**JavaScript - Assignment**

**JavaScript Introduction**

**Question 1**: What is JavaScript? Explain the role of JavaScript in web development.

**Answers 1:** JavaScript is synchronous language. JavaScript is client said, and server said scripting language. JavaScript is one type of interpreted programming language.

**The role of JavaScript in web development**

* Create Dynamics website,
* event handling
* client said validation
* DOM manipulation
* Logic building for a task on webpage
* Control the interaction with the server.
* Plug-ins Building

**Question 2**: How is JavaScript different from other programming languages like Python or java?

**Answers 2:** There are many differences between the languages and different situations that might Favour the choice of one over the other.

|  |  |
| --- | --- |
| **JavaScript** | **Python and java** |
| JavaScript is a scripting language used in the front-end development of websites and web applications. | Python and java is an object-oriented language that can be utilized for a variety of purposes: back-end web development, machine learning, data analysis, etc. |
| JavaScript runs in any browser | Python and java are **cross-platforms** and run on almost any operating system (Windows, macOS, Linux). |
| Limited set of utility | Comprehensive standard library |
| Prototype inheritance models | Class based internation model |
| Language of the web browser and easiest | More conservative programming paradigm like C , C++ |
|  |  |

|  |
| --- |
|  |

**Question 3:** Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

**Answers 3:** The **<script>** [HTML](https://developer.mozilla.org/en-US/docs/Web/HTML) element is used to embed executable code or data; this is typically used to embed or refer to JavaScript code.

For example: through the **<script >** tag you can add JavaScript to your HTML document

Script defined different attributes like;

* **Src :** It is used to specify the URL of an external script file.
* **Async:** It is used to specify the script is executed asynchronously
* **Defer:**
* **Type:** It is used to specify the media type of the script.

**How can you link in external JavaScript file to an HTML document**

<!DOCTYPE html>

<html>

<head>

<title>document </title>

</head>

<body>

<! -- HTML script Tag Starts Here -->

<script src= “main.js”> </script>

<! -- HTML Script Tag Ends Here -->

</body>

</html>

**Main.js**

A=5

B=10

Document. Write(“<br> This is my first JavaScript” +(a+b))

**Output:**

This is my first JavaScript 15

**Variables and Data Types**

**Question 1:** What are variables in JavaScript? How do you declare a variable using var, let, and const?

**Answers: 1** Variables are used to store and manage data that our JavaScript code needs to work with, such as numbers, strings, objects, and more.

You declare a variable using one of the following keywords:

* Let
* Var
* Const

**Let: The** let variable is not allowed to redecorate variables but allowing you to reassign values.

**Let Example:**

Let name = “nagma”

Document. Write(“<br> Name is”+name)

name = 1000

**Out put**

Name is nagma

1000

**Var:** The var variable is allowed to redeclare and reassign variables.

**Var example:**

Var a = 10

document. Write (“A is ”+a)

Var b = 20

document. Write (“B is”+b)

Var c = 30

document. Write (“C is”+c)

**Output:**

A is 10

B is 20

C is 30

**Const:** The var variable is not allowed to redeclare and reassign variables. Only constant value is allowed

**For example:**

Const pi = 3.14

Document. Write (“<br> pi is ”+pi)

**Output:**

Pi is 3.14

**Question 2:** Explain the different data types in JavaScript. Provide examples for each.

**Answers 2:** data type has two main type: primitive and non-primitive data type

JavaScript define 8 different type of data types like:

* String
* Number
* Bigint
* Boolean
* Null
* Symbol
* Object

String: A String in JavaScript is a series of characters that are surrounded by quotes. There are three types of quotes in JavaScript, which are: (“”) (‘’)

Example:

Var a = “ motha nagma “

Document. write(“<br> this is my string data type ”+a)

Document. write(“<by>” typeof a)

**Output**

This is my string data type motha nagma

string

Number: The Number data type contains both integer and floating-point numbers. JavaScript does not distinguish between integers and floats.

Examples:

Var a = 10 //integer

Var b = 12.2 // floating \_ point

Document. write(“<br> this is my number data type”+a)

Document. write(“<br> this is my number data type”+a)

Document. write(“<by>” typeof a)

**Output**

This is my number data type 10

This is my number data type 12.2

Number

**Boolean:** The Boolean type has only two values i.e. **true** and **false**.

**Example:** Var a = true

Var b = false

Document. write(“<br> this is my Boolean data type”+a)

Document. write(“<br> this is my Boolean data type”+b)

Document. write(“<by>” typeof a,)

Document. write(“<by>” typeof b,)

**Output:**

this is my Boolean data type true

this is my Boolean data type false

Boolean

Null: The special null value does not belong to any of the default data types. It forms a separate type of its own which contains only the null value.

