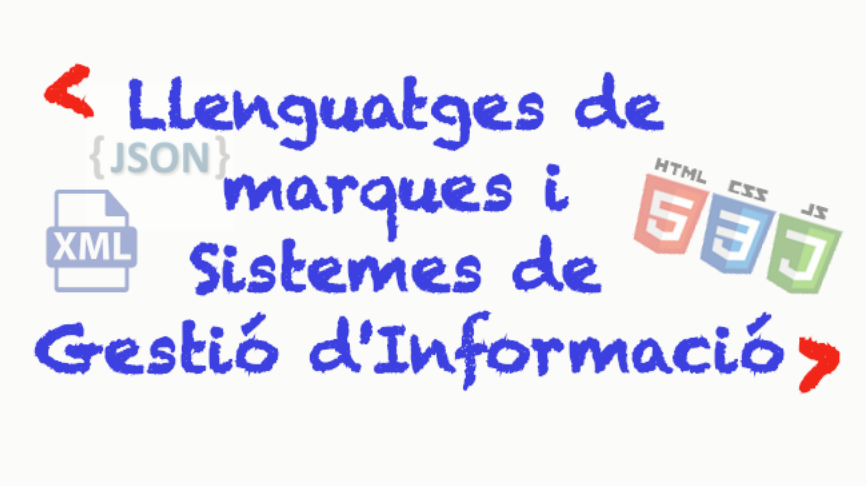
PROGRAMACIÓ AMB JAVASCRIPT

BOM i DOM



Contingut

1. Introducció

Normalment, quan programem en JavaScript, ens trobem dins del context d'un navegador (això no passa si desenvolupem amb NodeJS per exemple).

Dins d'aquest context, tenim alguns objectes predefinits i funcions que podem usar per a interactuar amb aquest navegador i amb el document HTML.

2. Browser Object Model (BOM)

2.1. Objecte window

L'objecte window representa el navegador i és l'objecte principal. Tot está contingut dins de window (variables globals, l'objecte document, l'objecte location, l'objecte navigation, etc.).

L'objecte window pot ser omés accedint a les seues propietats directament.

```
'use strict';
// Tamaño total de la ventana (excluye la barra superior del navegador)
console.log(window.outerWidth + " - " + window.outerHeight);
window.open("https://www.google.com");

// Propiedades de la pantalla
console.log(window.screen.width + " - " + window.screen.height); // Ancho de pantalla y alto (Resolución)
console.log(window.screen.availWidth + " - " + window.screen.availHeight); // Excluyendo la barra del S.O.

// Propiedades del navegador
console.log(window.navigator.userAgent); // Imprime la información del navegador
window.navigator.geolocation.getCurrentPosition(function(position) {
   console.log("Latitude: " + position.coords.latitude + ", Longitude: " + position.coords.longitude);
});

//  Podemos omitir el objeto window (está implícito)
console.log(history.length); // Número de páginas del history. Lo mismo que window.history.length
```

1 Exemple de codi JS. Objecte window

2.2. Timers

Hi ha dos tipus de "timers" que podem crear en JavaScript per a executar algun tros de codi en el futur (especificat en mil·lisegons), timeouts i intervals. El primer s'executa només una vegada (hem de tornar a crear-ho manualment si volem que es repetisca alguna cosa en el temps), i el segon es repeteix cada X mil·lisegons sense parar (o fins que siga cancel·lat).

- **timeout(función, milisegundos)** → Ejecuta una función pasados un número de milisegundos.

```
console.log(new Date().toString()); // Imprime inmediatamente la fecha actual
setTimeout(() => console.log(new Date().toString()), 5000);  // Se ejecutará en 5 segundos (5000 ms)
```

2 Exemple codi JS. Funció setTimeout

- **cleartimeout(timeoutId)** → Cancela un timeout (antes de ser llamado)

```
// setTimeout devuelve un número con el id, y a partir de ahí, podremos cancelarlo
let idTime = setTimeout(() => console.log(new Date().toString()), 5000);
clearTimeout(idTime); // Cancela el timeout (antes de que se ejecute)
```

3 Exemple codi JS. Funció clearTimeout

- **setInterval(funcion, milisegundo)** → La diferencia con timeout es que cuando el tiempo acaba y se ejecuta la función, se resetea y se repite cada X milisegundos automáticamente hasta que nosotros lo cancelemos.

```
let num = 1;
setInterval(() => console.log(num++), 1000); // Imprime un número y lo incrementa cada segundo
```

