01. Introducció JavaScript

IES L'ESTACIÓ 2021-2022

Índex

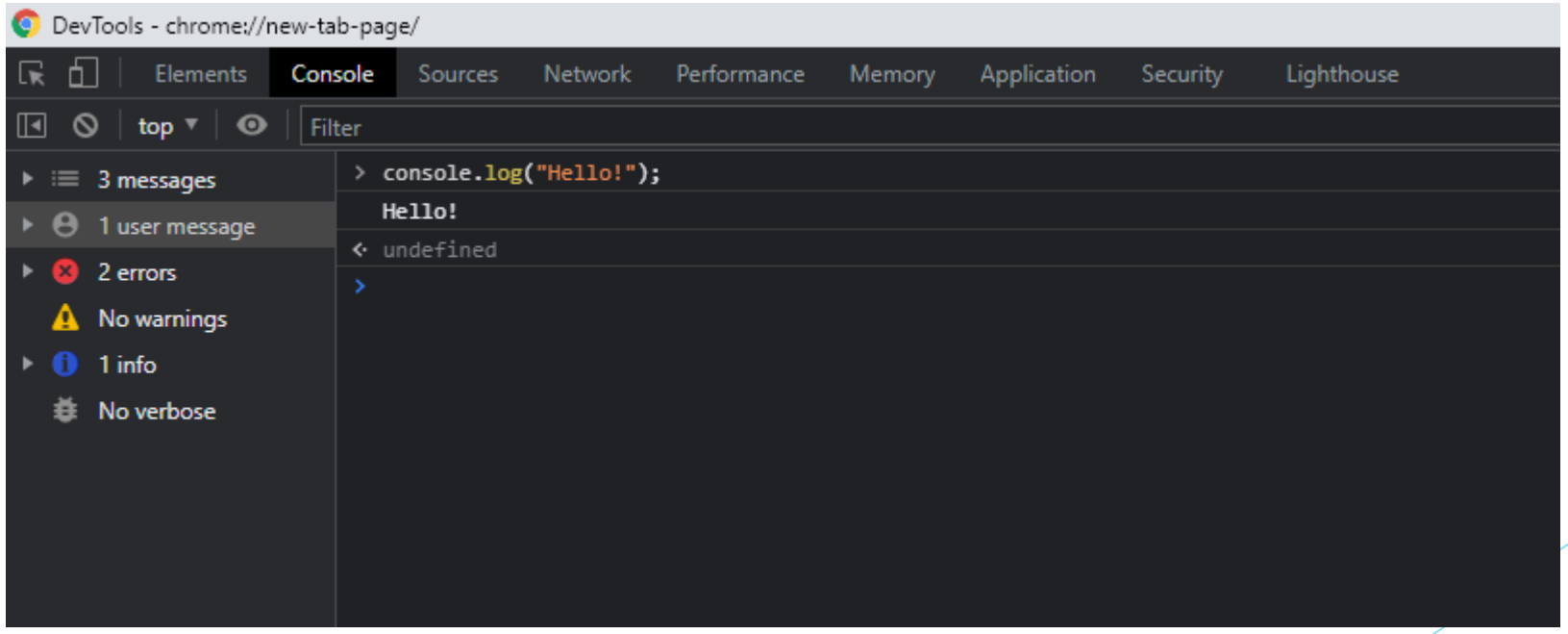
Índex

1. ¿Què és JavaScript?

