Page No :
Date : / /

JavaScript

COMPLETE NOTES

Prepared By : Topperworld

Follow us :

website : Topperworld.in
Linkedin : Topperworld
Instagram : Topperworld.in

Page No :.....
Date :.../.../....

JavaScript

INDEX

S.No	Title of Topic	Page No
1.	Introduction to Javascript - History of Javascript - Purpose of Javascript - Setting up the development environment	1-4
2.	Basic of Javascript - Variables and Datatypes - Operators and Expressions - Control structure	4-7
3.	Loops - for loop - while loop - Dowhile loop	7-8
4.	function and Scope - Defining functions - function parameters and return values - Scope and Closures	9-12

5.	Javascript Statement - If - If ... elseif - If else - Switch	12 - 16
6.	Arrays and Objects - working with Arrays - Creating and manipulating object - Arrays and object method	16 - 20
7.	DOM manipulation - Introduction to DOM - Accessing and Modifying html elements - Handling Object	20 - 23
8.	Asynchronous Javascript - Introduction to asynchronous programing - Callback functions - Promises and async/await	23 - 26
9.	Error Handling - Handling exceptions and errors - Handling Exceptions - Debugging Techniques	27 - 29

10.	ES6 and Beyond - New features of ES6 → let and const → Arrow function → classes → Template literals → Destructring Assignment → Spread and Rest operators → promises → Modules → Default parameters → Symbol → Iterator and Generator → Async / await	29-33
11.	Project Work	33-35
12.	working with dates and times → creating and manipulating date objects. → formatting and displaying dates → performing date calculations and comparisons. → working with time zones.	36-40
13.	AJAX and fetch API → Introduction to asynchronous programing. → Making HTTP requests with fetch API	40-43

→ Handling responces and data manipulation.
→ working with JSON data

14. Introduction to Javascript frame-work and Libraries. 43-48
→ Overview of popular frame works.
→ Introduction to Libraries
→ Using framework and library for building interactive web application
→ Pros and Cons of using Javascript framework.

Introduction to Javascript

- JavaScript is a high-level, dynamic and versatile programming language primarily used for web development.

- It is an Interperted, full-fledged programming language that enables interactivity on websites when applied to an an HTML document.

- It was created by Bredan Eich in 1995 while he was working at Netscape Communications Corporation. Initially, it was named "Mocha" and later "Live Script" before finally being called "JavaScript".

History of JavaScript

In 1993, Mosaic, the first popular web browser, came into existence. In the year 1994, Netscape was founded by Marc Andreessen. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part time programmers. Consequently, in 1995, the company recruited Breden Eich intending to implement and embed Scheme programming language to the

browser. But, before Brendan could start, the company merged with Sun Microsystems for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platform. Now, two languages were there: Java and the Scripting language.

furthur, Netscape decided to give a similar name to the scripting language as Java's. It led to 'Javascript'. Finally, in May 1995, Marc Andreessen coined the first code of Javascript named 'Mocha'. Later the marketing team replaced the name with 'LiveScript'. But due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript' from then, JavaScript came into existence.

Purpose of JavaScript

- JavaScript's primary purpose is to enable client side scripting on web pages. This mean it allows developers to manipulate the content and behaviour of web pages directly with in the users web browser.

- With JavaScript, we can dynamically update web pages elements, validate from input's and respond to user interactions like clicks, mouse movements, and Keyboard input.

• Additionally, Javascript can interact with web servers AJAX (Asynchronous JavaScript and XML) to fetch data without requiring a page reload.

Setting-up the development environment

1. Text Editor

To start coding in Javascript, we need a development environment set up

Here are the basic

Choose a text editor or an Integrated Development Environment (IDE) that supports JavaScript syntax highlighting Examples include Visual Studio code, Sublime text, or Atom.

2. Web browser

Java Script runs in Web browsers, So we need a modern browser like chrome, firefox or Edge to excute our JavaScript code.

3. HTML file

Create a new HTML file that will Serve as the Container for our Javascript code. This will include our HTML structure and link to the Javascript

code using the script tag

4. JavaScript Code

Write the JavaScript code within the script tags in the HTML file or in a separate, JS file, which we link to our HTML file.

