

# Module 3-5

Intro to JavaScript

# Objectives

- Variables
- Variable names
- Declaring variables
- var, let and const
- Datatypes
- Operators and arithmetic
- Type conversion



# Some Quick Facts

JavaScript is an interpreted scripting language that runs on internet browsers (client-side).

- It is not related in any way shape or form to that other language you are now familiar with.
  - Though it does share similar language syntax.
- Came into being in the mid-1990's.
- In recent times, JavaScript libraries and frameworks have greatly extended the language's capabilities.

# The Three Pillars of The World Wide Web

- **HTML:** The content being presented.
- **CSS:** How that content is formatted.
- **JavaScript:** Any actions or behaviors the content can provide.

## HTML



**BONES OF A WEB PAGE**  
HTML (Hypertext Markup Language) describes the structure and the content of a web page. It uses tags to indicate what type of content should be represented in the document.

ELEVATE YOURSELF

## CSS



ELEVATE YOURSELF

### LOOK OF A WEB PAGE

CSS (Cascading Style Sheets) enable the ability to provide visual effects for the HTML. We can use CSS to define characteristics such as font, border, color, and animation.



# JavaScript: Where does It Go?

JavaScript can be incorporated directly into a HTML page:

```
<html>
<head>
  <script>
    window.alert('Hello World. ');
  </script>
</head>
<body>Helpful Content.</body>
</html>
```

A block of JavaScript code is enclosed in a set of `<script>` tags.

# JavaScript: Where does It Go? (Preferred Method)

It is recommended that JavaScript logic be placed in a separate file and “included” in the HTML file.

```
<html>
<head>
<script src="thisScript.js"></script>
</head>
<body>Helpful Content.</body>
</html>
```

```
window.alert('Hello World.')
```

This is preferred over the first method we discussed.

# Loosely Typed

- In terms of data types, JavaScript is loosely typed, meaning we do not explicitly tell JavaScript what data type a variable will hold.
- These are the data types a variable can take on: **String**, **Number**, **Boolean**, **Arrays**, and **Objects**.

```
typeof 1 //> "number"
typeof "1" //> "string"

typeof [1,2,3] //> "object"
typeof {name: "john", country: "usa"} //> "object"

typeof true //> "boolean"
typeof (1 === 1) //> "boolean"

typeof undefined //> "undefined"
typeof null //> "object"

const f = () => 2
typeof f //> "function"
```





# Declaring Variables

- Declaring variables in JavaScript takes on the following form:

`let <<variable name>> = <<initial value>>;`

```
let myStrVariable = 'hammer';  
let myNumVariable = 3;  
let myOtherNumVariable = 3.14  
let myBoolean = true;
```

- In older texts you will see variables declared using `var`, i.e. `var myBoolean = true`. This should be avoided at all costs, **always use let**.
- Values that do not change are declared using **`const`**.

# typeof

- We can use typeof to ascertain the data type of a variable.

```
let myStrVariable = 'hammer';  
console.log(typeof myStrVariable); // string  
let myNumVariable = 3;  
console.log(typeof myNumVariable); // number  
let myOtherNumVariable = 3.14  
console.log(typeof myNumVariable); // number  
let myBoolean = true;  
console.log(typeof myBoolean); // boolean
```



Let's Code!



# Conditional Statements and Comparisons

These should also look familiar:

```
let x = -3;
let positive = (x > 0);
console.log(positive);
// Prints false

if (x < 0) {
  console.log(x + ' is a negative number.');
```

}

```
// Prints -3 is a negative number
```

# Conditional Statements and Comparisons

We can also apply AND / OR / XOR statements:

```
let x = -3;
let y = -4
let positive = (x > 0);

if (x < 0 && y < 0) {
  console.log('Both numbers are negative.');
```

}

```
else if ( x < 0 ^ y < 0) {
  console.log('Only one is negative.');
```

}

```
else {
  console.log('Both are positive');
```

}



# Declaring An Array

Here are a few examples of array declarations:

```
//Declaring an array with three strings:  
let myArray = ['Fiat Chrysler', 'Ford', 'GM'];  
  
//An empty array:  
let myEmptyArray = [];
```

# Iterating Through an Array

Our loopy friends are back:

```
let myArray = ['Fiat Chrysler', 'Ford', 'GM'];
for (let i=0; i < myArray.length; i++) {

    console.log(myArray[i]);
}
// Prints out Fiat Chrysler Ford GM
```

- Note that the for-loop is structurally similar to its Java counterpart.
- We access individual elements of an array in a similar way: myArray[0] for the first, myArray[1] for the second, etc.
- We can access the length of an array with the .length property.

