# Functions in JavaScript

JavaScript **function** is a set of statements that take inputs, do some specific computation, and produce output.

A JavaScript function is executed when “something” invokes it (calls it).

**Example 1:** A basic javascript function, here we create a function that divides the 1st element by the second element.

|  |
| --- |
| function myFunction(g1, g2) {      return g1 / g2;  }  const value = myFunction(8, 2); // Calling the function  console.log(value); |

**Output:**

4

You must already have seen some commonly used functions in JavaScript like [alert()](https://www.geeksforgeeks.org/html-window-alert-method/), which is a built-in function in JavaScript. But JavaScript allows us to create user-defined functions also. We can create functions in JavaScript using the keyword `function`.

**Syntax**: The basic syntax to create a function in JavaScript is shown below.

function functionName(Parameter1, Parameter2, ...)  
{  
 // Function body  
}

To create a function in JavaScript, we have to first use the keyword function, separated by the name of the function and parameters within parenthesis. The part of the function inside the curly braces {} is the body of the function.

In javascript, functions can be used in the same way as variables for assignments, or calculations.

## Function Invocation:

* Triggered by an event (e.g., a button click by a user).
* When explicitly called from JavaScript code.
* Automatically executed, such as in self-invoking functions.

## ****Function Definition:****

Before, using a user-defined function in JavaScript we have to create one. We can use the above syntax to create a function in JavaScript. A function definition is sometimes also termed a function declaration or function statement. Below are the rules for creating a function in JavaScript:

* Every function should begin with the keyword function followed by,
* A user-defined function name that should be unique,
* A list of parameters enclosed within parentheses and separated by commas,
* A list of statements composing the body of the function enclosed within curly braces {}.

**Example 2:** This example shows a basic declaration of a function in javascript.

|  |
| --- |
| function calcAddition(number1, number2) {      return number1 + number2;  }  console.log(calcAddition(6,9)); |

**Output**

15

In the above example, we have created a function named calcAddition,

* This function accepts two numbers as parameters and returns the addition of these two numbers.
* Accessing the function with just the function name without () will return the function object instead of the function result.

There are three ways of writing a function in JavaScript:

**Function Declaration:** It declares a function with a function keyword. The function declaration must have a function name.

**Syntax:**

function geeksforGeeks(paramA, paramB) {  
 // Set of statements  
}

## [****Function Expression:****](https://www.geeksforgeeks.org/javascript-function-expression/)

It is similar to a function declaration without the function name. Function expressions can be stored in a variable assignment.

**Syntax:**

let geeksforGeeks= function(paramA, paramB) {  
 // Set of statements  
}

**Example 3:** This example explains the usage of the Function expression.

|  |
| --- |
| const square = function (number) {        return number \* number;  };  const x = square(4); // x gets the value 16  console.log(x); |

**Output**

16

## Functions as Variable Values:

Functions can be used the same way as you use variables.

**Example:**

// Function to convert Fahrenheit to Celsius  
function toCelsius(fahrenheit) {  
 return (fahrenheit - 32) \* 5/9;  
 }  
   
 // Using the function to convert temperature  
 let temperatureInFahrenheit = 77;  
 let temperatureInCelsius = toCelsius(temperatureInFahrenheit);  
 let text = "The temperature is " + temperatureInCelsius + " Celsius";

## [****Arrow Function:****](https://www.geeksforgeeks.org/es6-arrow-function/)

It is one of the most used and efficient methods to create a function in JavaScript because of its comparatively easy implementation. It is a simplified as well as a more compact version of a regular or normal function expression or syntax.

**Syntax:**

let function\_name = (argument1, argument2 ,..) => expression

**Example 4:** This example describes the usage of the Arrow function.

|  |
| --- |
| const a = ["Hydrogen", "Helium", "Lithium", "Beryllium"];    const a2 = a.map(function (s) {      return s.length;  });    console.log("Normal way ", a2); // [8, 6, 7, 9]    const a3 = a.map((s) => s.length);    console.log("Using Arrow Function ", a3); // [8, 6, 7, 9] |

**Output**

Normal way [ 8, 6, 7, 9 ]

Using Arrow Function [ 8, 6, 7, 9 ]

## ****Function Parameters:****

Till now, we have heard a lot about function parameters but haven’t discussed them in detail. Parameters are additional information passed to a function. For example, in the above example, the task of the function calcAddition is to calculate the addition of two numbers. These two numbers on which we want to perform the addition operation are passed to this function as parameters. The parameters are passed to the function within parentheses after the function name and separated by commas. A function in JavaScript can have any number of parameters and also at the same time, a function in JavaScript can not have a single parameter.

**Example 4:** In this example, we pass the argument to the function.

|  |
| --- |
| function multiply(a, b) {      b = typeof b !== "undefined" ? b : 1;      return a \* b;  }    console.log(multiply(69)); // 69 |

**Output**

69

## ****Calling Functions****:

After defining a function, the next step is to call them to make use of the function. We can call a function by using the function name separated by the value of parameters enclosed between the parenthesis and a semicolon at the end. The below syntax shows how to call functions in JavaScript:

**Syntax:**

functionName( Value1, Value2, ..);

**Example 5:** Below is a sample program that illustrates the working of functions in JavaScript:

|  |
| --- |
| function welcomeMsg(name) {      return ("Hello " + name + " welcome to GeeksforGeeks");    }    // creating a variable  let nameVal = "Admin";    // calling the function  console.log(welcomeMsg(nameVal)); |

**Output:**

Hello Admin welcome to GeeksforGeeks

## ****Return Statement****:

There are some situations when we want to return some values from a function after performing some operations. In such cases, we can make use of the return statement in JavaScript. This is an optional statement and most of the time the last statement in a JavaScript function. Look at our first example with the function named as calcAddition. This function is calculating two numbers and then returns the result.

**Syntax:** The most basic syntax for using the return statement is:

return value;

The return statement begins with the keyword return separated by the value which we want to return from it. We can use an expression also instead of directly returning the value.

# A Basic Guide to JavaScript

JavaScript is a lightweight, open, and cross-platform programming language. It is omnipresent in modern development and is used by programmers across the world to create dynamic and interactive web content like applications and browsers. It is one of the core technologies of the World Wide Web, alongside HTML and CSS, and the powerhouse behind the rapidly evolving Internet by helping create beautiful and crazy-fast websites.

This article provides an in-depth JavaScript Cheat Sheet, a must-have for every web developer.



JavaScript Chear Sheet

## ****What is JavaScript Cheat Sheet?****

Our comprehensive **JavaScript Cheat Sheet** is designed to assist both beginners and experienced coders in mastering this versatile programming language. Whether you’re building interactive web pages, creating dynamic content, or enhancing user experiences, this quick reference guide provides essential concepts, syntax, and ready-to-use code snippets.