5. Testing

Open the HTML file in our web browser to see the result of our Javascript code.

As we progress in our Javascript learning journey, we may explore using more advanced development tools frame works, and libraries to stream our workflow and build complex applications

Basics of JavaScript

JavaScript, being a fundamental programming language, has several essential concepts that form its core.

Let's dive into the basics of JavaScript, including variables and data types, operators and expressions, and control structurs -

Variables and Data Types

Variables are containers used to store data values in JavaScript. We can declare variables using the 'var', 'let' or 'Const' keyword.

for Example


```
Var age = 30;
let name = 'John';
Const PI = 3.14;
```

- JavaScript has various data types, Including:

- Primitive data Types:
numbers, strings, boolean, null, undefined, and symbols.

- Complex data types
objects (arrays, functions, and Object themselves)

Operators and Expressions

Operators are symbols used to perform operations an variables and values.

- JavaScript supports various types of operators such as arithmetic assignment, comparision

logical etc

Examples

```
Var x = 10;
Var y = 5;

Var Sum = x + y;
Var difference = x - y;

Var is True = x > y;
```

- Expressions are combinations of Variables and values, and operators that evaluate to a single value

for Example;

```
Var result = (x+y) * 2;
```

Control structures

Control statements or structures allows us to control the flaw of our code, making decisions or repeating actions based on conditions

- If else

It allows us to execute a block of code if a

certain condition is true or another block of code if the condition is false

Example

```
Var age = 18;
```



```
if (age >= 18) {
  Console.log ("You can vote");
}
  else {
    console.log ("You can't vote");
```

- **Loops**

for Loop: It allows us to execute a block of data / code repeatedly based on a specified condition

Example

```
for (Var i=1 ; i<=5 ; i++) {
    console.log ("Iteration : " +i);
}
```

While loop: It executes a block of code as long as a specified condition is true.

Example :

```
var Count = 1;

while ( Count <= 5) {

    console.log (" Count: " + count);

    count ++ ;

}
```

do while loops

Similar to a while loop, but it executes the code at least once before checking the condition.

```
Var num = 1;

do {

        console . log (" Number ; " + num);

        num ++ ;

}

while (num <= 5);
```

3. Functions and Scope

Defining functions

In JavaScript, a function is a reusable block of code that performs a specific task. It allows us to encapsulate logic and execute it whenever needed.

functions are defined using the 'function' keyword, followed by a name, a set of parentheses '()' and curly braces '{ }' containing the function's body.

Example

```
function greet() {
    console.log("Hello!");
}
```

function Parameters and Return values

Parameters are placeholders for values that we can pass to a function when calling it. They enable us to customize the behaviour of the function dynamically Parameters are defined inside the parenthesses of the function declration

Example :

```
function greet (name) {
    console.log ("Hello," + name + "!");
}
```

Return values ; Functions can also return values using the 'return' statement . The returned value can then be used in other parts of the code.

Example


```
function add (a, b) {
    return a+b;
}
```

Scopes and closures :

Scope refers to the context in which variables and functions are accessible in our code . In JavaScript there are two main types of scope :

- **GLOBAL SCOPE**

Variables declared outside any function have global scope can be accessed from anywhere in the code.

LOCAL SCOPE

Variables declared inside a function have local scope and are only accessible within that function.

Example of local Scope

```
function myfunction() {
    var x = 10;
    console.log(x);
}

console.log(x);
```

Closures: A closure is a powerful feature in JavaScript that allow a function to remember and access its lexical scope even when it's executed outside that scope. This means a function can retain access to its parent function's variables even after the parent function has finished executing.

Example of a closure:

```
function outerfunction() {
    var outerVariable = "I am from the outer
                         function";

  function innerFunction() {
```

```
    console.log (outerVariable);
}
    return innerfunction;
}

var closureExample = OuterFunction ();
closureExample ();
```

JavaScript Statement

JavaScript if statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if (expression) {
// statement
}
```


Ex:-

```
<Script>
Var a = 20;
if (a>10) {
document. write (" Value of a is greater than 10");
}
</Script>
```

Output: Value of a is greater than 10

JavaScript if ... else if Statement

It evaluates the content only if expression is true from several expressions. The Signature of Javascript if else if statement is given below.