- JavaScript és un llenguatge de scripting multiplataforma i orientat a objectes.
- S'utilitza principalment per a crear pàgines web dinàmiques.
- Una pàgina web dinàmica és aquella que incorpora efectes:
  - Text que apareix i desapareix.
  - Animacions.
  - Accions en polsar botons.
  - Finestres amb missatges per l'usuari, etc.
- És un llenguatge interpretat, pot provar-se directament en qualsevol navegador. (F12).
- https://t.co/Fn1XedydVs https://t.co/4lbGp0N5zt

2. La consola JavaScript.

- Mostra informació sobre la pàgina web que està executant-se en eixe moment, i també inclou una linia de comando per executar expressions JavaScript en la página actual.
- La funció console.log() mostra la informació proporcionada en la consola JavaScript:



3. Comentaris.

- JavaScript té dos tipus de sintaxi de comentaris:
  - //Comentari d'una linia.
  - /* Comentari multi-linia*/
- /*no ets pot*//*anidar comentaris*/ -> SyntaxError

- Els cometaris són anotacions en el codi font d'un programa que són ignorats per l'intèrpret.

- Els comentaris han d'utilitzar-se per descriure aspectes importants, per exemple, aspectes que permetran una major compressió del codi.

4. Integrant JavaScript amb HTML

- Dins de l'etiqueta <script></script>.
- Dins del <head></head> o dins del <body></body>.
  - Exemple:

```
<!DOCTYPE>
<html>
    <head>
        <title>Ejemplo JS</title>
    </head>
    <body>
        <p>Hola Mundo!</p>
        <script>
            console.log("Hola Mundo!");
        </script>
    </body>
</html>
```

4. Integrant JavaScript amb HTML

- En un arxiu a part.
  - Arxiu: exemple1.html

```
<!DOCTYPE>
<html>
    <head>
        <title>Ejemplo JS</title>
    </head>
    <body>
        <p>Hola Mundo!</p>
        <script src="ejemplo1.js"></script>
    </body>
</html>
```

  - Arxiu exemple1.js

```
console.log("Hola Mundo!");
```

5. Variables.

- Una variable és un recurs de memòria reservat per emmagatzemar informació.
- El nom de la variable començarà per minúscula, subratllat (_nom) o per $ ($nom).
- JS diferencia entre majúscules i minúscules (CASE SENSITIVE).
- En JS les variables no tenen un tipus de dada explícit, pot canviar depenent del valor que assignen.
- Si declaren una variable i no li assignen un valor, tindrà un valor per defecte conegut com a undefined (aquest valor és diferent de null).

5. Variables.

- Hi ha 3 tipus de declaracions de variables en JS:
  - var: declara una variable, inicialitzant-la opcionalment a un valor. Podrà canviar el seu valor i scope. És local.
  - let: declara una variable local en un bloc d'àmbit, inicialitzant-la opcionalment a un valor. Podrà canviar el seu valor.
  - const: declara una variable de només lectura en un bloc d'àmbit. No es pot canviar el seu valor.
- Àmbit d'una variable:
  - Global: declarada fora d'una funció, és accessible per qualsevol altre codi en el document actual.
  - Local: declarada dins d'una funció, només està disponible per a eixa funció.

6. Tipus de dades.

- 6.1 Number.
  - No hi ha diferència entre números enters i decimals (float, double): number.

```
console.log(typeof 3); // Imprime number
console.log(typeof 3.56); // Imprime number
```

```
let num = 3.2e-3; // 3.2*(10^-3)
console.log(num); // Imprime 0.0032
```

  - Els números són objectes, si posem un punt després d'escriure un número, podrem accedir a alguns mètodes o propietats.

```
console.log(3.32924325.toFixed(2)); // Imrpime 3.33
console.log(5435.45.toExponential()); // Imrpime 5.43545e+3
console.log((3).toFixed(2)); // Imrpime 3.00 (Un entero necesita estar dentro de un paréntesis para poder acceder a sus propiedades)
```

  - Objecte global: Number (propietats molt útils per a treballar amb números).

```
console.log(Number.MIN_VALUE); // Imprime 5e-324 (El número más pequeño)
console.log(Number.MAX_VALUE); // Imprime 1.7976931348623157e+308 (El número más grande)
```

