

# Mètodes de console

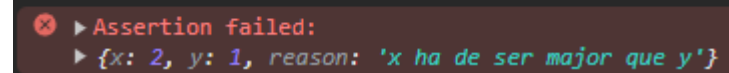
En el desenvolupament d'aplicacions web, la consola és una eina essencial per a la depuració i el seguiment del comportament del nostre codi. JavaScript proporciona l'objecte console, que conté una varietat de mètodes que permeten imprimir informació, errors, avisos i més a la consola del navegador.

En aquesta exploració, veurem diversos mètodes proporcionats per console en JavaScript i proporcionarem exemples pràctics per demostrar-ne l'ús. Analitzarem com utilitzar mètodes per millorar la depuració i facilitar l'anàlisi del nostre codi. Això ens ajudarà a entendre com utilitzar aquestes funcions per mostrar informació rellevant i millorar l'eficiència del nostre procés de desenvolupament.

## `console.assert()`

Aquesta funció s'utilitza per avaluacions condicionals. Si l'expressió proporcionada com a primer argument és falsa, l'assertió s'imprimeix a la consola amb el missatge proporcionat com a segon argument.

```
> let x = 2
   let y = 1
   let reason = 'x ha de ser major que y'
   console.assert(x < y, {x, y, reason})
```



## `console.clear()`

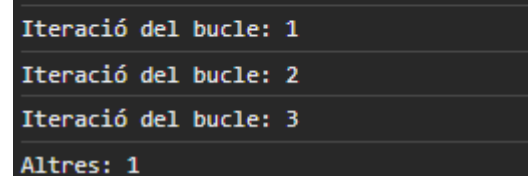
Neteja la consola.

## `console.count()`

Registra el número de vegades que s'ha cridat aquesta línia de codi des de l'inici de l'execució, utilitzant una etiqueta opcional.

```
for (let i = 0; i < 3; i++) {
  console.count('Iteració del bucle');
}
```

```
console.count('Altres');
```



## `console.countReset()`

Reinicia el valor del comptador per a l'etiqueta donada.

### `console.dir()`

Mostra un llistat interactiu de les propietats d'un objecte JavaScript específic. Aquest llistat permet fer servir els triangles de la llista desplegable per a examinar el contingut d'objectes fill.

Aquest mètode és la forma de veure totes les propietats d'un objecte JavaScript especificat per consola, mitjançant la qual els desenvolupadors poden fàcilment obtenir les propietats de l'objecte.

```
console.dir(document.location)

▼ Location ⓘ
  ▶ ancestorOrigins: DOMStringList {length: 0}
  ▶ assign: f assign()
    hash: ""
    host: "rogeriesmontsia.github.io"
    hostname: "rogeriesmontsia.github.io"
    href: "https://rogeriesmontsia.github.io/"
    origin: "https://rogeriesmontsia.github.io"
    pathname: "/"
    port: ""
    protocol: "https:"
  ▶ reload: f reload()
  ▶ replace: f replace()
    search: ""
```

### `console.dirxml()`

Mostra una representació en forma d'arbre d'un element XML/HTML si és possible o la vista de l'objecte JavaScript si no és possible.

```
console.dirxml(document.location)

VM155:1
Location {ancestorOrigins: DOMStringList, href:
  ▶ 'https://rogeriesmontsia.github.io/', origin:
    'https://rogeriesmontsia.github.io', protocol:
      'https:', host: 'rogeriesmontsia.github.io', ...}
```

### `console.error()`

Mostra un missatge d'error. Pot fer servir substitucions de cadenes i arguments addicionals amb aquest mètode.

```
> console.error('Això és un error!')

✖ ▶ Això és un error!
```

### `console.group()`

Crea un nou grup, indentant tots els missatges subsegüents en un nou nivell. Per a retrocedir un nivell, s'utilitza `groupEnd()`.

```
> console.log('Missatge 1 fora del grup');
Missatge 1 fora del grup

console.group('Agrupació 1');
console.log('Missatge 2 dins de 1\'Agrupació 1');
console.log('Missatge 3 dins de 1\'Agrupació 1');
console.groupEnd();

console.group('Agrupació 2');
console.log('Missatge 4 dins de 1\'Agrupació 2');
console.log('Missatge 5 dins de 1\'Agrupació 2');
console.groupEnd();

console.log('Missatge 6 fora del grup');
Missatge 6 fora del grup

▼ Agrupació 1
  Missatge 2 dins de 1'Agrupació 1
  Missatge 3 dins de 1'Agrupació 1
▼ Agrupació 2
  Missatge 4 dins de 1'Agrupació 2
  Missatge 5 dins de 1'Agrupació 2
```