- **clearInterval(idIntervalo)** → Cancela un intervalo (no se repetirá más).

```
let num = 1;
let idInterval = setInterval(() => {
  console.log(num++);
  if(num > 10) { // Cuando imprimimos 10, paramos el timer para que no se repita más
    clearInterval(idInterval);
  }
}, 1000);
```

- **setInterval/setTimeOut(nombreFuncion, milisegundos, argumentos...)** → Podemos pasarle un nombre función existente. Si se requieren parámetros podemos establecerlos tras los milisegundos.

```
function multiply(num1, num2) {
  console.log(num1 * num2);
}

setTimeout(multiply, 3000, 5, 7); // Después de 3 segundos imprimirá 35 (5*7)
```

4 Exemple codi JS. setInterval, clearInterval

2.3. Objecte location

Conté información sobre l'url actual del navegador

```
console.log(location.href); // Imprime la URL actual
console.log(location.host); // Imprime el nombre del servidor (o la IP) como "localhost" 192.168.0.34
console.log(location.port); // Imprime el número del puerto (normalmente 80)
console.log(location.protocol); // Imprime el protocolo usado (http ó https)

location.reload(); // Recarga la página actual
location.assign("https://google.com"); // Carga una nueva página. El parámetro es la nueva URL
location.replace("https://google.com"); // Carga una nueva página sin guardar la actual en el objeto history
```

5 Exemple codi JS. Objecte location

2.4. Objecte history

Para navegar a través de las páginas que hemos visitado en la pestaña actual, podemos usar el objeto **history**. Este objeto tiene métodos bastante útiles:

```
console.log(history.length); // Imprime el número de páginas almacenadas

history.back(); // Vuelve a la página anterior
history.forward(); // Va hacia la siguiente página
history.go(2); // Va dos páginas adelante (-2 iría dos páginas hacia atrás)
```

6 Exemple codi JS. Accés al objecte History

2.5. Diàlegs

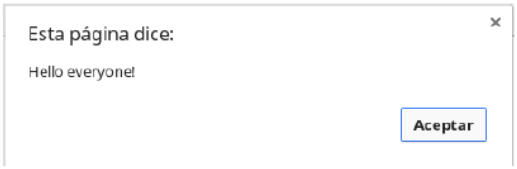
En cada navegador, tenim un conjunt de diàlegs per a interactuar amb l'usuari.

No obstant això, aquests no són personalitzables i per tant cada navegador implementa el seu a la seua manera. Per això no és recomanable usar-los en una aplicació en producció (uniformitat).

En canvi, són una bona opció per a fer proves (en producció hauríem d'usar diàlegs construïts amb HTML i CSS).

El diàleg **alert**, mostra un missatge (amb un botó d'Acceptar) dins d'una finestra. Bloqueja l'execució de l'aplicació fins que es tanca.

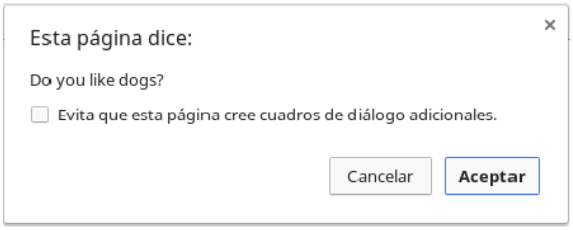
```
alert("Hello everyone!");
```



7 Exemple codi JS. Diàleg "alert"

El diàleg **confirm** és similar, però torna un valor de tipus boolean. Té dos botons (Cancel·lar -> false, Acceptar -> true). L'usuari elegirà entre alguna d'aquestes dues opcions.

```
if(confirm("Do you like dogs?")) {
   console.log("You are a good person");
} else {
   console.log("You have no soul");
}
```



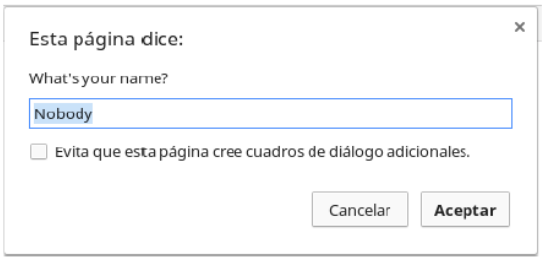
8 Exemple codi JS. Diàleg "confirm"

El diàleg **prompt** mostra un input després del missatge. S'utilitza per donar l'opció a l'usuari d'introduïr algun valor, retornant un string amb el valor introduït.

