LLENGUATGES DE MARQUES I SISTEMES DE GESTIÓ D'INFORMACIÓ

UNITAT 3. Manipulació de documents web amb scripts de client

1r DAM. 2024-2025
Elena Zamora
elena.zamora@cide.es

CONTINGUTS

- 3.1. Introducció.
- 3.2. Característiques i sintaxi bàsica de Javascript.
- 3.3. Selecció i accés als elements del DOM.
- 3.4. Creació, modificació i eliminació d'elements.
- 3.5. Manipulació d'estils

3.1. Introducció

Les **pàgines web** es poden classificar en pàgines estàtiques o dinàmiques, segons el comportament que presenten des que se les sol·licita fins que estan disponibles al navegador.

Les **estàtiques** són les que no ofereixen cap interacció ni modificació del seu contingut durant tot el procés de sol·licitud i presentació al navegador.

Les **dinàmiques** són les que ofereixen dinamisme, es dir, permeten aquesta interacció amb la pàgina o modificar el seu contingut, ja sigui al servidor en sol·licitar la pàgina o bé al navegador del cliente.

3.2. Característiques i sintaxi bàsica de Javascript

Què és JavaScript?

És un **Llenguatge de programació** utilitzat principalment per a la creació de pàgines i aplicacions **web dinàmiques**, permentent canvis, animacions i efectes.

S'executa principalment a la banda del client en un navegador web, que és l'encarregat d'interpretar-lo.

NO és el mateix que JAVA.

3.2. Característiques i sintaxi bàsica de Javascript

Què podem fer?

- Validar dades d'un formulari, per exemple, verificar que el format d'un email sigui correcte, que s'hagin completat tots els camps, etc.
- Mostrar missatges o advertències.
- Fer galeria d'imatges dinàmiques .
- Obrir finestres i pestanyes.
- Realitzat efectes.
- Moure objectes.
- Canviar la resolució d'una web .

3.2. Característiques i sintaxi bàsica de Javascript

Com es pot utilitzar?

- Generalment s'utilitza conjuntament amb HTML i CSS
- Igual que CSS es pot importar de manera externa o interna

Interna: s'incorpora directament entre les etiquetes <script>

```
<script>alert("Hola classe!")</script>
```

Externa: es crida a un fitxers JS extern

```
<script src="js/exemples.js"></script>
```

3.2. Característiques i sintaxi bàsica de Javascript

On podem utilitzar-lo?

- Es pot utilitzar **tant al <head> com al <body>** en funció de quan volem que es carregui l'script.
- El farem servir **al <head> quan necessitem que l'script es carregui abans que la pàgina** estigui completament visible. Això ens permet disposar de l'script abans que es carregui el contingut però pot bloquejar-ne la renderització si no es fa servir defer o async.
- L'usarem **al <body> quan vulguem evitar el bloqueig de la renderització** però no fer servir defer ni async, quan utilitzem scripts que manipulen elements ja renderitzats o si volem el que HMTL es carregui abans de l'executar el JS (en aquest cas ha d'estar col·locat al final del body). Però en aquest cas, l'script pot dependre de l'ordre de la càrrega de contingut i pot causar errors si es col·loca abans que alguns elements necessaris al JS del mateix HMTL.
- ***Defer*** és un atribut que retarda l'execució de l'script fins que el document HTML s'hagi interpretat completament.
- ***Async*** és un atribut que permet que l'script es descarregui i executi de manera asíncrona, independentment del procés d'anàlisi de l'HTML. (Es veurà més endavant quan s'expliqui DOM (Document Object Model)).

3.2. Característiques i sintaxi bàsica de Javascript

Activitat

Veurem com utilitzar un Javascript, tant de manera interna com externa, i com es carrega en funció d'on fem aquesta trucada.

Per això, crearem un HTML i el següent Javascript: `alert("Hola classe!");`

- Primer provarem d'inserir-lo de manera interna al codi i en el head.
- Després crearem el següent fitxer extern JS i el carregarem al final del body. Veurem com es carreguen tant l'HTML com el JS, des de les eines de desenvolupar del navegador:

```
const paragraf = document.getElementById("paragraf");
paragraf.textContent = "Paragraf modificar dins l'arxiu JS.";
```

- Finalment, canviarem la trucada a l'arxiu extern JS i el posarem al head. Què ha passat?