6. Tipus de dades.

- 6.1 Números.
  - Infinit i -Infinit

```
console.log(Number.MAX_VALUE * 2); // Imprime Infinity
console.log(Number.POSITIVE_INFINITY); // Imprime Infinity
console.log(Number.NEGATIVE_INFINITY); // Imprime -Infinity
console.log(typeof Number.POSITIVE_INFINITY); // Imprime number

let number = Number.POSITIVE_INFINITY / 2; // Sigue siendo todavía infinito!!
if(isFinite(number)) { // Es igual que (number !== Infinity && number !== -Infinity)
   console.log("El número es " + number);
} else { // Enters here
   console.log("El número no es finito");
}
```

6. Tipus de dades.

- **6.2 Operacions amb números.**
  - Podem realitzar qualsevol operació típica (+,-,*,/,%,…..).
  - ¿Què ocorre quan realitzen una operació numèrica amb valors que no són números? NaN (Not a number).

```
let a = 3;

let b = "asdf";
let r1 = a * b; // b es “asdf”, y no será transformado a número
console.log(r1); // Imprime NaN

let c;
let r3 = a + c; // c es undefined, no será transformado a número
console.log(r3); // Imprime NaN

let d = "12";
console.log(a * d); // Imprime 36. d puede ser transformado al  número 12
console.log(a + d); // Imprime 312. El operador + concatena si hay un string
console.log(a + +d); // Imprime 15.  El operador ‘+’ delante de un valor lo transforma en numérico
```

  - Number.isNaN(valor): retorna true si es NaN.

6. Tipus de dades.

- **6.3. Undefined i Null.**
  - En JS, si no assignen un valor en definir una variable o paràmetre, serà inicialitzada amb el valor undefined.
  - Null: denota valor nul. Ja que, JS es CASE SENSITIVE, null no es igual que Null o NULL.
  - No ens hem de confondre amb l'assignació del valor null a una variable.

```
let value; // Value no ha sido asignada (undefined)
console.log(typeof value); // Imprime undefined

value = null;
console.log(typeof value); // Imprime object
```

- **6.4 Boolean.**
  - true or false, per a negar-los utilitzaren el signe !

6. Tipus de dades.

- **6.5 String.**
  - Els valors string es representen dins de ‘cometes simples’ o “cometes dobles”. Podem utilitzar l'operador + per a concatenar cadenes.

```
let s1 = "Esto es un string";
let s2 = 'Esto es otro string';

console.log(s1 + " - " + s2); // Imrpime: Esto es un string – Esto es otro string
```

  - Per declarar ‘cometes simples’ o “cometes dobles” dins d’una cadena:

```
console.log("Hello 'World'"); // Imprime: Hello 'World'
console.log('Hello \'World\'); // Imprime: Hello 'World'

console.log("Hello \"World\""); // Imprime: Hello "World"
console.log('Hello "World"'); // Imprime: Hello "World"
```

6. Tipus de dades.

- **6.5 String.**
  - Els string també són objectes i podem utilitzar diferents funcions.

```
let s1 = "Esto es un string";
// Obtener la longitud del string
console.log(s1.length); // Imprime 17

// Obtener el carácter de una cierta posición del string (Empieza en 0)
console.log(s1.charAt(0)); // Imprime "E"

// Obtiene el índice de la primera ocurrencia
console.log(s1.indexOf("s")); // Imprime 1

// Obtiene el índice de su última ocurrencia
console.log(s1.lastIndexOf("s")); // Imprime 11

// Devuelve un array con todas las coincidencias en de una expresión regular
console.log(s1.match(/.s/g)); // Imprime ["Es", "es", " s"]

// Obtiene la posición de la primera ocurrencia de una expresión regular
console.log(s1.search(/[aeiou]/)); // Imprime 3

// Reemplaza la coincidencia de una expresión regular (o string) con un string (/g opcionalmente reemplaza
todas)
console.log(s1.replace(/i/g, "e")); // Imprime "Esto es un streng"
```

