# JavaScript CheatSheet


CODE HELP

In this Cheatsheet, we will cover the basics of JavaScript. We will provide examples to help you understand how JavaScript work and how to use them in your own web development projects. Whether you are a beginner or an experienced developer, this PDF can serve as a useful reference guide.


# JavaScript

JavaScript is a programming language that is widely used in web development. It is a client-side scripting language, which means that it is executed on the client side (in a web browser) rather than on the server side. JavaScript is used to create interactive web pages and is an essential component of modern web development. It is a high-level, dynamically typed language that is interpreted, which means that it is not compiled into machine code before it is run. JavaScript is used to add functionality to websites, such as handling user events (such as clicks and form submissions), animating page elements, and making asynchronous requests to servers. It is a versatile language that can be used for a wide range of applications, from simple scripts that add simple interactivity to websites to complex single-page applications.

## On page script:

```
<script type="text/javascript"> .... </script>
```

## Include external JS file:

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

## Delay - 1 second timeout:

```
setTimeout(function () {


}, 1000);
```

## Functions:

```
function addNumbers(a, b) {
return a + b; ;
}
x = addNumbers(1, 2);
```

## Edit DOM element:

```
document.getElementById("elementID").innerHTML = "Hello World!";
```

## Output:

```
console.log(a);          // write to the browser console.
document.write(a);       // write to the HTML.
alert(a);                // output in an alert box.
confirm("?");            // yes/no dialog, returns true/false depending on user click.
prompt("Age ?","0");     // input dialog box. (Initial Value = 0).
```

## Comments:

```
/* Multi line comment */
// One line
```

## Variables:

```
var v;                          // variable
var b = "CodeHelp";             // string
var c = "Code" + " " + "Help";  // = "Code Help"
var d = 1 + 4 + "5";            // = "55"
var e = [1,2,6,8];              // array
var f = true;                   // boolean
var g = /()/;                   // RegEx
var h = function(){};           // function object
const PI = 3.14;                // constant
var a = 7, b = 6, c = a + b;    // one line
let a = 'aaa';                  // block scope local variable
```

## Strict mode:

```
"use strict";          // Use strict mode to write secure code
x = 1;                 // Throws an error because variable is not declared
```

## Values:

```
false, true                    // Boolean
4, 3.14, 0b10011, 0xF6, NaN    // Number
"CodeHelp", 'Love'             // String
undefined, Infinity, null      // Special
```

## Basic Operators:

```
a = b + c - d;          // Addition, Subtraction.
a = b * (c / d);        // Multiplication, Division.
x = 10 % 4;             // Modulo, 10 / 4 Remainder = 2.
a++; b--;               // Postfix Increment and Decrement.
```

## Bitwise Operators:

| | | | | | |
|---|---|---|---|---|---|
| & | AND | 5 & 1 | (0101 & 0001) | 1 | (1) |
| \| | OR | 5 \| 1 | (0101 \| 0001) | 5 | (101) |
| ~ | NOT | ~ 5 | (~0101) | 10 | (1010) |
| ^ | XOR | 5 ^ 1 | (0101 ^ 0001) | 4 | (100) |
| << | Left Shift | 5 << 1 | (0101 << 1) | 10 | (1010) |
| >> | Right Shift | 5 >> 1 | (0101 >> 1) | 2 | (10) |
| >>> | Zero Fill Right Shift | 5 >>> 1 | (0101 >>> 1) | 2 | (10) |

## Arithmetic Operators:

```
x * (y + z)          // grouping
x = 4                // assignment
x == y               // equals
x != y               // unequal
x === y              // strict equal
x !== y              // strict unequal
x < y   x > y        // less and greater than
x <= y  x >= y       // less or equal, greater or equal
x += y               // x = x + y
x && y               // logical AND
x || y               // logical OR
!(x == y)            // logical NOT
x != y               // not equal
typeof x             // type of x
x << 2  x >> 3       // shifting
```

## Objects:

```
var student = {                                  // object name
        firstName: "Koushik",                    // list of properties and values
        lastName: "Sadhu",
        age: 20,
        height: 175,
        fullName: function() {                   // object function
                return this.firstName + " " + this.lastName;
        }
};
student.age = 19;                                // setting value
student[age]++;                                  // incrementing value
name = student.fullName();                       // call object function
```

