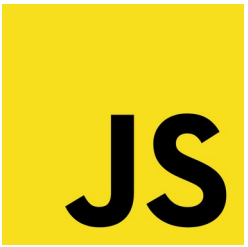
JavaScript : Conceptes bàsics



JavaScript és el llenguatge de programació front-end més popular actualment. És un llenguatge interpretat, és a dir que no requereix compilació abans d'executar-se. Els seus scripts s'executen del costat del client, en el navegador web. Aquesta característica permet a JavaScript afegir major interactivitat i efectes dinàmics a les pàgines web mitjançant la manipulació del contingut retornat des d'un servidor web.

Com afegir JavaScript a la teva pàgina web

Crearem un fitxer .js on hi escriurem totes les instruccions que ens permetran interactuar amb la pàgina web. Llavors el cridem des de qualsevol fitxer .html escrivint aquesta instrucció al final de l'etiqueta <*body*>:

```
<script src="ubicació_fitxer/nom_fitxer.js" type="text/javascript"></script>
```

Pots importar varis fitxers .js des del mateix fitxer .html.

Variables i constants

Les variables i les constants són contenidors de dades.
Hi ha 3 tipus de declaracions:

- **var** : pot canviar el seu valor, el seu àmbit (o scope) és global quan es declara fora d'una funció i local quan es declara dins una funció.
- **let** : pot canviar el seu valor, el seu àmbit és de bloc, només s'hi pot accedir des del bloc («{}») on s'ha declarat.
- **const** : defineix una regla general, ha de ser inicialitzada al declarar-se i el seu valor no pot canviar.

Diferència entre els àmbits d'acció de les variables var i let:
https://github.com/xbaubes/DAM/blob/main/MP4-Llenguatges-marques-SGI/JavaScript/exampleCode/VarLet.js

```
let language = "javascript";
let ImBestDev = true;
const pi = 3.14;
let emptyVar;
```

En aquest cas declarem una variable de tipus *string*, una altra de tipus *boolean* i una constant de tipus *number*. Per últim hem declarat una variable que actualment no té assignat cap valor i per tant cap tipus.

```
typeof language;
```

La funció *typeof* ens retorna el tipus de la variable. En el cas que la variable no tingui valor retorna *undefined*.

JavaScript és un llenguatge de tipus dinàmic, de manera que el tipus de la variable pot canviar amb l'assignació d'un nou valor.

Operadors

- Operadors d'assignació: Tal com hem vist, és la forma d'assignar un valor a una variable.

- Operadors aritmètics: Suma (+), resta (-), Multiplicació (*), Divisió (/), Mòdul (%).

```
valor = valor + 1;
valor += 1;
valor++; //retorna el valor sense modificar i després el modifica
++valor;
```

Quatre formes diferents de sumar 1 a un valor combinant-lo amb l'operador d'assignació.

L'operador «+» també pot ser usat per concatenar strings:

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

La variable resultant text3 conté «John Doe».

- Operadors de comparació: Aquest operador retorna un valor booleà.

a == b	els valors de a i b són iguals.
a === b	els valors i el tipus de a i b són iguals.
a != b	els valors de a i b són diferents.
a !== b	els valors o el tipus de a i b són diferents.
a > b	els valor de a és major que el de b.
a < b	els valor de a és menor que el de b.
a >= b	els valor de a és major o igual que el de b.
a <= b	els valor de a és menor o igual que el de b.

- Operadors lògics: Aquest operador retorna un valor booleà.

a && b	retorna true si a i b són true.
a || b	retorna true si a o b són true.
!a	nega el valor de a.

Taula de veritat → https://ca.wikipedia.org/wiki/Taula_de_veritat

- Operador ternari:

```
let condicio = true, a = 1, b = 2;
let resultat = condicio ? a : b;
```

Si *condicio* és true, *resultat* prendrà el valor de *a*; si *condicio* és false prendrà el de *b*.
En aquest cas: *resultat* = 1.

Els operadors s'executen segons un ordre de prioritat: https://www.w3schools.blog
Pots trencar aquestes regles usant parèntesis:

```
let resultat = (5+4)*3
```

resultat pren el valor 27. Sense parèntesis pren el de 17, ja que per defecte la multiplicació té preferència sobre la suma.

