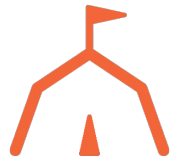
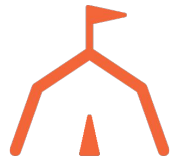
JavaScript

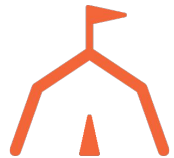


Objectives For JavaScript and DOM

- Create projects with JavaScript using VS Code.
- Code basic JavaScript with variables, conditionals, loops, and functions.
- Create and access arrays, objects, and arrays of objects.
- Write JS to respond to clicking buttons on the page.
- Use JS to modify text and styles on the page.



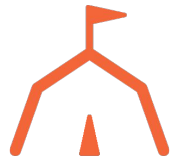
What is JavaScript



What is JavaScript

JavaScript is a dynamic, cross-platform language originally used for creating interactive websites.

JS is now popular server-side and client side-side.



JavaScript in the Browser

JavaScript runs in all browsers and most modern browsers have a JavaScript console that can be used for writing, running, and debugging code.

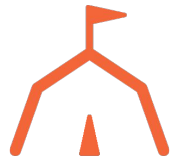
Dynamic Typing

JavaScript is a loosely typed or dynamic language, so you don’t have to declare a variable type ahead of time.

One Note on Javascript

It is an incredibly flexible language, in general there will be multiple ways of solving problems beyond what is typical of Java.

JavaScript Compared to C#



JavaScript Compared to Java

JavaScript	C#
Variable data types are not declared and can change (dynamic typing).	Variable data types must be declared and cannot change (static typing).
Code can exist outside of classes.	All code is written inside class methods.
Objects can be created with or without classes.	Objects can only be instantiated from a class.
Functions can exist outside of a class.	Functions only exist as methods of classes.

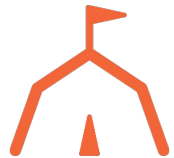
The Basics

VS Code

We will use a different IDE when working with JavaScript, TypeScript, and Angular.

Install Visual Studio Code. (Watch introduction video.)

To create a new project, simply make a folder on your computer and open that folder in VS Code.



Set up a webpage with JavaScript

In this bootcamp, we only use JavaScript within a webpage. So in the project folder, we need two files:

- index.html
- script.js

Link the files by adding the following line at the bottom of index.html, right before the closing </body> tag.

```
<script src="script.js"></script>
```

Live Serve

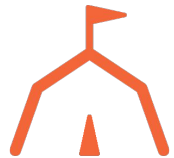
To run the project,

1. Make sure you have installed the "Live Server" extension by Ritwick Dey.
2. Right-click the index.html file and select "Open with Live Server".
3. When the page loads, open the JavaScript console to see the output and errors.

Printing to the Console

When programming in JavaScript, the output is tracked via the browser console.

```
console.log("The statement you want to output.");
```



Variables

Variables

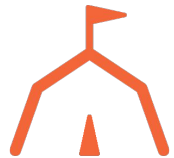
You must first declare a variable then initialize the variable. These can be done in one line or separate statements. The equals sign is used to assign a value.

Declare A Variable

```
let myVariable;
myVariable = value;
```

Initialize A Variable

```
let myVariable = value;
```



Variable Naming

Same naming rules as Java:

- Must begin with a letter, _, or $.
- Must contain letters, numbers, _, or $.

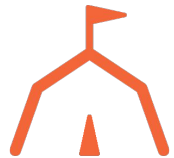
JavaScript also uses the same capitalization and indentation standards as Java: camelCase variables, TitleCase class names, etc.

Let vs Var

In modern Javascript, it is best practice to use **let** to define your variables, as let uses similar Scoping rules to Java and C#.

var will show up in older code snippets, but know its scoping rules are complex and it leads to many hard to track errors.

Data Types

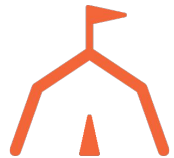


Data Types

JavaScript has five data types that you will often use: **string**, **number**, and **boolean**, **null**, **undefined**.

This bootcamp does not get into the two lesser-used primitive data types: bigint and symbol.

JavaScript also has **objects**, which includes a special type of object called an **array**.