```
If (expression 1) {
// Content to be evaluated if expression 1 is true
}
 else if (expression 2) {
// Content to be evaluated if expression 2 is true
 }
 else if ( expression 3) {
// Content to be evaluated if expression 3 is true
}
 else {
// Content to be evaluated if no expression is true
}
```

Example :-

```
 < Script>
Var a= 20;
If (a = = 10) {
document. write ("a is equal to 10");
}
 else if (a==15) {
  document. write (" a is equal to 15");
 }
  else if (a = = 20) {
  document . write ("a is equal to 20");
```

```
}
 else {
document.write("a is not equal to 10, 15 or 20");
}
</script>
```


Output: a is equal to 20

JavaScript if... else statement

It evaluates the content whether condition is true of false. The syntax of Javascript if- else statement is given below

```
if (expression) {
// Content to be evaluated if condition is true
 }
 else {
// Content to be evaluated if condition is fals.
 }
```

Example :-

```
var a = 20;
 if (a%. 2==0){
document. write ("a is even number");
 }
   else {
document. write ("a is odd number");
 }
 </Script>
```

Output : a is even number

JavaScript Switch

The JavaScript Switch statement is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if-- else-- if because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below

```
Switch (expression) {
case valve 1:
code to be executed;
break;
Case valve 2:
code to be executed;
break;

.....

default:

code to be executed if above values are not matched;

}
```

Example

```
< Script >
Var grade = 'B';
Var result ;
Switch (grade) {
 Case 'A' ;
result = "A Grade" ;
break ;
 Case 'B' ;
result = "B Grade" ;
break ;
Case 'C' ;
result = " C Grade" ;
break
default :
result = " No Grade" ;
}
document. write (result);
</ Script >
```


Output: B grade

Arrays and Objects

Working with Arrays

Array are data structures in JavaScript used to store collections of elements. They allow us to group multiple values into a single variable.

Arrays are created using square brackets '[]' and each element is separated by comma.

Example :

```
Var fruits = ['apple', 'banana', 'orange'];
```

- Accessing elements : We can access individual elements in an array using their index, which starts from 0.

Example :

```
Var first fruit = fruit[0];
```

- Modifying elements : We can modify elements in an array by assigning new values to their corresponding indices

Example :

```
fruits [1] = 'grape';
```

- Array methods : JavaScript provides various built in methods to manipulate arrays, such as 'push()', 'pop()', 'Shift()', unshift()', Splice()' and more.

Creating and manipulating objects

Objects are collections of key-value pairs. Each

Key is a property that represents a name or identifier and each value is the associated data.

Objects are enclosed in curly brases '{ }'

Example :


```
Var person = {
        name : 'John',
        age : 30,
        city : 'New York'
};
```

- Accessing Object properties : We can access properties of an object using dot notation or square brackets.

Example :

```
Var personName = person.name;
Var personAge = person['age'];
```

- Modifying object properties

We can modify object properties by assigning new values to them

Example :

```
var student = {
        name : 'Alice',
        age  : 25
        address : {  :
        Street  : '123 Main ST',
        city    : 'Los Angeles'
      }

};
```

Array and object methods

Arrays and objects have built In methods that allow us to perform various operations efficiently.

Some common methods include:

- **Array methods**

'push ( )' , 'pop' ( ) , 'Shift' ( ) ,
'unshift ( )' , 'Splice ( )', ' concat ( )',
'Slice ( )', 'index of ( )', 'for Each ( )',
'map ( )', 'filter ( )' , and more

- **object methods**

' Object . Keys ( )'
' object . values ( )',
'Object . entries ( ) ;

"UNLOCK YOUR POTENTIAL"

With- TOPPERWORLD

Explore More

topperworld.in

DSA TUTORIAL

C TUTORIAL

C++ TUTORIAL

JAVA TUTORIAL

PYTHON TUTORIAL

Follow Us On

E-mail

topperworld.in@gmail.com