# Comparison operators

## Identity vs. Equality:

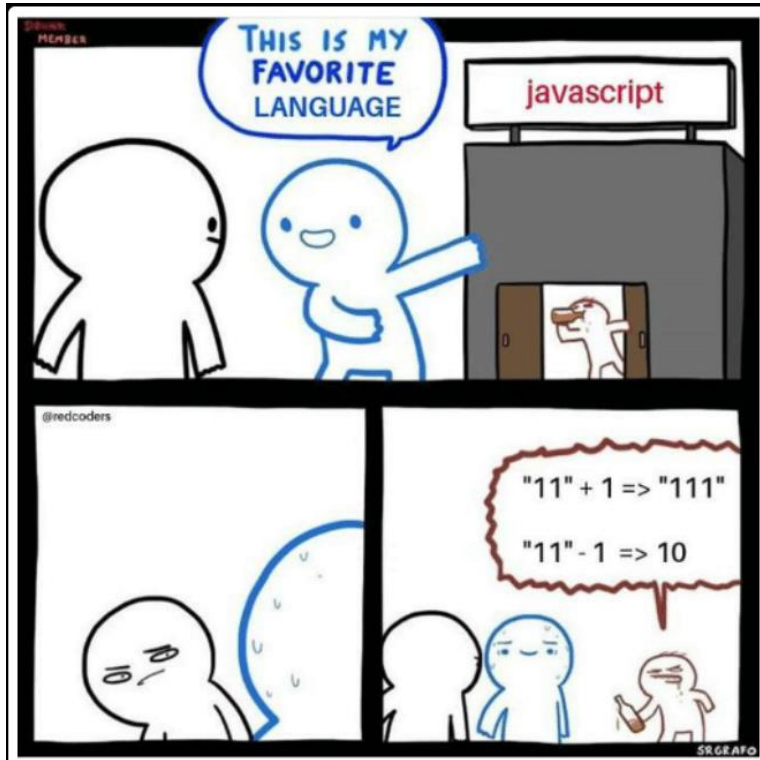
```
let x = 10;
if (x === "10"){

    console.log("equal");
}
// x is a number and "10" is a string so this
is false
```

- Identity (===) operator behaves identically to equality (==) operator except no type conversion is done
- == operator will compare for equality after doing any necessary type conversions
- === will not do the conversion and will only consider the two equal if they are of the same type and same values



# Truthy and Falsy



If you are coming from a strictly typed language like Java, there are some unusual things to consider with regards to data type, one of these is the idea of “truthy” and “falsy.”

# Truthy and Falsy

- These rules can sometimes strike one as bizarre, but we can derive an intuitive understanding of what's going on. Here is a good site with a more in-depth explanation: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality\\_comparisons\\_and\\_sameness](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness)
- For now, consider the following code:

```
let i = '1';  
let j = 1;  
console.log(i == j); // true  
console.log(i === j); // false
```

- The triple equals is to evaluate “strict equality” - meaning that not only do the values have to be the same, but the types must equal as well.

# Truthy and Falsy

The following values are **always falsy**:

- `false`
- `0` (zero)
- `''` or `""` (empty string)
- `null`
- `undefined`
- `NaN`

Everything else is **truthy**. That includes:

- `'0'` (a string containing a single zero)
- `'false'` (a string containing the text "false")
- `[]` (an empty array)
- `{}` (an empty object)
- `function(){}`  (an "empty" function)

```
function testTruthyFalsy (val)
{
    return val ? console.log('truthy') : console.log('falsy');
}

testTruthy(true);           // truthy
testTruthy(false);          // falsy
testTruthy(new Boolean(false)); // truthy (object is always true)

testTruthy('');              // falsy
testTruthy('Packt');          // truthy
testTruthy(new String(''));  // true (object is always true)

testTruthy(1);                // truthy
testTruthy(-1);               // truthy
testTruthy(NaN);              // falsy
testTruthy(new Number(NaN));  // truthy (object is always true)

testTruthy({});               // truthy (object is always true)

var obj = { name: 'John' };
testTruthy(obj);              // truthy
testTruthy(obj.name);         // truthy
```

# Objects

- JavaScript is not generally considered an object oriented language, it is instead a functional language (one that is based on functions).
  - Over time though, some OO features have been added to the language.
- JavaScript objects follow JSON notation, with the object itself surrounded by curly braces, and the object properties listed in comma delimited key-value pairs:

**{ prop1: <<prop1Value>>, prop2: <<prop2Value>> }**

# Objects Example

Let's look at a concrete example:

```
let crewMember = {  
  firstName: 'James',  
  lastName: 'Kirk',  
  rank: 'Captain'  
};  
  
console.log(crewMember.firstName);  
console.log(crewMember.lastName);  
console.log(crewMember.rank);  
crewMember.rank = 'Admiral';  
console.log(crewMember.rank);
```



# Let's Code!

The screenshot shows a web browser with multiple tabs. The active tab is a YouTube video titled "JavaScript: The Good Parts" by GoogleTechTalks. The video player displays a slide with the title "Transitivity? What's That?" and a list of JavaScript expressions and their boolean results:

- `' ' == '0'` // false
- `0 == ' '` // true
- `0 == '0'` // true
- `false == 'false'` // false
- `false == '0'` // true
- `false == undefined` // false
- `false == null` // false
- `null == undefined` // true
- `" \t\r\n " == 0` // true

Below the list, the text "so this all looks very mysterious." is displayed. The video player interface includes a search bar, a "Up next" sidebar with recommended videos, and a "SUBSCRIBE" button.