Data Types

```
let myVariable = 25;          //number
let myVariable = "word";      //string
let myVariable = "25";        //string
let myVariable = true;        //boolean
let myVariable = null;        //null -- no value
let myVariable;               //undefined
```

String Concatenation

JavaScript has the + concatenation operator.

```
let name = firstName + " " + lastName;
```

It also has template string literals with backticks (beside "1" key) which allow insertion of values into the string.

```
let name = `${firstName} ${lastName}`;
```

Arrays

JavaScript Arrays are like Lists in Java. They can grow and shrink, and they have many useful methods.

```
let cats = []; // create an empty array
let dogs = [ "Toto", "Lassie", "Beethoven" ];
dogs.push("Marley"); // add to the end

console.log( dogs.length ); // 4
console.log( dogs[0] ); // Toto
dogs[1] = "Rin Tin Tin"; // Replace Lassie
```

Objects

JavaScript objects can be created without a class. Dot notation is used for properties.

```
let car = { model: "Mustang", year: 1964 };

console.log( car.model ); // Mustang
console.log( car["model"] ); // alternate syntax
car.year = 2019; // Change year
car.make = "Ford"; // add new property
```

Arrays of Objects

It is common, as with Java, to have an array of objects.

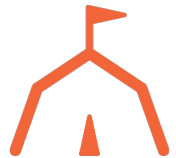
```
let headings = [
 { type: "h1", fontSize: 32 },
 { type: "h2", fontSize: 24 },
 { type: "h3", fontSize: 18 }
];
headings.push( { type: "h4", fontSize: 16 } );
console.log( headings[0].fontSize ); // 32
headings[2].fontSize = 18.7; // replace 18
```

Keywords

JavaScript has similar syntax, keywords, and operators as Java, with a few differences. Here are some of JavaScript's keywords.

break	continue	debugger
do...while	for	function
if..else	return	switch
try...catch	let, const	

Operators

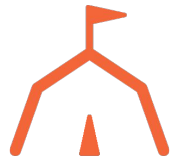


Arithmetic

Addition	+
Subtraction	-
Division	/
Multiplication	*
Modulus	%
Increment	++
Decrement	--

Assignment

standard assignment	=
plus equals	+=
minus equals	-=
assignment by multiplication	*=
assignment by division	/=



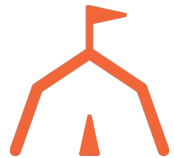
Comparison

Strict Equality (preferred)	===
Strict Inequality (preferred)	!==
Equality (with coercion)	==
Inequality (with coercion)	!=
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=

Double Equals

- 'Shallow' equals ==
- 'Shallow' inequals !=

Performs a type coercion before checking equality.



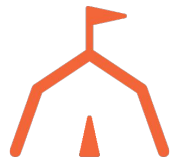
Type Coercion

Don't use double equals.

The equivalent to Java's == is ===.

```
true == "true"   // > false
true === "true"  // > false
"1" == 1         // > true
"1" === 1        // > false
"1" != 1         // > false
```

Conditionals & Loops



If/Else

Same as C# and Java

```
if (condition) {
 // do something
} else {
 // do something else
}
```

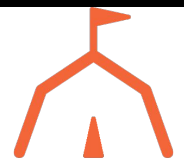
```
if (condition) {
 // statement 1
} else if (condition2) {
 // statement 2
} else if (condition3) {
 // statement 3
} else {
 // statement 4
}
```

For, While, Do...While

Same as C# and Java

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

```
let count = 0;
while (count < 5) {
  console.log(count);
  count++;
}
```



For...of

Used for looping through an array, like C#'s for each loop.

```
// headings is an array
// heading is new variable for each element
for (let heading of headings) {
  console.log(heading);
  console.log(heading.type);
  console.log(`${heading.fontSize}px`);
}
```

Functions

Function Declaration

JavaScript functions are equivalent to Java methods, but simpler.

- They are declared with the **function** keyword.
- They do not need to be inside a class.
- The do not specify parameter types or a return type.
- They do not have access modifiers (e.g. public, private)
- They never use the static keyword.

Function Declaration

```
function sayHi() {
 console.log("Hello World!");
}
sayHi(); // Call the function
```

```
function greet(name, place) {
 console.log(`Welcome, ${name} to ${place}!`);
}
greet("Dr. Grant", "Jurassic Park");
```

Function Declaration

Functions can return a value, just like C#.

```
function multiply(x, y) {
  return x * y;
}
let result = multiply(3, 4);
```