DAY 3 ASSIGNMENT

Video 1: History of JS

- **1990s Web Development**: Initially, web pages were static, created using HTML and CSS, with no interactivity.
- **Mosaic Browser (1993)**: Introduced for better viewing of web pages.
- **Netscape and Internet Explorer**: Netscape developed Netscape Navigator and attempted to use Java applets, but this was unsuccessful. Brendan Eich from Netscape created JavaScript (initially called Mocha, then LiveScript).
- **Microsoft's Response**: Microsoft reverse-engineered JavaScript, creating JScript for Internet Explorer, leading to browser compatibility issues.
- **Standardization**: In 1997, JavaScript was standardized by ECMA International, resulting in ECMAScript.
- **Netscape's Decline**: Netscape was acquired by AOL in 2000, leading to the development of Firefox from its open-source code.
- **Key Developments**: AJAX (2005) enabled single-page applications, jQuery (2008) simplified JavaScript, and Node.js (2009) allowed JavaScript for both front-end and back-end development.
- **ECMAScript Versions**: ES5 (2009) began modern JavaScript, and ES6 (2015) introduced significant new features with annual updates. ES12 (2021) is the current version.
- **Compatibility**: JavaScript is backward-compatible but not forward-compatible.

Video 2:Data Types in JavaScript-1

- **Data Importance**: Data is crucial in software for storage, processing, and deriving useful information.
- **Value and Type**: Every value in JavaScript has a type assigned to it, not to the variable.
- **Primitive Data Types**: Include number, string, boolean, null, undefined, and symbol.
- **Object Data Type**: The only non-primitive data type.
- **Numbers in JavaScript**: The maximum safe integer is 9007199254740991, and BigInt is used for larger numbers.
- **Floating-Point Representation**: Numbers can be represented in exponential notation, and underscores can be used for readability.
- **Infinity and NaN**: JavaScript supports Infinity, -Infinity, and NaN (Not a Number).
- **BigInt**: Allows handling of large integers by appending n to the number.

```
// Maximum Safe Integer
console.log(Number.MAX_SAFE_INTEGER); // Output: 9007199254740991

// BigInt
let bigNumber = 123456789012345678901234567890n;
console.log(bigNumber); // Output: 123456789012345678901234567890n

// Floating-Point Representation
let floatNumber = 1.23e4; // Exponential notation
```

```
console.log(floatNumber); // Output: 12300

// Underscores in Numbers
let readableNumber = 1_000_000; // Using underscores for readability
console.log(readableNumber); // Output: 1000000

// Infinity and -Infinity
console.log(7/0);    // Output: Infinity
console.log(-7/0);   // Output: -Infinity

// NaN (Not a Number)
console.log(0 / 0);       // Output: NaN
console.log(NaN + 1);     // Output: NaN
console.log(typeof NaN); // Output: number
```

Video 3: Data Types in JavaScript-2

- **String Literals**: Strings are represented using quotes, and escape characters are used for special characters.

```
//Example 1 : Using Double Quotes
let message = "Hello, World!";
console.log(message); // Output: Hello, World!

// Example 2 : Using Single Quotes
let quoste = 'It\'s a beautiful day!';
console.log(quoste); // Output: It's a beautiful day!

//Example 3 : Using Escape Characters
let text = "She said, \"Hello!\" and then left.\nIt's a new line.";
console.log(text);
// Output:
// She said, "Hello!" and then left.
// It's a new line.
```

- **Combining Strings**: Strings can be concatenated using the + operator.

```
// Example 1: Combining Two Strings
let firstName = "Kattekola";
let lastName = "Vidya";
let fullName = firstName + " " + lastName;
console.log(fullName); // Output: Kattekola Vidya
```

```
// Example 2: Adding Numbers as Strings
let age = "25";
let message = "Age: " + age;
console.log(message); // Output: Age: 25

// Example 3: Concatenating Strings with Other Data Types
let item = "apple";
let quantity = 5;
let statement = "I have " + quantity + " " + item + "s.";
console.log(statement); // Output: I have 5 apples.
```