**Table of Content**

* [Fundamentals](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#fundamentals)
* [Variables](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#variables)
* [Datatypes](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#datatypes)
* [Operators](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#operators)
* [JS scope and scope chain](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#js-scope-and-scope-chain)
* [Functions](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#functions)
* [Arrays](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#arrays)
* [Loops](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#loops)
* [If-else](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#ifelse)
* [Strings](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#strings)
* [Regular Expressions](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#regular-expressions)
* [Data Transformation](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#data-transformation)
* [Date objects](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#date-objects)
* [DOM](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#dom)
* [Numbers and Math](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#numbers-and-math)
* [Events](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#events)
* [Error](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#error)
* [Window Properties](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/#window-properties)

## ****Fundamentals****

To use JavaScript on website we need to attach the JavaScript file to the HTML file. To do a better code, we should also do the commenting during the code. There are two types of commenting single-line and multiline.

* For a browser to know the code is written in JavaScript and execute it we must enclose the code within the <script> tag to include it on the HTML page.

<script type="text/javascript">   
 // Your js code goes here   
</script>

* An external JavaScript file can also be written separately and included within our HTML file using the script tag as:

<script src="filename.js"></script>.

* JavaScript comments can be extremely useful to explain JavaScript code and help understand what’s going on in your code and make it more readable.
  + **Single-line comments:** Start a single-line comment with “//”.
  + **Multi-line comments:** Wrap the comment in /\* and\*/ if it spans multiple lines.

## ****Variables****

[Variables](https://www.geeksforgeeks.org/variables-datatypes-javascript/) in JavaScript are containers for storing data. JavaScript allows the usage of variables in the following three ways:

| **Variable** | **Description** | **Example** |
| --- | --- | --- |
| [**var**](https://www.geeksforgeeks.org/javascript-var/) | Used to initialize to value, redeclared and its value can be reassigned. | var x= value; |
| [**let**](https://www.geeksforgeeks.org/javascript-let/) | Similar to var but is block scoped | let y= value; |
| [**const**](https://www.geeksforgeeks.org/javascript-const/) | Used to declare a fixed value that cannot be changed. | const z= value; |

console.log("Using var Keyword");

var x = 1;

if (x === 1) {

var x = 2;

console.log(x); // Output: 2

}

console.log(x); // Output: 2

// Using let keyword

console.log("Using let Keyword");

let x1 = 1;

if (x1 === 1) {

let x1 = 2;

console.log(x1); // Output: 2

}

console.log(x1); // Output: 1

// Using const keyword

console.log("Using const Keyword");

const number = 48;

// Changing const value will display TypeError

try {

const number = 42;

} catch (err) {

console.log(err);

}

console.log(number); // Output: 48

## ****Datatypes****

There are different types of [datatypes](https://www.geeksforgeeks.org/variables-datatypes-javascript/) that can be stored in JavaScript variables. For the machine to be able to operate on variables, and correctly evaluate the expressions it is important to know about the type of variables involved. There are following primitive and non-primitive datatypes in JavaScript:

| **Datatype** | **Description** | **Example** |
| --- | --- | --- |
| [**Number**](https://www.geeksforgeeks.org/javascript-numbers/) | Numeric values can be real number or integers. | var x= number; |
| [**String**](https://www.geeksforgeeks.org/javascript-strings/) | Series of multiple characters written in quotes. | var x= “characters”; |
| [**Boolean**](https://www.geeksforgeeks.org/javascript-boolean/) | Has only two values true or false. | var x= true/false; |
| [**Null**](https://www.geeksforgeeks.org/null-in-javascript/) | Special value that represents that the variable is empty. | var x= null; |
| [**Undefined**](https://www.geeksforgeeks.org/undefined-in-javascript/) | Represents a variable which is declared but not assigned any value. | let x; / let x= undefined; |
| [**Object**](https://www.geeksforgeeks.org/objects-in-javascript) | Complex data type that allows us to store a collection of data. | var x= {     key: “value”;     key: “value”; } |
| [**Array**](https://www.geeksforgeeks.org/arrays-in-javascript/) | Stores multiple values of same type in a single variable. | var x =[‘y1’, ‘y2′,’y3′,’y4’];  y: any datatype |
| [**Function**](https://www.geeksforgeeks.org/functions-in-javascript/) | Functions are objects that can be called to execute a block of code. | function x(arguments){      block of code  } |

// String

let str = "hello geeks";

console.log(str);

// Number

const num = 10;

console.log(num);

// Boolean

const x = "true";

console.log(x);

// Undefined

let name;

console.log(name );

// Null

const number = null;

console.log(number);

// Symbol

const value1 = Symbol("hello");

const value2 = Symbol("hello");

console.log(value1);

console.log(value2);

// Here both values are different

// as they are symbol type which

// is immutable object

const object = {

firstName: "geek",

lastName: null,

batch: 2,

};

console.log(object);

## ****Operators****

JavaScript [operators](https://www.geeksforgeeks.org/javascript-operators/) are symbols used to perform various operations on variables (operands). Following are the different types of operators:

| **Operators** | **Description** | **Symbols** |
| --- | --- | --- |
| [**Arithmetic**](https://www.geeksforgeeks.org/javascript-arithmetic-operators/?ref=lbp) | Used to perform basic arithmetic operations on variables(operands). | +,-,\*,/,%,++,– |
| [**Comparison**](https://www.geeksforgeeks.org/javascript-comparison-operators/?ref=lbp) | Comparison operator is used to compare two operands. | ==, ===,!=,>,<,>=,<= |
| [**Bitwise**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/) | Used to perform bitwise operations. | &, | , ^,~,<<, >>, >>> |
| [**Logical**](https://www.geeksforgeeks.org/javascript-course-logical-operators-in-javascript/) | There are three logical operators in javascript.   * logical AND: When all the operands are true. * logical OR: When one or more than one operands are true. * logical NOT: Converts true to false. | exp1&&exp2,exp1 ||exp2, !exp |
| [**Assignment**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/%3ca%20href=) | Assignment operators assign values to JavaScript variables. | =, +=,-=,\*=,/=,%= |

let x = 5;

let y = 3;

// Addition

console.log("x + y = ", x); // 8

// Subtraction

console.log("x - y = ", x - y); // 2

// Multiplication

console.log("x \* y = ", x \* y); // 15

// Division

console.log("x / y = ", x / y);

// Remainder

console.log("x % y = ", (x % y)); // 2

// Increment

console.log("++x = ", ++x); // x is now 6

console.log("x++ = ", x++);

console.log("x = ", x); // 7

// Decrement

console.log("--x = ", --x); // x is now 6

console.log("x-- = ", x--);

console.log("x = ", x); // 5

// Exponentiation

console.log("x \*\* y =", x \*\* y);