6. Tipus de dades.

- **6.5 String.**

```
// Devuelve un substring (posición inicial: incluida, posición final: no incluida)
console.log(s1.slice(5, 7)); // Imprime "es"

// Igual que slice
console.log(s1.substring(5, 7)); // Imprime "es"

// Como substring pero con una diferencia (posición inicial, número de caracteres desde la posición inicial)
console.log(s1.substr(5, 7)); // Imprime "es un s"

// Transforma en minúsculas, toLowerCase no funciona con caracteres especiales (ñ, á, é, ...)
console.log(s1.toLocaleLowerCase()); // Imprime "esto es un string"

// Transforma a mayúsculas
console.log(s1.toLocaleUpperCase()); // Imprime "ESTO ES UN STRING"

// Devuelve un string eliminando espacios, tabulaciones y saltos de línea del principio y final
console.log("   String con espacios   ".trim()); // Imprime "String con espacios"

// Devuelve si una cadena empieza por una determinada subcadena
console.log(s1.startsWith("Esto")); // Imprime true

// Devuelve si la cadena acaba en la subcadena recibida
console.log(s1.endsWith("string")); // Imprime true

// Devuelve si la cadena contiene la subcadena recibida
console.log(s1.includes("es")); // Imprime true
```

6. Tipus de dades.

- **6.6 Objecte.**
  - Object {}: Pot contindre qualsevol tipus de variable: text, números, arrays, booleans, altres objectes i fins i tot funcions.
  - Per a accedir al valor emmagatzemat en una propietat d'un objecte hem d'escriure «persona.nombrePropiedad»
  - Una particularitat molt important és que els objectes no necessiten tindre les seues propietats ordenades.
  - Cada propietat se separa per coma «,» a excepció de l'última.

```
var persona = {
    nombre:"Francesc",
    apellido:"Ricart",
    aficiones:["html","css","javascript"],
    inscrito:1
}

console.log(persona.nombre);          // devuelve Francesc
console.log(persona.apellido);        // devuelve Ricart
console.log(persona.aficiones[0]);    // devuelve html
console.log(persona.aficiones[persona.aficiones.length-1]);
                                      // devuelve javascript
console.log(persona.inscrito)         // devuelve 1
```

6. Tipus de dades.

- **6.6 Objecte.**
  - Object {}: Pot contindre qualsevol tipus de variable: text, números, arrays, booleans, altres objectes i fins i tot funcions.
  - Per a accedir al valor emmagatzemat en una propietat d'un objecte hem d'escriure «persona.nombrePropiedad»
  - Una particularitat molt important és que els objectes no necessiten tindre les seues propietats ordenades.
  - Cada propietat se separa per coma «,» a excepció de l'última.

```
var persona = {
    nombre:"Francesc",
    apellido:"Ricart",
    aficiones:["html","css","javascript"],
    inscrito:1
}

console.log(persona.nombre);          // devuelve Francesc
console.log(persona.apellido);        // devuelve Ricart
console.log(persona.aficiones[0]);    // devuelve html
console.log(persona.aficiones[persona.aficiones.length-1]);
                                      // devuelve javascript
console.log(persona.inscrito)         // devuelve 1
```

7. Funcions.

- En JS declaren funcions utilitzant la paraula reservada function abans del nom de la funció. Els arguments van dins del parèntesi darrere del nom de la funció.

```
function sayHello(name) {
   console.log("Hello " + name);
}

sayHello("Tom"); // Imprime "Hello Tom"
```

- No cal declarar la funció abans de cridar-la, ja què l'interpret de JS primer procesa les declaracions de variables i funcions y després executa la resta de codi.
- Retorn de valors.
  - Podem utilitzar la paraula reservada return per tornar un valor en una funció. Si intentem obtindre un valor d'una funció que no retorna res, obtindrem undefined.

```
function totalPrice(priceUnit, units) {
   return priceUnit * units;
}
   let total = totalPrice(5.95, 6);
   console.log(total);  // Imprime 35.7
```

