

This property is used to reflect the HTML Checked attribute.

**document.getElementById("GFG").checked;**

If the CheckBox is checked then it returns True.

JavaScript is also known as the browser’s language.

JavaScript supports automatic type conversion.

We can place the <script> tag inside the head tag or the body tag, both the techniques are correct.

Only prompt creates an input field where the user can enter the data.

**It is not necessary for the external script file to contain a <script> tag.**

What is the correct syntax for referring to an external script called ‘gfg.js’?

**<script src=”gfg.js”>**

The**onerror** event handler was the first feature to facilitate error handling in JavaScript.

**Navigator used to obtain browser vendor and version info.**

Calling **setTimeout** with a delay of **0** (**zero**) milliseconds doesn't execute the callback function after the given interval. The execution depends on the number of waiting tasks in the queue.

[window.location==documment.location](https://www.google.com/search?safe=active&sxsrf=ACYBGNSlYjzewBCb-F2r9VcYH-ovh9-UfQ:1574271262591&q=window.location%3D%3Ddocumment.location&nfpr=1&sa=X&ved=2ahUKEwii5ou7qfnlAhXQXSsKHfsqDf4QvgUoAXoECA4QKA) returns ‘True”

A JavaScript **window** is a global object, The **document** object comes under the window object

Window.open()//to open window

Window.close()//close

window.print();//print on window

Is a variable named ‘apple’ same as ‘Apple’ in javascript?

Javascript is case sensitive.(PHP is case sensitive only for variable not for function)

**Which company developed JavaScript ?**  
Netscape developed the JavaScript and created by Brenden Eich in the year of 1995.

**What is closure and how do you use it?**

1. When a function returns the other function the returning function will hold its environment and this is known as closure.

2. A closure is a variable that is created inside the function. It is combination of two things, a function and any local variable.

1. [**What are undeclared and undefined variables ?**](https://www.geeksforgeeks.org/what-are-undeclared-and-undefined-variables-in-javascript/)
   * **Undefined:** It occurs when a variable has been declared but has not been assigned with any value. **Undefined is not a keyword.**
   * **Undeclared:** It occurs when we try to access any variable which is not initialized or declared earlier using var or const keyword. If we use ‘typeof’ operator to get the value of an undeclared variable, we will face the runtime error with return value as “undefined”. **The scope of the undeclared variables is always global.**

**What do you mean by NULL in JavaScript ?**  
The NULL value represents the no value or no object. It can be called as empty value/object.

**How to delete property specific value ?**  
The delete keyword is used to delete the whole property and all the values at once like

var gfg={Course: "DSA", Duration:30};

delete gfg.Course;

**How can you submit a form using JavaScript ?**  
You can use **document.form[0].submit()** method to submit the form in JavaScript.

### **we detect OS of the client machine using “navigator.appVersion”**

**Data Types:**

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

var x;           // Now x is undefined  
x = 5;           // Now x is a Number  
x = "John";      // Now x is a String

2. You can use single or double quotes for string

var carName1 = "Volvo XC60";   // Using double quotes  
var carName2 = 'Volvo XC60';

**3. Array**

Creating array in 3 ways:

1. By array literal

var emp=["Sonoo","Vimal","Ratan"];

1. By creating instance of Array directly (using new keyword)

var emp = new Array();

emp[0] = "Arun";

1. By using an Array constructor (using new keyword)

var emp=new Array("Jai","Vijay","Smith");

**4. Javascript Object**

Creating object 3 ways

1. By object literal(for simplicity)

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

1. By creating instance of Object **directly** (using new keyword)

**<script>**

var emp=new Object();

emp.id=101;

emp.name="Ravi Malik";

emp.salary=50000;

document.write(emp.id+" "+emp.name+" "+emp.salary);

**</script>**

1. By using an object constructor (using new keyword)

By using constructor **you need to create function** with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object.

**<script>**

function emp(id,name,salary){

this.id=id;

this.name=name;

this.salary=salary;

}

e=new emp(103,"Vimal Jaiswal",30000); //object constructor

document.write(e.id+" "+e.name+" "+e.salary);

**</script>**

5. The typeof operator returns the type of a variable or an expression:

The typeof operator returns "object" for **objects, arrays, and null.**

**JavaScript arrays are objects.**

**typeof {name:'John', age:34}** // Returns "object"  
**typeof [1,2,3,4]**             // Returns "object" (not "array”)  
**typeof null**                  // Returns "object"  
**typeof function myFunc(){}**   // Returns "function"

typeof "John"              // Returns "string"  
**typeof 3.14**                **// Returns "number**"  
typeof true                // Returns "boolean"  
typeof false               // Returns "boolean"  
**typeof x**                   // Returns "undefined"

