DATA STRUCTURES

- Which **data type** is used specifically to **store sequences of values**? Show an example in the following programming languages:

  The arrays

  - Javascript: let colors = ["Blue", "Orange", "Green", "Red", "Black"];
  - Java: int array[]=new int[5]
  - C#: int[] intArray = new int[] { 3, 4, 5 };

- Which are the main **properties** of **arrays**? Search for the properties that arrays have in the following programming languages:
    - Javascript

constructor: Returns the function that created the Array object's prototype

length: Sets or returns the number of elements in an array

prototype: Allows you to add properties and methods to an Array object

- Which are the main **properties** of **arrays**? Search for the properties that arrays have in the following programming languages:
    - Java: Array.length

- Which are the main **properties** of **arrays**? Search for the properties that arrays have in the following programming languages:

    - C#

IsFixedSize: Obtiene un valor que indica si la interfaz Array tiene un tamaño fijo.

IsReadOnly: Obtiene un valor que indica si Array es de solo lectura.

IsSynchronized: Obtiene un valor que indica si el acceso a la interfaz Array está sincronizado (es seguro para subprocesos).

Length: Obtiene el número total de elementos de todas las dimensiones de Array.

LongLength: Obtiene un entero de 64 bits que representa el número total de elementos de todas las dimensiones de Array.

MaxLength: Obtiene el número máximo de elementos que pueden estar contenidos en una matriz.

Rank: Obtiene el rango (número de dimensiones) de Array. Por ejemplo, una matriz unidimensional devuelve 1, una matriz bidimensional devuelve 2, y así sucesivamente.

SyncRoot: Obtiene un objeto que se puede usar para sincronizar el acceso a Array.

- What is an **object**? Show a real life example to better understand this concept

An object is an unordered collection of properties, each of which has a name and a value.

```
const myObject = new Object();

const anotherObject = {};
```

- What **data types** can you assign as values of **objects** in Javascript? Show an example of an object with the different data types assigned to it

A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects.

```
const myObject  = {

name: “Name”,

year: 34,

hobbies: [“football”, “music”, “e-sports”]

}
```

- What are **methods**?

block of code that has a set of instructions defined within it.

- How can you create an object in the following languages?

Javascript

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

Java

```
class Student{
        int id;
        String name;
 public static void main(String args[]){
         Student s1=new Student();
```

C#

```
public class Student{
        int id;
        String name;
public static void Main(string[] args){
        Student s1 = new Student();
        s1.id = 101;
        s1.name = "Sonoo Jaiswal";
```

- What is the **delete operator** of objects in Javascript? Show an example

The delete operator deletes a property from an object.

```
let person = {
  firstName:"John",
  age:50,
};

delete person.age;

// let person = {
// firstName:"John",
//  };
```

- What is the **in operator** of objects in Javascript? Show a practical example of using the **in operator**

This operator returns true if the property is in the specified object.

```
let object{
    property1: “nice”,
}
console.log(“wow”, property1 in object); //true
console.log(“wow”, property2 in object); //false
```

- What is the difference between the **in operator** and the **hasOwnProperty method** in Javascript?

The key difference is that **in** will return true for inherited properties, whereas **hasownProperty**() will return false for inherited properties.

- What **built-in array methods** exist in the following programming languages?
    - Javascript

concat() Joins two or more arrays, and returns a copy of the joined arrays

filter() Creates a new array with every element in an array that pass a test

find() Returns the value of the first element in an array that pass a test

forEach() Calls a function for each array element

join() Joins all elements of an array into a string

map() Creates a new array with the result of calling a function for each array element

pop() Removes the last element of an array, and returns that element

push() Adds new elements to the end of an array, and returns the new length

shift() Removes the first element of an array, and returns that element

slice() Selects a part of an array, and returns the new array

sort() Sorts the elements of an array

toString() Converts an array to a string, and returns the result

unshift() Adds new elements to the beginning of an array, and returns the new length

- What **built-in array methods** exist in the following programming languages?
    - Java

asList() Returns a fixed-size list backed by the specified Arrays

binarySearch() Searches for the specified element in the array with the help of the Binary Search Algorithm

compare(array 1, array 2) Compares two arrays passed as parameters lexicographically

equals(array1, array2) Checks if both the arrays are equal or not.

fill(originalArray, fillValue) Assigns this fill value to each index of this arrays.

hashCode(originalArray) Returns an integer hashCode of this array instance.

mismatch(array1, array2) Finds and returns the index of the first unmatched element between the two specified arrays.

parallelSort(originalArray) Sorts the specified array using parallel sort.

sort(originalArray)Sorts the complete array in ascending order.

spliterator(originalArray) Returns a Spliterator covering all of the specified Arrays.

stream(originalArray) Returns a sequential stream with the specified array as its source.

toString(originalArray) It returns a string representation of the contents of this array.