**Example:**

Var a = null

Document. write(“<br>” +a)

Document. write(“<by>” typeof a)

**Output:**

Null

Null

**Undefined:** A variable that has been declared but not initialized with a value is automatically assigned the undefined value.

**Example:** Var a = true

Var b;

Document. write(“<br>” +a)

Document. write(“<by>” typeof a)

**Output:**

Undefined

### **Symbol:**

A Symbol is a unique and immutable primitive value that is used as an identifier for object properties. Symbols are primarily used for creating unique property keys in objects, ensuring there are no conflicts with other properties.

**Example:**

Var a = Symbol("Geeks");

Var b = Symbol("Geeks");

console.log (a == b);

Document. write(“<by>” typeof a)

**Output:**

False

**BigInt :** BigInt is a built-in object that provides a way to represent whole numbers greater than 253. The largest number that JavaScript can reliably represent with the Number primitive is 253

**Example:**

let b = BigInt("0b1010101001010101001111111111111111");

console.log(b);

**Output:**

11430854655n

**Object:** JavaScript objects are fundamental data structures used to store collections of data. They consist of key-value pairs and can be created using curly braces {} or the new keyword.

**Example:**

Let a = {

code

}

**Arrays:** An Array is a special kind of object used to store an ordered collection of values, which can be of any data type.

**Example:**

let a = [1, 2, 3,4,5];

Documents. Write (“<br>” +a)

**Output:**

1

2

3

4

5

**Question 3:** What is the difference between undefined and null in JavaScript?

|  |  |
| --- | --- |
| **Undefined** | **null** |
| Undefined means the variable has been declared but has not yet been assigned a value | Null means the value has been set to be empty |
| For example:  let user  Console .log(user) | For example:  Let a = null  Console .log(a) |
| It is datatype itself | It is an object |
| Is a value automatically assigned by JavaScript when variable is declared but not assigned any value | Is a value that must be explicitly assigned by programmers |
| The null value is a primitive value which represents the null, empty, or non-existent reference. | The undefined value is a primitive value, which is used when a variable has not been assigned a value |
| Null indicates the absence of a value for a variable. | Undefined indicates the absence of the variable itself. |
| Null is converted to zero (0) while performing primitive operations. | Undefined is converted to NaN while performing primitive operations |

**JavaScript Operators**

**Question 1:** What are the different types of operators in JavaScript? Explain with examples.

**Answers 1:** JavaScript has a variety of operators, categorized as follows:

1. Arithmetic Operators:

* + (Addition): give addition of declare value

Let a = 10

Let b= 20

Document. write (‘<br>’+(a+b))

* - (Subtraction): give subtraction of declared value
* Let a = 10
* Let b= 20
* Document. write (‘<br>’+(a-b))
* \* (Multiplication): give multiplication of declared value

Let a = 10

Let b= 20

Document. write (‘<br>’+(a\*b))

* / (Division): give division of declared value

Let a = 10

Let b= 20

Document. write (‘<br>’+(a/b))

* % Modulus -: returns the remainder after division

Let a = 10

Let b= 20

Document. write (‘<br>’+(a%b))

* ++ : Increments the value of a variable by 1

Let ch = 5

Ch++

Ch=6

Let ch = 5

++ch

Output: Ch=5

* -- Decrements the value of a variable by 1

Let ch = 8

--ch

Output: ch=7

2. Assignment Operators:

* == : identify equal to value

Example:

10 = 20: false

10=10: true

* ===: identify equal to and data type

Example:

Let a =

3. Comparison Operators

|  |  |  |
| --- | --- | --- |
| **Comparision operator** | **Use of operator** | Example |
| == | Chack equal value | A==10 true |
| === | Chack equal value and data type | A ===”10” false |
| != | Chack Not equal value | A != 10 |
| !== | Check not equal value and data type | A !== 10 |
| > | Greater than | A > 10 |
| < | Less than | A < 10 |
| >= | Greater than or equal to | A>=10 |
| <= | Less than or equal to | A<=10 |

4. Logical Operators:

|  |  |
| --- | --- |
| **Logical operator** | **Description** |
| && | Logical and |
| || | Logical or |
| ! | Logical not |

5. String Operators

6. Conditional (Ternary) Operator

7. Type Operators: Return data type

**Question 2:** What is the difference between == and === in JavaScript?

**Answers 2: =**= (double equal to) it is useful before comparing values.

For example: 1== ”1” it is true because "1" is converted to the number 1 before comparison

=== (triple equal to) it is useful before comparing values and data

For example: 1===”1” it is false because both types are different

**Control Flow (If-Else, Switch)**

**Question 1:** What is control flow in JavaScript? Explain how if-else statements work with an example.

**Answers 2:** Control flow refers to the order in which statements within a program are executed. While programs typically follow a sequential flow from top to bottom, there are scenarios where we need more flexibility. This article provides a clear understanding of everything you need to know about Control Flow Statements.

