JavaScript : Arrays

Un array és una variable que emmagatzema varis valors ordenats per posicions. L'avantatge dels arrays és que es poden recórrer per accedir a tots els valors.
Pot emmagatzemar qualsevol tipus de dada, inclús de diferents tipus, dins del mateix array.
Tingues en compte que els arrays es passen per referència, de manera que si modifiques un array dins una funció també quedarà modificat fora.

Sintaxis

```
const idArray = [item1, item2, ...];
```

```
const idArray = new Array(item1, item2, ...);
```

Dos formes diferents de generar un array, si les dades són les mateixes el resultat també ho serà.

Accés a les dades

```
const coses = [1, "Persona", false];
let cosa = coses[0];
```

D'aquesta manera accedim al primer dels tres elements emmagatzemats a l'array.

Modificació de les dades

```
const coses = [1, "Persona", false];
coses[0] = true;
```

D'aquesta manera modifiquem el primer element emmagatzemat a l'array.

Mètodes per consultar arrays

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
```

- **length** : Retorna el nombre d'elements que conté l'array.
  ```
  let length = fruits.length;
  ```
  En aquest cas retorna 4.

- **toString** : Obtenim un string amb els elements de l'array.
  ```
  let fruitsString = fruits.toString();
  ```
  Aquest mètode s'aplica automàticament quan intentem usar un array on es requereix un tipus bàsic.

- **join** : Igual que toString però podem especificar el separador.
  ```
  let fruitsStringSeparator = fruits.join("*");
  ```

Mètodes per manipular arrays

```
const vegetables = ["Carrot", "Pumpkin", "Onion", "Potato"];
```

- **push** : Afegeix un o més elements al final de l'array i en retorna la llargada actual.
```
let vegetablesLength = vegetables.push("Cabbage");
```

- **pop** : Elimina i retorna l'últim element de l'array.
```
let vegetable = vegetables.pop();
```

- **unshift** i **shift** afegeixen i eliminen a l'inici de l'array.

- **sort** : Ordenar alfabèticament un array.
```
vegetables.sort();
```

Iterar arrays

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
```

- **for** :
```
let fLen = fruits.length;
for (let i = 0; i < fLen; i++) {
    console.log(fruits[i]);
}
```

- **forEach** : Cada element de l'array s'envia com a paràmetre d'una funció:
```
fruits.forEach(myFunction);
function myFunction(value) {
    console.log(value);
}
```

- **for** ... **in** : Es recorre l'array complet, en cada iteració es genera una clau per accedir al següent valor.
```
for (let i in fruits) {
    console.log(fruits[i]);
}
```

Mètodes per generar arrays a partir d'altres

```
const myGirls = ["Cecilie", "Lone"];
```

- **concat** : Crea un nou array unint-ne varis.

```
const myBoys = ["Emil", "Tobias", "Linus"];
const myChildren = myGirls.concat(myBoys);
```

- **map** : Crea un nou array aplicant una funció a cada element d'un array.

```
const myPrettyGirls = myGirls.map(myPrettyFunction);
function myPrettyFunction(value) {
    return value + " ♥"
}
```

- **filter** : Crea un nou array amb els elements que compleixen la condició indicada.

```
const over4 = myGirls.filter(myLengthFunction);
function myLengthFunction(value) {
    return value.length > 4;
}
```

En aquest cas retorna un array amb els strings amb més de 4 caràcters.

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

ACTIVITATS

```
const numerosSaltejats = [53, 364, -87, 29, 635, 571];
```

Crea un fitxer JavaScript (*arrays.js*) i dissenya les següents funcions:

1. Crea una funció que rebi un array de números i dos valors numèrics. Recorre cada element de l'array, indica per cada valor si és més gran que el primer valor numèric, si és més petit que el segon valor numèric o si està entremig.

   Per l'array de referència el resultat hauria de ser similar a:

   ```
   53 no es mes gran que 100 i no es negatiu
   364 es mes gran que 100
   -87 es negatiu
   29 no es mes gran que 100 i no es negatiu
   635 es mes gran que 100
   571 es mes gran que 100
   ```

2. Crea una funció que rebi un array de números i en retorni un altre només amb els valors positius.

3. Crea una funció que rebi un array de números i en retorni la suma.

   Per l'array de referència el resultat és: 1565.

4. Crea una funció que rebi un array de números i un valor numèric i retorni un booleà per indicar si el valor apareix a l'array. Tingues en compte que un cop el trobi el bucle ha de finalitzar.