- **Boolean**: Represents true or false values.

```
// Example 1: Basic Boolean Assignment
let isJavaScriptFun = false;
console.log(isJavaScriptFun); // Output: false

// Example 2: Boolean Expression Evaluation
let a = 10;
let b = 5;
let isGreater = a > b;
console.log(isGreater); // Output: true
```

- **Null and Undefined**: Null represents the absence of a value, while undefined indicates a declared variable with no value assigned.

```
// Example 1
let item;
console.log(item); // Output: undefined
// Example 2
let car = { make: "Toyota" };
console.log(car.model); // Output: undefined
//Example 3
let user = null;
console.log(user); // Output: null
```

- **NaN**: Represents a value that is not a valid number.

```
//Example 1
let result1 = "vidya" * 3;
console.log(result1); // Output: NaN
```

```
//Example 2
let result3 = "vidya" - 5;
console.log(result3); // Output: NaN
//Example 3
let result = 10 / "vidya";
console.log(result); // Output: NaN
```

Video 4: Type Conversion and Coercion

- **Type Conversion**: Converting data from one type to another, such as a string to a number or vice versa. Explicit conversion can be done using String(), Number(), etc.

```
// Example 1: Converting a number to a string
let num = 123;
let numToString = String(num);
console.log(numToString); // Output: "123"
console.log(typeof numToString); // Output: string

// Example 2: Converting a string to a number
let str = "456";
let strToNumber = Number(str);
console.log(strToNumber); // Output: 456
console.log(typeof strToNumber); // Output: number

// Example 3: Converting a boolean to a string
let bool = true;
let boolToString = String(bool);
console.log(boolToString); // Output: "true"
console.log(typeof boolToString); // Output: string
```

- **Type Coercion :** Occurs when JavaScript automatically converts one data type to another.

```
// Example 1: Adding a number to a string
let num = 5;
let str = " apples";
let result = num + str;
console.log(result); // Output: "5 apples"
console.log(typeof result); // Output: string

// Example 2: Subtracting a string from a number
```

```
let strNumber = "10";
let diff = 20 - strNumber;
console.log(diff); // Output: 10
console.log(typeof diff); // Output: number

// Example 3: Comparing a string to a number
let strNum = "7";
let isEqual = strNum == 7;
console.log(isEqual); // Output: true
console.log(typeof isEqual); // Output: boolean
```

Video 5: Arithmetic Operators

- **Basic Arithmetic Operations**: JavaScript supports addition, subtraction, multiplication, and division.

```
// Example 1: Addition
let num1 = 10;
let num2 = 5;
let sum = num1 + num2;
console.log(sum); // Output: 15

// Example 2: Subtraction
let num3 = 20;
let num4 = 7;
let difference = num3 - num4;
console.log(difference); // Output: 13

// Example 3: Multiplication
let num5 = 6;
let num6 = 4;
let product = num5 * num6;
console.log(product); // Output: 24
```

- **Increment and Decrement**: ++ and -- are used to increase or decrease a number by 1.

```
// Example 1: Post-Increment
let a = 5;
console.log(a++); // Output: 5 (value before increment)
console.log(a);   // Output: 6 (value after increment)
```

```
// Example 2: Pre-Increment
let b = 10;
console.log(++b); // Output: 11 (value after increment)
// Example 3: Post-Decrement
let c = 8;
console.log(c--); // Output: 8 (value before decrement)
console.log(c);   // Output: 7 (value after decrement)
```

- **Exponentiation**: The ** operator or Math.pow() function is used for exponentiation.

```
// Example 1: Using the ** Operator
let base1 = 2;
let exponent1 = 3;
let result1 = base1 ** exponent1;
console.log(result1); // Output: 8 (2 raised to the power of 3)
// Example 2: Using Math.pow() Function
let base2 = 5;
let exponent2 = 4;
let result2 = Math.pow(base2, exponent2);
console.log(result2); // Output: 625 (5 raised to the power of 4)
// Example 3: Using Math.pow() Function with Decimal Exponents
let base3 = 4;
let exponent3 = 0.5;
let result4 = Math.pow(base4, exponent4);
console.log(result4); // Output: 2 (Square root of 4)
```