Si l'usuari prem el botó de Cancel·lar o tanca el diàleg retornarà null. Es pot establir un valor per defecte (segon paràmetre).

```
let name = prompt("What's your name?", "Nobody");

if(name !== "null") {
   console.log("Your name is: " + name);
} else {
   console.log("You didn't answer!");
}
```



9 Exemple codi JS. Diàleg "prompt"

3. Document Object Model (DOM)

El Document Object Model és una estructura en arbre que conté una representació de tots els nodes de l'HTML incloent els seus atributs. En aquest arbre, tot es representa com a node i podem afegir, eliminar o modificar-los.

L'objecte principal del DOM és **document**. Aquest és un objecte global del llenguatge.

Cada node HTML contingut dins del document és un objecte **element**, i aquests elements contenen altres nodes, atributs i estil.

Manipular el DOM usant només JavaScript és una mica més complicat que amb l'ajuda de llibreries com JQuery o frameworks com Angular.

A continuació, veurem alguns mètodes bàsics i propietats dels elements del DOM en JavaScript.

document.documentElement -> Torna l'element <html>

document.head -> Torna l'element <head>

document.body -> Torna l'element <body>

document.getElementById("id") -> Torna l'element que té definit l'id especificat, o null si no existeix

document.getElementByClassName("classe") -> Torna una llista (array) d'elements que tinguen la classe especificada. Si la crida a aquesta funció es realitza desde un node en lloc de des de document, la cerca d'elements es farà a partir d'aquest node.

document.getElementByTagName("etiqueta HTML") -> Torna una llista (array) amb els elements amb l'etiqueta HTML especificada. Per exemple, podem cercar pàrragrafs (p), títols (h1), etc.

element.childNodes -> Torna un llistat amb els elements descendents (fills) del node. Els nodes de tipus text i comentaris també s'obtenen al llistat.

element.children -> Torna un llistat amb els elements descendents (fills) del node. (Exclou comentaris i etiquetes de text)

element.parentNode -> Torna l'element pare del node

element.nextSibling -> Torna el següent node del mateix nivell. Podem utilizar nextElementSibling per obtener sols els elements de tipus HTML.

3.1. Selector Query

Una de les principals característiques que JQuery va introduir quan es va llançar (en 2006) va ser la possibilitat d'accedir als elements HTML basant-se en selectors CSS (classe, id, atributs,….). Des de fa anys, els navegadors han implementat aquesta característica de manera nativa (selector query) sense la necessitat d'usar jQuery.

document.querySelector("selector") -> Torna el primer element coincident

document.querySelectorAll("selector") -> Torna una llista (array) amb tots els elements que coincideixen amb el selector.

```
a → Elementos con la etiqueta HTML <a>
.class → Elementos con la clase "class"
#id → Elementos con el id "id"
.class1.class2 → Elementos que tienen ambas clases, "class1" y "class2"
.class1,.class2 → Elementos que contienen o la clase "class1", o "class2"
.class1 p → Elementos <p> dentro de elementos con la clase "class1"
.class1 > p → Elementos <p> que son hijos inmediatos con la clase "class1"
#id + p → Elemento <p> que va después (siguiente hermano) de un elemento que tiene el id "id"
#id ~ p → Elementos que son párrafos <p> y hermanos de un elemento con el id "id"
.class[attrib] → Elementos con la clase "class" y un atributo llamado "attrib"
.class[attrib="value"] → Elementos con la clase "class" y un atributo "attrib" con el valor "value"
.class[attrib^="value"] → Elementos con la clase "class" y cuyo atributo "attrib" comienza con "value"
.class[attrib*="value"] → Elementos con la clase "class" cuyo atributo "attrib" en su valor contiene "value"
.class[attrib$="value"] → Elementos con la clase "class" y cuyo atributo "attrib" acaba con "value"
```

10 Exemples de selectors CSS que podem utilitzar per trobar elements

File: example1.html

```
<!DOCTYPE>
<html>
  <head>
    <title>JS Example</title>
  </head>
  <body>
    <div id="div1">
      <p>
        <a class="normalLink" href="hello.html" title="hello world">Hello World</a>
        <a class="normalLink" href="bye.html" title="bye world">Bye World</a>
        <a class="specialLink" href="helloagain.html" title="hello again">Hello Again World</a>
      </p>
    </div>
    <script src="./example1.js"></script>
  </body>
</html>
```

