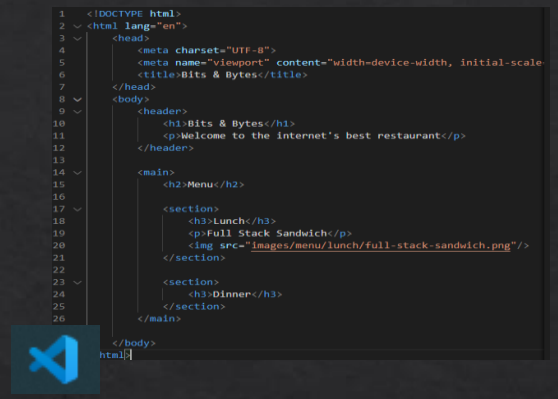
Module 3

Introduction to JavaScript



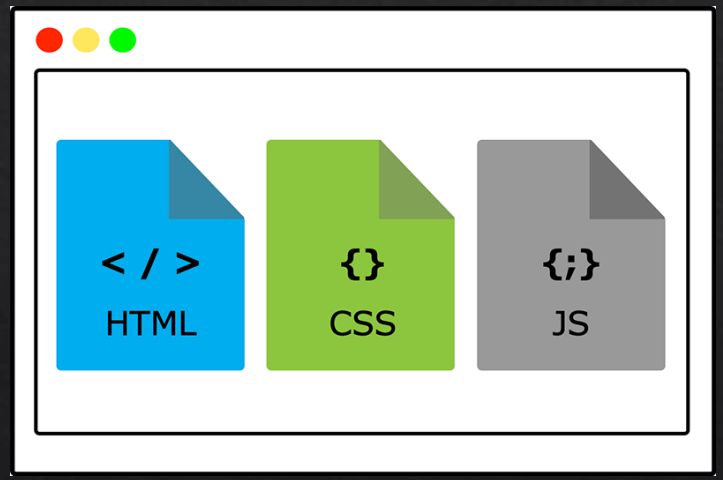
Session Objectives:

- Variables
  - Variables Names
  - Declaring Variables
  - var, let & const
- Datatypes
- Operators and Arithmetic
- Type Conversion

Module 3
Introduction to JavaScript

HTML, **CSS**, and **JavaScript** are the building blocks of the web. Each of these languages play their own significant roles in building the web:

- **HTML**: Provides the basic structure or markup of a document.
- **CSS**: Provides formatting of the document to control presentation and layout.
- **JavaScript**: Provides behavior to the document.



Module 3

Introduction to JavaScript

JavaScript is a general-purpose programming language originally intended as a way to bring dynamic behavior to the static content of HTML. JavaScript has evolved considerably from its earliest days and is now used in ways not thought of back in the mid-1990s. From the beginning, JavaScript was designed to be a quickly learned, forgiving language that was approachable for beginners.

Standardized under the name, "ECMAScript".

JavaScript is executed from within the user's browser.

Module 3
Introduction to JavaScript

Variables - Variable Names

Naming rules for JavaScript variables:

- Variable names consist 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 not be a reserved keyword.

The following are considered best practices in JavaScript:

- Use camelCase for multi-word variable names.
- Use uppercase for constants and separate words with an underscore, _.
- Boolean variable names should begin with is.

Module 3
Introduction to JavaScript

Variables - Declaring Variables

There are two ways to declare a variable in JavaScript. In either case, the basic form is the reserved words, **let**, or **const**, a variable name, and a semi-colon, **;**

Using let
You use let when you know the value of the variable needs to be changeable. This is its basic form:

```
let age; // Declare without initializing
let breed = 'Poodle'; // Declare and initialize
```

Using const
The reserved word const—short for "constant"—is the alternative to let. Variables declared with const must be initialized with a value, and can't be reassigned later. Here's an example:

```
const PI = 3.14159;
```

Avoid using var
Avoid variable shadowing

Module 3
Introduction to JavaScript

Data types

The most important JavaScript data types to be familiar with are:

- Number
- String
- Boolean
- Object
- null
- Undefined

In JavaScript, variables aren't associated with any particular data type when you declare them. This makes JavaScript a loosely typed language.