// Comparison

console.log(x > y); // true

// Equal operator

console.log((2 == 2)); // true

// Not equal operator

console.log((3 != 2)); // true

// Strict equal operator

console.log((2 === 2)); // true

// Strict not equal operator

console.log((2 !== 2)); // true

// Logical Operator

// Logical AND

console.log((x < 6 && y < 5)); // true

// Logical OR

console.log((x < 6 || y > 6)); // true

// Logical NOT

console.log(!(x < 6)); // false

## ****JS scope and scope chain****

* **Scope:** [Scope](https://www.geeksforgeeks.org/all-about-functions-and-scopes-in-javascript/) defines the accessibility or visibility of variables in JavaScript. That is, which sections of the program can access a given variable and where the variable can be seen. There are usually three types of scopes.
* **Scope chain:** The [scope chain](https://www.geeksforgeeks.org/explain-scope-and-scope-chain-in-javascript/) is used to resolve the value of variable names in JavaScript. Without a scope chain, the JavaScript engine wouldn’t know which value to pick for a certain variable name if there are multiply defined at different scopes. If the JavaScript engine could not find the variable in the current scope, it will look into the outer scope and will continue to do so until it finds the variable or reaches the global scope. If it still could not find the variable, it will either implicitly declare the variable in the global scope (if not in strict mode) or return an error.

| **scope** | **Description** |
| --- | --- |
| **function** | Variables declared inside a function is inside the function scope also known as local scope. |
| **global** | The variables in global scope can be accessed from anywhere in the program. |
| **block** | Block scope restricts the access of a variable outside its scope |

let z = 3;

function foo() {

if (true) {

var x = '1'; // Exist in function scope

const y = '2'; // Exist in block scope

}

console.log(x);

console.log(y);

console.log(z); // Exist in global scope

}

foo();

## ****Functions****

A [JavaScript function](https://www.geeksforgeeks.org/functions-in-javascript/) is a block of code designed to perform a particular task. It is executed when invoked or called. Instead of writing the same piece of code again and again you can put it in a function and invoke the function when required. JavaScript function can be created using the functions keyword. Some of the functions in JavaScript are:

| **Function** | **Description** |
| --- | --- |
| [**parseInt()**](https://www.geeksforgeeks.org/javascript-parseint-function/) | Parses an argument passed to it and returns an integral number. |
| [**parseFloat()**](https://www.geeksforgeeks.org/javascript-parsefloat-function/) | Parses the argument and returns a floating-point number. |
| [**isNaN()**](https://www.geeksforgeeks.org/javascript-isnan-function/) | Determines if a given value is Not a Number. |
| [**Number()**](https://www.geeksforgeeks.org/javascript-number-function/) | Returns an argument after converting it to number. |
| [**eval()**](https://www.geeksforgeeks.org/javascript-eval-function/) | Used for evaluating JavaScript programs presented as strings. |
| [**prompt()**](https://www.geeksforgeeks.org/javascript-window-prompt-method/) | Creates a dialogue box for taking input from the user. |
| [**encodeURI()**](https://www.geeksforgeeks.org/javascript-encodeuri-decodeuri-and-its-components-functions/) | Encodes a URI into a UTF-8 encoding scheme. |
| [**match()**](https://www.geeksforgeeks.org/javascript-match-function/) | Used to search a string for a match against regular expression. |

// JS parseInt function

const num1 = parseInt("3.14");

console.log('Using parseInt("3.14") = ' + num1);

// JS parsefloat function

// It returns floating point Number until

// it encounters Not a Number character

const num2 = parseFloat("2018.12@geeksforgeeks");

console.log('parseFloat("2018@geeksforgeeks") = ' + num2);

// JS isNAN function

console.log(isNaN(12));

// JS number() function

const num3 = Number(true);

console.log("Value of 'true': " + num3);

// JS eval() function

function evalfn() {

const a = 4;

const b = 4;

const value = eval(new String(a \* b));

console.log(value);

}

evalfn();

// JS encode URI function

const url = "https://www.google.com/search?q=geeks for geeks";

const encodedURL = encodeURI(url);

console.log(encodedURL);

## ****Arrays****

In JavaScript, [array](https://www.geeksforgeeks.org/arrays-in-javascript/) is a single variable that is used to store different elements. It is often used when we want to store list of elements and access them by a single variable. Arrays use numbers as index to access its “elements”.  
Declaration of an Array: There are basically two ways to declare an array.

**Example:**   
var House = [ ]; // Method 1  
var House = new Array(); // Method 2

There are various operations that can be performed on arrays using JavaScript methods. Some of these methods are:

| **Method** | **Description** |
| --- | --- |
| [**push()**](https://www.geeksforgeeks.org/javascript-array-push-method/) | Adds a new element at the very end of an array. |
| [**pop()**](https://www.geeksforgeeks.org/javascript-array-pop-method/) | Removes the last element of an array. |
| [**concat()**](https://www.geeksforgeeks.org/javascript-array-concat-method/) | Joins various arrays into a single array. |
| [**shift()**](https://www.geeksforgeeks.org/javascript-array-shift-method/) | Removes the first element of an array |
| [**unShift()**](https://www.geeksforgeeks.org/javascript-array-unshift-method/) | Adds new elements at the beginning of the array |
| [**reverse()**](https://www.geeksforgeeks.org/javascript-array-reverse-method/) | Reverses the order of the elements in an array. |
| [**slice()**](https://www.geeksforgeeks.org/javascript-array-slice-method/) | Pulls a copy of a part of an array into a new array. |
| [**splice()**](https://www.geeksforgeeks.org/javascript-array-splice-method/) | Adds elements in a particular way and position. |
| [**toString()**](https://www.geeksforgeeks.org/javascript-array-tostring-method/) | Converts the array elements into strings. |
| [**valueOf()**](https://www.geeksforgeeks.org/javascript-array-valueof-method/) | Returns the primitive value of the given object. |
| [**indexOf()**](https://www.geeksforgeeks.org/javascript-array-indexof-method/) | Returns the first index at which a given element is found. |
| [**lastIndexOf()**](https://www.geeksforgeeks.org/javascript-array-lastindexof-method/) | Returns the final index at which a given element appears. |
| [**join()**](https://www.geeksforgeeks.org/javascript-array-join-method/) | Combines elements of an array into one single string and then returns it |
| [**sort()**](https://www.geeksforgeeks.org/javascript-array-sort-method/) | Sorts the array elements based on some condition. |

// Declaring and initializing arrays

// Number Array

let arr = [10, 20, 30, 40, 50];

let arr1 = [110, 120, 130, 140];

// String array

let string\_arr = ["Alex", "peter", "chloe"];

// push: Adding elements at the end of the array

arr.push(60);

console.log("After push op " + arr);

// unshift() Adding elements at the start of the array

arr.unshift(0, 10);

console.log("After unshift op " + arr );

// pop: removing elements from the end of the array

arr.pop();

console.log("After pop op" + arr);

// shift(): Removing elements from the start of the array

arr.shift();

console.log("After shift op " + arr);

// splice(x,y): removes x number of elements

// starting from index y

arr.splice(2, 1);

console.log("After splice op" + arr);

// reverse(): reverses the order of elements in array

arr.reverse();

console.log("After reverse op" + arr);

// concat(): merges two or more array

console.log("After concat op" + arr.concat(arr1));

