

# INTRO TO JAVASCRIPT

# TODAY'S OBJECTIVES

- **What is Client-Side Scripting and Why Use it?**
- **What is Javascript?**
  - Compiled languages vs. interpreted languages
  - Statically typed vs. dynamically typed
- **Variables in JavaScript**
  - Declaring variables
  - Naming
- **Data Types**
  - Strict vs. loose equality
  - Number, String, Boolean, Object (includes arrays), undefined
  - Type coercion
  - Null vs undefined
- **Branching**
  - if/else if/else
  - switch
- **Loops**
  - for/while/do

# TODAY'S OBJECTIVES

- **Arrays**
  - push/pop
  - unshift/shift
  - indexOf/lastIndexOf
- **JavaScript Objects**
- **Functions in JavaScript**
  - Signature
  - Variable scope
- **Built-In Functions**
  - String methods
  - Numbers, Math, and Dates

# WHAT IS CLIENT-SIDE SCRIPTING?

- Executes code on the user's browser, allowing us to interact with the HTML rendered and the CSS sent by the server.
- Interacts with HTML on the page (the DOM - Document Object Model).
- JavaScript is the scripting language all browsers understand.

# WHY DO WE USE CLIENT-SIDE SCRIPTING?

- Creates less stress on the server and more interactive engaging experiences for users.
  - Allows client (browser) to perform validation immediately.
  - Fewer calls to server.
- Allows page interaction/manipulation.
  - Can respond to user event.
  - Can make calls to web services/APIs to dynamically update page.
  - Can update page without page refresh DOM manipulation.
- Separation of Concerns.
  - HTML: Presentation content
  - CSS: Presentation styling
  - JavaScript: Behavior and logic

# WHAT IS JAVASCRIPT?

- Programming language with similarities to Java and C#.
- How is JavaScript different than Java/C#
  - Java/C# require a runtime while JavaScript requires a browser.
  - Java/C# are compiled while JavaScript is interpreted.
  - Java/C# are statically typed while JavaScript is dynamically typed.

# DECLARING JAVASCRIPT VARIABLES

- JavaScript doesn't require data type in declaration.
- Declare variables that will change using `let`.

```
let myText = 'Hello world!';  
  
// can be changed  
  
myText = 'Howdy world!';  
  
// can also be declared without value,  
  
let myOtherText;  
  
// then assigned later  
  
myOtherText = 'Hello other world!';
```

# DECLARING JAVASCRIPT VARIABLES

- Declare variables that will not change using **const**.

```
const MY_CONST_TEXT = 'Hello world!';  
  
// CANNOT change: below will throw an error  
  
MY_CONST_TEXT = 'Howdy world!';  
  
// CANNOT be declared without value:  
// below will throw an error  
  
let myOtherText;
```



# DECLARING JAVASCRIPT VARIABLES

- Avoid using **var** - considered **harmful!**
  - Used in older versions of JavaScript
  - Allows multiple declarations without warning
  - Function scope (vs. block scope)
  - Use **let** or **const** instead

# JAVASCRIPT VARIABLE NAMING

- Variable names are comprised of letters **A-Z**, **a-z**, characters **\_**, **\$**, and **digits 0-9**.
- Variable names must start with a **letter**, **\_**, or **\$**.
- Variable names are case-sensitive.
- Variable names may be not be a reserved keyword.
- Follow best practice conventions:
  - Use camelCase for multi-word variable names.
  - Use uppercase for constants and separate words with an underscore, **\_**.
  - Boolean variable should begin with **is**

# JAVASCRIPT DATA TYPES

- Number
  - integer
  - floating-point
  - **NaN**
- String
  - Zero or more characters enclosed in double(") or single (') quotation marks ("foo" or 'foo').
  - Build larger strings from smaller ones in code with string concatenation using the concatenation operator, +, just as you do in Java and C#.
- Boolean

# JAVASCRIPT DATA TYPES

## **null vs. undefined:**

- **null** is a value of type **Object**
- **undefined** is a value of type **undefined**
- **null** must be assigned. It means **nothing**.
- **undefined** occurs from the "**let var\_name;**" statement
  - It also may be assigned

# JAVASCRIPT DATA TYPES

## JavaScript is loosely typed

- Variables aren't associated with any particular data type when declared and are free to hold any type of value.
- Variables can be assigned and re-assigned values of any datatype.
- JavaScript does type coercion as necessary.