## Strings:

```
var a = "Codehelp";
var b = 'I don\'t \n know';      // \n new line.
var len = a.length;              // string length.
a.indexOf("h");                  // find substring, -1 if doesn't contain.
a.lastIndexOf("e");              // last occurrence.
a.slice(3, 6);                   // cut out "ehe", negative values count from behind.
a.replace("help","love");        // find and replace, takes regular expressions.
a.toUpperCase();                 // convert to upper case.
a.toLowerCase();                 // convert to lower case.
a.concat(" ", str2);             // Codehelp + " " + str2.
a.charAt(4);                     // character at index 4: "h".
abc.split(" ");                  // splitting a string on space, and stores in an array.
```

## Numbers and Math:

```
var pi = 3.14;
pi.toFixed(0);                  // returns 3
pi.toFixed(2);                  // returns 3.14, working with money
pi.toPrecision(2)               // returns 3.1
pi.valueOf();                   // returns number
Number(true);                   // converts to number
Number(new Date())              // number of milliseconds since, 1970
parseInt("3 months");           // returns the first number 3
parseFloat("3.5 days");         // returns 3.5
Number.MAX_VALUE                // largest possible JS number
Number.MIN_VALUE                // smallest possible JS number
Number.NEGATIVE_INFINITY        // Minus Infinity
Number.POSITIVE_INFINITY        // Infinity
var pi = Math.PI;               // 3.141592653589793
Math.round(4.5);                // 5
Math.pow(2,8);                  // 256 - 2 to the power of 8
Math.sqrt(49);                  // 7 - square root
Math.abs(-3.14);                // 3.14 - absolute, positive value
Math.ceil(3.14);                // 4 - rounded up
Math.floor(3.99);               // 3 - rounded down
Math.sin(0);                    // 0 - sine
Math.cos(Math.PI);              // OTHERS: tan, atan, asin, acos,
Math.min(0, 3, -2, 2);          // -2 the lowest value
Math.max(0, 3, -2, 2);          // 3 the highest value
Math.log(1);                    // 0 natural logarithm
Math.exp(1);                    // 2.7182pow(E,x)
Math.random();                  // random number between 0 and 1
Math.floor(Math.random() * 5) + 1;  // random integer, from 1 to 5
```

## Array:

```
var flowers = ["Rose", "Sunflower", "Lotus", “Lily”];                    // Array Declaration
var flowers = new Array ("Rose", "Sunflower", "Lotus", “Lily”);   //Alternate method.
alert(flowers[1]);                                                  // access value at index.
dogs[0] = "Hibiscus";                                               // change the first item to “Hibiscus”.
for (var i = 0; i < flowers.length; i++) {                          // Accessing array element using loop.
        console.log(flowers[i]);
}
```

## Array Methods:

```
flowers.toString();                                  // convert the array to string.
flowers.join(" _ ");                                 // join between two array element.
flowers.pop();                                       // remove last element.
flowers.push("Tulip");                               // add new element to the end.
flowers [flowers.length] = "Tulip";                  // the same as push.
flowers.shift();                                     // remove first element.
flowers.unshift("Tulip");                            // add new element to the beginning.
delete.flowers[0];                                   // change element to undefined.
flowers.splice(2, 0, "ABC", "DEF");                  // add elements.
var f = flowers.concat(a,b);                         // join two arrays (a followed by b).
flowers.slice(1,4);                                  // elements from [1] to [4-1].
flowers.sort();                                      // sort string alphabetically.
flowers.reverse();                                   // sort string in descending order.
x.sort(function(a, b){return a - b});                        // numeric sort.
x.sort(function(a, b){return b - a});                        // numeric descending sort.
x.sort(function(a, b){return 0.5 - Math.random()});          // random order sorting.
```

## Regular Expressions:

```
var a = str.search(/CheatSheet/i);
```

## Modifiers:

- **i** perform case-insensitive matching.
- **g** perform a global match.
- **m** perform multiline matching.

