Lecture 3.1

JavaScript Arrays

Topics

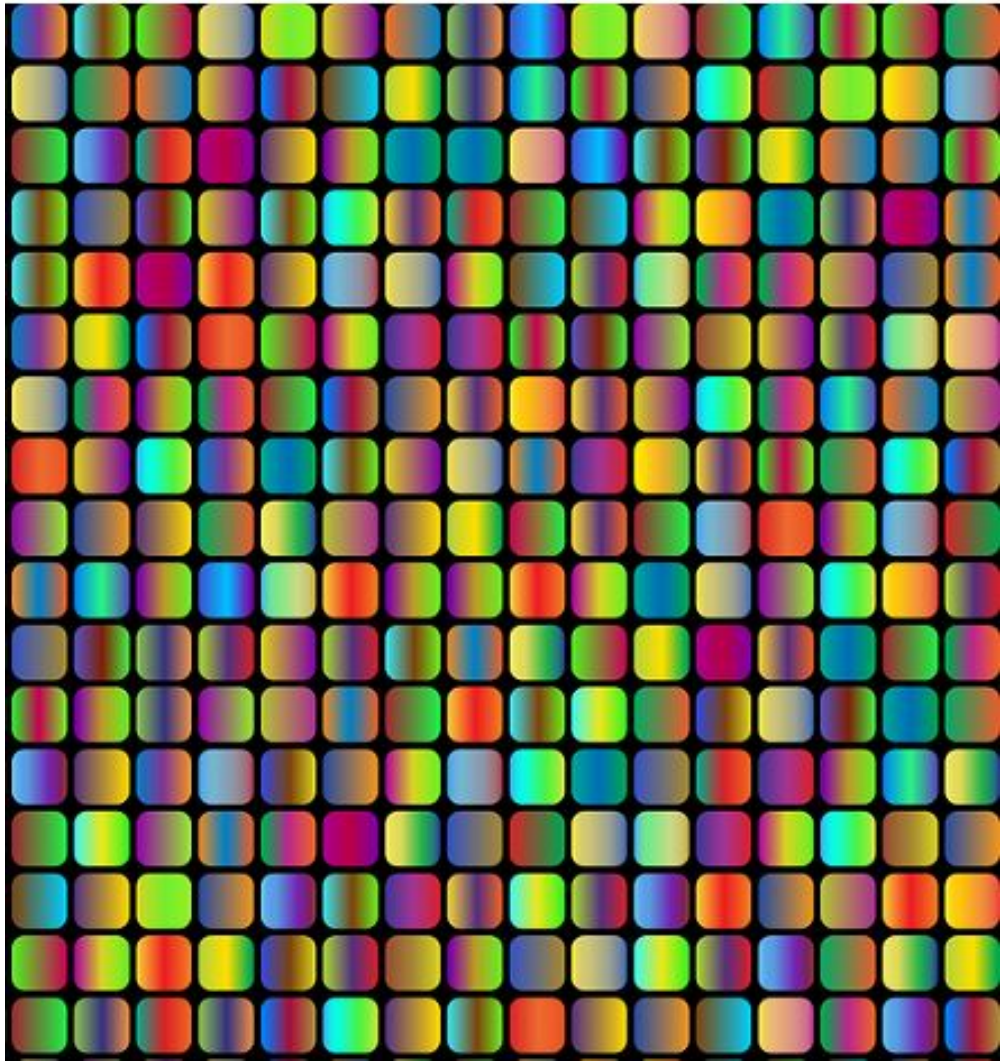
- Intro to Arrays
  - Memory
- JavaScript Arrays
- Creating Arrays
- Array Indices
- Array Size
- Array Basic Operations

Intro to Arrays

Data Structure 101

To understand how they work, it's very helpful to visualize your computer's memory as a grid, just like the one below.

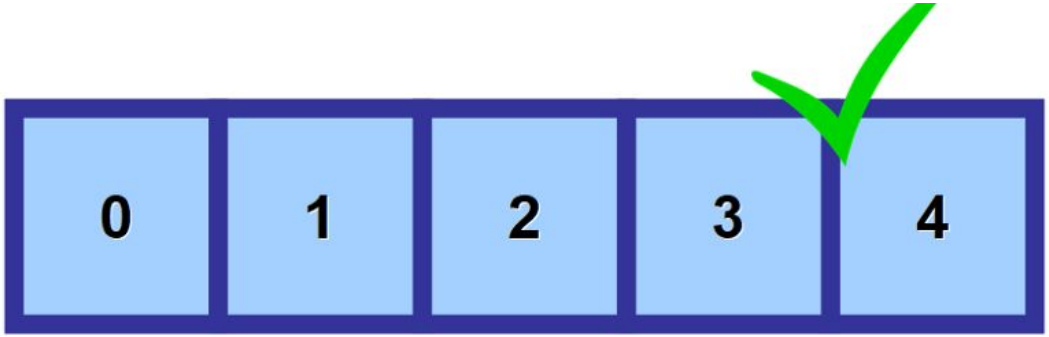
Each piece of information is stored in one of those small elements (squares) that make the grid.