Module 3
Introduction to JavaScript

Operators and Arithmetic

JavaScript provides +, -, *, /, and % arithmetic operators.

Closely associated with basic arithmetic are the shorthand assignments operators (+=, -=, *=, /=, and %=)

Module 3
Introduction to JavaScript

Type Conversion

The Number object has several useful functions that can help you convert strings to numbers or numbers to strings.

Number to string

- toString()
- toFixed()

ℹ Note: Methods are functions
JavaScript refers to methods as functions. The terms are commonly used interchangeably.

Module 3
Introduction to JavaScript

Type Conversion

String to number
There are two options for converting, or parsing a string into a number:

- parseInt()
- parseFloat()

```
Number.parseInt(string)
Number.parseFloat(string)
```

```
const myDecimal = Number.parseFloat('97.5'); // Result: 97.5
const myInteger = Number.parseInt('97.5'); // Result: 97
```

ℹ Note: NaN
There is one potential issue that exists when parsing strings. It's always possible that the string you attempt to parse can't be converted into a number. In these cases, JavaScript returns NaN (Not-a-Number):

Module 3
Introduction to JavaScript

Logical Branching

Boolean expressions:
boolean expressions evaluate to true or false. There are two common ways that boolean expressions are built:

- Comparison operators (to compare two values)
- Logical operators (to create relationships between one or more boolean values)

Module 3
Introduction to JavaScript

Logical Branching

Comparison operators
In addition to the relational operators (>, <, >=, <=), JavaScript has four equality operators:

Operator	Meaning
`==`	Equal To
`===`	Strictly equal to
`!=`	Not Equal To
`!==`	Strictly not equal to

The equality (==) and inequality (!=) operators attempt to convert operands of different types before comparing them, while the strict versions require values to be the same type to be considered equal:

```
7 == 7;  // true
7 == '7'; // true
7 === 7; // true
7 === '7'; // false
```

Module 3
Introduction to JavaScript

Logical Branching

Logical operators
The logical operators are AND (&&), OR (||), and NOT (!). These operators are used with boolean values to create boolean expressions.

Module 3
Introduction to JavaScript

Conditional code

JavaScript includes an if statement for logical branching that works the same way an if statement does in Java or C#:

```
if (condition) {

} else if (condition) {

} else {

}
```

The condition specified for the if may be a boolean expression or any other truthy or falsy value:

```
if (100) {
  // The code here will be executed because 100
 is a truthy value.
}
```

Module 3
Introduction to JavaScript

Arrays

Declaring and initializing an array

```
const testScores = [];
```

```
const testScores = [ 85, 96, 80, 98, 89, 70, 93, 84, 66, 96 ];
```

Determining the length of an array

```
const size = testScores.length;
```

Accessing elements within an array

```
const testScores = [ 85, 96, 80, 98, 89, 70, 93, 84, 66, 96 ];

testScores[0] = 82; // update the value at index 0 to 82
testScores[1] = 72; // update the value at index 1 to 72
testScores[4] = 80; // update the value at index 4 to 80

const highScore = testScores[3]; //set highScore to 98
```

Module 3
Introduction to JavaScript

Loops

for loops

```
let sum = 0; // the sum of all our scores

for(let i = 0; i < testScores.length; i++) {
    sum = sum + testScores[i];  // add each score to
the sum
}

const average = sum / testScores.length;
```

Module 3
Introduction to JavaScript

Loops

for...of loops

```
let sum = 0;

for (const score of testScores) {
    sum += score;
}

const average = sum / testScores.length;
```

Module 3
Introduction to JavaScript

Loops

while and do...while loops

```
while (condition) {
  //body (which never runs if condition is false or a
 falsy value)
}
```

```
do {
  //body (which always runs at least once)
} while (condition);
```

Module 3
Introduction to JavaScript