3.2. Característiques i sintaxi bàsica de Javascript

Sintaxi

Comentaris

```
//Comentari en una línia

/*
Comentari de varies línies
*/
```

Impressió per consola

Podem visualitzar missatges de Javascript per la consola del navegador (eines del desenvolupador dels navegadors)

```
console.log('Hola classe!');
```

3.2. Característiques i sintaxi bàsica de Javascript

Tipus de dades

JS és un llenguatge fortament tipat, per tant, NO cal que els especifiquem, però sí que cal tenir en compte l'ús que donarem de cadascun d'ells:

- *String*; per a caràcters alfanumèrics, amb cometes simples o dobles.
- *Number*: per a valors numèrics, amb o sense decimals.
- *BigInt*: per a valors numèrics més grans que 9007199254740992 o menors que -9007199254740992. Per evitar problemes a l'hora de treballar amb números grans, podem posar una *n* al final per forçar que sigui un Bigint.
- *Boolean*: permet dos valors possibles, true o false.
- *Null*: contenen el valor null.
- *Undefined*: s'utilitza per a variables que encara no tenen valors definits.
- *Object*: pot contenir tant objectes integrats (matrius, dades, arrays), com objectes definits per l'usuari.

3.2. Característiques i sintaxi bàsica de Javascript

Variables

Són espais de memòria on es poden emmagatzemar dades de diferents tipus. **JS es case sensitive, es a dir, diferencia entre majúscules i minúscules.**

3 tipus de variables:

- Var: declaració d'una variable

  ```
  var nombre;
  ```

- Let: declaració d'una variable però que NOMÉS es pot utilitzar dins de determinats blocs de codi (generalment els limitats per claus), per exemple, si es defineix la variable let dins d'un if, aquesta variable no es podrà utilitzar fora de l'if.

  ```
  let nombre;
  ```

- Const: constant, un cop assignat un valor aquest no podrà canviar.

  ```
  const nombre;
  ```

3.2. Característiques i sintaxi bàsica de Javascript

Assignació de Variables i concatenació de Strings

L'operador per a l'assignació és =

No cal especificar el tipus de dades a l'hora de declarar i assignar una variable, sinó que JavaScript detecta el tipus de dada.

```
const nom = "Elena";
const present = true;
const notaExamen = 9.5;
const numeroGran = 34343n;
```

Si necessitem saber el tipus de dada que està agafant JavaScript podem fer-ho mitjançant ***typeof***.

```
console.log('Nombre: ' + nombre + " " + typeof nombre); // concatenació amb +, converteix tot a String
console.log('Nombre: ',nombre,typeof nombre);  // concatenació , posa un espai directament i es mostrar l'objecte.
                                                //Només en algunes instruccions como console.log.
nombre += " Zamora"; // concatenació i assignació a variable.
```

Alertes i diàlegs

Funció ***alert***: mostra un missatge al navegador.

```
alert('Hola món!');
```

Funció ***prompt***: obre un quadre de diàleg perquè l'usuari pugui introduir un text i es pot assignar a una variable per recollir-lo.

```
var text = prompt('Introdueix un text');
```

Funció ***confirm***: obre un quadre de text informatiu amb dos botons per Acceptar o Cancel·lar. El resultat del botó premut (true o false) es pot recollir en una variable.

```
var continuar = confirm("Vols continuar?");
```

3.2. Característiques i sintaxi bàsica de Javascript

Como es visualitza cada constant/variable per consola? Quin tipus es visualitza?

```
const nom = "Elena";
const present = true;
const notaExamen = 9.5;
const numeroGran = 34343n;
const valorNull = null;
var variableSenseAsignar;
const dadesAlumne = {nom:"Elena", cognom:"Zamora", edat:30};
const assignatures = ["LDM", "SGE", "PMiDM"];
const data = new Date("2024-12-01");
```

Operadors

Operadors aritmètics

- Suma +
- Resta –
- Multiplicació *
- Divisió /
- Mòdul (resta d'una divisió) %
- Increment ++
- Decrement –

Operadors Condicionals o lògics

- AND &&
- OR ||
- NOT !

