Array introduction

An ***array*** is an ordered collection of values called ***elements***. Each array element is stored in a numeric location called an ***index***. An array is initialized by assigning an array variable with brackets [] containing comma-separated values.

Array elements may be of the same type or different types. Arrays increase in size as elements are added and decrease as elements are removed.

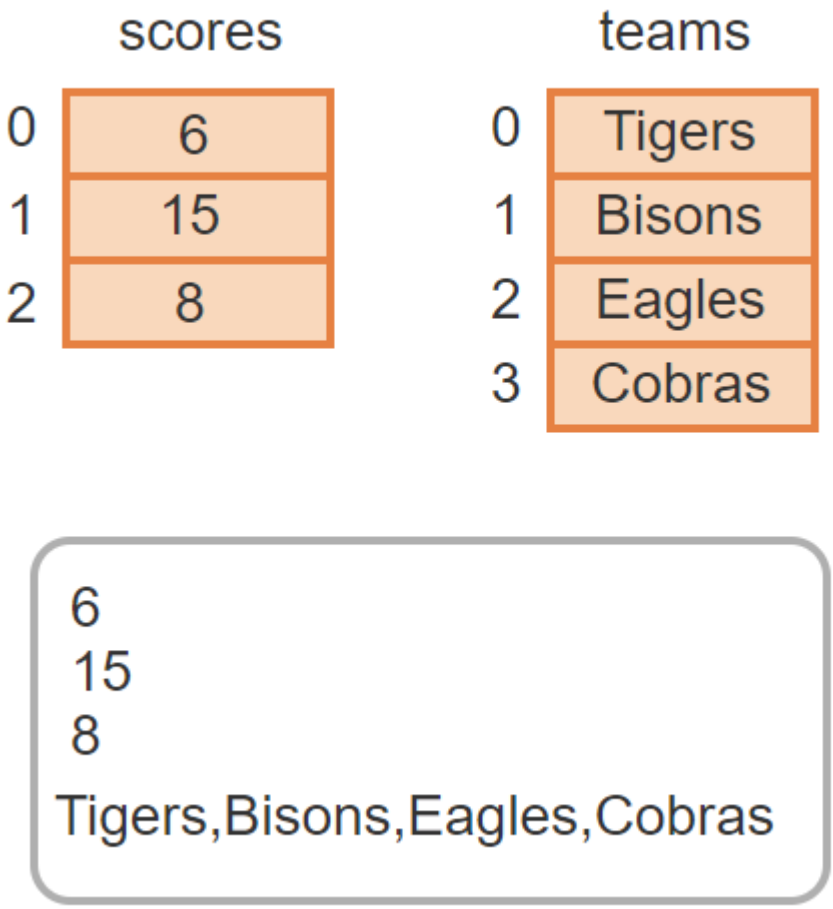
Initializing and displaying array elements.

```
let scores = [];

scores[0] = 6;
scores[1] = 15;
scores[2] = 8;

console.log(scores[0]);
console.log(scores[1]);
console.log(scores[2]);

let teams = ["Tigers", "Bisons",
             "Eagles", "Cobras"];

console.log(teams);
```

Explanation :

1. An empty array called "scores" is declared with [ ].
2. Three elements are added to the scores array at indexes 0, 1, and 2.
3. The three elements are output to the console.
4. The teams array is initialized with four elements.
5. All four elements are output to the console separated by commas

Output visualization :



Adding and removing array elements

An array is an ***Array object*** that defines numerous methods for manipulating arrays. A ***method*** is a function that is attached to an object and operates on data stored in the object. Methods are called by prefacing the method with the object. Ex: myArray.method();.

Table : Array methods for adding and removing array elements.