# JAVASCRIPT DATA TYPES

## Strict and loose equality

- === vs. ==
- === means types and values are equal (strict equality)
- == means values are equal (loose equality)
- Types are coerced
- !== and != are the "not equal" equivalents
- **Falsy** values:
  - When coerced to Boolean, value is false
  - false, 0, "", null, undefined, NaN
- All other values are **Truthy**
- More craziness: <https://codeburst.io/javascript-double-equals-vs-tripleequals-61d4ce5a121a>

# LOGICAL BRANCHING

- if
- else if
- else
- switch

# LOOPING

- for
- while
- do

```
for (let i = 0; i < 5; i++) {  
    console.log("Hello world!");  
}  
  
let i = 0;  
while (i < 5) {  
    console.log("Hello world!");  
    i++;  
}  
  
let i=0;  
do {  
    console.log("Hello world!");  
    i++;  
} while (i < 5);
```



# STRING INTERPOLATION IN JAVASCRIPT

- String interpolation in JavaScript use the ` mark (know as a tick) to enclose the literal template.
- Values enclosed in `\${}` are populated with the variable name with the {}.

```
let birthDate = '03/15/1970';  
  
console.log(`Birthdate is ${birthDate }`);
```

# JAVASCRIPT SCOPE

- Can declare a variable at any point in a block, but you must declare it before you use it.
- Once declared, the variable is in scope.
- Variables are in scope until the end of the block when they are discarded and go out of scope.
- Nested blocks:
  - Each nested block can declare and use its own set of local variables.
  - Statements within the inner block can use both variables from the inner and outer scope
  - JavaScript allows a variable in an inner block to have the same name as a variable in an outer block (this is called variable shadowing) but this should be avoided.

# JAVASCRIPT ARRAYS

- Defining arrays:
  - `let scores = [];`
  - `let scores = [10, 20, 30];`
- Accessing arrays
  - `scores[2];`
  - **index is 0 based.**
- Array size can be modified in JavaScript!
- Can check size of array with **length** property.

# JAVASCRIPT ARRAY FUNCTIONS

- **push** adds element to end of array
- **pop** removes element from end of array
  - returns element removed
- **unshift** adds element before first element of array
- **shift** removes element at first element of array
  - returns element removed
- **includes** indicates whether an array contains a given value
- **indexOf** returns the index of first occurrence of value in array, or -1 if not found
- **lastIndexOf** returns the index of last occurrence of value in array, or -1 if not found

# JAVASCRIPT OBJECT LITERALS

- { } denotes an object
- Key : value pairs, separated by commas

```
const person = {  
  firstName: 'Lisa',  
  lastName: 'Simpson',  
  age: 42,  
  relatives: [  
    'Marge Simpson',  
    'Homer Simpson',  
    'Bart Simpson'  
  ]  
};
```

- Access element: person.firstName

# JAVASCRIPT FUNCTIONS (JAVASCRIPT VERSION ON METHODS)

- no access modifier
- function keyword
- function name
  - usually camel-case
- no return type
- parameter names
  - no type defined
- return statement

```
function sumVals(val1, val2)
{
    return val1 + val2;
}
```

# BUILT-IN FUNCTIONS

## String methods

- [https://www.w3schools.com/js/js\\_string\\_methods.asp](https://www.w3schools.com/js/js_string_methods.asp)
- [https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Text\\_formatting](https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Text_formatting)

## Numbers, Math and Dates

- [https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Numbers\\_and\\_dates](https://developer.mozilla.org/enUS/docs/Web/JavaScript/Guide/Numbers_and_dates)
- [https://www.w3schools.com/js/js\\_number\\_methods.asp](https://www.w3schools.com/js/js_number_methods.asp)
- [https://www.w3schools.com/js/js\\_math.asp](https://www.w3schools.com/js/js_math.asp)
- [https://www.w3schools.com/js/js\\_dates.asp](https://www.w3schools.com/js/js_dates.asp)

# JAVASCRIPT VS. C#

	JavaScript	C#
Variables	<pre>let age = 35; const PI = 3.14159; • Also var (don't use)</pre>	<pre>int age = 35;</pre>
Data Types	Number, String, Boolean, Symbol, Object, null, undefined	int, short, long, float, decimal, single, double, string bool...
Strings	Single or double quotes	Double quotes
Undefined data type	"undefined" is both a datatype, and a value	Not applicable
String interpolation (template literals)	<code>`My name is \${firstname} \${lastname}`</code>	<code> \$"My name is {firstname} {lastname}"</code>
Equality Comparison	<code>==, ===, !=, !==</code>	<code>==, !=</code>
Array	Size can change Multiple types Directly address even new elements	Size fixed and set when allocated Single data type Must address in-bounds
Functions / methods	<code>function myFunction(param1)</code>	<code>public int myMethod(string param1)</code>