Condicionals

Una sentència condicional agrupa un conjunt d'instruccions que s'executen només si la condició és verdadera.

- **if** : Comprova si la condició es compleix, si és així s'executa el codi que conté entre {}. La declaració pot incloure **else** i **else if** per seguir avaluant altres condicions si l'if no s'ha complert.

```
if (condicio) {
        //  bloc de codi que s'executarà si la condició és true
}
else (condicio) {
        //  bloc de codi que s'executarà si la condició és false
}
```

- **switch** : Permet realitzar diferents accions en funció de diferents estats de les variables. S'avalua si es compleix l'expressió del **case**, si es compleix s'executa el codi sinó s'avalua el següent case; **default** s'executa si tots els altres casos s'han resolt com a falsos.

```
switch (expressio) {
        case x:
                //  bloc de codi que s'executarà si expressio compleix la condició
                break;
        case y:
                //  bloc de codi
                break;
        default:
                //  bloc de codi que s'executarà si expressio no compleix cap de les
                condicions estipulades
}
```

Exemples funcionals: https://github.com/xbaubes/DAM/blob/main/MP4-Llenguatges-marques-SGI/JavaScript/exampleCode/IfSwitch.js

Bucles i iteracions

Permeten l'execució d'un fragment de codi tantes vegades com es vulgui. Si els bucles no es dissenyen correctament pot ser que la condició de finalització mai es compleixi i siguin infinits.

- **for** : Generalment s'utilitzen per realitzar recorreguts.

```
for (expressioInicial ; condicioDeContinuacio ; expressioIncrement)
{
        //  bloc de codi que s'executarà en cada iteració
}
```

  Ordre d'execució:
  1. *expressioInicial* : Inicialitza un o més contadors del bucle. S'executa un sol cop.
  2. *condicioDeContinuacio* : Si el resultat és true continua el bucle i s'executa el codi, sinó finalitza.
  3. S'executa el codi de dins el bucle.
  4. *expressioIncrement* : Augmenta o disminueix el valor de *expressioInicial* i torna al pas 2.

- **while** : Generalment s'utilitzen per realitzar cerques. El bloc de codi s'executa mentre la condició es compleixi, la condició s'ha d'inicialitzar prèviament al bucle; dins el bloc de codi cal incrementar o decrementar alguna variable de la condicio.

```
while (condicio) {
        //  bloc de codi que s'executarà en cada iteració
}
```

- **do ... while** : En la primera iteració el bloc de codi s'executa sense que la condició sigui avaluada, llavors continuarà executant-se mentre la condició es compleixi.

```
do {
        //  bloc de codi que s'executarà en cada iteració
} while (condicio);
```

Exemples funcionals: https://github.com/xbaubes/DAM/blob/main/MP4-Llenguatges-marques-SGI/JavaScript/exampleCode/ForWhile.js

Funcions

Una funció encapsula un bloc de codi dissenyat per una acció concreta.
Les funcions s'executen a l'invocar-les. Se'ls pot passar valors si a l'invocar-les s'escriuen dins el parèntesi, el fet de passar els arguments com a valor provoca que al modificar una variable dins la funció no alteri la variable original.
Poden retornar un valor si usem la sentència «return».

```
function testingFunction (argument) {
        //  bloc de codi que s'executarà al cridar la funció
}
```

Les funcions s'utilitzen perquè permeten reutilitzar el codi fàcilment i tantes vegades com es vulgui, sense haver de reescriure'l.

Exemple funcional d'una funció amb dos paràmetres d'entrada i amb retorn de valor:
https://github.com/xbaubes/DAM/blob/main/MP4-Llenguatges-marques-SGI/JavaScript/exampleCode/Function.js

BIBLIOGRAFIA I WEBGRAFIA

«JavaScript Tutorial». https://www.javascripttutorial.net/
«JAVASCRIPT.INFO». https://javascript.info/
«Tutorials Point». https://www.tutorialspoint.com/javascript/index.htm
«W3Schools». https://www.w3schools.com/js/js_intro.asp
«Yeison Daza». https://yeisondaza.com/entendiendo-scopes-de-variables-en-javascript

---

Autor: Xavier Baubés Parramon
Aquest document es llicència sota Creative Commons versió 4.0.
Es permet compartir i adaptar el material però reconeixent-ne l'autor original.