6. Unfortunately, in JavaScript, the data type of null is an object.

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

*person = null;*

*document.write(typeof(person));//object*

*7.Null and Undefined*

typeof undefined           // undefined  
typeof null                // object  
null === undefined         // false  
null == undefined          // true

In JavaScript, all the declarations (including variable declarations) are processed before program execution. Thus, **declaring a variable anywhere is equivalent to declaring it at the top of the code. This provides JavaScript with the functionality to use the variables before their declarations in the code.** We call this behavior of variables **“var hoisting”.**

Program:1

<script>

        // calling x after definition

        var x = 5;

        document.write(x, "\n");

        // calling y after definition

        let y = 10;

        document.write(y, "\n");

**// calling var z before definition will return undefined**

**document.write(z, "\n");**

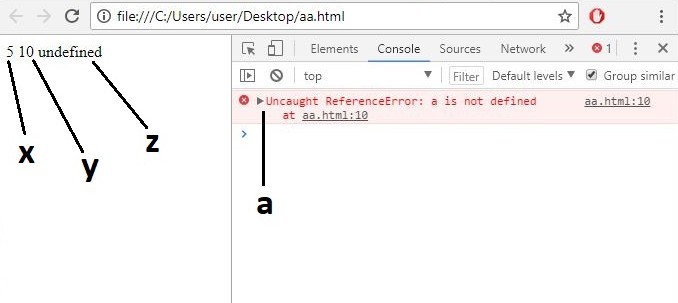
**var z = 2;**

**// calling let a before definition will give error**

**document.write(a);**

**let a = 3;**

    </script>



**undefined x 1**

**arr = [1, 2, 3].**

If I add any value at position 4, position 3 is left empty. When I display the array **arr** in Chrome’s developer console, I get the value: **[1, 2, 3, undefined x 1, 5].** Or undefined, empty, etc. in other browser consoles. **1** indicates the number of consecutive undefined or empty

**multi-line string in JavaScript**

**Using backticks**

1. var string = `line1
2. line2
3. line3`
4. **Using + operator**
5. var string = “line1” +
6. “line2” +
7. “line3”;

**Using \ (backslash)**

1. var string = “line1 \
2. line2 \
3. line3”;

**Array:**

1. How to make empty array in javascript: [https://data-flair.training/blogs/javascript-interview-questions-and-answers/#](https://data-flair.training/blogs/javascript-interview-questions-and-answers/) Q\_No:4

2. Output of given code:

1. var courses = ["JavaScript","Java","C","C++","Python"];
2. delete courses[2];
3. console.**log**(courses.length);

we will get **(5) (** deletes only the value present at the position,no affect on length)

**[“JavaScript”, “Java”, empty, “C++”, “Python”].**

3. var myArray = [[[]]];   is a three-dimensional array.

**Closure:**

(**function**(x)

{

**return** (**function**(y)

{

console.**log**(x);

})(2)

})(1);

 the inner function has access to the outer function’s variables. the interpreter didn’t find the value of x in the inner function, it searched for its defined value in the outside function. That’s where it found x to have a value of 1.

**The window.history object contains the browsers history.**

history.back() -same as clicking back in the browser

history.forward() -same as clicking forward in the browser

history.go() -loads the given page number.

history.back();//for previous page

history.forward();//for next page

history.go(2);//for next2nd page

history.go(-2);//for previous 2nd page

1.What will be the result of the following code?

<script>

  document.write( true && 1 && 3);

</script>

OutPut:3

The && (and operator) returns the last (right-side) value as long as the chain is True.

2.What will be the output of the following code?

<script>

  document.write((0 && 1) || (1 || 0));

</script>

OutPut:1

The && (and operator) returns the last (right-side) value as long as the chain is “truthy”. The || (or operator) returns true if either of the value is true.

3.What will be the output of the following code?

<script>

 let ans = 0 / 0;

 document.write(ans);

</script>

Outpuit:NaN

0/0 is undefined for the real number and is therefore represented by NaN (Not a Number) in Javascript.

4.<script>

  let ans = 1;

  document.write( ans === '1');//strickly check data type and value

</script>

Output:false

5.What will be the output of the following code?

|  |
| --- |
| <script>    let y = 1;    y = typeof x;  document.write(typeof x);//undefined    document.write(typeof y);// string  </script> |

Output:

|  |  |
| --- | --- |
| 6.What will be the output of the following code?   |  | | --- | | <script>     var x = [typeof x, typeof y][1];     document.write(typeof typeof x);  </script> | |

Output:string

<script>

document.write(typeof('1' + 2));

</script>

Output: ‘string’

The ‘+’ operator will perform concatenation if either of the variables is a string.

let ans = 6 / "3";

Ans:2

**A number, when divided by a string, will always result in a number.**

**7.**

console.**log**(typeof typeof 1);

The output of the code above in the console will be ‘string’. The value returned by typeof 1 is ‘number’; typeof ‘number’ will return a string.

Read PPT 1.operator