7. Funcions.

- **Funcions anònimes**
  - La manera de declarar una funció anònima és no assignar-li cap nom.
  - Podem assignar aquesta funció com a valor a una variable, ja que és un tipus de valor (com pot ser un string o número), per tant, pot ser assignada a (o referenciada des de) múltiples variables.
  - S'utilitza igual que una funció clàssica.
  - Les funcions anònimes ens permeten no assignar-li un nom a un conjunt d'instruccions que desitgem executar sense necessitat d'associar-lo.
  - El podem utilitzar comunament quan tenim un "**callback**" o quan volem aïllar una funció d'algun altre element

```
var miFuncion = function(valor){
        valor *= 2;
        return valor;
};
var total = 2;
for (i = 0; i <10; i++)
{
        total = miFuncion(total);
}
alert(total);
```

7. Funcions.

- **Funcions lambda (arrow functions).**
  - Aquestes expressions ofereixen la possibilitat de crear funcions anònimes però amb alguns avantatges.
  - Veurem les diferències que existeixen creant dues funcions equivalents (una anònima i una altra lambda que fan el mateix):

```
let sum = function(num1, num2) {
   return num1 + num2;
}
console.log(sum(12,5)); // Imprime 17

let sum = (num1, num2) => num1 + num2;
console.log(sum(12,5)); // Imprime 17
```

  - Quan declarem una funció lambda, la paraula reservada function no s'usa. Si només es rep un paràmetre, els parèntesis poden ser omesos. Després dels paràmetres ha d'anar una fletxa (=>), i el contingut de la funció.

```
let square = num => num * num;
console.log(square(3)); // Imprime 9
```

```
let sumInterest = (price, percentage) => {
   let interest = price * percentage / 100;
   return price + interest;
}
console.log(sumInterest(200,15)); // Imprime 230
```

8. Exercici.

- Treballarem tots els coneixements de JS, CSS i HTML programant jocs. El primer joc amb el qual treballarem serà el 3 en ratlla.
- Segons avancen en la teoria i amb els coneixements del llenguatge, la funcionalitat i el codi han d'anar optimitzant-se.
- Objectius a aconseguir:
  - - Arrays.
  - - DOM.
  - - Events.

9. Estructures.

- 9.1 condicional (if).

```
let price = 65;

if(price < 50) {
    console.log("Esto es barato!");
} else if (price < 100) {
    console.log("Esto no es barato...");
} else {
    console.log("Esto es caro!");
}
```

9. Estructures.

- 9.2 switch.

```
let userType = 1;

switch(userType) {
   case 1:
   case 2: // Tipos 1 y 2 entran aquí
      console.log("Puedes acceder a esta zona");
      break;
   case 3:
      console.log("No tienes permisos para acceder aquí");
      break;
   default: // Ninguno de los anteriores
      console.error("Tipo de usuario erróneo!");
}
```

9. Estructures.

- 9.3 switch (condicional).
  - **switch** es comporta com un **if**.

```
let age = 12;

switch(true) {
   case age < 18:
      console.log("Eres muy joven para entrar");
      break;
   case age < 65:
      console.log("Puedes entrar");
      break;
   default:
      console.log("Eres muy mayor para entrar");
}
```

10. Bucles.

- 10.1 while.

```
let value = 1;

while (value <= 5) { // Imprime 1 2 3 4 5
   console.log(value++);
}
```

- 10.2 do … while.

```
let value = 1;

do { // Imprime 1 2 3 4 5
   console.log(value++);
} while (value <= 5);
```

10. Bucles.

- 10.3 for.

```
let limit = 5;

for (let i = 1; i <= limit; i++) { // Imprime 1 2 3 4 5
   console.log(i);
}

  let limit = 5;

  for (let i = 1, j = limit; i <= limit && j > 0; i++, j--) {
     console.log(i + " - " + j);
  }
  /* Imprime
  1 - 5
  2 - 4
  3 - 3
  4 - 2
  5 - 1
  */
```