**If else:** Executes a block of code if a specified condition is true, and another block if the condition is false.

Example :

Let a = 15

If (a<=10){

Documents. write (“<br> this is true” +a)

}

Else {

documents . Write (“<br> this is false”)

}

**Question 2:** Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

**Answers 2:** The switch statement executes a block of code depending on different cases.

The switch statement is a part of JavaScript's "Conditional" Statements, which are used to perform different actions based on different conditions. Use switch to select one of many blocks of code to be executed. This is the perfect solution for long, nested [if/else](https://www.w3schools.com/jsref/jsref_if.asp) statements.

The switch statement evaluates an expression. The value of the expression is then compared with the values of each case in the structure. If there is a match, the associated block of code is executed.

The switch statement is often used together with a break or a default keyword (or both). These are both optional:

The **break** keyword breaks out of the switch block. This will stop the execution of more execution of code and/or case testing inside the block. If the break is omitted, the next code block in the switch statement is executed.

The **default** keyword specifies some code to run if there is no case match. There can only be one default keyword in a switch. Although this is optional, it is recommended that you use it, as it takes care of unexpected cases.

Example:

Let num = 4

Let newnum;

Switch (a) {

case 1:

newnum = "5";

break;

case 2:

newnum = "6";

break;

case 3:

newnum = "7";

break;

case 4:

newnum= "8";

break;

case 5:

newnum = "10";

break;

default:

New num = "Invalid day";

}

**console.log(newnum); // Output: 8**

**Question 1:** Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

**Answers 2:** three different types of loops: for, while and do while

For : The java script for loop provides a concise way of writing the loop structure. The for loop contains initialization, condition, and increment/decrement in one line thereby providing a shorter, easy-to-debug structure of looping.

Example: hello word print 5 time

For (I =1; I<=5; I++)

{

Document. write (“hello word”)

}

While : The [JS while loop](https://www.geeksforgeeks.org/javascript-while-loop) is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Sntyx : while(condition) {

}

* While loop starts with checking the condition. If it is evaluated to be true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason, it is also called the Entry control loop
* Once the condition is evaluated to be true, the statements in the loop body are executed. Normally the statements contain an updated value for the variable being processed for the next iteration.
* When the condition becomes false, the loop terminates which marks the end of its life cycle.

Example :

Let i = 1

While(i<=6) {

Document. Write(“<br>”+i)

}

Output : 1

2

3

4

5

6

**Do while :** The [JS do-while loop](https://www.geeksforgeeks.org/javascript-do-while-loop) is similar to the while loop with the only difference is that it checks for the condition after executing the statements and therefore is an example of an **Exit Control Loop.** It executes loop content at least once even if the condition is false.

Syntax :

Do

{

Statement

}

While (condition)

Example :

Let a = 1

Do {

Document. write (“<br>” +a)

}

While(a<=5);

Output:

1

2

3

4

5

**Question 2:** What is the difference between a while loop and a do-while loop?

**Answers 2:**

|  |  |
| --- | --- |
| **while** | **Do while** |
| **While loop is entry control loop** | **Do while exact control loop** |
| **A while loop is pre-test loop** | **Do while loop is a post-test loop** |
| **It is testing condition before executing the loop body** | **It tests the condition at the end of the loop body** |
| **There is no semicolon at end of the while loop** | **The semicolon is compulsories the end of the loop** |
| **Ex: while(a<=10)** **{**  **Document. write (“<br>+I")**  **}** | **Ex: do**  **{**  **Document. write (“<br>+a")**  **}**  **While(a<10);** |
| **It might have occurred statement executed zero time. If condition is false** | **At list statement executed if condition is false** |

**Functions**

**Question 1:** What are functions in JavaScript? Explain the syntax for declaring and calling a function.

**Answers 2:** JavaScript function is group of cade Diegan to perform particulars task. JavaScript function can be called Anyware in the program.

It must be case sensitive.

It must start with alphabetical character (A-Z) or an underscore symbol.

It cannot contain space.

It cannot be used as a reserve word.

We are declaring and calling function by four types which are mentions below

Tack nothing and return nothing:

Here we cannot tack any value and not return.