5. Crea una funció que se li passin tres arrays i en retorni un de nou amb els elements comuns dels tres arrays.

   Per l'array de referència i els següents exemples:

   ```
   const numerosSaltejats2 = [736, 29 ,520, 46, 727, 571, 835];
   const numerosSaltejats3 = [78, 205, 4, 53, 571, 358, 454, 11];
   ```

   L'array resultant seria: [571]

6. Crea una funció que se li passi un array i un valor numèric. Començant pel final anirà esborrant els elements fins que el buidi o fins que trobi un element que coincideixi amb el valor passat.

   Per l'array de referència el resultat és: [53, 364, -87, 29]

7. Un array pot ser de qualsevol tipus, inclús un array d'arrays, en aquest cas els anomenem matrius:

```
let matriuNumerica = [
        [72, 111, 108, 97],
        [70, 82, 73, 75, 73],
        [33]
];
```

L'accés a les dades d'una matriu es fa de la següent manera:

```
matriuNumerica[0][1];
```

Amb el primer valor selecciones l'array i amb el segon la posició dins de l'array seleccionat, en aquest exemple obtindries el valor 111.

Crea una funció que generi una nova matriu, a partir de l'original però sense alterar-la, en aquesta nova matriu els valors numèrics serà substituïts per valors ASCII (https://www.ascii-code.com/).

Per convertir valors numèrics en caràcters ASCII pots utilitzar el següent mètode estàtic de l'objecte String:
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String/fromCharCode

Crea un segon fitxer JavaScript (*test.js*) per testejar les funcions creades.
Codifica un joc de proves complet que mostri per consola el resultat de les execucions de cada exercici, de forma similar al de l'exercici 1. Si és necessari crea funcions específiques per mostrar els resultats.
El fitxer amb el testeig de les funcions és el que hauràs d'importar des de l'HTML, afegeix «type="module"» ja que l'importarem com a mòdul per poder cridar funcions JavaScript d'un fitxer extern, les funcions que has creat al fitxer *arrays.js* per resoldre els exercicis.

```
<script type="module" src="test.js"></script>
```

Perquè una constant o funció sigui exportable li hem d'afegir «export» davant:

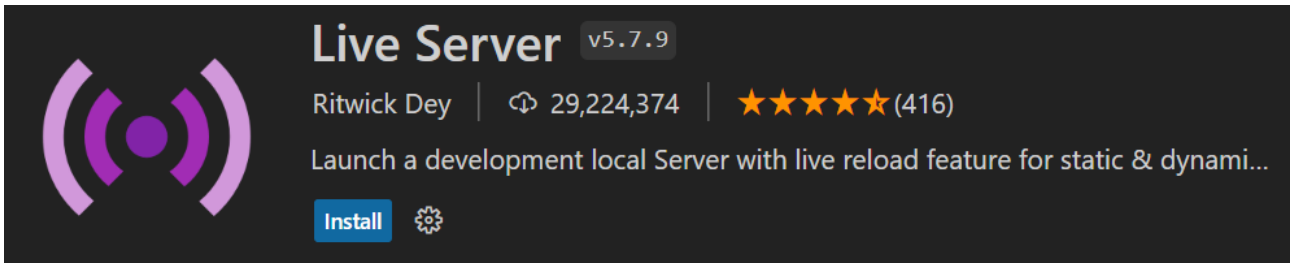
```
export function funcio1 ()
```

Per importar-la des d'un altre fitxer, a les primeres línies hem de fer un «import» del que es vol importar tot indicant-ne el fitxer:

```
import { funcio1, funcio2 } from "./arrays.js";
```

El navegador web no ens permetrà obrir un arxiu JavaScript que no provingui d'un servidor HTTPS, ja que és una vulneració de les polítiques de seguretat de CORS (Cross-Origin Requests).

Haurem d'instal·lar una extensió de Visual Studio Code, un petit servidor per les execucions del nostre codi:



Per que arranqui, al menú inferior cal clickar: 

Llavors s'obrirà el navegador amb la URL localhost.

Recorda que perquè el servidor detecti automàticament el fitxer HTML principal l'has d'anomenar «index.html».

BIBLIOGRAFIA I WEBGRAFIA

«Visual Studio Marketplace». https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
«MDN Web Docs». https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import
«W3Schools». https://www.w3schools.com/js/js_intro.asp