## ****Loops****

[Loops](https://www.geeksforgeeks.org/loops-in-javascript/) are a useful feature in most programming languages. With loops you can evaluate a set of instructions/functions repeatedly until certain condition is reached. They let you run code blocks as many times as you like with different values while some condition is true. Loops can be created in the following ways in JavaScript:

| **Loop** | **Description** | **Syntax** |
| --- | --- | --- |
| [**for**](https://www.geeksforgeeks.org/javascript-for-loop/) | Loops over a block of with conditions specified in the beginning. | for (initialization condition; testing condition;increment/decrement) {     statement(s) } |
| [**while**](https://www.geeksforgeeks.org/javascript-while-loop/) | Entry control loop which executes after checking the condition. | while (boolean condition) {     loop statements… } |
| [**do-while**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/) | Exit Control Loop which executes once before checking the condition. | do {     statements.. }  while (condition); |
| [**for-in**](https://www.geeksforgeeks.org/javascript-for-in-loop/?ref=lbp) | Another version of for loop to provide a simpler way to iterate. | for (variableName in Object) {     statement(s) } |

// Illustration of for loop

let x;

// for loop begins when x=2

// and runs till x <=4

for (x = 2; x <= 4; x++) {

console.log("Value of x:" + x);

}

// Illustration of for..in loop

// creating an Object

let languages = {

first: "C",

second: "Java",

third: "Python",

fourth: "PHP",

fifth: "JavaScript",

};

// Iterate through every property of

// the object languages and print all

// of them using for..in loops

for (itr in languages) {

console.log(languages[itr]);

}

// Illustration of while loop

let y = 1;

// Exit when x becomes greater than 4

while (y <= 4) {

console.log("Value of y:" + y);

// Increment the value of y for

// next iteration

x++;

}

// Illustration of do-while loop

let z = 21;

do {

// The line while be printer even if

// the condition is false

console.log("Value of z:" + z);

z++;

} while (z < 20);

## ****If-else****

[If-else](https://www.geeksforgeeks.org/if-else-statement-in-javascript/) is used in JavaScript to execute a block of codes conditionally. These are used to set conditions for your code block to run. If certain condition is satisfied certain code block is executed otherwise else code block is executed. JavaScript allows us to nest if statements within if statements as well. i.e, we can place an if statement inside another if statement.

if ( condition ) {  
 // Executes this block if  
 // condition is true  
}  
else {  
 // Executes this block if  
 // condition is false  
}

// JavaScript program to illustrate if-else statement

const i = 10;

if (i < 15)

console.log("Value of i is less than 10");

else

console.log("Value of i is greater than 10");

## ****Strings****

[Strings](https://www.geeksforgeeks.org/javascript-strings/) in JavaScript are primitive and immutable data types used for storing and manipulating text data which can be zero or more characters consisting of letters, numbers or symbols. JavaScript provides a lot of methods to manipulate strings. Some most used ones are:

| **Methods** | **Description** |
| --- | --- |
| [**concat()**](https://www.geeksforgeeks.org/javascript-string-prototype-concat-function/) | Used for concatenating multiple strings into a single string. |
| [**match()**](https://www.geeksforgeeks.org/javascript-match-function/) | Used for finding matche of a string against a provided pattern. |
| [**replace()**](https://www.geeksforgeeks.org/javascript-string-replace-method/) | Used for finding and replacing a given text in string. |
| [**substr()**](https://www.geeksforgeeks.org/javascript-string-substr/) | Used to extract length characters from a given string. |
| [**slice()**](https://www.geeksforgeeks.org/javascript-string-slice/) | Used for extracting an area of the string and returs it |
| [**lastIndexOf()**](https://www.geeksforgeeks.org/javascript-string-prototype-lastindexof-function/) | Used to return the index (position) of the last occurrence of a specified value. |
| [**charAt()**](https://www.geeksforgeeks.org/javascript-string-charat-method/) | Used for returning the character at a particular index of a string |
| [**valueOf()**](https://www.geeksforgeeks.org/javascript-string-valueof/) | Used for returning the primitive value of a string object. |
| [**split()**](https://www.geeksforgeeks.org/javascript-string-prototype-split-function/) | Used for splitting a string object into an array of strings. |
| [**toUpperCase()**](https://www.geeksforgeeks.org/javascript-string-touppercase/) | Used for converting strings to upper case. |
| [**toLoweCase()**](https://www.geeksforgeeks.org/javascript-string-tolowercase-method/) | Used for converting strings to lower case. |

let gfg = 'GFG ';

let geeks = 'stands-for-GeeksforGeeks';

// Print the string as it is

console.log(gfg);

console.log(geeks);

// concat() method

console.log(gfg.concat(geeks));

// match() method

console.log(geeks.match(/eek/));

// charAt() method

console.log(geeks.charAt(5));

// valueOf() method

console.log(geeks.valueOf());

// lastIndexOf() method

console.log(geeks.lastIndexOf('for'));

// substr() method

console.log(geeks.substr(6));

// indexOf() method

console.log(gfg.indexOf('G'));

// replace() method

console.log(gfg.replace('FG', 'fg'));

// slice() method

console.log(geeks.slice(2, 8));

// split() method

console.log(geeks.split('-'));

// toUpperCase method

console.log(geeks.toUpperCase(geeks));

// toLowerCase method

console.log(geeks.toLowerCase(geeks));

## ****Regular Expressions****

A [regular expression](https://www.geeksforgeeks.org/javascript-regular-expressions/) is a sequence of characters that forms a search pattern. The search pattern can be used for text search and text to replace operations. A regular expression can be a single character or a more complicated pattern.

**Syntax:**

/pattern/modifiers;

**You can also use regEx() to create regular expression in javascript:**

const regex1 = /^ab/;  
const regex2 = new Regexp('/^ab/');

Let us look at how JavaScript allows Regular Expressions:

### ****Regular Expression Modifiers:****

Modifiers can be used to perform multiline searches. Some of the pattern modifiers that are allowed in JavaScript:

| **Modifiers** | **Description** |
| --- | --- |
| **[abc]** | Find any of the character inside the brackets |
| **[0-9]** | Find any of the digits between the brackets 0 to 9 |
| **(x/y)** | Find any of the alternatives between x or y separated with | |

### ****Regular Expression Patterns:****

Metacharacters are characters with a special meaning. Some of the metacharacters that are allowed in JavaScript:

| **Metacharacters** | **Description** |
| --- | --- |
| **.** | Used for finding a single character, except newline or line terminator |
| **\d** | Used to find a digit. |
| **\s** | Used to find a whitespace character |
| **\uxxxx** | Used to find the Unicode character specified by the hexadecimal number |

### ****Quantifiers:****

They provide the minimum number of instances of a character, group, or character class in the input required to find a match. Some of the quantifiers allowed in JavaScript are:

| **Quantifiers** | **Description** |
| --- | --- |
| **n+** | Used to match any string that contains at least one n |
| **n\*** | Used to match any string that contains zero or more occurrences of n |
| [**n?**](https://www.geeksforgeeks.org/javascript-regexp-quantifier/) | Used to matches any string that contains zero or one occurrences of n |
| **n{x}** | Matches strings that contain a sequence of X n’s |
| **^n** | Matches strings with n in the first place |

Here is an example to help you understand regular expression better.

