JS

{ JavaScript }

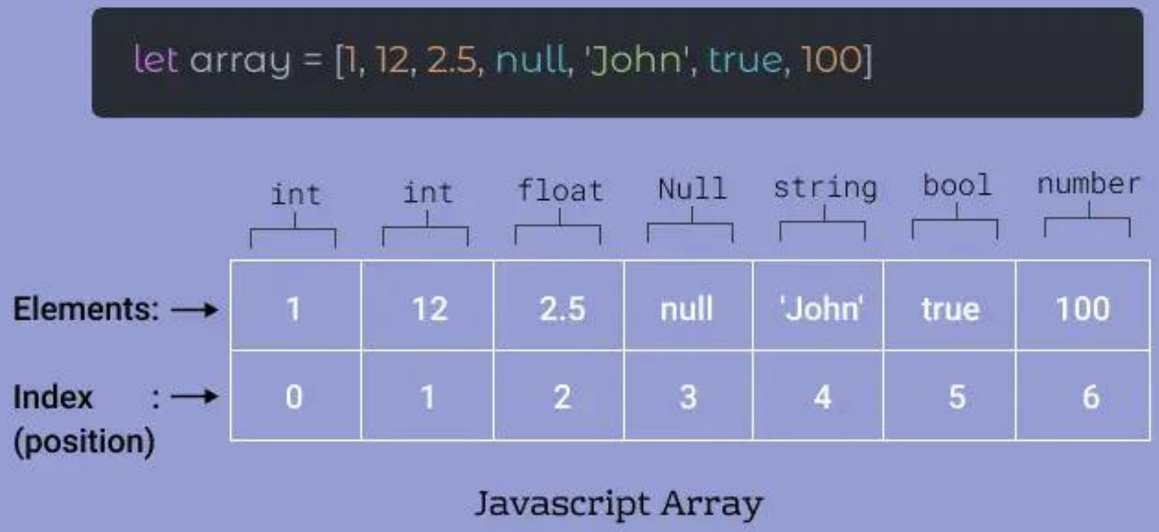
CRAFT KNOWLEDGE

Objectives

JavaScript Array

- WHAT IS JS ARRAY
- CREATEING JS ARRAY
- ACCESSING ARRAY ELEMENTS
- ITERATE AN ARRAY
- ARRAY METHODS MULTI-
- DIMENSIONAL ARRAYS

What is Javascript Array



Javascript Array

- In JavaScript, an array is an ordered list of values. Each value is called an element specified by an index:
- variable can hold only one value. We cannot assign multiple values to a single variable. JavaScript array is a special type of variable, which can store multiple values using a special syntax.
- JavaScript array can store a mixed data formats in a single array.
- JavaScript arrays are dynamic, which means that they grow or shrink as needed.

Creating JavaScript arrays

- JavaScript provides you with two ways to create an array.

1. literal way
2. Array () constructor

- **Literal way:**
  let array_name = [item1, item2, ...];
- The array literal form uses the square brackets [] to wrap a comma-separated list of elements.
- You can also create an array, and then provide the elements:

```
let array_name = [];
cars[0]= "item1";
cars[1]= "item2";
cars[2]= "item3";
```

- It is not required to store the same type of values in an array. It can store values of different types as well.

```
let data = [1, "Steve", "DC", true, 255000, 5.5];
```

- **Array () constructor:**
    - let numArray= new Array(); // The numArray array is empty, which does hold any elements.
    - let numArray= Array(10); // we can create an array with an initial size
    - let numArray= new Array(1,2,3,4,5);//array and initialize it with some elements,

- if you use the Array() constructor to create an array and pass **a number (one number)** into it, you are creating an array with an initial size.
- However, when you pass a value of another type like **string** into the Array() constructor, you create an array with an element of that value.

- JavaScript allows you to omit the new operator when you use the Array() constructor.

- Note: There is no need to use new Array(). For simplicity, readability and execution speed, use the array literal method.

```
let numArr = [10, 20, 30, 40, 50];

console.log(numArr[0]); // 10
console.log(numArr[1]); // 20
console.log(numArr[2]); // 30
console.log(numArr[3]); // 40
console.log(numArr[4]); // 50

console.log(numArr.at(0)); // 10
console.log(numArr.at(1)); // 20
console.log(numArr.at(2)); // 30
console.log(numArr.at(3)); // 40
console.log(numArr.at(4)); // 50

console.log(numArr.at(-1)); // 10
console.log(numArr.at(-2)); // 20
console.log(numArr.at(-2)); // 30
console.log(numArr.at(-3)); // 40
console.log(numArr.at(-4)); // 50
```

Accessing Array Elements

- Array elements (values) can be accessed using an **index**.
- Specify an index in square brackets with the array name to access the element at a particular index.
- like arrayName[index].

