzem **setAttribute** per canviar l'atribut alt de l'element a "Nova descripció".

- **`Nivell d'Element`**: El nivell d'element fa referència a la relació jeràrquica entre els elements del document, que es representa com un arbre en el **DOM**. Cada element té un pare (excepte el **`document`**), i pot tenir fills i germans.

```
// Accedim al pare d'un element
const fill = document.getElementById('fill');
const pare = fill.parentNode;
console.log('El pare de l\'element fill és:', pare.tagName);
```

En este cas, accedim a l'element **fill** amb l'id "**`fill`**". A continuació, utilitzem la propietat **parentNode** per obtenir una referència al pare de l'element. Finalment, mostrem el **tag name** del pare amb **tagName**.

- **`Selecció d'Elements:`** La selecció d'elements és la capacitat de trobar i accedir als elements del DOM utilitzant mètodes com **`getElementById`**, **`getElementsByClassName`**, **`getElementsByTagName`**, **`querySelecto`**r i **`querySelectorAll`**. Estos mètodes permeten seleccionar elements específics segons els seus **identificadors**, **classes**, **noms d'etiqueta**, etc.

```
// Seleccionem un element per id
const elementId = document.getElementById('idElement');

// Seleccionem elements per classe
const elementsClasse = document.getElementsByClassName('classeElement');

// Seleccionem elements per etiqueta
const elementsEtiqueta = document.getElementsByTagName('p');

// Utilitzem selectors CSS
const elementSelector = document.querySelector('.selectorClasse');
const elementsSelector = document.querySelectorAll('p');
```

A l'exemple, es mostren diferents per seleccionar elements.

  - **`getElementById`** selecciona un únic element per l'ID especificat.
  - **`getElementsByClassName`** selecciona tots els elements amb la classe especificada.
  - **`getElementsByTagName`** selecciona tots els elements amb l'etiqueta especificada.
  - **`querySelector`** selecciona el primer element que coincideix amb el selector CSS especificat.
  - **`querySelectorAll`** selecciona tots els elements que coincideixen amb el selector CSS especificat.

- **Manipulació del DOM**: Manipular del DOM és la capacitat de modificar la estructura, l'estil i el contingut del document, afegint o eliminant elements, canviar atributs i text, modificar estils CSS, i altres canvis que poden afectar la representació visual o la funcionalitat de la pàgina web.

```
// Creem un nou element
const nouElement = document.createElement('div');

// Afegim el nou element com a fill d'un altre element
pare.appendChild(nouElement);

// Eliminem un element
pare.removeChild(fill);
```

En este cas, estem creant un nou element utilitzant **createElement** i l'afegim com a fill d'un altre element utilitzant **appendChild**. Després, eliminem un element del seu pare utilitzant **removeChild**.

- **Esdeveniments del DOM**: Els esdeveniments són accions que es produeixen en el navegador, com clics de ratolí, premses de tecles o càrregues de pàgines. La **vinculació de funcions** de gestió d'esdeveniments als elements HTML permet gestionar aquests esdeveniments i interactuar dinàmicament amb els usuaris.

```
// Afegim un esdeveniment de clic a un element
const meuElement = document.getElementById('meuElement');
meuElement.addEventListener('click', function() {
    alert('S\'ha clicat l\'element!');
});
```

En este exemple, afegim un gestor d'esdeveniments de clic a l'element amb **l'id** "**meuElement**". Quan aquest element siga clicat, es mostrarà un missatge d'alerta que indica que s'ha clicat l'element.

Exercicis

- **Exercici 1**

Canvia el color de fons del títol a un color verd fosc (#006400).

Pista: Seleccionem l'element del títol utilitzant getElementById i després accedim a la propietat style.backgroundColor per canviar el color de fons a verd fosc.

- **Exercici 2**

Afegir un nou element al menú. Crea un nou element li al menú amb un enllaç que digui "Sobre nosaltres" i afegix-lo al final de la llista.

Pista: Creem un nou element li i un enllaç a, configurem l'enllaç amb l'atribut href i el text contingut, després afegim l'enllaç com a fill del li i finalment afegim el nou li com a fill de la llista del menú.

- **Exercici 3**

Canvia el color de fons de tots els articles a un color groc clar (#FFFF99).

Pista: Utilitzem querySelectorAll per seleccionar tots els elements article, i després iterem a través de cada element utilitzant forEach per canviar el color de fons a groc clar.

- **Exercici 4**

Canviar el color del text del menú a roig (#ee0808).

Pista: Utilitzem querySelectorAll per seleccionar tots els enllaços (a) dins del menú, i després iterem a través de cada enllaç per canviar el color del text a roig.

- **Exercici 5**

Crea un nou paràgraf amb el text "Segueix-nos a les xarxes socials!" i afegix-lo al final del main.

Pista: Creem un nou element p amb el text especificat, i després afegim aquest nou paràgraf com a fill del main utilitzant appendChild.

- **Exercici 6**

Eliminar l'enllaç "Serveis" del menú

Pista: Utilitzem querySelector per seleccionar l'enllaç amb l'atribut href igual a "#", que conté el text "Serveis". Després, accedim al seu pare utilitzant parentNode dues vegades per eliminar l'element li del DOM.

- **Exercici 7**

Canvia el color de fons del footer a un color gris fosc (#9b9797).

Pista: Seleccionem l'element del footer i accedim a la seua propietat style.backgroundColor per canviar el color de fons a gris fosc.

- **Exercici 8**

Canvia la mida de la font dels paràgrafs de l'article 1 a 36 pixels.

Pista: Utilitzem querySelectorAll per seleccionar tots els paràgrafs (p) dins del primer article, i després iterem a través de cada paràgraf per canviar la mida de la font a 36 píxels. Utilitzem l'argument 'article:nth-child(1) p'.

- **Exercici 9**

Canviar el color de fons del títol i del menú quan es fa clic. Canvia el color de fons del títol a blau (#336699) i el color de fons del menú a vermell (#FF0000).

Pista: Afegim un event listener a l'element del títol i al menú per escoltar els esdeveniments de clic. Quan es fa clic, canviem el color de fons corresponent utilitzant style.backgroundColor.