Arrays in Memory

Arrays take advantage of this “grid” structure to **store lists of related information in adjacent memory locations** to guarantee extreme efficiency for finding those values.

Their elements are next to each other in memory. If you need to access more than one of them, the process is extremely optimized because your computer already knows where the value is located.



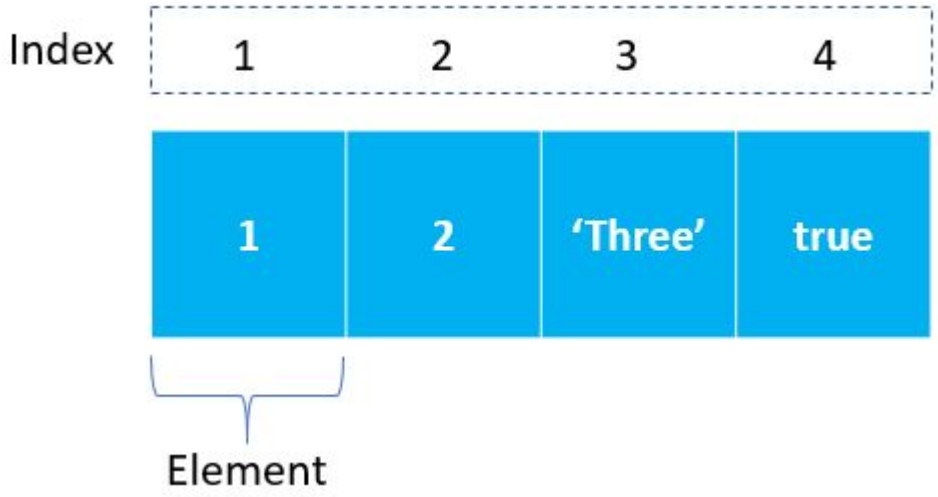
JavaScript Arrays

JavaScript Arrays

In JavaScript, an array is an ordered list of values.

Each value is called an element specified by an index.

An **array** is a data structure, which can store a fixed-size collection of elements



JavaScript Arrays cont..

An JavaScript array has the following characteristics:

1. First, an array can hold values of different types. For example, you can have an array that stores the number and string, and boolean values.

2. Second, the length of an array is dynamically sized and auto-growing. In other words, you don’t need to specify the array size upfront.

Creating Arrays

Creating Arrays

When you create an array, you:

- Assign it to a variable.
- Define the type of elements that it will store.
- Define its size (the maximum number of elements).

Creating JavaScript Arrays

JavaScript provides you with two ways to create an array.

The first one is to use the Array constructor as follows: *(The scores array is empty ie. no elements)*

```
let scores = new Array();
```

If you know the number of elements that the array will hold, you can create an array with an initial size as shown in the following example:

```
let scores = Array(10);
```

Creating JavaScript Arrays cont..

To create an array with some elements, you pass the elements as a comma-separated list into the Array() constructor.

For example, the following creates the scores array that has five elements (or numbers):

```
let scores = new Array(9,10,8,7,6);
```

Initial size of Arrays

It’s important to notice that if you use the array constructor to create an array and pass into a number, 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. For example:

```
let athletes = new Array(3); // creates an array with initial size 3
let scores = new Array(1, 2, 3); // create an array with three numbers 1,2 3
let signs = new Array('Red'); // creates an array with one element 'Red'
```

Optional Array() constructor

JavaScript allows you to omit the `new` operator when you use the array constructor. For example, the following statement creates the `artists` array.

```
let artists = Array();
```

In practice, you'll rarely use the `Array()` constructor to create an array.

The more preferred way to create an array is to use the array literal notation:

```
let arrayName = [element1, element2, element3, ...];
```

New Array Examples

The array literal form uses the square brackets [] to wrap a comma-separated list of elements.

The following example creates the colors array that hold three strings:

```
let colors = ['red', 'green', 'blue'];
```

To create an empty array, you use square brackets without specifying any element like this:

```
let emptyArray = [];
```