// Program to validate the email address

function validateEmail(email) {

// Regex pattern for email

const re = /\S+@\S+\.\S+/g;

// Check if the email is valid

let result = re.test(email);

if (result) {

console.log("The email is valid.");

} else {

console.log("The email is not valid.");

}

}

// Input Email Id

let email = "abc@gmail.com"

validateEmail(email);

email = "abc#$#@45com"

validateEmail(email);

## ****Data Transformation****

Data transformation is converts data from one format to another. It can be done with the usage of higher-order functions which can accept one or more functions as inputs and return a function as the result. All higher-order functions that take a function as input are map(), filter(), and reduce().

| **Method** | **Description** | **Syntax** |
| --- | --- | --- |
| [**map()**](https://www.geeksforgeeks.org/javascript-array-map-method-2/) | Iterates over an array and calls function on every element of array. | array.map(function(currentValue, index, arr), thisValue) |
| [**filter()**](https://www.geeksforgeeks.org/javascript-array-filter-method/) | Create a new array from a given array after applying a condition. | array.filter(callback(element, index, arr),  thisValue) |
| [**reduce()**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/%3ca%20href=) | Reduces the array to single value using a function | array.reduce( function(total, currentValue, currentIndex, arr), initialValue ) |

const num = [16, 25];

/\* Using JS map() Method \*/

console.log(num.map(Math.sqrt));

const ages = [19, 37, 16, 42];

/\* Using JS filter() Method \*/

console.log(ages.filter(checkAdult));

function checkAdult(age) {

return age >= 18;

}

/\* Using JS reduce() Method \*/

const numbers = [165, 84, 35];

console.log(numbers.reduce(myFunc));

function myFunc(total, num) {

return total - num;

}

## ****Date objects****

The [Date object](https://www.geeksforgeeks.org/javascript-date-objects/) is an inbuilt datatype of JavaScript language. It is used to deal with and change dates and times. There are four different way to declare a date, the basic things is that the date objects are created by the new Date() operator.  
**Syntax:**

new Date()  
new Date(milliseconds)  
new Date(dataString)  
new Date(year, month, date, hour, minute, second, millisecond)

There are various methods in JavaScript used to get date and time values or create custom date objects. Some of these methods are:

| **Method** | **Description** |
| --- | --- |
| [**getDate()**](https://www.geeksforgeeks.org/javascript-date-getdate-method/) | Used to return the month’s day as a number (1-31) |
| [**getTime()**](https://www.geeksforgeeks.org/javascript-date-gettime-method/) | Used to get the milliseconds since January 1, 1970 |
| [**getMinutes()**](https://www.geeksforgeeks.org/javascript-date-getminutes-method/) | Returns the current minute (0-59) |
| [**getFullYear()**](https://www.geeksforgeeks.org/javascript-date-getfullyear-method/) | Returns the current year as a four-digit value (yyyy) |
| [**getDay()**](https://www.geeksforgeeks.org/javascript-date-getday-method/) | Returns a number representing the weekday (0-6) to |
| [**parse()**](https://www.geeksforgeeks.org/javascript-date-parse-method/) | Returns the number of milliseconds since January 1, 1970 |
| [**setDate()**](https://www.geeksforgeeks.org/javascript-date-setdate-method/) | Returns the current date as a number (1-31) |
| [**setTime()**](https://www.geeksforgeeks.org/javascript-date-settime-function/) | Sets the time (milliseconds since January 1, 1970) |

// Here a date has been assigned by creating a date obj

let DateObj = new Date("October 13, 1996 05:35:32");

// getDate()

let A = DateObj.getDate();

// Printing date of the month

console.log(A);

// getTime()

let B = DateObj.getTime();

// Printing time in milliseconds.

console.log(B);

// getMinutes()

let minutes = DateObj.getMinutes();

// Printing minute.

console.log(minutes);

// getFullYear()

let C = DateObj.getFullYear();

// Printing year

console.log(C);

// getDay()

let Day = DateObj.getDay();

// Printing day of the week

console.log("Number of Day: " + Day);

// setDate

DateObj.setDate(15);

let D = DateObj.getDate();

// Printing new date of the month

console.log(D);

// parse(), taking wrong date string as input.

let date = "February 48, 2018 12:30 PM";

// calling parse function.

let msec = Date.parse(date);

console.log(msec);

## ****DOM****

[DOM](https://www.geeksforgeeks.org/dom-document-object-model/) stands for **Document Object Model.** It defines the logical structure of documents and the way a document is accessed and manipulated. JavaScript can not understand the tags in HTML document but can understand objects in DOM.Below are some of the methods provided by JavaScript to manipulate these nodes and their attributes in the DOM:

| **Method** | **Description** |
| --- | --- |
| [**appendChild()**](https://www.geeksforgeeks.org/html-dom-appendchild-method/) | Adds a new child node as the last child node. |
| [**cloneNode()**](https://www.geeksforgeeks.org/html-dom-clonenode-method/) | Duplicates an HTML element. |
| [**hasAttributes()**](https://www.geeksforgeeks.org/html-dom-hasattributes-method/) | Returns true If an element has any attributes otherwise,returns false. |
| [**removeChild()**](https://www.geeksforgeeks.org/html-dom-removechild-method/) | Removes a child node from an element using the Child() method. |
| [**getAttribute()**](https://www.geeksforgeeks.org/html-dom-getattribute-method/) | Returns the value of an element node’s provided attribute. |
| [**getElemetsByTagName()**](https://www.geeksforgeeks.org/html-dom-getelementsbytagname-method/) | Returns a list of all child elements. |
| [**isEqualNode()**](https://www.geeksforgeeks.org/html-dom-isequalnode-method/) | Determines whether two elements are same. |

<!DOCTYPE html>

<html>

<head>

/\* CSS is used to make the output looks good \*/

<style>

#sudo {

border: 1px solid green;

background-color: green;

margin-bottom: 10px;

color: white;

font-weight: bold;

}

h1,

h2 {

text-align: center;

color: green;

font-weight: bold;

}

</style>

</head>

<body>

<h1>GeeksforGeeks</h1>

<h2>DOM appendChild() Method</h2>

<div id="sudo">

The Good Website is learning for Computer Science is-

</div>

<button onclick="geeks()">Submit</button>

<br />

<div style="border: 3px solid green">

<h1>GeeksforGeeks</h1>

<h2>A computer science portal for geeks</h2>

</div>

<h2>DOM cloneNode() Method</h2>

<button onclick="nClone()">

Click here to clone the above elements.