Operadors de comparació o relacionals

- Igualtat ==
- Desigualtat (diferent) !=
- Igualtat estricta ===
- Desigualtat estricta !==
- Major i menor > <
- Major o igual i menor o igual >= <=

Concatenació

- +
- +=

Activitat 1. Mitjana

Realitza un programa a JavaScript on se sol·licita a l'usuari les notes de 3 exàmens per pantalla, es calcula la mitjana i que després es mostri la nota final per la consola.

Has de tenir en compte que:

- El fitxer JavaScript ha de ser extern i s'ha d'importar en un HTML.
- L'HTML no cal que contingui cap tipus d'informació, només la importació del JS.

Declaracions condicionals

A JavaScript tenim les següents declaracions condicionals:

- **if**: per especificar un bloc de codi que cal executar, si una condició especificada és certa.
- **else**: per especificar un bloc de codi que cal executar, si la mateixa condició és falsa.
- **else if**: per especificar una nova condició per provar, si la primera condició és falsa.
- **switch**: per especificar molts blocs alternatius de codi.

3.2. Característiques i sintaxi bàsica de Javascript

Declaracions condicionals

```
var hora = 11;
if (hora < 14) {
    console.log("Bon dia");
}else if (hora < 21){
    console.log("Bones tardes");
}else{
    console.log("Bona nit");
}
```

```
var color = prompt("Escriu un color");
switch (color) {
    case "Groc":
        console.log("Color groc");
        break;
    case "verd":
    case "Verd":
        console.log("Color verd");
        break;
    default:
        console.log("El color no és vàlid.");
        break;
}
```

Activitat 2. Notes

Continuant amb l'activitat 1, modifica el programa perquè guardi la mitjana calculada anteriorment sobre els 3 exàmens en una variable (si no està fet ja), i mostri a la consola la nota final no numèrica seguint les regles següents:

- Si la mitjana és menor que 5, ha de mostrar "Suspès".
- Si més gran o igual que 5 i menor que 7, "Aprovat".
- Si és més gran o igual que 7 i menor que 9, "Notable".
- Si és més gran o igual que 9, "Excel·lent".

Has de tenir en compte que:

- El fitxer JavaScript ha de ser extern i s'ha d'importar en un HTML.
- L'HTML no cal que contingui cap tipus d'informació, només la importació del JS.

Estructures repetitives

A JavaScript tenim les següents estructures repetitives :

- **for**: repetició d'un bucle una determinada quantitat de vegades.
- **while**: repetició mentre es compleixi una determinada condició. Aquesta condició s'avalua a l'inici del bucle.
- **do-while**: semblant al while, però en aquest cas, la condició s'avalua al final del bucle, de manera que, sempre, com a mínim, el bucle s'executarà una vegada.

Estructures repetitives

```
var i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

```
var i = 0;
do {
  console.log(i);
  i++;
}
while (i < 5)
```

```
for (var i = 0; i < 5; i++) {
    console.log(i);
}
```

Funcions

Tant si la funció retorna un valor o no, la sintaxi serà la mateixa, incloent la instrucció *return*

```
function hola (){
    console.log("Hola Classe");
}
hola();


function sumar (num1, num2){
   return num1+num2;
}
let suma = sumar(5,6);
```

Arrays

Són estructures de dades que permeten emmagatzemar múltiples valors en una sola variable.

Hi ha dos tipus: vectors (unidimensionals) i matrius (multidimensionals).

```
var vector = [1,2,3,4,5];
var matriu= [[1,2,3],[4,5,6],[7,8,9]];

