JavaScript

**How can use JavaScript in code?**

1. Javascript code use between <script> ….. </script> tag.

Example:<script> document.getElementById("main").innerHTML = "JavaScript"; </script>

\*\* Old JavaScript examples may use a type attribute: <script type="text/javascript">. The type attribute is not required. JavaScript is the default scripting language in HTML.

2. You can place any number of scripts in an HTML document.

3. Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

4. for using external script file, file name use in src (source) attribute of a <script> tag:

<script src="myScript.js"></scrip

\*\* External script files cannot contain <script> tags.

**JavaScript Output**

1. Writing into an HTML element, using innerHTML.
2. Writing into the HTML output using document.write().

\*\* Using document.write() after an HTML document is loaded, will delete all existing HTML:

1. Writing into an alert box, using window.alert().
2. Writing into the browser console, using console.log().

**JavaScript Statements**

1. JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

2. Semicolons separate JavaScript statements.

3. JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

**JavaScript Values**

The JavaScript syntax defines two types of values:

1. Fixed values (Literal)
2. Variable values (Variables)

\*\* Numbers are written with or without decimals:

\*\* Strings are text, written within double or single quotes

\*\* All JavaScript identifiers are case sensitive.

\*\* Hyphens are not allowed in JavaScript. They are reserved for subtractions.

**4 Ways to Declare a JavaScript Variable:**

1. Using var
2. Using let
3. Using const
4. Using nothing

**Let**

1. Variables defined with let cannot be Redeclared in same block.
2. Variables defined with let must be Declared before use.
3. Variables defined with let have Block Scope.

**Const**

1. Variables defined with const cannot be Redeclared.
2. Variables defined with const cannot be Reassigned.
3. Variables defined with const have Block Sco
4. JavaScript const variables must be assigned a value when they are declared:

Use const when you declare:

1. A new Array
2. A new Object
3. A new Function
4. A new RegExp

The keyword const is a little misleading. It does not define a constant value. It defines a constant reference to a value.

Because of this you can NOT:

1. Reassign a constant value
2. Reassign a constant array
3. Reassign a constant object

But you CAN:

1. Change the elements of constant array
2. Change the properties of constant object

**Data Types**

1. Number
2. String
3. Boolean
4. Object
5. Undefined

\*\* JavaScript has dynamic types. This means that the same variable can be used to hold different data

**Strings**

1. You can use single or double quotes
2. You can use quotes inside a string, as long as they don't match the quotes surrounding the string

Comparing two JavaScript objects **always** returns **false**.

Code Result Description

\' ' Single quote

\" " Double quote

\\ \ Backslash

Code Result

\b Backspace

\f Form Feed

\n New Line

\r Carriage Return

\t Horizontal Tabulator

\v Vertical Tabulator

\*\* Do not create Strings objects.

**String Methods**

1. slice(start, end) : slice() extracts a part of a string and returns the extracted part in a new string.
2. substring(start, end): substring() is similar to slice().
3. substr(start, length)
4. replace()The replace() method replaces a specified value with another value in a string.
5. replace() method is case sensitive
6. toUpperCase():
7. toLowerCase():
8. concat()
9. concat() joins two or more strings:
10. trim()
11. padEnd()
12. charAt()
13. split()

\*\*All string methods return a new string. They don't modify the original string.

**String Search**

1. String indexOf()
2. String lastIndexOf()
3. String startsWith()
4. String endsWith()

## **Number Methods**

toString()

toExponential()

toFixed()

## **toPrecision()**

## **valueOf()**

## **Converting Variables to Numbers**

* The Number() method
* The parseInt() method
* The parseFloat() method

If first string is number then it return otherwise NaN return .

## **Number Properties**

MAX\_VALUE

MIN\_VALUE

POSITIVE\_INFINITY

NEGATIVE\_INFINITY

NaN

**JSON**

1. JSON is a format for storing and transporting data.
2. JSON is often used when data is sent from a server to a web page.
3. JSON stands for JavaScript Object Notation
4. JSON is a lightweight data interchange format
5. JSON is language independent \*
6. JSON is "self-describing" and easy to understand

**JSON Example:**

{

"employees": [

 {"firstName":"John", "lastName":"Doe"},

{"firstName":"Anna", "lastName":"Smith"},

{"firstName":"Peter", "lastName":"Jones"}

]

}

**JSON Syntax Rules**

1. Data is in name/value pairs
2. Data is separated by commas
3. Curly braces hold objects
4. Square brackets hold arrays

**JSON strings into JavaScript objects:**

Use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

let text = '{ "employees" : [' +

'{ "firstName":"John" , "lastName":"Doe" },' +

'{ "firstName":"Anna" , "lastName":"Smith" },' +

'{ "firstName":"Peter" , "lastName":"Jones" }

]

}';

const obj = JSON.parse(text);

**converting an object into a JSON string:**

Use the JavaScript built-in function JSON.stringify() to convert object into a JSON string

**Objects**

1. All JavaScript values, except primitives, are objects.
2. Primitive values are immutable (they are hardcoded and cannot be changed)
3. Object values are written as name : value pairs (name and value separated by a colon).
4. Objects are mutable: They are addressed by reference, not by value.

**There are different ways to create new objects:**

1. Create a single object, using an object literal.

const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

1. Create a single object, with the keyword new.

const person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

1. Define an object constructor, and then create objects of the constructed type.
2. Create an object using Object.create().

**Object Properties**

1. Properties are the values associated with a JavaScript object.
2. A JavaScript object is a collection of unordered properties.
3. Properties can usually be changed, added, and deleted, but some are read only.

**The syntax for accessing the property of an object is:**

const person = {

name : “shuvo”,

age : 25

}

1. objectName.property

person.name

1. objectName["property"]

person[“name”]

1. objectName[expression]

int a = ‘age’

person[a];

## **Adding New Properties**

person.nationality = "English";

## **Deleting Properties**

delete person.age;

The delete keyword deletes both the value of the property and the property itself.

After deletion, the property cannot be used before it is added back again.

The delete operator is designed to be used on object properties. It has no effect on variables or functions.

The delete operator should not be used on predefined JavaScript object properties. It can crash your application.

## **Nested Arrays and Objects**

const myObj = {  
  name: "John",  
  age: 30,  
  cars: [  
    {name:"Ford", models:["Fiesta", "Focus", "Mustang"]},  
    {name:"BMW", models:["320", "X3", "X5"]},  
    {name:"Fiat", models:["500", "Panda"]}  
  ]  
}

REG

A regular expression is a sequence of characters that forms a **search pattern**.

/pattern/modifiers;

Flag of REG:

I =   
Perform case-insensitive matching

G =   
Perform a global match (find all matches rather than stopping after the first match)

M = Perform multiline matching

[abc] = Find any of the characters between the brackets