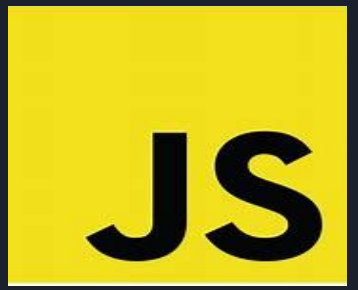
WEEK 1
CLASS 3



WEB PAGE

1. HTML

   Markup language that defines the structure of the web page.

2. CSS

   StyleSheet language that is used to style HTML elements or to specify layout of the webpage

3. JavaScript

   Scripting language that enables you to create dynamically updating content, control multimedia, animate images. All scripting languages are programming languages.

JavaScript Almost Everywhere ...

Where can we use JavaScript ?

1. Interactive Web Pages.

   Ex : Beautiful animations, Form validation, Countdown

2. Making server-side apps

   JavaScript is also used as a server-side language to build backend using **node.js**

3. Framework/Library Implementation

   React, angular, Vue, jquery

4. Mobile Apps and Game Development

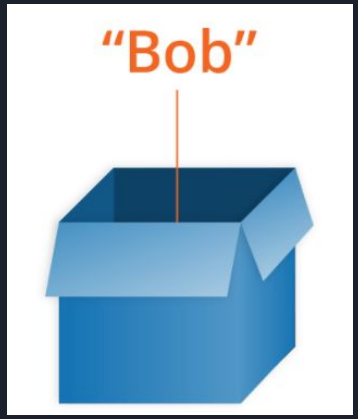
   Using **React Native** and games can be built using libraries like **three.js**

Basics : Variable Declaration

A variable can be declared in one of the three ways:

1. let
2. const
3. var

Scope difference




```
let myName;
let myAge;
```

Basics : Variable Initialization

Once we have declared a variable we can initialize it with a value.

```
myName = 'Chris';
myAge = 37;
```

Declaring and Initializing at same time

```
let myDog = 'Rover';
```

Basics : Comments

```
// a one line comment

/* this is a longer,
 * multi-line comment
 */

/* You can't, however, /* nest comments */ SyntaxError */
```

There are two ways to add comments in JavaScript :

1. One line comment
2. Multi-line comment

Data Types

Primitive

1. Boolean - true or false
2. null - null is a special value that represents an empty or unknown value.
3. undefined - The value which is automatically assigned to variables that have been declared but not assigned
4. Number - An integer or floating point number.
5. BigInt - The BigInt type is a numeric primitive in JavaScript that can represent integers with arbitrary precision.
6. String - A sequence of characters that represent a text value
7. Symbol - A data type whose instances are unique and immutable.

Examples : Primitive Data Types

1. JavaScript Boolean

```
let x = 5;
let y = 5;
let z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

2. null

```
var saveButton = document.getElementById("save");

if (saveButton !== null)
        saveButton.submit();
```

Examples : Primitive Data Types

3. undefined

```
var myVar;

alert(myVar); // undefined
```

4. Number

```
let x1 = 34.00;     // Written with decimals
let x2 = 34;        // Written without decimals
```

Examples : Primitive Data Types

5. BigInt

-(2^53 - 1) and 2^53 - 1

```
let maxNumber = Number.MAX_SAFE_INTEGER;
console.log(maxNumber+' = '+typeof maxNumber) // 9007199254740991 = number
maxNumber=maxNumber+1
console.log(maxNumber+' = '+typeof maxNumber) //9007199254740992 = number
maxNumber=maxNumber+1
console.log(maxNumber+' = '+typeof maxNumber) // 9007199254740992 = number
```

6. String

```
let carName1 = "Volvo XC60";   // Using double quotes
let carName2 = 'Volvo XC60';   // Using single quotes
```

7. Symbol

```
let id1 = Symbol("id");
let id2 = Symbol("id");

alert(id1 == id2); // false
```

Data Types : Non-Primitive

Non-Primitive are collectively referred as Objects

1. Objects

   Objects are used to store various keyed collections and more complex entities.

2. Arrays

   JavaScript arrays are used to store multiple values in a single variable

3. Functions

   A JavaScript function is a block of code designed to perform a particular task.

Data Type: Non-Primitive Examples

1. Objects

Properties

- color
- brand
- Weight
- model

Methods

- start
- drive
- stop
- brake

Object Syntax