</button>

<br />

<h2>DOM hasAttributes() Method</h2>

<p id="gfg">

Click on the button to check if that

body element has any attributes

</p>

<button type="button" onclick="hasAttr()">

Submit

</button>

<br />

<h2>DOM removeChild() Method</h2>

<p>Sorting Algorithm</p>

<ul id="listitem">

<li>Insertion sort</li>

<li>Merge sort</li>

<li>Quick sort</li>

</ul>

<button onclick="Geeks()">

Click Here!

</button>

<br />

<h2>DOM getAttribute() Method</h2>

<br />

<button id="button" onclick="getAttr()">

Submit

</button>

<p id="gfg1"></p>

<br />

<h2>DOM getElementsByTagName()</h2>

<p>A computer science portal for geeks.</p>

<button onclick="getElememt()">

Try it

</button>

<h3>DOM isEqualNode() method .</h3>

<!-- 3 div elements-->

<div>GeeksforGeeks</div>

<div>GfG</div>

<div>GeeksforGeeks</div>

<button onclick="isequal()">

Check

</button>

<p id="result"></p>

<script>

function geeks() {

var node = document.createElement("P");

var t = document.createTextNode("GeeksforGeeks");

node.appendChild(t);

document.getElementById("sudo").appendChild(node);

}

function nClone() {

// Accessing div attribute using a variable geek

var geek = document.getElementsByTagName("DIV")[0];

// Cloning geek variable into a variable named clone

var clone = geek.cloneNode(true);

// Adding our clone variable to end of the document

document.body.appendChild(clone);

}

function hasAttr() {

var s = document.body.hasAttributes();

document.getElementById("gfg").innerHTML = s;

}

function Geeks() {

var doc = document.getElementById("listitem");

doc.removeChild(doc.childNodes[0]);

}

/\* Using getElementById \*/

function getAttr() {

var rk = document.getElementById("button").getAttribute("onClick");

document.getElementById("gfg1").innerHTML = rk;

}

/\* Using getElementsByTagName \*/

function getElement() {

var doc = document.getElementsByTagName("p");

doc[0].style.background = "green";

doc[0].style.color = "white";

}

/\* Cheacking the equality \*/

function isequal() {

var out = document.getElementById("result");

var divele = document.getElementsByTagName("div");

out.innerHTML +=

"element 1 equals element 1: " +

divele[0].isEqualNode(divele[0]) +

"<br/>";

out.innerHTML +=

"element 1 equals element 2: " +

divele[0].isEqualNode(divele[1]) +

"<br/>";

out.innerHTML +=

"element 1 equals element 3: " +

divele[0].isEqualNode(divele[2]) +

"<br/>";

}

</script>

</body>

</html>

## ****Numbers and Math****

JavaScript provides various properties and methods to deal with Numbers and Maths.

[Number Properties](https://www.geeksforgeeks.org/javascript-math-object/) include MAX value, MIN value, NAN(not a number), negative infinity , positive infinity etc. Some of the methods in JavaScript to deal with numbers are:

| **Method** | **Description** |
| --- | --- |
| [**valueOf()**](https://www.geeksforgeeks.org/number-valueof-method-javascript/) | Returns a number in its original form. |
| [**toString()**](https://www.geeksforgeeks.org/javascript-tostring-function/) | Returns string representation of an integer. |
| [**toFixed()**](https://www.geeksforgeeks.org/javascript-tofixed-function/) | Returns a number’s string with a specified number of decimals. |
| [**toPricision()**](https://www.geeksforgeeks.org/javascript-toprecision-function/) | Converts a number to a string of a specified length. |
| [**toExponential()**](https://www.geeksforgeeks.org/javascript-toexponential-function/) | Returns a rounded number written in exponential notation. |

<script type="text/javascript">

var num = 213;

var num1 = 213.3456711;

// JS valueof() Method

console.log("Output : " + num.valueOf());

// JS tostring() Method

console.log("Output : " + num.toString(2));

// JS tofixed() Method

console.log("Output : " + num1.toString(2));

// JS topricision() Method

console.log("Output : " + num1.toPrecision(3));

// JS toexponential() Method

console.log("Output : " + num1.toExponential(4));

</script>

Javascript provides math object which is used to perform mathematical operations on numbers. There are many math object properties which include euler’s number, PI, square root, logarithm. Some of the methods in JavaScript to deal with math properties are:

| **Method** | **Description** |
| --- | --- |
| [**max(x,y,z…n)**](https://www.geeksforgeeks.org/javascript-math-max-method/) | Returns the highest-valued number |
| [**min(x,y,z…n)**](https://www.geeksforgeeks.org/javascript-math-min-method/) | Returns the lowest-valued number |
| [**exp(x)**](https://www.geeksforgeeks.org/javascript-math-exp-method/) | Returns x’s exponential value. |
| [**log(x)**](https://www.geeksforgeeks.org/javascript-math-log-method/) | Returns the natural logarithm (base E) of x. |
| [**sqrt(x)**](https://www.geeksforgeeks.org/javascript-math-sqrt-method/) | Returns x’s square root value. |
| [**pow(x,y)**](https://www.geeksforgeeks.org/javascript-math-pow-method/) | Returns the value of x to the power of y |
| [**round(x)**](https://www.geeksforgeeks.org/javascript-math-round-function-2/) | Rounds the value of x to the nearest integer |
| [**sin(x)**](https://www.geeksforgeeks.org/javascript-math-sin-method/) | Finds the sine value of x(x is in radians). |
| [**tan(x)**](https://www.geeksforgeeks.org/javascript-math-tan-method/) | Finds the angle’s(x) tangent value. |

<script>

document.getElementById("GFG").innerHTML =

"Math.LN10: " + Math.LN10 + "<br>" +

"Math.LOG2E: " + Math.LOG2E + "<br>" +

"Math.Log10E: " + Math.LOG10E + "<br>" +

"Math.SQRT2: " + Math.SQRT2 + "<br>" +

"Math.SQRT1\_2: " + Math.SQRT1\_2 + "<br>" +

"Math.LN2: " + Math.LN2 + "<br>" +

"Math.E: " + Math.E + "<br>" +

"Math.round: " + Math.round(5.8) + "<br>" +

"Math.PI: " + Math.PI + "<br>" +

"

< p > <b>Math.sin(90 \* Math.PI / 180):</b> " +

Math.sin(90 \* Math.PI / 180) + "</p>

" +

"

< p > <b>Math.tan(90 \* Math.PI / 180):</b> " +

Math.tan(90 \* Math.PI / 180) + "</p>

" +

"

< p > <b>Math.max(0, 150, 30, 20, -8, -200):</b> " +

Math.max(0, 150, 30, 20, -8, -200) + "</p>

" +

"

< p > <b>Math.min(0, 150, 30, 20, -8, -200):</b> " +

Math.min(0, 150, 30, 20, -8, -200) + "</p>

" +

"

< p > <b>Math.pow(3,4):</b> " + Math.pow(3, 4) + "</p >

";