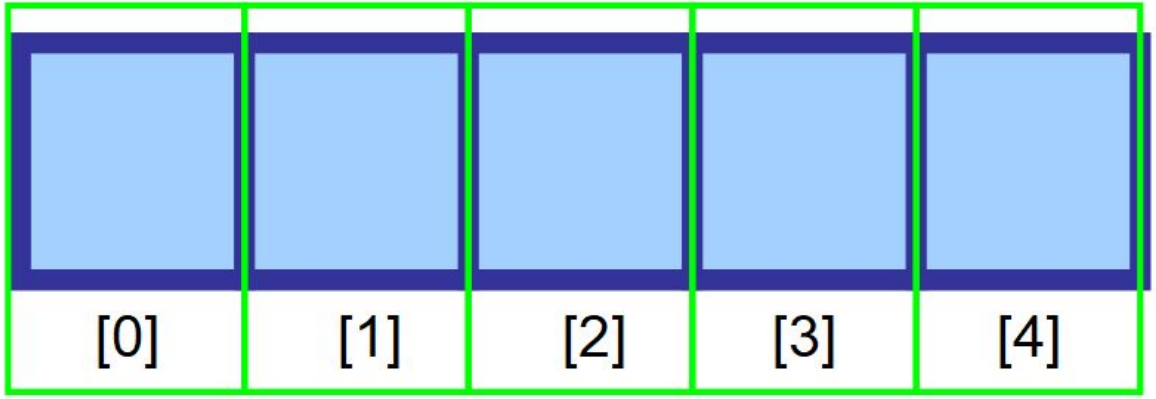
Video - Array Basics

Array Indices

Array Indices

You use what it's called an "index" ("indices" in plural) to access a value in an array. This is a number that refers to the location where the value is stored.

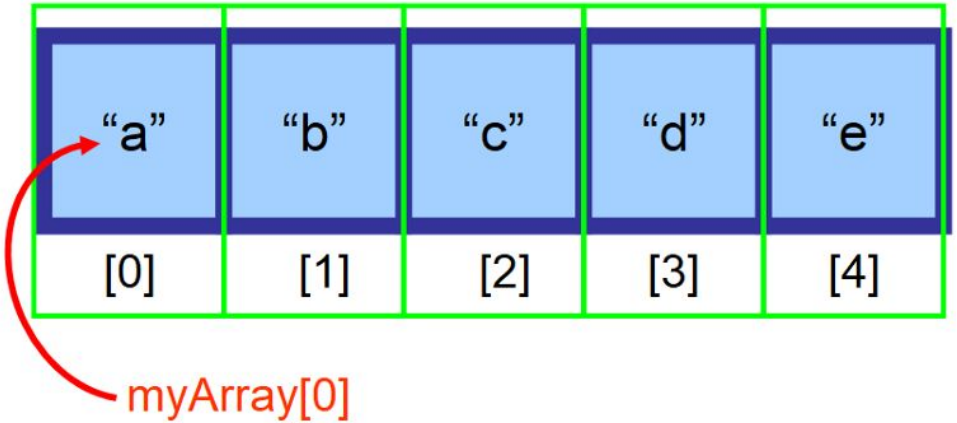
As you can see in the diagram below, the first element in the array is referred to using index 0. As you move further to the right, the index increases by one for each space in memory.



Zero Based Numbering

Note: I know that it seems strange at first to start counting from 0 instead of 1, but this is called Zero-Based Numbering. It's very common in computer science.

For Example: If your array is stored in the variable **myArray** and you want to access the first element (at index 0), you would use **myArray[0]**



Accessing JavaScript Array Elements

JavaScript arrays are zero-based indexed. In other words, the first element of an array starts at index 0, the second element starts at index 1, and so on.

To access an element in an array, you specify an index in the square brackets []:

Accessing JavaScript Array Elements cont..

The following shows how to access the elements of the mountains array:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];

console.log(mountains[0]); // 'Everest'
console.log(mountains[1]); // 'Fuji'
console.log(mountains[2]); // 'Nanga Parbat'
```

Changing value of Array Elements

To change the value of an element, you assign that value to the element like this:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
mountains[2] = 'K2';

console.log(mountains);
```

Output:

```
[ 'Everest', 'Fuji', 'K2' ]
```

Video - Array

Array Size

Getting the Array Size

Typically, the length property of an array returns the number of elements.

The following example shows how to use the length property:

```
let mountains = ['Everest', 'Fuji', 'Nanga Parbat'];
console.log(mountains.length); // 3
```

Array Basic Operations

1. Adding an element to the end of an Array

To add an element to the end of an array, you use the push() method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
seas.push('Red Sea');

console.log(seas);
```

```
[ 'Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea', 'Red Sea' ]
```

2. Adding an element to the beginning of an Array

To add an element to the beginning of an array, you use the unshift() method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
seas.unshift('Red Sea');

console.log(seas);
```

```
[ 'Red Sea', 'Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea' ]
```

3. Removing an element to the end of an Array

To remove an element from the end of an array, you use the pop() method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
const lastElement = seas.pop();
console.log(lastElement);
```

```
Baltic Sea
```

4. Removing an element to the beginning of an Array

To remove an element from the beginning of an array, you use the shift() method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
const firstElement = seas.shift();

console.log(firstElement);
```

```
Black Sea
```

5. Finding an index of an element in Array

To find the index of an element, you use the indexOf() method:

```
let seas = ['Black Sea', 'Caribbean Sea', 'North Sea', 'Baltic Sea'];
let index = seas.indexOf('North Sea');

console.log(index); // 2
```

6. Check if a value is an Array

To check if a value is an array, you use Array.isArray() method:

```
console.log(Array.isArray(seas)); // true
```