//Inicialitzen una o més variables

11. Operadors aritmètics.

- 11.1 Aritmètics.
  - Suma (+): pot utilitzar-se per sumar números o concatenar cadenes.

```
console.log(4 + 6); // Imprime 10
console.log("Hello " + "world!"); // Imprime "Hello world!"
console.log("23" + 12); // Imprime "2312"
console.log("42" + true); // Imprime "42true"
console.log("42" + undefined); // Imprime "42undefined"
console.log("42" + null); // Imprime "42null"
console.log(42 + "hello"); // Imprime "42hello"
console.log(42 + true); // Imprime 43 (true => 1)
console.log(42 + false); // Imprime 42 (false => 0)
console.log(42 + undefined); // Imprime NaN (undefined no puede ser convertido a number)
console.log(42 + null); // Imprime 42 (null => 0)
console.log(13 + 10 + "12"); // Imprime "2312" (13 + 10 = 23, 23 + "12" = "2312")
```

  - Si un operand és string, sempre es realitzarà una concatenació, per tant s'intentarà transformar l'altre valor en un string (si no ho és). En cas contrari, intentarà fer una suma (convertint valors no numerics a número). Si la conversió del valor a número falla, retornarà NaN (Not a Number).

12. Operadors aritmètics.

- 11.1 Aritmètics.
  - Resta (-), multiplicació (*), divisió (/) i resto (%) operen sempre amb números, per tant, cada operand ha de ser convertit a número (si no ho era previament).

```
console.log(4 * 6); // Imprime 24
console.log("Hello " * "world!"); // Imprime NaN
console.log("24" / 12); // Imprime 2 (24 / 12)
console.log("42" * true); // Imprime 42 (42 * 1)
console.log("42" * false); // Imprime 0 (42 * 0)
console.log("42" * undefined); // Imprime NaN
console.log("42" - null); // Imprime 42 (42 - 0)
console.log(12 * "hello"); // Imprime NaN ("hello" no puede ser convertido a número)
```

11. Operadors aritmètics.

- 11.2 Unaris.
  - En Javascript podem preincrementar (++variable), postincrementar (variable++), predecrementar (--variable) i postdecrementar (variable--).

```
let a = 1;
let b = 5;
console.log(a++); // Imprime 1 y incrementa a (2)
console.log(++a); // Incrementa a (3), e imprime 3
console.log(++a + ++b); // Incrementa a (4) y b (6). Suma (4+6), e imprime 10
console.log(a-- + --b); // Decrementa b (5). Suma (4+5). Imprime 9. Decrementa a (3)
```

  - També, podem usar els signes – i + davant d'un número per a canviar o mantindre el signe del número. Si apliquem aquests operadors amb una dada que no és un número, aquest serà convertit a número primer. Per això, és una bona opció usar +value per a convertir a número, la qual cosa equival a usar Number(value).

```
let a = "12";
let b = "13";
let c = true;
console.log(a + b); // Imprime "1213"
console.log(+a + +b); // Imprime 25 (12 + 13)
console.log(+b + +c); // Imprime 14 (13 + 1). True -> 1
```