</script>

## ****Events****

Javascript has [events](https://www.geeksforgeeks.org/javascript-events/) to provide a dynamic interface to a webpage. When a user or browser manipulates the page events occur. These events are hooked to elements in the Document Object Model(DOM). Some of the events supported by JavaScript:

| **Events** | **Description** |
| --- | --- |
| [**onclick()**](https://www.geeksforgeeks.org/html-dom-onclick-event/) | Triggers an event when an element is clicked. |
| [**onkeyup()**](https://www.geeksforgeeks.org/html-dom-onkeyup-event/) | Executes instructions whenever a key is released after pressing. |
| [**onmouseover()**](https://www.geeksforgeeks.org/html-onmouseover-event-attribute/) | Triggers an event when mouse pointer is hovered over an element |
| [**onmouseout()**](https://www.geeksforgeeks.org/html-onmouseout-event-attribute/) | Triggers an event when mouse pointer is moved away from an element. |
| [**onchange()**](https://www.geeksforgeeks.org/html-dom-onchange-event/) | Detects the change in value of any element listing to this event. |
| [**onload()**](https://www.geeksforgeeks.org/html-dom-onload-event/) | Evokes an event when an element is completely loaded. |
| [**onfocus()**](https://www.geeksforgeeks.org/html-dom-onfocus-event/) | Triggers when an aspect is brought into focus. |
| [**onblur()**](https://www.geeksforgeeks.org/html-dom-onblur-event/) | Evoked an event when an element loses focus. |
| [**onsubmit()**](https://www.geeksforgeeks.org/html-dom-onsubmit-event/) | Evokes an event when a form is submitted |
| [**ondrag()**](https://www.geeksforgeeks.org/html-ondrag-event-attribute/) | Invokes an event when an element is dragged. |
| [**oninput()**](https://www.geeksforgeeks.org/html-oninput-event-attribute/) | Triggers when an input field gets any value. |

<!DOCTYPE html>

<html>

<head>

/\* CSS is used to make the output looks good \*/

<style>

#geeks {

border: 1px solid black;

padding: 15px;

width: 60%;

}

h1 {

color: green;

}

</style>

<script>

function hiThere() {

alert("Hi there!");

}

function focused() {

var e = document.getElementById("inp");

if (confirm("Got it?")) {

e.blur();

}

}

/\* Mouseover event \*/

document.getElementById("hID").addEventListener("mouseover", over);

/\* Mouseoout event \*/

document.getElementById("hID").addEventListener("mouseout", out);

/\* Over on green \*/

function over() {

document.getElementById("hID").style.color = "green";

}

/\* Leaving Out Black \*/

function out() {

document.getElementById("hID").style.color = "black";

}

function Geeks() {

var x = document.getElementById("GFG").value;

document.getElementById("sudo").innerHTML = "Selected Subject: " + x;

}

/\* Success alert \*/

function Geek() {

alert("Form submitted successfully.");

}

function Function() {

document.getElementById("geeks").style.fontSize = "30px";

document.getElementById("geeks").style.color = "green";

}

</script>

</head>

<body>

<!-- onload event -->

<img onload="alert('Image completely loaded')" alt="GFG-Logo"

src="https://media.geeksforgeeks.org/wp-content/cdn-uploads/GeeksforGeeksLogoHeader.png" />

<br />

<!-- onclick event -->

<h2>onclick event</h2>

<button type="button" onclick="hiThere()" on>

Click me

</button>

<!-- onfocus event -->

<h2>onfocus event</h2>

<p>Take the focus into the input box below:</p>

<input id="inp" onfocus="focused()" />

<!-- onblur Event -->

<h2>onblur event</h2>

<p>

Write something in the input box and

then click elsewhere in the document

body.

</p>

<input onblur="alert(this.value)" />

<!-- onmouseover and onmouseout event -->

<h2 id="hID">onmouseover event</h2>

<h2>onchange Event</h2>

<p>Choose Subject:</p>

<select id="GFG" onchange="Geeks()">

<option value="Data Structure">

Data Structure

</option>

<option value="Algorithm">

Algorithm

</option>

<option value="Computer Network">

Computer Network

</option>

<option value="Operating System">

Operating System

</option>

<option value="HTML">

HTML

</option>

</select>

<p id="sudo"></p>

<!-- onsubmit event -->

<h2>onsubmit event</h2>

<form onsubmit="Geek()">

First Name:<input type="text" value="" />

<br />

Last Name:<input type="text" value="" />

<br />

<input type="submit" value="Submit" />

</form>

<!--ondrag event -->

<h2>ondrag event attribute</h2>

<div id="geeks" ondrag="Function()">

GeeksforGeeks: A computer science portal for geeks

</div>

</body>

</html>

## ****Error****

When executing JavaScript code, [errors](https://www.geeksforgeeks.org/javascript-errors-throw-and-try-to-catch/) will most definitely occur when the JavaScript engine encounters a syntactically invalid code. These errors can occur due to the fault from the programmer’s side or the input is wrong or even if there is a problem with the logic of the program. Javascript has a few statements to deal with these errors:

| **Statement** | **Description** |
| --- | --- |
| **try** | Tests a block of code to check for errors. |
| **catch** | Handles the error if any are present. |
| **throw** | Allows construction of new errors. |
| **finally** | Executes code after try and catch. |

<!DOCTYPE html>

<html>

<body>

<h2>

JavaScript throw try catch finally keywords

</h2>

<p>Please enter a number:</p>

<input id="demo" type="text" />

<button type="button" onclick="myFunction()">

Test Input

</button>

<p id="p01"></p>

<script>

function myFunction() {

const message = document.getElementById("p01");

message.innerHTML = "";

let x = document.getElementById("demo").value;

/\* Using try.. catch.. with conditions\*/

try {

if (x == "") throw "is empty";

if (isNaN(x)) throw "is not a number";

x = Number(x);

if (x > 20) throw "is too high";

if (x <= 20) throw "is too low";

} catch (err) {

message.innerHTML = "Input " + err;

} finally {

document.getElementById("demo").value = "";

}

}

</script>

</body>

</html>

## ****Window Properties****

The [window object](https://www.geeksforgeeks.org/properties-of-window-object/) is the topmost object of DOM hierarchy. Whenever a window appears on the screen to display the contents of document, the window object is created. To access the properties of the window object, you will specify object name followed by a period symbol (.) and the property name.