Video 6 : Relational Operators

- **Comparison of numbers :** : Relational operators compare two values, resulting in a boolean output.
- **Comparison of Strings**: JavaScript compares strings lexicographically using ASCII values.
- **Comparison of Mixed Types**: JavaScript performs type coercion when comparing different types.
- **Strict Equality (===) vs. Loose Equality (==)**: === compares both value and type, while == performs type conversion.

```
//Example 1 : Comparison of Numbers
let num1 = 20;
let num2 = 15;
console.log(num1 > num2); // Output: true (20 is greater than 15)
```

```
//Example 2 :  Comparison of Strings
let str1 = "apple";
let str2 = "orange";
console.log(str1 < str2); // Output: true ("apple" comes before "orange" lexicographically)
// Example 3 : Comparison of Mixed Types
let str3 = "5";
let num3 = 5;
console.log(str3 == num3); // Output: true (string "5" is coerced to number 5)
//Example 4 : Strict Equality vs. Loose Equality
let num4 = 10;
let str4 = "10";
console.log(num4 === str4); // Output: false (number 10 is not strictly equal to string "10")
let num5 = 0;
let bool1 = false;
console.log(num5 == bool1); // Output: true (number 0 is loosely equal to boolean false)
```

Video 7: Logical Operators

- Logical operators are used to combine multiple comparisons or conditions in a single expression.
- **AND (&&), OR (||), NOT (!)**: Used to combine multiple conditions or invert boolean values.

A	B	A AND B	A OR B	NOT A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

```
//Example 1 : AND (&&) Operator
let a = 10;
let b = 20;
console.log(a < b && b < 30); // Output: true (both conditions are true)
//Example 2 :  OR (||) Operator
let x = 5;
let y = 15;
console.log(x > 10 || y > 10); // Output: true (y > 10 is true)
```

```
//Example 3 :  NOT (!) Operator
let isAvailable = false;
console.log(!isAvailable); // Output: true (NOT false is true)
```

Video 8: Terinary Operators

- **Ternary Operators**: The ternary operator (`? :`) provides a shorthand way to evaluate a condition and assign a value based on that condition.

  Syntax :

  condition ? value_if_true : value_if_false

  //Example 1:

```
let age = 18;
let access = (age >= 18) ? "Allowed" : "Not Allowed";
console.log(access); // Output: Allowed (age is 18, which is >= 18)
//Example 2:
let score = 75;
let grade = (score >= 60) ? "Pass" : "Fail";
console.log(grade); // Output: Pass (score is 75, which is >= 60)
//Example 3 :
let temperature = 30;
let weather = (temperature > 25) ? "Hot" : "Cool";
console.log(weather); // Output: Hot (temperature is 30, which is > 25)
```

Video 9: Template Literals

- Template literals provide an easier and more readable way to include variables and expressions inside strings. They allow for multiline strings and embedded expressions without the need for string concatenation.

  Syntax:

  - Use backticks ` instead of quotes " or ' .
  - Embed expressions inside `${}`

  //Example 1:

```
let name = "Vidya";
let age = 22;
let greeting = `Hello, my name is ${name} and I am ${age} years old.`;
```

```
console.log(greeting); // Output: Hello, my name is Vidya and I am 25 years old.
//Example 2 :
let num1 = 5;
let num2 = 10;
let sum = `The sum of ${num1} and ${num2} is ${num1 + num2}.`;
console.log(sum); // Output: The sum of 5 and 10 is 15.
//Example 3 :
let multiline = `This is a string
that spans multiple
lines.`;
console.log(multiline);
// Output:
// This is a string
// that spans multiple
// lines.
```