- we can use the **arrayName.at(pos)** method to get the element from the specified index.
- This is the same as arr[index] except that the **.at()** returns an element from the last element if the specified index is negative.

- How get the array size?
  - **length property** of an array returns the number of elements.

Iterate an array

```
let numArray = [10, 20, 30, 40, 50];

numArray.forEach(i =>  console.log(i));

for(let i=0; i<numArray.length; i++)
  console.log(numArray[i]);

for(let i of numArray)
  console.log(i);

for(let i in numArray)
  console.log(numArray[i]);
```

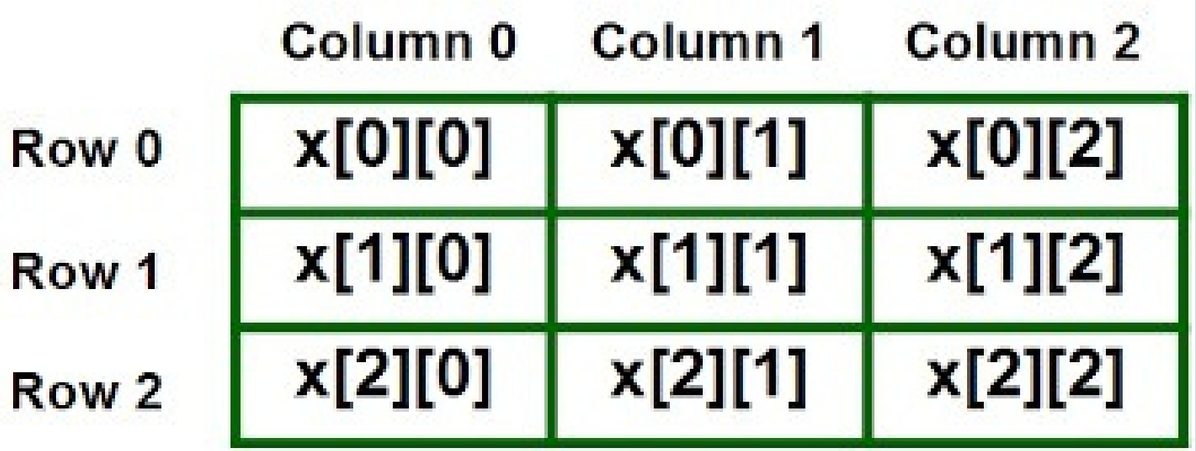
- we can iterate an array using
  Array.forEach(),
  for loop,
  for-of, and
  for-in loop,

Array Methods

The following explains some basic operations on arrays using array method.

1) Adding an element to the end of an array using array **push()** method.
2) Adding an element to the beginning of an array using array **unshift()** method.
3) Removing an element from the end of an array using array **pop()** method.
4) Removing an element from the beginning of an array using array **shift()** method.
5) Finding an index of an element in the array using array **indexOf()** method.
6) Check if a value is in array using array **includes()** method.
7) Merging (Concatenating) Arrays using array **concat()** method.

Multi-dimensional arrays



- JavaScript does not provide the multidimensional array natively.
- However, you can create a multidimensional array by defining an array of elements,
- For this reason, we can say that a JavaScript multidimensional array is an array of arrays.
- defines a two-dimensional array named activities:

```
let number= [
['one', 1],
 ['two', 2],
 ['three', 3],
 ['foure', 4],
['five', 5]
 ];
```

(index)	0	1
0	'one'	1
1	'two'	2
2	'three'	3
3	'foure'	4
4	'five'	5

- In the number array, the first dimension represents the number in word and the second one shows the corresponding number.
- To show the number array in the console, you use the **console.table()** method as follows:

  console.table(activities);

- To access an element of the multidimensional array, you first use square brackets to access an element of the outer array that returns an inner array; and then use another square bracket to access the element of the inner array.
- console.log(activities[0][1]);