JavaScript For Web

Week 2, Lecture 4 - JavaScript Fundamentals: Arrays

Instructor: Jason Xu

Today's Overview

- Review: Functions, Objects and String
- What are arrays useful for?
- Access/Change an array element
- Some useful array properties/methods
- **Tutorial**: Readings and Reviews

Review: Functions

Functions are reusable blocks of code.

```
let sum = function(a, b) {
  return a + b
}

console.log(sum(1, 2)) // 3
console.log(sum(2, 2)) // 4
console.log(sum(100, 2)) // 102
```

This arrow function is a shorter form of function expression

```
let sum = (a, b) => a + b
```

and it's the same as:

```
let sum = (a, b) => {
  return a + b
}
```

What are Objects

Think about in a real life, a car is an object. A car has **properties** like weight and color, and **methods** like start and stop:

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

- All cars have the same **properties**, but the property **values** differ from car to car.
- All cars have the same **methods**, but the methods are performed **at different times**.

What are Objects continued

In JavaScript, objects are variables too. But objects can contain many **properties(values)** and **methods(functions)**. For example:

```
const myCar = {
  type: "Fiat",
  model: 500,
  color: "white",
  ...
  start: function() {...},
  drive: function() {...}
  ...
}
```

- We can access the properties or run the methods using `variable name` with a dot notation `.`:

```
console.log(myCar.type)
myCar.start()
```

- You don't have to know too much details about the object at this moment. You just need to know what are **properties** and **methods**, and how we can access them using dot notation `.`.
- Just like this car object, **arrays** also have **properties** and **methods**.

What are arrays

The Array, as with arrays in other programming languages, enables storing **a collection of multiple items under a single variable name**.

- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
let car1 = "Saab"
let car2 = "Volvo"
let car3 = "BMW"
```

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- An array can hold many values under a single name, and you can access the values by referring to an index number.

```
const cars = ["Saab", "Volvo", "BMW"]
```

Creating an Array

Here is the syntax:

```
const array_name = [item1, item2, ...]
```

For example:

```
const cars = ["Saab", "Volvo", "BMW"]
```

You can also use `new` keyword to create an array:

```
const cars = new Array("Saab", "Volvo", "BMW")
```

You can also create an empty array, and add elements later:

```
const cars = []
cars[0] = "Saab"
cars[1] = "Volvo"
cars[2] = "BMW"
```

Create an Array

```
const cars = ["Saab", "Volvo", "BMW"]
```

It is a common practice to declare arrays with the `const` keyword.

- The keyword `const` is a little misleading.
- It does NOT define a constant array. It defines a constant reference to an array.
- Because of this, we can still change the elements of a constant array.

```
cars[0] = "Toyota"
```

An array declared with `const` cannot be reassigned:

```
cars = ["Toyota", "Volvo", "Audi"] // ERROR
```

Accessing/Changing Array Elements

You access an array element by referring to the index number

- JavaScript arrays are **zero-indexed**
- The first element of an array is at **index 0**.
- The second is at **index 1**, and so on.
- The last element is at the value of the **array's length property minus 1**.

```
const cars = ["Saab", "Volvo", "BMW"]
let firstCar = cars[0]
let secondCar = cars[1]
let lastCar = cars[cars.length - 1]
```

You can change the value of the first element in cars like this:

```
cars[0] = "Opel"
```

Converting an Array to a String

The JavaScript method `toString()` converts an array to a string of (comma separated) array values.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"]
console.log(fruits.toString())
```

With JavaScript, the full array can be accessed by referring to the array name:

```
console.log(fruits)
```

Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.

```
console.log(typeof fruits)
```

Array Properties and Methods

Array Properties and Methods

The length property: The `length` property of an array returns the length of an array (the number of array elements).

```
const fruits = ["Banana", "Orange", "Apple", "Mango"]
let length = fruits.length
```

Accessing the First Array Element

```
let fruit = fruits[0]
```

Accessing the Last Array Element

```
let fruit = fruits[fruits.length - 1]
```

Array Properties and Methods continued

Adding Array Elements

The easiest way to add a new element to an array is using the `push()` method:

```
const fruits = ["Banana", "Orange", "Apple"]
fruits.push("Lemon");  // Adds a new element (Lemon) to fruits
```

New element can also be added to an array using the length property:

```
fruits[fruits.length] = "Lemon"  // Adds "Lemon" to fruits
```

WARNING! Adding elements with high indexes can create undefined "holes" in an array:

```
const fruits = ["Banana", "Orange", "Apple"]
fruits[6] = "Lemon"  // Creates undefined "holes" in fruits
```

My suggestion: Just use `push()`

How to Recognize an Array

A common question is: How do I know if a variable is an array?

The problem is that the JavaScript operator `typeof` returns `object`:

```
const fruits = ["Banana", "Orange", "Apple"]
console.log(typeof fruits)
```

To solve this problem ECMAScript 5 (JavaScript 2009) defined a new method Array.isArray():

```
console.log(Array.isArray(fruits))
```

The `instanceof` operator returns true if an object is created by a given constructor

```
console.log(fruits instanceof Array)
```

Thank you