5. Introduction to JavaScript

5.1 Introduction

- Low level languages
    - Closer to being understood by a computer's CPU
- High Level Languages
    - Needs to be interpreted → Changed to CPU-understandable language
- JavaScript
    - Language used to incorporate interactivity into web pages
    - Allows direction interaction with the webpages dynamically
    - Backwards compatible
    - * New Tip * - In browser when we write %c in console log statement - the next line is treated as the CSS

5.2 Data Types

- **Primitive Data Types**
    - String
    - Number - Integers and decimal points

    - Boolean - *true* or *false*
    - Null - Absence of Value
    - Undefined - Variable not assigned a Value
    - Symbol - Use as unique identifier
    - Big Int - To accommodate a greater range of number

5.3 Operators

- **Arithmetic**
    - Add (+)
    - Subtraction (-)
    - Multiplication (*)
    - Divide (/)
- **Logical**
    - && (AND)
    - | | (OR)
    - ! (NOT)
- Comparisons
    - > (Greater Than)
    - < (Less Than)
    - == (Equal)
    - === (Strict Equality)
    - != (Inequality)
    - !== (Strict Inequality)

5.4 Objects

- Collection of related properties
    - Each property can be specified as key-value pair
- Dot can be used to add new properties

```
# Method 1

var house = {}

house2.address = 'Ave E';
house2.type = 'Condo';

# Method 2

var house2 = {
  house2.address : 'Ave E',
  house2.type: 'Apartment',
}

# Method 3

var house3 = {}

house3['address'] = 'Ave E';
house3['type'] = 'Studio'
house3['number of members'] = 5
```

- With Bracket Notation → It is possible to add space between the property names.
- **Math object**
  - ceil
  - floor
  - round
  - trunc
  - pow
  - sqrt
  - cbrt
  - abs

5.5 Closer look at Strings

- For-loop can be executed over strings.
- Some common methods
  - Length

    - chat At
    - Concat
    - index of
    - split
    - to Upper Case
    - to Lower Case

5.6 Bugs and Error

- **Bug** - The program keeps on running in an unintended way

```
function addTwo(num1, num2) {
  return num1 + num2;
}

let input = addTwo("1", 2);
console.log(input);

//Output: 12
```

- **Error** - The program stops execution and no further lines are executed

```
console.log(c + d);
console.log("This line never runs");

// ReferenceError : c is not defined
```

- **Types**
    - Syntax Error
        - Piece of code that JavaScript cannot read.
    - Type Error
        - Running a method that does not exist.
    - Reference Error
    - Range Error
        - A Range Error is thrown when we're giving a value to a function, but that value is out of the allowed range of acceptable input values.

- **Try-Catch Block**
  - Basic format -

```
try {
  // main execution
}
catch(err){
  // do something here
}
```

  - Using the *throw* keyword → we can throw the keyword to be caught by catch block

```
try {
  // main execution
  throw new Error();
}
catch(err){
  // do something here
}
```

  - Here, the program continues to run even after an error was observed

```
try {
  console.log(a + b);
} catch (err) {
  // console.log(`The Error: ${err}`);
  console.log(err);
  console.log("There was an error");
}
console.log("Program Continues...");
```

- **Types of Empty Values**
  - *Null*
    - Intentional absence of object
  - *Undefined*
    - Can only hold one value - *undefined*
    - All functions return undefined by default
    - Unless a specific return value has been specified - we see this in console log statement

- *Empty*
    - Empty Strings

5.7 Defensive Programming

- Assumes that the function arguments are always wrong - type or value

```
function letterFinder(word, match) {
  var condition1 = typeof word == "string" && word.length >= 2;
  var condition2 = typeof match == "string" && match.length == 1;

  if (condition1 == true && condition2 == true) {
    for (i = 0; i < word.length; i++) {
      if (word[i] == match) {
        //if the current character at position i in the word is equal to the match
        console.log("Found the", match, "at", i);
      } else {
        console.log("---No match found at", i);
      }
    }
  } else {
    console.log("Please pass correct arguments to the function.");
  }
}
```