var vector = new Array(2);
vector[0] = "1";
vector[1] = "2";
```

```
var contador = 0;
var matriu= new Array(3);
for (let i=0;i<3;i++) {
    matriu[i] = new Array (3);
    for (let j=0;j<3;j++) {
        matriu[i][j]=contador;
        contador++;
    }
}
console.log(matriu);
```

Arrays

Hi ha molts mètodes per no haver de treballar manualment amb els elements dels arrys.

Per exemple, amb els mètodes *push* i *pop* podem afegir i eliminar un element a la darrera posició de l'array.

Podeu trobar aquests mètodes a les següents URLs:

https://www.w3schools.com/js/js_array_methods.asp

https://www.w3schools.com/js/js_array_search.asp

https://www.w3schools.com/js/js_array_sort.asp

https://www.w3schools.com/js/js_array_iteration.asp

Activitat 3. Assignatures

Realitza un programa a JavaScript per visualitzar les assignatures de DAM. Per això:

- Crea un array d'assignatures de primer amb els valors següents: Llenguatges, Sistemes, Programació.
- Crea un array d'assignatures de segon amb els valors següents: Sistemes de Gestió, Interficies, Programació Multimedia.
- Crear un nou array d'assignatures DAM amb els dos arrays anteriors.
- Mostra per consola els valors.
- Un cop creat, sol·licita per pantalla una nova assignatura de primer, afegeix-lo a l'array i mostra per consola els valors de nou.
- Elimina l'assignatura de interficies de segon i torna a mostrar els valors per consola.
- Sol·licita per pantalla una nova assignatura i comprova que existeix.

Has de tenir en compte que:

- El fitxer JavaScript ha de ser extern i s'ha d'importar en un HTML.
- L'HTML no cal que contingui cap tipus d'informació, només la importació del JS.

Strings i números

Igual que passa amb els Arrays, podem trobar diversos mètodes per treballar amb strings i amb números.

Per als string en trobem alguns com length, substring(), toUpperCase(), toLowerCase(), split(), concat(), entre d'altres que es pot consultar a https://www.w3schools.com/js/js_string_methods.asp

En el cas dels números, podem trobar toString(), parseFloat(), parseInt(), entre d'altres que es pot consultar a https://www.w3schools.com/js/js_number_methods.asp

Dates

Els objectes de tipus data es creen amb el constructor Date(). Hi ha 9 possibles constructors:

```
new Date()
new Date(date string) → const d = new Date("October 13, 2014 11:13:00"); // Date("2022-03-25");
new Date(year,month) → → const d = new Date(2025, 0); → JS compta els mesos de 0 a 11
new Date(year,month,day)
new Date(year,month,day,hours)
new Date(year,month,day,hours,minutes) → const d = new Date(2025, 11, 10, 13, 33);
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes,seconds,ms)
new Date(milliseconds)
```

Dates

Com en els casos anteriors, tenim diversos mètodes per manejar les dates, entre els quals trobem:

- getFullYear()
- getMonth()
- getDate()
- getDay() que cerca el dia de la setmana entre 0 i 6
- getHours()
- getMilliseconds()
- setDate()
- setMinutes()
- setMilliseconds()
- etc.

Objectes

A JavaScript, podem definir un objecte de diverses maneres diferents, per exemple, creant directamente l'assignació d'una manera simple o mitjançant la classe Objecte.

```
const persona = {
    nom: "Elena",
    cognom: "Zamora",
    edat: 30,
    saludar: function() {
      console.log("Hola! El meu nom es", this.nom);
    }
};

console.log("Nom:", persona.nom);
persona.saludar();
```

```
const persona = new Object();
persona.nom = "Elena";
persona.cognon = "Zamora";
persona.edat = 30;
persona.saludar = function() {
    console.log("Hola! El meu nom es", this.nom);
};

console.log("Nom:", persona.nom);
persona.saludar();
```

Esdeveniments HTML

Un esdeveniment HTML és **qualsevol interacció o acció que ocorre en una pàgina web, ja sigui pel navegador o l'usuari**, i que es pot detectar i manipular mitjançant codi JavaScript.

Els esdeveniments **permeten als desenvolupadors fer que una pàgina sigui interactiva, responent a accions de l'usuari o canvis a l'estat de la pàgina**. Per exemple, quan s'ha acabat de carregar una pàgina web, s'ha canviat un camp d'entrada o s'ha fet clic a un botó.

Podem trobar d**iferents tipus d'esdeveniments**, com són esdeveniments del ratolí, del teclat, del formulari, de la finestra o del document.

3.2. Característiques i sintaxi bàsica de Javascript

Esdeveniments HTML

Podem trobar molts esdeveniments. Aquí els més comuns:

- onchange: s'ha canviat un element HTML.
- onclick: l'usuari fa clic en un element HTML.
- onmouseover: l'usuari mou el ratolí sobre un element HTML.
- onmouseout: l'usuari allunya el ratolí d'un element HTML.
- onkeydown: l'usuari prem una tecla del teclat.
- onload: el navegador ha acabat de carregar la pàgina.
- onsubmit: es fa l'enviament d'un formulari.
- onfocus: un element obté el focus.