Method	Description	Example
push(value)	Adds a value to the end of the array	`let nums = [2, 4, 6];` `nums.push(8);  // nums = [2, 4, 6, 8]`

Method	Description	Example
pop()	Removes the last array element and returns the element	`let nums = [2, 4, 6];` `let x = nums.pop(); // returns 6, nums = [2, 4]`
unshift(value)	Adds a value to the beginning of the array	`let nums = [2, 4, 6];` `nums.unshift(0); // nums = [0, 2, 4, 6]`
shift()	Removes the first array element and returns the element	`let nums = [2, 4, 6];` `let x = nums.shift(); // returns 2, nums = [4, 6]`

splice(startingIndex, numElemToDelete, valuesToAdd)	Adds or removes elements from anywhere in the array and returns the deleted elements (if any)	`let nums = [2, 4, 6, 8, 10];`
		
		`// Deletes all elements from index 3 to the end`
		`nums.splice(3); // nums = [2, 4, 6]`
		
		`// Deletes 2 elements starting at index 0`
		`nums.splice(0, 2); // nums = [6]`
		
		`// Adds 3, 5 starting at index 0`
		`nums.splice(0, 0, 3, 5); // nums = [3, 5, 6]`
		
		`// Adds 7, 9, 11 starting at index 2`
		`nums.splice(2, 0, 7, 9, 11); // nums = [3, 5, 7, 9, 11, 6]`

Destructuring Assignment :

Destructuring Assignment is a JavaScript expression that allows to **unpack values** from arrays, or properties from objects, into distinct variables data can be extracted from *arrays, objects,*

nested objects and **assigning to variables**. In Destructuring Assignment on the left-hand side defined that which value should be unpacked from the sourced variable. In general way implementation of the *extraction* of the array is as shown below:

```
<script>
   var names = ["alpha", "beta", "gamma", "delta"];

   var firstName = names[0];
   var secondName = names[1];

   console.log(firstName);//"alpha"
   console.log(secondName);//"beta"
</script>
```

Output:
alpha

beta

Syntax:

Array destructuring:

var x, y;

[x, y] = [10, 20];

console.log(x); // 10

console.log(y); // 20

or

[x, y, ...restof] = [10, 20, 30, 40, 50];

console.log(x); // 10

console.log(y); // 20

console.log(restof); // [30, 40, 50]

Example 1: When using **destructuring assignment** the same extraction can be done using below implementations.

```
<script>
   var names = ["alpha", "beta", "gamma", "delta"];
   var [firstName, secondName] = names;
```

```
    console.log(firstName);//"alpha"
    console.log(secondName);//"beta"

    //Both of the procedure are same
    var [firstName, secondName] = ["alpha", "beta", "gamma", "delta"];

    console.log(firstName);//"alpha"
    console.log(secondName);//"beta
</script>
```

Output:

alpha

beta

alpha

beta

Example 2: The array *elements* can be *skipped* as well using a *comma* separator. A single comma can be used to skip a single array element. One key difference between the **spread operator** and array destructuring is that the spread operator unpacks all array elements into a comma-separated list which does not allow us to pick or choose which elements we want to assign to variables. To skip the whole array it can be done using the number of commas as there is a number of array elements.

```
<script>
var [firstName,,thirdName] = ["alpha", "beta", "gamma", "delta"];

console.log(firstName);//"alpha"
console.log(thirdName);//"gamma"
</script>
```

Output:

alpha

gamma

Example 3: In order to assign some array elements to variable and rest of the array elements to only a single variable can be achieved by using **rest operator (…)** as in below implementation. But one limitation of rest operator is that it works correctly only with the last elements implying a subarray cannot be obtained leaving the last element in the array.

```
<script>
var [firstName,,...lastName] = ["alpha", "beta", "gamma", "delta"];
```

```
console.log(firstName);//"alpha"
console.log(lastName);//"gamma, delta"
</script>
```

Output:

alpha

gamma

Example 4: Values can also be **swapped** using destructuring assignment as below:

```
<script>
    var [firstName, lastName] = ["alpha", "beta", "gamma", "delta"];

    console.log(firstName);//"alpha"
    console.log(secondName);//"beta"

    //After swapping
    [firstName, secondName] = [secondName, firstName]

    console.log(firstName);//"beta"
    console.log(secondName);//"alpha"
</script>
```

Output:

alpha

beta

beta

alpha