11. Operadors aritmètics.

- 11.3 Relacionals.
  - L'operador de comparació, compara dos valors i retorna un booleà (true o false) Aquests operadors són pràcticament els mateixos que en la majoria de llenguatges de programació, a excepció d'alguns, que veurem a continuació.
  - Podem usar == o === per a comparar la igualtat (o el contrari !=, !==). La principal diferència és que el primer, no té en compte els tipus de dades que estan sent comparats, compara si els valors són equivalents. Quan usem ===, els valors a més han de ser del mateix tipus. Si el tipus de valor és diferent (o si és el mateix tipus de dada però diferent valor) retornarà false. Retorna true quan tots dos valors són idèntics i del mateix tipus.

```
console.log(3 == "3"); // true
console.log(3 === "3"); // false
console.log(3 != "3"); // false
console.log(3 !== "3"); // true
// Equivalente a falso (todo lo demás es equivalente a cierto)
console.log("" == false); // true
console.log(false == null); // false (null no es equivalente a cualquier boolean).
console.log(false == undefined); // false (undefined no es equivalente a cualquier boolean).
console.log(null == undefined); // true (regla especial de JavaScript)
console.log(0 == false); // true
console.log({} >= false); // Object vacío -> false
console.log([] >= false); // Array vacío -> true
```

11. Operadors aritmètics.

- 11.3 Relacionals.
  - Altres operadors relacions per a números o strings són: menor que (<), major que (>), menor o igual que (<=), i major o igual que (>=).
  - Quan comparem un string amb aquests operadors, es va comparant caràcter a caràcter i es compara la seua posició en la codificació Unicode per a determinar si és menor (situat abans) o major (situat després).
  - A diferència de l'operador de suma (+), quan un dels dos operands és un número, l'altre serà transformat en número per a comparar.
  - Per a poder comparar com string, tots dos operands han de ser string.

```
console.log(6 >= 6); // true
console.log(3 < "5"); // true ("5" → 5)
console.log("adiós" < "bye"); // true
console.log("Bye" > "Adiós"); // true
console.log("Bye" > "adiós"); // false. Las letras mayúsculas van siempre antes
console.log("ad" < "adiós"); // true
```

11. Operadors aritmètics.

- 11.4 Booleans.
  - Els operadors booleans són negació (!), i (&&), o (||). Aquests operadors normalment són usats de forma combinada amb els operadors relacionals formant una condició més complexa, la qual retorna true o false.

```
console.log(!true); // Imprime false
console.log(!(5 < 3)); // Imprime true (!false)

console.log(4 < 5 && 4 < 2); // Imprime false (ambas condiciones deben ser ciertas)
console.log(4 < 5 || 4 < 2); // Imprime true (en cuanto una condición sea cierta, devuelve cierta y deja de comparar)
```

11. Operadors aritmètics.

- 11.4 Booleans.
  - Es pot usar l'operador i, o l'operador o amb valors que no són booleans, però es pot establir equivalència.
  - Amb l'operador or, en trobar-se un true o equivalent, ho retornarà sense continuar avaluant la resta.
  - L'operador and en avaluar les condicions, si alguna d'elles és falsa o equivalent no continua avaluant.
  - Sempre es retorna l'última expressió avaluada.

```
console.log(0 || "Hello"); // Imprime "Hello"
console.log(45 || "Hello"); // Imprime 45
console.log(undefined && 145); // Imprime undefined
console.log(null || 145); // Imprime 145
console.log("" || "Default"); // Imprime "Default"
```

11. Operadors aritmètics.

- 11.4 Booleans.
  - Usem la doble negació !! per a transformar qualsevol valor a booleà. La primera negació força el càsting a boolean i nega el valor. La segona negació, torna a negar el valor deixant-lo en el seu valor equivalent original.

```
console.log(!!null); // Imprime false
console.log(!!undefined); // Imprime false
console.log(!!undefined === false); // Imprime true
console.log(!!""); // Imprime false
console.log(!!0); // Imprime false
console.log(!!"Hello"); // Imprime true
```

  - L'operador boolean || (or) pot ser utilitzat la per a simular valors per defecte en una funció. Per exemple

```
function sayHello(name) {
  // Si nombre es undefined o vacío (""), Se le asignará "Anonymous" por defecto
    let sayName = name || "Anonymous";
    console.log("Hello " + sayName);
}

sayHello("Peter"); // Imprime "Hello Peter"
sayHello(); // Imprime "Hello Anonymous"
```