```
<button onclick="alert('Hola des d\'HTML!')">Saludar des d'HTML</button>
<button onclick="saludar()">Saludar des de JS</button>
```

3.3. Selecció i accés als elements del DOM.

Què és el DOM?

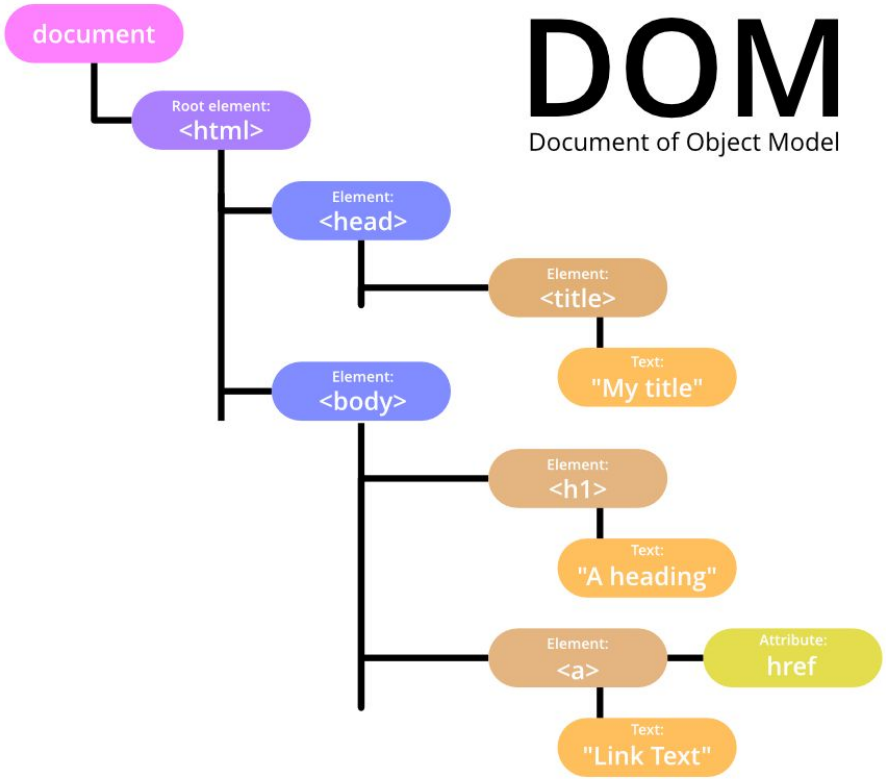
El DOM (Document Object Model) és un **estàndard del W3C** (World Wide Web Consortium). És com un mapa o una representació estructurada d'una pàgina HTML.

Quan el navegador carrega una pàgina web, crea una **estructura d'objectes** a partir del codi HTML, com un **arbre de nodes**. **Cada element HTML**, com ara títols, paràgrafs, imatges, etcètera, es converteix en **un objecte al DOM**, és a dir, en **un node de l'arbre**.

Document: el fitxer HTML.
Object: cada element dins del fitxer HTML.
Model: l'estructura organitzada que tenen aquests objectes entre si (gairebé sempre en forma d'arbre).

3.3. Selecció i accés als elements del DOM.

Què és el DOM?



3.3. Selecció i accés als elements del DOM.

Quan treballem amb JS en una pàgina web, es pot interactuar amb els objectes del DOM per canviar el contingut de la pàgina, modificar estils, afegir o treure elements, etc. **La navegació del DOM permet accedir als nodes i modificar-los de manera dinàmica**.

És important entendre la diferència entre elements i nodes de text.

Un *node* és la unitat bàsica d'un document HTML dins del DOM. Cada node pot representar un element, un atribut, el text o un comentari, dins d'una pàgina web.

Un *element*, és un tipus de node que representa un element HTML concret dins del document. Poden tenir atributs i nodes fills (com un node de text).

3.3. Selecció i accés als elements del DOM.

L'objecte *document* representa tota la nostra pàgina web, per tant, per poder accedir a qualsevol element d'una pàgina HTML sempre hem de fer la trucada a aquest objecte. Disposem dels mètodes següents:

- ***document.getElementById(id)***: cerca un element per l'id de l'element al HTML.
- ***document.getElementsByTagName(name)***: cerca els elements amb l'etiqueta *HTML* indicada.
- ***document.getElementsByClassName(name)***: cerca els elements amb la classe CSS indicada.
- ***document.querySelector(selector)***: cerca el primer element especificat, ja sigui id, etiqueta o clase.
- ***document.querySelectorAll(selector)***: cercar tots els elements especificats. Acepta qualsevol especificació CSS i les seves combinacions.

3.3. Selecció i accés als elements del DOM.

A l'hora d'executar el nostre Javascript, **si necessitem accedir a elements d'HTML, és important que l'HTML s'hagi acabat de carregar** perquè, si no, el nostre Javascript fallarà o no funcionarà com nosaltres esperem, perquè **els elements encara no existeixen en el DOM**.

Per exemple, si l'HTML no s'ha carregat, la funció document.getElementById(id) no es podrà executar perquè no trobareu l'element que esteu intentant cercar. Per evitar-lo tenim diverses opcions:

- Si el nostre Javascript és al <head> de l'HTML, hem d' afegir la propietat defer, ja comentada.
- Afegir el nostre Javascript al final de l'HTML.
- Truqueu a l'escoltador DOMContentLoaded perquè no executi el codi Javascript fins que el document HTML s'hagi carregat completament. Seria de la següent manera:

```
document.addEventListener("DOMContentLoaded", function () {
// Codi JavaScript que volem executa
});
```

3.3. Selecció i accés als elements del DOM.

```html
<body>
    <h1 class="colorBlau">Javascript</h1>
    <p id="paragraf">Paragraf inicial escrit a l' HTML.</p>
    <p id="segonParagraf">Un segon paràgraf escrit a l' HTML.</p>
    <p id="tercerParagraf" class="colorBlau">Un tercer paràgraf escrit a l' HTML.</p>
    <p id="quartParagraf" class="colorBlau">Un quart paràgraf escrit a l' HTML.</p>
</body>
```

```javascript
const paragrafById = document.getElementById("paragraf");
const paragrafByTag = document.getElementsByTagName("p");
const paragrafByClass = document.getElementsByClassName("colorBlau");
const querySelector = document.querySelector("p");
const querySelectorAll = document.querySelectorAll("p.colorBlau");
```

3.3. Selecció i accés als elements del DOM.

Per poder **navegar entre nodes del DOM** amb JavaScript, trobem les següents propietats:

- ***parentNode***: obté el node pare de l'element.
- ***childNodes[nodenumber]***: retorna una llista de nodes fills d'un element (inclou text, comentaris, etc.).
- ***firstChild*****:** obté el primer fill d'un element.
- ***lastChild***: obté l'últim fill d'un element.
- ***nextSibling***: obté el següent node germà d'un element.
- ***previousSibling***: obté l'element anterior al node actual.

3.3. Selecció i accés als elements del DOM.

```
<html>
<head><title>Exemples</title></head>
<body>
    <h1 class="colorBlau">Javascript</h1>
    <p id="paragraf">Hola classe.</p>
    <p id="segonParagraf">Un segon paràgraf.</p>
</body>
</html>

var element = document.getElementById('paragraf');
var pare = element.parentNode;
var fills = element.childNodes;
var primerFill = element.firstChild;
var darrerFill = element.lastChild;
var seguent = element.nextSibling;
var anterior = element.previousSibling;
```

`<html>` és el node arrel i no té nodes pares

`<html>` és el pare de `<head>` i `<body>`

`<head>` és el primer fill de `<html>`

`<body>` és el darrer fill de `<html>`

`<head>` té un fill: `<title>`

`<title>` té un fill (un node de text): "Exemples"

`<body>` té tres fills: `<h1>` i dos `<p>`

`<h1>` té un fill (un node de text): "Javascript"

`<p>` té un fill (un node de text): "Hola classe!"

`<h1>` i `<p>` són germans.

3.4. Creació, modificació i eliminació d'elements.

Tenim diferents mètodes i propietats que ens permeten crear, modificar i eliminar elements i nodes del DOM

- **createElement(tagName):** permet crear un nou node d'un tipus específic.
- **appendChild(node)**: afegeix un node com a fill últim d'un altre node.
- **innerHTML**: modifica el contingut HTML d'un element.
- **textContent**: modifica només el text d'un element. No interpreta el codi HTML
- **replaceChild(*new, old*)**: substitueix un element HTML
- **remove(element)**: elimina un node del DOM.
- **removeChild()**: elimina un node fill d'un altre node.
- ***<attribute>***: canvia el valor de l'atribut d'un element HTML directament.

