JavaScript

Chapter 1

Introduction to JavaScript

What is JavaScript?

JavaScript is a commonly used programming language for developing interactive and dynamic websites. It is a flexible programming language that can be used on both the client and server sides. From basic form validation to intricate web-based games, JavaScript is a must when developing interactive web applications.

History of JavaScript

In May 1995, Brendan Eich, a programmer at Netscape Communications Corporation, developed JavaScript in ten days. JavaScript was known by many names at first, including "Mocha" and "LiveScript," until Netscape decided on the name "JavaScript." JavaScript was first used in Netscape Navigator 2.0, which at the time was the most popular browser. Netscape presented JavaScript for standardization to the European Computer Manufacturers Association (ECMA) in 1996. The standardized version was known as ECMAScript.

Features of JavaScript

- **Interpreter:** JavaScript is an interpreted language, which means it is run immediately by a web browser without the need for any further compilation.
- **DOM Manipulation:** JavaScript has the ability to manipulate the Document Object Model (DOM), which describes the structure of a web page.
- **Event Handling:** JavaScript allows developers to construct event handlers to respond to different user interactions and events, such as button clicks and form submissions.

- **Case Sensitive:** JavaScript is a case-sensitive language.
- **Control Statements:** Javascript has control statements such as if-else-if, switch case, and loop. These control statements allow users to create complicated programs.
- **Dynamic Typing:** JavaScript is a dynamically typed language, which implies that when declaring variables, you do not need to declare data types.

How to add JavaScript in html document ?

The **<script>** tag is used to add JavaScript code in an HTML document.

Example:

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

JavaScript Comments

JavaScript comments include explanations or notes in the code that the browser does not execute or process. They serve only to help developers understand the code.

There are two types of comments in JavaScript:

- Single Line Comments
- Multi-line Comments

Single Line Comments

Single line comments start with `//`. Any information between `//` and the end of the line will be ignored by JavaScript.

Example:

```
let name = john;
// This is a single line comment
console.log(name);
```

Multi-line Comments

Multi-line comments start with `/*` and end with `*/`. Any information between `/* and */` will be ignored by JavaScript.

Example:

```
let name = john;
/* This is a
multi line
comment */
console.log(name);
```

Chapter 2

JavaScript Variables

Variables are used in JavaScript to store data.

Declaring Variables

To declare a variable in JavaScript use the var or let keyword followed by the name of the variable.

Example:

```
var x;
let y;
```

Variable Naming Rules

The rules for naming variables are as follows.

- Variable names must begin with either a letter, an underscore `(_)`, or the dollar symbol `($)`.
- Variable names cannot start with numbers.
- Variable names are case-sensitive.

Example:

```
//valid variable name
let name = john;
let _name = john;
let $name = john;

//invalid variable name
let 2name = john;
```

JavaScript Data Types

JavaScript provides various data types that you can use to store different types of data.

There are mainly two types of data types in JavaScript:

- Primitive data types.
- Non-primitive data types.

Primitive Data Types

There are six primitive data types in JavaScript:

- String
- Number
- Boolean
- Undefined
- Null
- Symbol

String

String is used to represents textual data, enclosed within single `(' ')` or double `(" ")` quotes.

Example:

```
// Double quotes
let name = "James";
// Single quotes
let name = 'John';
```

Number

Number represents integer and floating-point numbers.

Example:

```
let number1 = 24;
let number2 = 4.33;
```

Boolean

Boolean data types represent logical values that are either true or false.

Example:

```
let isTrue = true;
let isFalse = false;
```

Undefined

The undefined data type represents a value that has not been assigned.

Example:

```
let num;
console.log(num); // undefined
```

Symbol

The Symbol data type represents a unique identifier.

Example:

```
let val1 = Symbol("Hello");
let val2 = Symbol("Hello");
```

Non-primitive data types

There are mainly three types of non-primitive data types in JavaScript:

- Object

- Array

Object

JavaScript objects are separated by curly braces {}.

Example:

```
let student = {
   name: "John",
   age: 22,
   education: "3rd year"
};
```

Array

Arrays are written with square brackets. Array items are separated by commas.

Example:

```
let numbers = [20, 40, 60, 80]
console.log(numbers[0]);
```

Chapter 3
JavaScript Operators

An operator is a special symbol in JavaScript that performs operations on operands.

There are several types of operators in JavaScript, including:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators

Arithmetic Operators

Arithmetic Operators are used to perform mathematical operations

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Example:

```
let x = 6;
let y = 4;

console.log('x + y = ', x + y);
console.log('x - y = ', x - y);
console.log('x * y = ', x * y);
console.log('x / y = ', x / y);
```

Output:

```
10
2
24
1.5
```

Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Description	Example
=	Assignment operator	a = 5;
+=	Addition assignment	a += 5;
-=	Subtraction Assignment	a -= 3;
*=	Multiplication Assignment	a *= 2;
/=	Division Assignment	a /= 2;
%=	Modulus Assignment	a %= 2;

Example:

```
let x = 6;
```

Comparison Operators

Comparison operators are used to compare two values.