### `console.groupCollapsed()`

Crea un grup nou, indentant tots els missatges subsegüents en un nivell nou. A diferència de `group()`, s'inicia amb la línia de grup col·lapsada, requerint l'ús d'un botó d'obertura per expandir el grup. Per retrocedir un nivell, s'utilitza `groupEnd()`.

```
> console.log('Missatge 1 fora del grup');

console.groupCollapsed('Agrupació Col·lapsada 1');
console.log('Missatge 2 dins de 1\Agrupació Col·lapsada 1');
console.log('Missatge 3 dins de 1\Agrupació Col·lapsada 1');
console.groupEnd();

console.groupCollapsed('Agrupació Col·lapsada 2');
console.log('Missatge 4 dins de 1\Agrupació Col·lapsada 2');
console.log('Missatge 5 dins de 1\Agrupació Col·lapsada 2');
console.groupEnd();

console.log('Missatge 6 fora del grup');
Missatge 1 fora del grup
▶ Agrupació Col·lapsada 1
▶ Agrupació Col·lapsada 2
Missatge 6 fora del grup
```

### `console.groupEnd()`

Finalitza el grup actual (s'ha fet ús als exemples vistos anteriorment)

### `console.info()`

Mostra un missatge d'informació a la consola. Pot fer servir substitucions de cadenes i arguments addicionals amb aquest mètode.

```
> console.info('Estic informant!')
Estic informant!
```

### `console.log()`

Per a sortida general de la informació enregistrada. Pot fer servir substitucions de cadenes i arguments addicionals amb aquest mètode.

### `console.table()`

Mostra les dades en forma de taula.

Ja sigui d'un Array:

```
let vehicles = ['BMW', 'Audi', 'Mercedes']
console.table(vehicles)
```

VM653:3	
(index)	Value
0	'BMW'
1	'Audi'
2	'Mercedes'

▶ Array(3)

Com d'un Array d'objectes:

```
let vehicles = [  
  {  
    marca: 'BMW',  
    model: 'M5'  
  },  
  {  
    marca: 'Audi',  
    model: 'R8'  
  }  
];  
  
console.table(vehicles);
```

(index)	marca	model
0	'BMW'	'M5'
1	'Audi'	'R8'

► Array(2)

`console.time()`

Inicia un temporitzador amb un nom especificat com a paràmetre.

```
console.time('temporitzador');
```

`console.timeEnd()`

Detén el temporitzador especificat i registra el temps transcorregut en mil·lèsimes de segon des de que ha estat iniciat.

```
console.timeEnd('temporitzador')  
temporitzador: 105680.34399414062 ms
```

`console.timeLog()` (en-US)

Mostra el valor del temporitzador especificat a la consola.

```
console.timeLog('temporitzador')  
temporitzador: 42120.01513671875 ms
```

### `console.trace()`

Aquesta funció és una eina que proporciona una representació de la pila de trucades en ordre invers. En altres paraules, mostra l'ordre en què les funcions s'han executat o cridat, començant per la funció més recent i anant cap a enrere fins a la funció inicial.

```
function funcioA() {
  console.log('Cridem a funció A:');
  console.trace();
}

function funcioB() {
  console.log('Cridem a funció B:');
  funcioA();
}

function funcioC() {
  console.log('Cridem a funció C:');
  funcioB();
}

funcioC();
```

Cridem a funció C:  
Cridem a funció B:  
Cridem a funció A:

▼ console.trace

funcioA	@ VM311:3
funcioB	@ VM311:8
funcioC	@ VM311:13
(anonymous)	@ VM311:16

### `console.warn()`

Mostra un missatge d'advertència. Pot fer servir substitucions de cadenes i arguments addicionals amb aquest mètode.

```
> console.warn('Això és un warning!')
```

⚠ ▶ Això és un warning!

## Diferència entre == i ===

L'operador d'estricta igualtat (===) és utilitzat per comparar dos operands i produeix un resultat booleà. A diferència de l'operador d'igualtat convencional (==), l'operador d'estricta igualtat considera sempre que operands amb diferents tipus de valor són diferents i mai similars.

Com podem veure en l'exemple següent, l'operador d'igualtat convencional intenta igualar els valors realitzant conversions de tipus abans de la comparació.

```
console.log(5 == "5")
console.log(1 == true)
console.log(0 == false)

true
true
true
```

Mentre que l'operador d'estricta igualtat retorna fals en veure que són els mateixos valors però de tipus diferents

```
console.log(5 === "5")
console.log(1 === true)
console.log(0 === false)

false
false
false
```

# Webgrafia

<https://developer.mozilla.org/es/docs/Web/API/console> [13/09/2023]

[https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Strict\\_equality](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Strict_equality) [13/09/2023]

<https://midu.dev/los-ocho-metodos-de-console-que-debes-conocer/> [17/09/2023]