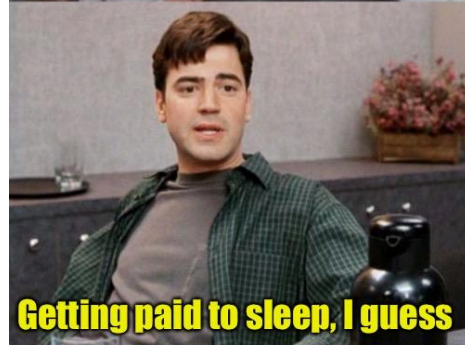
# Objectives

- Variables



# Objectives

- Variables
- Variable names





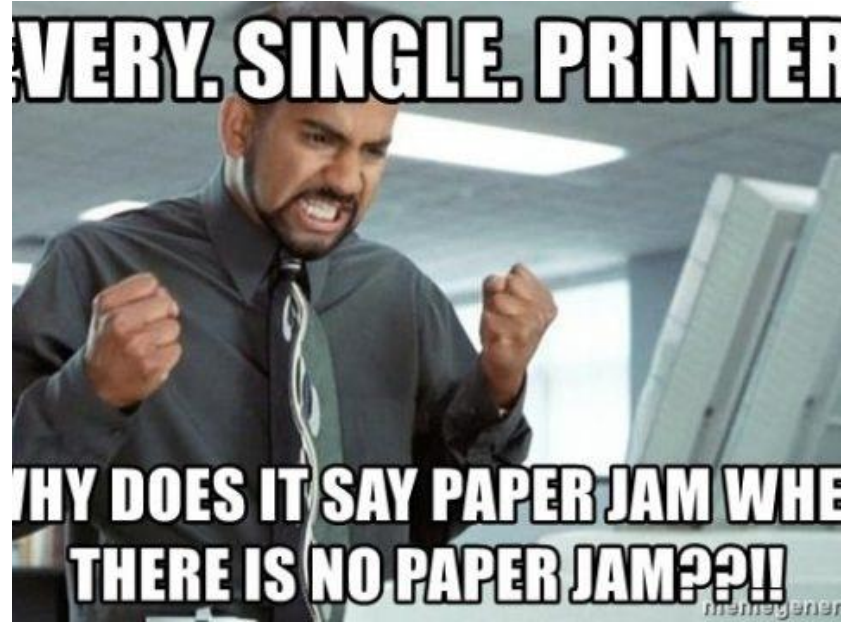
# Objectives

- Variables
- Variable names
- Declaring variables



# Objectives

- Variables
- Variable names
- Declaring variables
- var, let and const



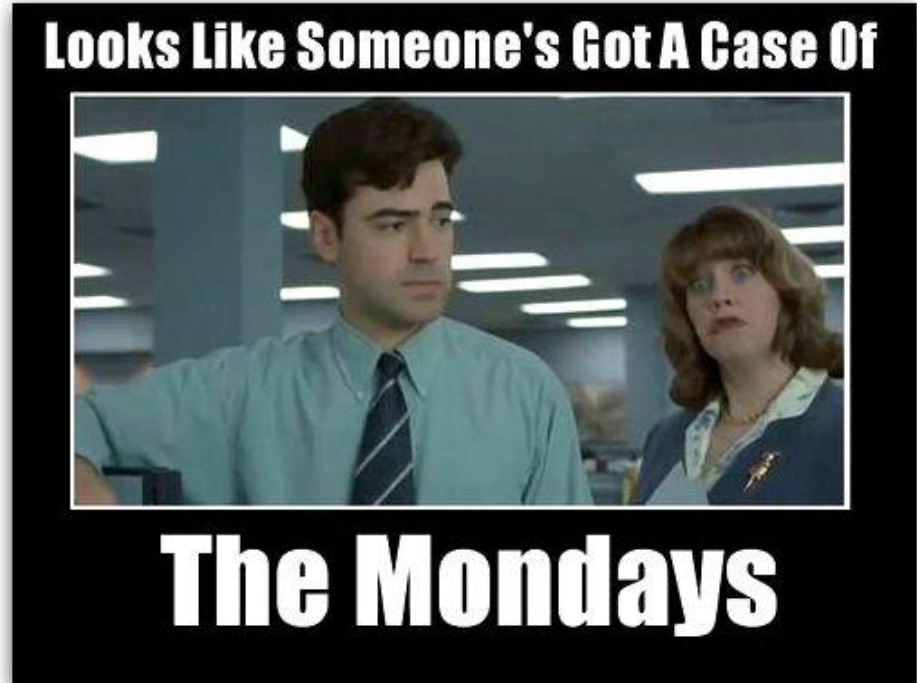
# Objectives

- Variables
- Variable names
- Declaring variables
- var, let and const
- Datatypes



# Objectives

- Variables
- Variable names
- Declaring variables
- var, let and const
- Datatypes
- Operators and arithmetic



# Objectives

- Variables
- Variable names
- Declaring variables
- var, let and const
- Datatypes
- Operators and arithmetic
- Type conversion