Operator	Description
==	Equal
!=	Not Equal
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Example:

```
let x = 5;
let y = 5;

console.log(x == y);
console.log(x != y);
console.log(x > y);
console.log(x >= y);
console.log(x < y);
console.log(x <= y);
```

Output:

```
true
false
false
true
false
true
```

Logical Operators

Logical operators perform logical operations and return a boolean value.

Operator	Description	Example
&&	Logical AND	x && y
||	Logical OR	x || y
!	Logical NOT	!x

Example:

```
let x = 5;
let y = 5;

console.log(x && y);
console.log(x || y);
```

Output:

```
5
5
```

Bitwise Operators

Bitwise operators are used to deal with binary operations.

Operator	Description
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
<<	Left shift

Example:

```
let x = 10;
let y = 12;

result = x & y;
console.log(result);
```

Output:

```
8
```

Chapter 4

JavaScript Control Flow

if...else Statement

In JavaScript, the if..else statement is used to make decisions. It allows you to execute a block of code if a specified condition is true and another block of code if the condition is false.

There are three types of if-else statements in JavaScript:

- if statement
- if...else statement
- if...else ladder statement

If Statement

The if statement is used to execute a block of code if a given condition is true.

Syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Example:

```
let x = 8;
if (x > 5) {
  console.log("x is greater than 5");
}
```

Output:

```
x is greater than 5
```

If...else statement

The If...else statement is used to execute a block of code if a specified condition is true and another block of code if the condition is false.

Syntax:

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

Example:

```
let x = 8;
if (x > 5) {
  console.log("x is greater than 5");
} else {
  console.log("x is not greater than 5");
}
```

Output:

```
x is greater than 5
```

If else ladder

The "if-else ladder" is a control structure in JavaScript that allows you to execute a different block of code depending on multiple conditions.

Syntax:

```
if (condition) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and
condition2 is true
} else {
  // block of code to be executed if the condition1 is false and
condition2 is false
}
```

Example:

```
let x = 10;
if (x > 15) {
  console.log("x is greater than 15");
} else if (x > 10) {
  console.log("x is greater than 10 but less than or equal to 15");
} else {
  console.log("x is equal to 10");
}
```

Output:

```
x is equal to 10
```

Switch Statement

The "switch" statement in JavaScript is another control structure that allows you to execute a different block of code depending on a specific value.

Syntax:

```
switch (expression) {
  case value1:
    // block of code
    break;
  case value2:
    // block of code
    break;
  case value3:
    // block of code
    break;
  default:
    // block of code
}
```

Example:

```
let x = 3;
switch (x) {
  case 1:
    console.log("I am Statement 1");
    break;
  case 2:
    console.log("I am Statement 2");
    break;
  case 3:
    console.log("I am Statement 3");
    break;
  default:
    console.log("I am default");
}
```

Output:

```
I am Statement 3
```

For Loop

A for loop In JavaScript is used to execute a piece of code a specified number of times.

Syntax:

```
for (initialization; testExpression; increment/decrement) {
// block of code
}
```

Example:

```
let x = 5;
for (let i = 1; i <= 5; i++) {
   console.log("Hello JavaScript");
}
```

Output:

```
Hello JavaScript
Hello JavaScript
Hello JavaScript
Hello JavaScript
Hello JavaScript
```

For-of loop

The for-of loop is used to iterate over the values of an iterable object, such as an array or a string.

Syntax:

```
for (variable of object) {

  // code to be executed

}
```

Example:

```
let numbers = [1, 2, 3, 4, 5];


for (let number of numbers) {

  console.log(number);

}
```

Output:

```
1

2

3

4

5
```

While Loop

The while loop in JavaScript is used to execute a block of code as long as a specified condition is true.

Syntax:

```
while (condition) {
  // block of code
}
```

Example:

```
let i = 1;

while (i <= 8) {
  console.log(i);
  i++;
}
```

Output:

```
1
2
3
4
5
6
7
8
```

Chapter 5

JAVASCRIPT Types

Strings

A string in JavaScript is a primitive data type that represents a sequence of characters. It is enclosed by single quotes (`' '`), double quotes (`" "`), or backticks (`' '`).

Example:

```
let name1 = "Peter";
let name2 = 'Perker';
let greeting = `Hello ${name}!`;
```

String Methods

JavaScript has several built-in methods for manipulating strings.

length

The length method returns the length of a string.

Example:

```
let str = "Hello JavaScript";
console.log(str.length);
```

Output:

```
16
```

concat

The `concat()` method method is used to concatenate two or more strings.

Example:

```
let str1 = "Hello";
let str2 = " JavaScript";
console.log(str1.concat(str2));
```

Output:

```
Hello JavaScript
```

indexOf

The `indexOf()` method is used to find the index of a specific character in a string.

Example:

```
let str = "Hello JavaScript";
console.log(str.indexOf("J"));
```

Output:

```
6
```

toUpperCase and toLowerCase

The `toUpperCase()` and `toLowerCase()` methods are used to convert a string to uppercase or lowercase letters.

