FSD Assignment – 3

Watch the following YouTube lectures on JavaScript – Values, Types and Operators. Please feel free to watch more videos of your choice.

1. History of JS

   Definition:

   JavaScript is a high-level, dynamic programming language used primarily for creating interactive and responsive elements on websites. It runs in web browsers and enables developers to control multimedia, animate images, and handle user inputs. JavaScript is versatile, allowing it to be used on both the client-side and server-side.

   Examples:

   1. **Interactive Forms:** Validating user input in real-time before submission.
   2. **Animations:** Creating moving graphics or transitions on web pages.
   3. **Real-Time Data Updates:** Displaying live sports scores or chat messages without page reloads.

2. Data Types in JavaScript - 1

Primitive Data Types: These are the most basic types of data. They include:

- **String:** Represents textual data (`"Hello, World!"`).
- **Number:** Represents numerical data (both integers and floating-point numbers, like `42` or `3.14`).
- **Boolean:** Represents logical values (`true` or `false`).
- **Undefined:** Represents a variable that has been declared but not assigned a value.
- **Null:** Represents an intentional absence of any object value.
- **Symbol:** A unique and immutable primitive data type in JavaScript, often used as keys for object properties to avoid name collisions.

Object Type.

Examples:

console.log(typeof 9.09) //number

console.log(typeof 10/0) // Infinity

console.log(typeof -10/0) //Infinity

3. Data Types in JavaScript - 2

String: Text data enclosed in quotes

Examples

```
string1 = "Hello, world,,"
string2 = "This is a new line.\nHere is the next line."
result = string1 + string2
print(result)

string1 = "Name \t Ramya"
string2 = "Age \t 20"
result = string1 + string2
print(result)
```

```
let firstName = "ramya";
let lastName = "sree";
let fullName = firstName + " " + lastName;
console.log(fullName); // "ramya sree"
```

Boolean: A value that is either `true` or `false`.

Examples:
console.log(3>2)
console.log(9<10)
console.log(45>90)

Null: A special value that means "no value."
Example:
let object=null
console.log(typeof object)

Undefined: A value that means a variable has been declared but not yet assigned a value.
Example:
let object
console.log(object)

4. Type Conversion and Coercion

Type Conversion

Definition: Type conversion, also known as type casting, is the explicit conversion of a value from one type to another. It is performed by the programmer using functions or methods.

Examples:

let str = "123";

let num = Number(str);

console.log(num); //123

let num = 456;

```
let str = String(num);

console.log(str);  // Output: "456"


let bool = true;

let num = Number(bool);

console.log(num);  // Output: 1
```

Type Coercion

Definition: Type coercion is the automatic or implicit conversion of values from one type to another by the JavaScript engine during operations. It happens without explicit conversion by the programmer.

Examples:

```
let result = "The number is " + 42;

console.log(result);  // Output: "The number is 42"


let result = 5 + " apples";

console.log(result);  // Output: "5 apples"
```

```
let result = 1 + true;

console.log(result);  // Output: 2 (true is coerced to 1)
```

5. Arithmetic Operators

Arithmetic Operators

Definition: Arithmetic operators perform mathematical operations such as addition, subtraction, multiplication, division, and modulus. They work with numerical values to produce results based on the operation performed.

Examples:

```
let power = 2 ** 3;

console.log(power);  // Output: 8

let i = 5;

console.log(++i);  // Output: 6 (value of `i` is incremented first, then used)

let i = 5;

console.log(i++);  // Output: 5 (value of `i` is used first, then incremented)

console.log(i);    // Output: 6 (value of `i` after increment)
```

6. Relational Operators

Relational operators are symbols used to compare two values or expressions. They determine the relationship between these values, such as whether one is greater than, less than, equal to, or not equal to the other. The outcome of using a relational operator is always a boolean value: `true` if the specified condition is met, and `false` if it is not.

Examples:

Here are three examples of relational operators in JavaScript:

1. `5 > 3;` // `true` - Checks if 5 is greater than 3.
2. `7 == 7;` // `true` - Checks if 7 is equal to 7.
3. `4 <= 2;` // `false` - Checks if 4 is less than or equal to 2.

7. Logical Operators

Logical operators are used in programming to perform logical operations on boolean values (`true` or `false`). They allow for combining multiple conditions and are essential for controlling the flow of a program, especially in decision-making structures like `if` statements. The three main logical operators are:

1. **&& (Logical AND):** Returns `true` if both operands are `true`. If either operand is `false`, it returns `false`.
2. **|| (Logical OR):** Returns `true` if at least one of the operands is `true`. If both are `false`, it returns `false`.
3. **! (Logical NOT):** Inverts the boolean value of its operand. If the operand is `true`, it returns `false`, and vice versa.

Examples:

```
let a = 5;

let b = 10;

console.log(a > 3 && b < 15); // true


let a = 5;

let b = 10;

console.log(a > 3 || b > 15); // true


let isRaining = true;

console.log(!isRaining); // false
```

8. Ternary Operators

Ternary Operator is a shorthand conditional operator in programming that allows you to evaluate a condition and return one of two values depending on whether the condition is `true` or `false`. It is called "ternary" because it takes three operands: a condition, a value to return if the condition is `true`, and a value to return if the condition is `false`.
The syntax is condition ? expressionIfTrue : expressionIfFalse;

Examples:

```
let number = 5;
let result = number > 0 ? "Positive" : "Negative";
console.log(result); // "Positive"

let age = 18;
let isAdult = age >= 18 ? "Yes, an adult" : "No, not an adult";
console.log(isAdult); // "Yes, an adult"

let number = 7;
let result = (number % 2 === 0) ? "Even" : "Odd";
console.log(result); // "Odd"
```

9. Template Literals

Template Literals are a feature in JavaScript that allows you to work with strings in a more flexible way. They are enclosed by backticks (`\`) instead of regular quotes, and they allow for embedding expressions, multi-line strings, and string interpolation using ${expression}`.

Examples:

```
let name = "Ramya";
let age = 25;
let message = `My name is ${name} and I am ${age} years old.`;
console.log(message); // "My name is Ramya and I am 25 years old."

let a = 5;
let b = 10;
let result = `The sum of ${a} and ${b} is ${a + b}.`;
console.log(result); // "The sum of 5 and 10 is 15."

let fruit = "apple";
let quantity = 5;
```

```
let message = `I have ${quantity} ${fruit}s.`;
console.log(message); // "I have 5 apples."
```

Once you complete watching those videos, document each topic you learnt with a proper definition(in your own words, don’t copy paste what was told in the video) and at least 3 examples for each topic.