Syntax:

Function add (){

}

Add ()

Example:

Let a = 10

Let b = 20

Function add () {

Document. Write (“<br>” +(a+b))

}

add ()

Tack something returns nothing

syntax

Function add (a,b) {

Document. Write (“....” +(a+b))

}

Add (10,20)

Tack nothing and return something:

Function area() {

Return x\*x\*x

}

Area ()

Example :

Function area () {

Let r = 20

Const pi = 3.14

Return pi\*r\*r

}

Document. Write(“<br>”area())

Tack something and return something:

Function cube (a) {

Return a\*\*3

}

Document. Write(“<br>” cube ())

**Question 2:** What is the difference between a function declaration and a function expression?

**Answers 2:**

|  |  |
| --- | --- |
| **Function declaration** | **Function expression** |
| A function declaration must have a function name | Function expression is same as function declaration without name it is anonymous. |
| Function declaration does not require variable assignment | Function expression can be stored in variable assignment |
| These are executed before any other code | Function expression load and execute only when program interpreter reaches the line of code |
| Function in function declaration can be accessed before and after function definition | Function in function declaration can be accessed after function definition |
| Function are hoisted | Function expressions are not hoisted |
| Syntax: function sum (a,b) {  } | Syntax: let sum= function (a,b) {  } |

**Question 3:** Discuss the concept of parameters and return values in functions.

**Answers 2:**

**Parameters:**

Parameters are variables listed in the function definition that act as placeholders for values passed into the function when it is called. they allow functions to be flexible and work with different inputs.

Function user (name) {

Console.log (“Hello”+user)

}

User (nagma)

User (abc)

**Return Values:** A return value is the result that a function sends back after it has completed its execution. It allows you to capture the output of a function and use it elsewhere in your code

Function user (a,b) {

Return a+b;

}

Let ans = user (10,23)

Console.log (ans())

**Question 1:** What is an array in JavaScript? How do you declare and initialize an array?

**Answers 2:** an array is a data structure used to store a collection of values. These values can be of any data type, such as numbers, strings, objects, or even other arrays.

Example:

Let array = [1,2,3,4,5]

**Initializing an array in JavaScript** involves creating a variable and assigning it a literal array. The array items are enclosed in square brackets with comma-separated elements. These elements can be of any data type and can be omitted for an empty array.

Example:

Ler arr1 = []

Console.log (“this is empty array “+arr1)

Let arr2 = [10,20,30,40,50]

Console.log (“array with Iteam” +arr2)

**Question 2:** Explain the methods push (), pop (), shift (), and unshift () used in arrays.

**Answers 2:**

Push: this is used for adding element in array at the end of the array

Example:

Let array= [1,2,3,4,5]

Array. Push(6)

Output: 1,2,3,4,5,6

Pop(): this is used for deleting element in array at the end of the array

Let array= [1,2,3,4,5]

Array. Pop (4)

Output: 1,2,3,5

Shift (): shift () function add element in array at first position

Let array= [1,2,3,4,5]

Array. Shift (0)

Output: 0,1,2,3,4,5

Unshift (): unshift function remove element at first position

Let array= [1,2,3,4,5]

Array. unshift (1)

Output: 2,3,4,5

**Question 1:** What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

**Answers 1:**

the DOM (Document Object Model) is a programming interface that represents the structure of an HTML or XML document. It allows you to access and manipulate the elements, attributes, and content of a web page using JavaScript.

* The HTML document is a tree-like structure with elements like <html>, <body>, <div>, <p>, etc.
* DOM is a representation of this structure in the browser's memory, which JavaScript can interact with.
* Each element in the HTML document becomes a node in the DOM, and you can access and manipulate these nodes using JavaScript.

What can you do with the DOM?

* Modify content: Update the text content of elements, add or remove elements, and change attribute values.
* Style elements: Change the appearance of elements by modifying CSS properties.
* Handle events: Respond to user actions like clicks, mouse movements, and keyboard input.

Question 2: Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.

**Answers 2:**

getElementById() : This method is used when developers have defined certain HTML elements with **IDs** that uniquely identify the same elements in the whole document. It returns an **Element object** which matches the specified ID in the method. If the ID does not exist in the document, it simply returns **null**.

Example:

<div id = “Demo”> </div>

<script> document .getElementbyid (“demo”).innerhtml = “hello sir” </script>

getElementsByClassName():

This method is used when there are multiple HTML elements with the same class name. It returns a **collection** of all objects which match the specified class in the method.

**querySelector() :**

This method returns the **first match** of an element that is found within the HTML document with the specific selector. If there is no match**, null** is returned.

**Syntax:**

document.querySelector(selector);

GetElementsByClassName()

The getElementsByClassName() method returns a collection of elements with a specified class name(s).

**Syntax:**

document.getElementbyclass(selector);