Example:

```
let str = "Hello JavaScript";
console.log(str.toUpperCase());
console.log(str.toLowerCase());
```

Output:

```
HELLO JAVASCRIPT
hello javascript
```

JavaScript Number

Numbers are a fundamental data type in JavaScript that represents numerical values. They can be either integer or floating-point numbers.

Example:

```
let x = 4;
let y = 4.13;
```

JavaScript Number Methods

There are various methods available in JavaScript Date object.

Method	Description
isNaN()	It determines whether the provided value is NaN.
isInteger()	It determines whether the provided value is an integer.
parseFloat()	It parses a string argument and returns a floating-point number.
parseInt()	It parses a string argument and returns an integer number.
toFixed()	Formats a number using fixed-point notation.

toString()	It converts a number to a string.

Example:

```
let x = 10;
console.log(Number.isInteger(a));

let y = NaN;
console.log(Number.isNaN(b));

let c = parseInt('05');
console.log(c);
```

Output:

```
true
true
6
```

JavaScript Arrays

Arrays in JavaScript are a fundamental data structure used to store multiple values in a single variable.

Example:

```
let cars = ["Thar", "Scorpio", "BMW", "Ferrari"];
```

Array Methods

JavaScript has many built-in methods for manipulating arrays.

length

The length method returns the number of elements in an array.

Example:

```
let myArr = ["Thar", "Scorpio", "Ferrari"];
console.log(myArr.length);
```

Output:

```
3
```

push

The `push()` method is used to add an element to the end of an array.

Example:

```
let myArr = ["Thar", "Scorpio", "Ferrari"];
myArr.push("Audi");
console.log(myArr);
```

Output:

```
[ 'Thar', 'Scorpio', 'Ferrari', 'Audi' ]
```

pop

The `pop()` method is used to remove the last element of an array.

Example:

```
let myArr = ["Thar", "Scorpio", "Ferrari"];
myArr.pop();
console.log(myArr);
```

Output:

```
[ 'Thar', 'Scorpio' ]
```

JavaScript Date

The `Date` object in JavaScript is used to interact with dates and times. You can create a new `Date` object to represent either the current date and time or a specific date and time.

JavaScript Date Methods

There are various methods available in JavaScript Date object.

Method	Description
getDate()	Gets the day of the month (1-31) according to local time
getFullYear()	Gets the year according to local time
getMonth()	Gets the month, from 0 to 11 according to local time
setDate()	Sets the day of the month according to local time
setFullYear()	Sets the full year according to local time
setMonth()	Sets the month according to local time

Example:

```
let date = new Date();
let day = date.getDate();
console.log(day);

let date = new Date();
let year = date.getFullYear();
console.log(year);

let date = new Date();
let month = date.getMonth()+1;
console.log(month);

let date = new Date();
date.setDate(26);
let D = date.getDate();
console.log(D);
```

Output:

```
25
2024
3
26
```

JavaScript Booleans

A boolean is a data type in JavaScript representing either true or false.

Example:

```
let isTrue = true;
let isFalse = false;
```

JavaScript Boolean Methods

Here is a list of boolean methods in JavaScript.

Method	Description
toString()	Converts Boolean into String.
valueOf()	Returns the value of a boolean.

Example:

```
// toString Method
let bool = new Boolean(10);
console.log(bool.toString());

// valueOf Method
let bool = new Boolean(true);
console.log(bool.valueOf());
```

Output:

```
true
true
```

JavaScript Map

JavaScript Maps is a data structure in JavaScript that was introduced in ECMAScript 6 (ES6). They contain key-value pairs.

Example:

```
let map1 = new Map();
console.log(map1); // Map {}
```

JavaScript Map Methods

There are various methods available in JavaScript Map object.

Method	Description
set()	Adds or updates a key-value pair in the Map.
get()	Returns the value associated with the specified key.
delete()	Removes the specified element from a Map object.
has()	Checks if the Map contains a specific key.
forEach()	Calls a function for each key/value pair in a Map
entries()	Returns an iterator with the key-value pairs in a Map

set() Method

To add elements to a Map, use the `set()` method:

Example:

```
let myMap = new Map();

myMap.set('name', 'John');
myMap.set('age', 22);

console.log(myMap.get('name'));
```

Output:

```
John
```

get() Method

The `get()` method gets the value of a key in a Map.

Example:

```
let myMap = new Map();

myMap.set(1, 'JavaScript');

console.log(myMap.get(1));
```

Output:

```
JavaScript
```

Chapter 6

JavaScript Objects

JavaScript object is a variable that can store multiple data in key-value pairs.

Example:

```
const student = {

    firstName: "Jack",

    rollNo: 32

};

console.log(student);
```

Output:

```
{ firstName: 'Jack', rollNo: 32 }
```

JavaScript Object Methods

Object methods are actions that can be performed on objects.

Example:

```
const person = {

    name: "Bob",

    age: 30,

    greet: function () {

        console.log("Bob says Hi!");

    }

};

person.greet();
```

Output:

```
Bob says Hi!
```