- What **built-in array methods** exist in the following programming languages?
  - C#

AsReadOnly<T>(T[]) It returns a read-only wrapper for the specified array.

BinarySearch(Array,Int32,Int32,Object) It is used to search a range of elements in a one-dimensional sorted array for a value.

Clear(Array,Int32,Int32) It is used to set a range of elements in an array to the default value.

Clone() It is used to create a shallow copy of the Array.

Copy(Array,Array,Int32) It is used to copy elements of an array into another array by specifying starting index.

CreateInstance(Type,Int32) It is used to create a one-dimensional Array of the specified Type and length.

Empty<T>() It is used to return an empty array.

Finalize() It is used to free resources and perform cleanup operations.

Find<T>(T[],Predicate<T>) It is used to search for an element that matches the conditions defined by the specified predicate.

IndexOf(Array,Object) It is used to search for the specified object and returns the index of its first occurrence in a one-dimensional array.

Initialize() It is used to initialize every element of the value-type Array by calling the default constructor of the value type.

Reverse(Array) It is used to reverse the sequence of the elements in the entire one-dimensional Array.

Sort(Array) It is used to sort the elements in an entire one-dimensional Array

ToString() It is used to return a string that represents the current object.

- Find the **built-in methods** that **objects** have in Javascript and show an example of each one of them

```
const object1 = {
  a: 'somestring',
  b: 42,
  c: false
};
```

- Object.keys(obj): console.log(Object.keys(object1)); expected output: Array ["a", "b", "c"]
- Object.values(obj): console.log(Object.values(object1)); expected output: Array ["somestring", 42, false]
- Object.entries(obj): for (const [key, value] of Object.entries(object1)) {console.log(`${key}: ${value}`);} expected output:"a: somestring" "b: 42" “c: false”

```
const entries = new Map([
['foo', 'bar'],
['baz', 42]
]);
const obj = Object.fromEntries(entries);
```

- Object.fromEntries(arr) console.log(obj); expected output: Object { foo: "bar", baz: 42 }

- What is the difference between an **array** and an **object** in Javascript?
    - Explain when it is recommended to use each one of them.

They have very different methods of iterating their values.

Array can be called a more standard organization of data and values.

Objects are more focused in defining a set of data and this data having properties that can be called by their name.

Arrays have their data ordered contrary to Objects.

- Explain the difference between **mutability** and **immutability**

The opposite of immutability is mutability, i.e., the ability to change the value or state of elements

- Find an example of a **mutable** and an **immutable** data type in Javascript and show a practical example

Mutable: Objects and arrays

```
var arr = [1,2,3,4]
arr.push(5); //MUTATES & ADDS VALUE
console.log(arr) // [1,2,3,4,5]
```

Inmutable: strings and numbers.

```
var statement = "I am a string";
var otherStr = statement.slice(0, 4);
console.log(statement);    // I am a string
console.log(otherStr);      // I am
```

- **Write a function that receives the person object as a parameter and that changes all the values of the object to upper case and adds a new property of the object named age with the value of 24 (age: 24). The function should return a new object with the values changed and the new property without mutating the original object.**

- The initial object:

```
const person = {     firstName: 'Ana',     secondName: 'Marks',     role: 'admin' }
const newPerson = (object) => {
    let newPerson = object;
    newPerson.age = '24';
    for (const value in newPerson) {
            let element = newPerson[value];
            if (typeof (element) === 'string') {
                newPerson[value] = element.toUpperCase();
            }     }
    return console.log(newPerson);
}
newPerson(person);
```

1. **Write a function that returns an array with all the keys of the following object:**

- const product= { name: “Sony WH-1000XM3”, price: ”300.00”, itemsInStock: 23 };

```
const returnArray = (object) =>{
  console.log(Object.keys(object));
}

returnArray(product);
```

1. **Write a function that returns an array with all the keys and values of the following object:**

- product: { name: “Sony WH-1000XM3”, price: ”300.00”, itemsInStock: 23 };

```
const returnArray = (object) =>{
  console.log(Object.keys(object));
  console.log(Object.values(object));
}
returnArray(product);
```