File: example1.js

```
console.log(document.querySelector("#div1 a").title); // Imprime "hello world"
console.log(document.querySelector("#div1 > a").title); // ERROR: No hay un hijo inmediato dentro de <div
id="div1"> el cual sea un enlace <a>
console.log(document.querySelector("#div1 > p > a").title); // Imprime "hello world"
console.log(document.querySelector(".normalLink[title^='bye']").title); // Imprime "bye world"
console.log(document.querySelector(".normalLink[title^='bye'] + a").title); // Imprime "hello again"

let elems = document.querySelectorAll(".normalLink");
elems.forEach(function(elem) { // Imprime "hello world" y "bye world"
  console.log(elem.title);
});

let elems2 = document.querySelectorAll("a[title^='hello']"); // Atributo title empieza por "hello..."
elems2.forEach(function(elem) { // Imprime "hello world" y "hello again"
  console.log(elem.title);
});


let elems2 = document.querySelectorAll("a[title='hello world'] ~ a"); // Hermanos de <a title="hello world">
elems2.forEach(function(elem) { // Imprime "bye world" y "hello again"
  console.log(elem.title);
});
```

11 Exemples utilitzant querySelector i querySelectorAll

3.2. Manipulant el DOM

document.createElement("etiqueta") → Crea un element HTML. No estará en el DOM, fins que l'inserim (utilitzant appendChild, per exemple) dins d'un altre element del DOM.

document.createTextNode("text") → Crea un node de text que podem introduir dins d'un element. Equival a element.innerText = "text".

element.appendChild(childElement) → Afegeix un nou element fill al final de l'elemento pare.

element.insertBefore(newChildElement, childElem) → Inserta un nou element fill abans de l'element fill rebut com segon paràmetre.

element.removeChild(childElement) → Elimina el node fill que rep per paràmetre.

element.replaceChild(newChildElem, oldChildElem) → Substitueix un node fill amb un nou node.

3.3. Atributs

Dins dels elements HTML hi ha atributs com name, id, href, src, etc. Cada atribut té nom (name) i valor (value), i aquest pot ser llegit o modificat.

element.attributes → Retorna el array amb els atributs d'un element

element.className → S'usa per a accedir (llegir o canviar) a l'atribut class.

Altres atributs als quals es pot accedir directament són: element.id, element.title, element.style (propietats CSS), … .

element.classList → Array de classes CSS de l'element. A diferència de className, que és una cadena amb les classes separades per espai, aquest atribut te les retorna en forma de array o llista. Té mètodes molt útils per a consultar i modificar classes com:

- **classList.contains("classe"):** true si té la classe.
- **classList.replace("clase1","clase2"):** Lleva la classe "clase1" i la substitueix per la classe "clase2".
- **classList.add("clase1"):** Afig la classe "clase1" a l'element.
- **classList.remove("clase1"):** Li lleva la classe "clase1".
- **classList.toggle("clase1"):** Si no té "clase1", l'afig. En cas contrari, la lleva.

element.hasAttribute("attrName") → Retorna cert si l'element té un atribut amb el nom especificat

element.getAttribute("attrName") → Retorna el valor de l'atribut

element.setAttribute("attrName", "newValue") → Canvia el valor

File: example1.html

```
<!DOCTYPE>
<html>
  <head>
    <title>JS Example</title>
  </head>
  <body>
    <p><a id="toGoogle" href="https://google.es" class="normalLink">Google</a></p>
    <script src="./example1.js"></script>
  </body>
</html>
```

File: example1.js

```
let link = document.getElementById("toGoogle");
link.className = "specialLink"; // Equivale a: link.setAttribute("class", "specialLink");
link.setAttribute("href", "https://twitter.com");
link.textContent = "Twitter";
if(!link.hasAttribute("title")) { // Si no tenía el atributo title, establecemos uno
  link.title = "Ahora voy aTwitter!";
}

/* Imprime: <a id="toGoogle" href="https://twitter.com" class="specialLink"
                title="Ahora voy a Twitter!">Twitter</a> */
console.log(link);
```

12 Exemple codi JS manipulació d'atributs

4. Enllaços d'interés

Objecte window: https://www.w3schools.com/jsref/obj_window.asp

Objecte location: https://www.w3schools.com/jsref/obj_location.asp

Objecte document: https://www.w3schools.com/jsref/dom_obj_document.asp

Objecte history: https://www.w3schools.com/jsref/obj_history.asp

window.navigator: https://www.w3schools.com/jsref/obj_navigator.asp

Objecte Screen: https://www.w3schools.com/jsref/obj_screen.asp

5. Bibliografia

Bernal Mayordomo, Arturo. Curso Programación con JavaScript. Cefire 2020

W3Schools - JavaScript Tutorial