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, això es realitza mitjançant un índex que comença per 0.
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

```
let 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 indicar el separador usat entre els elements de l'array.
  ```
  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);
}
```
També es pot usar amb funcions anònimes:
```
fruits.forEach(function(fruit) {
    console.log(fruit);
});
```

- **for** ... **of** : Recorre l'array complet, s'utilitza per iterar sobre elements iterables com arrays o strings.
```
for (const i of fruits) {
    console.log(i);
}
```

- **for** ... **in** : Recorre l'array complet, en cada iteració es genera una clau per accedir al següent valor indexat.
```
for (const 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. L'operador «+» els converteix en string, no crea un array.

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

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

- **reduce** : Aplica una funció acumulativa a cada element de l'array per reduir-lo a un únic valor.

```
const inicials = myGirls.reduce((acc, nom, index) => {
	if (index === 0) {
		return nom[0];  // Per al primer element, només afegim la inicial
	} else {
		return acc + '+' + nom[0];  // Per a la resta, afegim la inicial amb un '+'
	}
}, "");
```

En aquest cas, retorna un string amb el primer caràcter de cada element separats per «+». Comença amb un valor inicial: "", i concatena la primera lletra de cada nom (nom[0]) a l'acumulador acc.
L'exemple usa una funció anònima de fletxa.

https://github.com/xbaubes/DesenvolupamentWeb/blob/main/Frontend/JavaScript/exampleCode/Array.js

ACTIVITATS

Enllaç amb tots els mètodes de l'objecte Array:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Array de referència:

```
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 i retorni un string. Ha de recórrer cada element de l'array, comparar cada valor de l'array amb els valors passats per decidir si és més gran que el primer valor passat, si és més petit que el segon valor passat o si està entremig.
   Valida primer que el primer valor passat és major que el segon, retorna un string buit si no és així.

   Per l'array de referència i passant 100 i 0, l'string resultant hauria de ser similar a:

   ```
   53 esta entre 100 i 0
   364 es mes gran que 100
   -87 es mes petit que 0
   29 esta entre 100 i 0
   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];
   ```

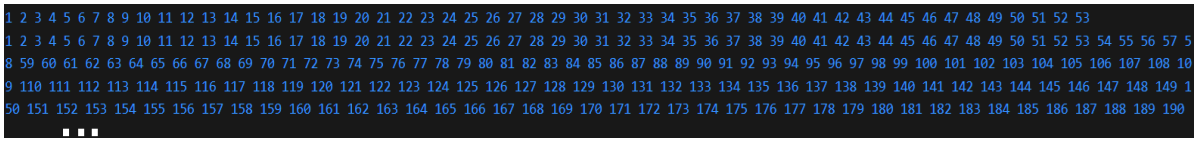
   L'array resultant seria: [571]

6. Crea una funció que rebi 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 i passant el valor 29 el resultat és: [53, 364, -87, 29]

7. Crea una funció que rebi un array i retorni un string on cada número de l'array es representi mitjançant una seqüència numèrica ascendent, que comença a 1 i continua indefinidament. Si el número és negatiu s'usarà el valor absolut.

   Per l'array de referència, l'string resultant hauria de ser similar a:



8. Crea una funció que rebi dos arrays i retorni un booleà per indicar si contenen els mateixos elements o no. Has de comprovar que tinguin la mateixa longitud i que cada valor sigui igual. Tingues en compte que un cop trobis una diferència el bucle ha de finalitzar. En JavaScript, no existeix una funció nativa específica per comparar arrays directament, això és perquè els arrays són objectes, i quan compares objectes, el comparador només comprova si les referències a aquests objectes són les mateixes, no si el contingut és idèntic.

9. Crea una funció que rebi un array i un valor numèric i retorni un array que contingui l'array rebut però amb un «null» on hi havia el valor numèric rebut, seguidament el valor numèric rebut i el lloc que ocupava a l'array. Si no el troba, com a posició guardarà «null». Pots utilitzar *indexOf() per trobar la posició*.
   No modifiquis l'array rebut, un array és un objecte, quan el passes a una funció passes una referència no una còpia, si modifiques l'array dins de la funció, els canvis es reflectiran fora de la funció. No pots copiar un array utilitzant «=», assignar un array a una nova variable crea una referència al mateix array, no una còpia. Pots utilitzar *slice()* per copiar l'array.

   Per l'array de referència i passant 29 el resultat és: [53, 364, -87, null, 635, 571, 29, 3]

10. Crea una funció que rebi un array i retorni un array que contingui dos arrays. El primer ha de ser l'array original i el segon l'array ordenat.

11. 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 un dels arrays de l'array principal i amb el segon la posició dins de l'array seleccionat, en aquest exemple obtindries el valor 111.

Per recórrer una matriu necessites usar un bucle dins un altre, amb el bucle exterior recorres l'array principal i amb el bucle interior els arrays secundaris.

Crea una funció que generi i retorni 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

[Test Frontend]

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. Si és necessari crea funcions específiques per mostrar els resultats.
Idealment, si la funció retorna un resultat, comparar aquest resultat amb el resultat esperat que prèviament hauràs de crear.
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:

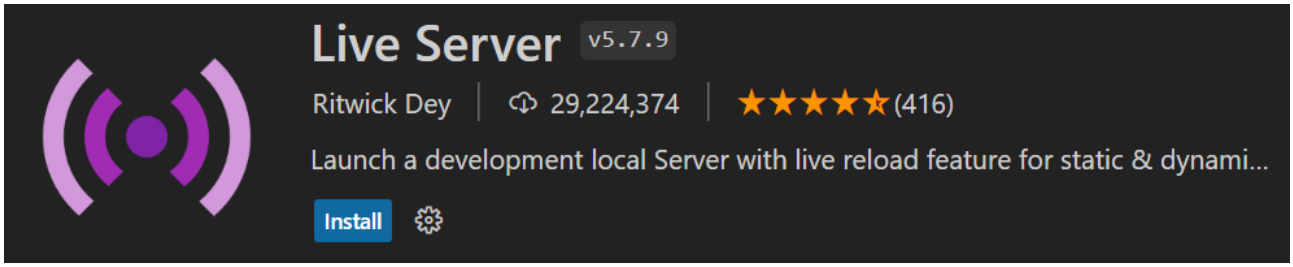
```
export function funcio1 ()
```

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, al fitxer test.js hi afegiríem:

```
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: 

S'aplica a tota la carpeta, de manera que el projecte ha d'estar obert amb: «File/Open Folder».

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».

[Test Backend]

Crea una funció (*testComplet*) per testejar les funcions creades.

Codifica un joc de proves complet que mostri per consola, i per cada exercici, tant el resultat de les execucions com el resultat esperat.
Compara el resultat de la funció amb el resultat esperat que prèviament hauràs de crear.

El resultat de cada funció i el resultat esperat seran enviats a una funció amb la següent capçalera:

```
function testIndividual(esperat, resultat)
```

Dins d'aquesta funció hem de comparar els dos elements passats, tant si són strings, arrays o matrius; si és un array o matriu utilitzarem la funció creada per comparar arrays. Per saber si l'element és un array podem usar el mètode estàtic *Array.isArray()*.
Un cop finalitzat l'anàlisi llançarem un missatge informant del resultat, si són iguals usarem *console.log*, si no ho són: *console.error*.

Registre, i mostra al final, el número de testos realitzats i el número de testos superats.

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