JavaScript Fundamentals

JavaScript Methods

JavaScript Methods

What are methods?

- Methods are functions that are *attached* to the JS types.
- Methods are special functions that reference the values of the data stored in the object.
- We call a method using the object *dot notation*.

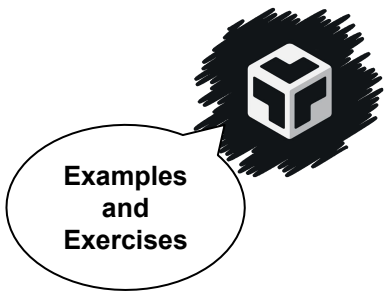
JS String Methods

notes...

- Strings are *immutable*.
- Methods will return a *new* version of the string that can then be used.

```
// common string methods

.length
.replace()
.slice()
.split()
.indexOf()
```



JS Array Methods

There are many methods for manipulating an array.

```
.includes()
.slice()

.indexOf()
.push()

.pop()
.sort()

.shift()
.unshift()

.reverse()
.lastIndexOf()

.splice()
.toString()
```

Group activity

In a breakout room, you will take a pair of array methods and

1. Find an explanation/definition
2. Rewrite it in your own words
3. Create 2 examples of each method in use.

JavaScript Fundamentals

JavaScript Functions

JavaScript Functions

- Functions are the main “building blocks” of a program. They allow the code to be called many times without repetition.
- They should take some input and **return** an output.
- They should also be considered black boxes. They take an input, or many, and output *something*.
- Functions need to be **called**.

A function that is never called is like a computer that is never turned on.

Functions (Examples)

Example 1

```
// Formula for area of rectangle


// Turn that into a more math-like function...


// Define JavaScript function



// Call the JavaScript function
```

Example 2

```
// Formula for area of circle


// Turn that into a more math-like function...


// Define JavaScript function



// Call the JavaScript function
```

JavaScript Functions

- Functions let you group and reuse code.
- *Define* a function with parameters between the brackets.
- To *call* the function (i.e. use it), you pass it arguments whose values take the place of the parameters in the function definition.
- The function acts just like any other expression when called.
- It produces the value you'd expect from the body of the function.

A function that is never called is like a light bulb without electricity.

Functions (Exercises)

Use pen and paper to write out these functions. You have 15 minutes!

```
// Q1. Write a function that returns the sum of 3
// numbers.




// Q2. Write a function that returns the square
// of a number minus twice the number.




// Q3. Write a function that returns the a
// person's full name, given their first and last
// names.
```

```
// Q4. Write a function that returns the value of
// the tax (15%) for a given amount.




// Q5. Write a function that returns the value
// 42.




// Q6. Write a function that returns "Hello!".
```

JavaScript Functions

- Functions are key in implementing software development principles.
- Complexity emerges from simplicity.

A function that is never called is like a poem that is never read.

JavaScript Fundamentals

Complex Array methods

JS Array methods

- Some array methods need a function as a parameter.
- This function is called a **callback**
- These are arguably the most flexible and useful methods.
- But they are definitely more complex.

```
const myArray = [1, 2, 4, 7];

function myFunction(arrItem) {
  // do something with each item of the array
  console.log(arrItem);
}
// this is one way of writing it.
myArray.forEach(myFunction);

// We usually just declare the function inside the
// method argument.

myArray.forEach(function(arrItem) {
  // do something with each item of the array
  console.log(arrItem);
}
```

It will become clear in time. Let’s look at some of these methods and see how they’re used.

.forEach()

The **.forEach()** method calls a function once for every element of the array.

The callback DOES NOT return anything, ever!

```
let words = ['The', 'large', 'shaggy', 'dog'];

words.forEach(function(word) {
  console.log(word);
});
```

.map()

The **.map()** method calls the provided function once for every element in the array and returns *a new array with the result*.

This method ALWAYS returns an array!

```
let words = ['The', 'large', 'shaggy', 'dog'];

const allCaps = words.map(function(word) {
  return word.toUpperCase();
});


console.log(allCaps);
```

.filter()

The **.filter()** method returns a new array with the values that meet the requirement(s).

This method ALWAYS returns an array!

The callback cannot modify the array item values.

```
let words = ['The', 'large', 'shaggy', 'dog'];

const longWords = words.filter(function(word) {
  return word.length > 4;
});


console.log(longWords);
```

.every()

The **.every()** method checks if all elements in an array pass a test.

This method ALWAYS returns either true or false

```
let words = ['The', 'large', 'shaggy', 'dog'];

const validateArray= words.every(function(word) {
  return typeof word === 'string';
});


console.log(validateArray);
```

JS Array Methods

Let’s go over these examples again in Codesandbox.