```
var logo = document.getElementById('logo');
logo.src = "img/imatge2.png";
```

3.4. Creació, modificació i eliminació d'elements.

```
var llista = document.createElement('ul');
var item1 = document.createElement('li');
var item2 = document.createElement('li');
var item3 = document.createElement('li');
item1.innerHTML = '<b>Element 1</b>';
llista.appendChild(item1);
item2.textContent = '<b>Element 2</b>';
llista.appendChild(item2);
item3.innerHTML = '<b>Element 3</b>';
llista.appendChild(item3);

document.body.appendChild(llista);

var primerElement = llista.querySelector('li');
primerElement.remove();
```

3.5. Manipulació d'estils

De la mateixa manera que a l'apartat anterior, podem modificar els estils de la nostra web mitjançant mètodes i propietats de Javascript.

- **style.*property*:** canvia l'estil d'un element HTML indicant la propietat que es vol modificar.
- **classList**: propietat que retorna una llista (de tipus DOMTokenList) amb totes les classes d'un element HTML. Permet utilitzar mètodes específics per afegir, eliminar, alternar o verificar classes.
    - **add**: afegir una classe
    - **remove**: eliminar una classe
    - **toggle**: alterna una classe, es a dir, afegeix la classe si no existeix o l'elimina si ja existeix
    - **contains**: comproba si existeis una classe
    - **replace**: Substitueix una classe existent per una altra.

```
document.getElementById('titol').style.color = 'red';
var element = document.getElementById('segonParagraf');
element.classList.add('actiu');
element.classList.remove('inactiu');
```

3.6. Validació Formularis

Quan creem formularis en HTML, és fonamental assegurar-nos que les dades introduïdes pels usuaris siguin correctes abans d'enviar-les.

L'ús de JavaScript permet validar dades de manera dinàmica abans de l'enviament, millorant la interacció amb l'usuari i reduint la càrrega del servidor.

Podem fer la validació l'atribut ***onsubmit*** del formulari, cridant a una funció de JavaScript, que si retorna false, evitará que el formulari s'enviï.

També podem cridat en l'atribut ***oninput*** de cada camps per validar mentre l'usuari escriu.

Una altra opció és capturar l'esdeveniment submit del formulari amb **addEventListener()** i si hi ha errors, es cancel·la l'enviament del formulari amb **event.preventDefault()**.

3.6. Validació Formularis

```
<form name="form1" action="" onsubmit="return sumar()" method="post">
      <input type="number" name="num1" required>
      <input type="number" name="num2" required>
      <input type="submit" value="Sumar Form 1">
</form>
function sumar(){
    let num1 = parseFloat(document.forms["form1"]["num1"].value);
    let num2 = parseFloat(document.forms["form1"]["num2"].value);
    if (num1 == 0 || num2 == 0){
        alert("Introduïu dos números diferents de 0");
        return false;
    } else {
        alert("Resultat: " + parseFloat(num1 + num2));
        return true;
    }
}
```

3.6. Validació Formularis

```
<form id="form2" name="form2">
      <input type="number" name="num3" required>
      <input type="number" name="num4" required>
      <button type="submit">Sumar Form 2</button>
</form>
document.getElementById("form2").addEventListener("submit", function(event) {
    let valid = true;
    let num3 = parseFloat(document.forms["form2"]["num3"].value);
    let num4 = parseFloat(document.forms["form2"]["num4"].value);
    if (num3 == 0 || num4 == 0){
        alert("Introduïu dos números diferents de 0");
        valid = false;
    } else alert("Resultat: " + parseFloat(num3 + num4));

    if (!valid) event.preventDefault();
});
```

GRÀCIES

Moltíssimes gràcies per la vostra atenció!