**Syntax:**

window.property\_name

The properties and methods of Window object that are commonly used are listed in the below tables:

| **Property** | **Description** |
| --- | --- |
| [**window**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/) | Returns the current window or frame. |
| [**screen**](https://www.geeksforgeeks.org/javascript-bom-window-screen/) | Returns the window’s Screen object. |
| [**toolbar**](https://www.geeksforgeeks.org/javascript-cheat-sheet-a-basic-guide-to-javascript/) | Creates a toolbar object, whose visibility can be toggled in the window. |
| [**Navigator**](https://www.geeksforgeeks.org/html-dom-window-navigator-property/) | Returns the window’s Navigator object. |
| [**frames[]**](https://www.geeksforgeeks.org/html-dom-window-frames-properties/) | Returns all <iframe> elements in the current window. |
| [**document**](https://www.geeksforgeeks.org/html-dom-window-document-property/) | Returns a reference to the document object. |
| [**closed**](https://www.geeksforgeeks.org/javascript-window-close-method/) | Boolean used to check whether the window is closed or not. |
| [**length**](https://www.geeksforgeeks.org/html-window-length-property/) | Represents the number of frames in the current window. |
| [**History**](https://www.geeksforgeeks.org/javascript-history-object/) | Provides the window’s History object. |

<!DOCTYPE html>

<html>

<body>

<h1>The Window properties</h1>

<h2>The origin Property</h2>

<p id="demo"></p>

<br />

<button type="button" onclick="getResolution();">

Get Resolution

</button>

<br />

<button type="button" onclick="checkConnectionStatus();">

Check Connection Status

</button>

<br />

<button type="button" onclick="getViews();">

Get Views Count</button>

<br />

<p>

<button onclick="closeWin()">

Close "myWindow"

</button>

</p>

<script>

// JS location property

let origin = window.location.origin;

document.getElementById("demo").innerHTML = origin;

// JS screen property

function getResolution() {

alert("Your screen is: " + screen.width + "x" + screen.height);

}

// JS toolbar property

var visible = window.toolbar.visible;

// JS navigator property

function checkConnectionStatus() {

if (navigator.onLine) {

alert("Application is online.");

} else {

alert("Application is offline.");

}

}

// JS history property

function getViews() {

alert(

"You've accessed " + history.length + " web pages in this session."

);

}

// JS close property

let myWindow;

function closeWin() {

if (myWindow) {

myWindow.close();

}

}

</script>

</body>

</html>

| **Method** | **Description** |
| --- | --- |
| [**alert()**](https://www.geeksforgeeks.org/html-window-alert-method/) | Shows a message and an OK button in an alert box. |
| [**print()**](https://www.geeksforgeeks.org/javascript-window-print-method/) | Prints the current window’s content. |
| [**blur()**](https://www.geeksforgeeks.org/javascript-window-blur-and-window-focus-method/) | Removes the current window’s focus. |
| [**setTimeout()**](https://www.geeksforgeeks.org/java-script-settimeout-setinterval-method/) | Evaluates an expression after a specified time interval. |
| [**clearTimeout()**](https://www.geeksforgeeks.org/javascript-cleartimeout-clearinterval-method/) | Removes the timer that was set with setTimeout() |
| [**setInterval()**](https://www.geeksforgeeks.org/java-script-settimeout-setinterval-method/) | Evaluates an expression at intervals defined by the user. |
| [**prompt()**](https://www.geeksforgeeks.org/javascript-window-prompt-method/) | Shows a conversation window asking for feedback from the visitor. |
| [**close()**](https://www.geeksforgeeks.org/javascript-window-close-method/) | Closes the currently open window. |
| [**focus()**](https://www.geeksforgeeks.org/javascript-window-blur-and-window-focus-method/) | Sets the current window’s focus. |
| [**resizeTo()**](https://www.geeksforgeeks.org/html-window-resizeto-method/) | Resizes the window to the width and height supplied. |

<!DOCTYPE html>

<html>

<head>

<title>JavaScript Window Methods</title>

/\* CSS is used to make the output looks good \*/

<style>

.gfg {

font-size: 36px;

}

form {

float: right;

margin-left: 20px;

}

</style>

</head>

<body>

<div class="gfg">JavaScript Window Methods</div>

<br />

<button onclick="windowOpen()">

JavaScript window Open

</button>

<button onclick="resizeWin()">

JavaScript window resizeTo

</button>

<button onclick="windowBlur()">

JavaScript window Blur

</button>

<button onclick="windowFocus()">

JavaScript window Focus

</button>

<button onclick="windowClose()">

JavaScript window Close

</button>

<br />

<br />

<p id="g"></p>

<form>

<button onclick="setTimeout(wlcm, 2000);">

Alert after 2 Second

</button>

<button onclick="geek()">Click me!</button>

<input type="button" value="Print" onclick="window.print()" />

</form>

<br /><br />

<button id="btn" onclick="fun()" style="color: green">

JavaScript Used setTimeOut

</button>

<button id="btn" onclick="stop()">

JavaScript clearTimeout

</button>

<script>

var gfgWindow;

// Function that open the new Window

function windowOpen() {

gfgWindow = window.open(

"https://www.geeksforgeeks.org/",

"\_blank",

"width=200, height=200"

);

}

// Function that Resize the open Window

function resizeWin() {

gfgWindow.resizeTo(400, 400);

gfgWindow.focus();

}

// Function that Closes the open Window

function windowClose() {

gfgWindow.close();

}

// Function that blur the open Window

function windowBlur() {

gfgWindow.blur();

}

// Function that focus on open Window

function windowFocus() {

gfgWindow.focus();

}

// Alert function

function wlcm() {

alert("Welcome to GeeksforGeeks");

}

// Prompt function

function geek() {

var doc = prompt("Please enter some text", "GeeksforGeeks");

if (doc != null) {

document.getElementById("g").innerHTML = "Welcome to " + doc;

}

}

// Function setTimeout and clearTimeout

var t;

function color() {

if (document.getElementById("btn").style.color == "blue") {

document.getElementById("btn").style.color = "green";

} else {

document.getElementById("btn").style.color = "blue";

}

}

function fun() {

t = setTimeout(color, 3000);

}

function stop() {

clearTimeout(t);

}

</script>

</body>

</html>

## Benefits of Using JavaScript Cheat Sheet

A JavaScript Cheat Sheet is an essential tool that simplifies the process of writing JavaScript code, helps your web applications function smoothly, and enables you to create interactive and dynamic web content.

**Here are some key benefits of a JavaScript Cheat Sheet:**

* **Efficient Web Development:** A JavaScript Cheat Sheet provides a quick reference guide for web developers, enabling faster and more efficient coding. It helps in reducing the time spent on searching for syntax or functions, thereby increasing productivity.
* **Comprehensive Function Reference:** The cheat sheet includes an extensive collection of JavaScript functions, methods, and properties, covering everything from basic variables and operators to complex objects and events. This makes it a valuable resource for both beginners and experienced developers.
* **Dynamic Content Creation:** With the inclusion of JavaScript functions in the cheat sheet, developers can create more interactive and dynamic web content. It aids in enhancing the user experience and engagement on web pages.
* **Interoperability:** JavaScript is a core technology of the web. A JavaScript Cheat Sheet can be beneficial when working with other web technologies like HTML, CSS, and various web development frameworks.
* **Optimized for Performance:** A well-structured JavaScript code using the correct functions and methods can significantly improve the performance of web applications. The cheat sheet can guide developers in creating performance-optimized scripts.
* **Event Handling:** The cheat sheet covers methods for handling user interactions and browser events, enabling more responsive and interactive web content.