```
var objectname = {
                        //Properties and Methods here
                    }
```

Let's start building our car object

1. Adding Properties

```
let car = {
            color : “white” ,
            brand : “ford” ,
            weight : “mustang” ,
             model  : “2021”
          }
```

Object Syntax

2. Adding Methods

Methods are actions that can be performed on objects.

A method is function as a property

```
let car = {
            color : “white” ,
             brand : “ford” ,
             model  : “2021”,
             printInfo : function(){
                   console.log( ‘ Car  is :  ’ + this.brand +  ‘of the year  ’  + this.model ‘ ) ;
             }
       }
```

Accessing Properties/Methods

So, our final car object is:

```
let car = {
                color : “white” ,
                 brand : “ford” ,
                 model  : “2021”,
                 printInfo : function(){
                       console.log( ‘ Car  is :  ’ + this.brand + ‘of the year  ’  + this.model ) ;
                 }
           }
```

1. Property Access

   This can be done in one of two ways : a) Dot b) Square

Accessing Properties (dot vs [ ])

```
let car = {
            color : “white” ,
             brand : “ford” ,
             model  : “2021”,
             printInfo : function(){
                   console.log( ‘ Car  is :  ’ + this.brand +  ‘of the year : ’  + this.model ) ;
             }
          }
```

```
console.log(car.color);                    // white
console.log(car.brand)                     // ford
console.log(car[“model”])                  // model
console.log( car.printInfo())              // Car is : ford of the year: 2021
```

Arrays and Array Methods

An array is a special variable, which can hold more than one value at a time.

```
const cars = ["Saab", "Volvo", "BMW"];
```

Accessing Items in Array

```
cars[0] ;          // Saab
cars[1] ;          // Volvo
cars[2];           // BMW
```

Modifying Array Items

```
cars[0] = “Jeep”;
cars[2] = “MS”;
console.log(cars);          // [ “Jeep”, “Volvo”, “BMW” ]
```

Array Properties and Methods

Just like in the car examples we defined properties i.e. model, year, color , Since array is an object too, it gives use access to some built-in properties and methods

Array Inbuilt Properties

1. length
2. prototype

Array built-in Methods

1. Push
2. Pop
3. join
4. Shift
5. Unshift
6. Slice
7. concat

String and String Methods

String also has inbuilt length property just like an array.

Methods String

1. slice(start, *end)

   extracts a part of a string and returns the extracted part in a new string.

2. substring(start, *end)

   works similar to slice() but it cannot accept negative indexes.

3. substr(start, *length)

   works similar to slice() but the difference is that in the 2nd parameter we specify the length of extracted object.

* - optional parameter

String and String Methods

String Search Methods

- Indexof (index | -1)

  Returns the index of the first occurrence of a specified text in a string

- lastIndexof (index | -1)

  Returns the index of the last occurrence of a specified text in a string

- startWith (true | false)

  Return true if a string begins with a specified value

- endsWith (true | false)

  Returns true if a string ends with a specified value

More Numbers

We can store number as :

1. Decimal | Exponent | Floating Values

```
let x = 3.14;    // A number with decimals
let y = 3;       // A number without decimals
```

```
let x = 123e5;    // 12300000
let y = 123e-5;   // 0.00123
```

Numbers

Adding Numbers and String

Data Type	Data Type	Operation	Result Data Type
NUMBER	NUMBER	ADDITION	NUMBER
NUMBER	STRING	CONCAT	STRING
STRING	NUMBER	CONCAT	STRING
STRING	STRING	CONCAT	STRING

NaN (Not a Number)

Indicates that a number is not a legal value, Ex : 1 * “name”

Functions

SYNTAX :

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}
```

Why Functions ?

You can reuse code : Define the code once, and use it many times

Flex Froggy : Let’s Play

Discussion :

Assignment - 1

GitHub