## Patterns:

- **** Escape character
- **\d** find a digit
- **\s** find a whitespace character
- **\b** find match at beginning or end of a word
- **n+** contains at least one n
- **n*** contains zero or more occurrences of n
- **n?** contains zero or one occurrences of n
- **^** Start of string
- **$** End of string
- **\uxxxx** find the Unicode character
- **.** Any single character
- **(x | y)** x or y
- **( ... )** Group section
- **[xyz]** In range (x, y or z)
- **[0-9]** any of the digits between the brackets
- **[^xyz]** Not in range
- **\s** White space

- **a?** Zero or one of a
- **a*** Zero or more of a
- **a*?** Zero or more, ungreedy
- **a+** One or more of a
- **a+?** One or more, ungreedy
- **a{2}** Exactly 2 of a
- **a{2,}** 2 or more of a
- **a{,10}** Up to 10 of a
- **a{1,6}** 1 to 6 of a
- **a{4,6}?** 4 to 6 of a
- **[:punct:]** Any punctuation symbol
- **[:space:]** Any space character
- **[:blank:]** Space or tab

## If–Else Statements:

```
if ((age >= 10) && (age < 20)) {          // logical condition
    status = "Permitted.";                // executed if condition is true
}
else {                                    // else block is optional
    status = "Not Permitted.";            // executed if condition is false
}
```

## Switch Statement:

```
switch (day) {                          // Input is current day in numeric.
case 1:                                 // if (day == 1)
        text = "Monday";
        break;
case 2:                                 // if (day == 2)
        text = "Tuesday";
        break;
case 3:                                 // if (day == 3)
        text = "Wednesday";
        break;
case 4:                                 // if (day == 4)
        text = "Thursday";
        break;
case 5:                                 // if (day == 5)
        text = "Friday";
        break;
case 6:                                 // if (day == 6)
        text = "Saturday";
        break;
case 7:                                 // if (day == 7)
        text = "Sunday";
        break;
default:                                // else....
        text = "Please enter valid day number.";
   }
```

## For Loop:

```
for (var i = 0; i < 5; i++) {
        document.write(i + ": " + i*i+ "<br/>");
}
```

```
var sum = 0;
for (var i = 0; i < a.length; i++) {
        sum + = a[i];
}
```

```
html = "";
for (var i of stud) {
        html += "<li>" + i + "</li>";
}
```

## While Loop:

```
var i = 1;                          // initialize
while (i < 20) {                    // enters the cycle if statement is true
        i += i;                     // increment to avoid infinite loop
        document.write(i + ", ");   // output
}
```

## Do While Loop:

```
var i = 1;                          // initialize
do {                                // enters loop at least once
      i += 1;                       // increment to avoid infinite loop
document.write(i + ", ");           // output
} while (i < 20)                    // repeats loop, if statement is true
```

## Break Statement:

```
for (var i = 0; i < 20; i++) {
        if (i == 10) {
                break;                          // terminates the loop
        }
        document.write(i + ", ");
}
```

## Continue Statement:

```
for (var i = 0; i < 20; i++) {
        if (i == 10) {
                continue;                               // skips
        }
        document.write(i + ", ");
}
```

## Dates:

```
var d = new Date();
a = d.getDay();                          // getting the weekday
getDate();                               // day as a number (1-31)
getDay();                                // weekday as a number (0-6)
getFullYear();                           // four digit year (yyyy)
getHours();                              // hour (0-23)
getMilliseconds();                       // milliseconds (0-999)
getMinutes();                            // minutes (0-59)
getMonth();                              // month (0-11)
getSeconds();                            // seconds (0-59)
getTime();                               // milliseconds since 1970
d.setDate(d.getDate() + 7);              // adds a week to a date
setDate();                               // day as a number (1-31)
setFullYear();                           // year (optionally month and day)
setHours();                              // hour (0-23)
setMilliseconds();                       // milliseconds (0-999)
setMinutes();                            // minutes (0-59)
setMonth();                              // month (0-11)
setSeconds();                            // seconds (0-59)
setTime();                               // milliseconds since 1970)
```

## Global Functions:

```
eval();                          // executes a string as if it was scriptcode
String(44);                      // return string from number
(44).toString();                 // return string from number
Number("44");                    // return number from string
decodeURI(enc);                  // decode URI. Result: "page.asp"
encodeURI(uri);                  // encode URI. Result: "page.asp"
decodeURIComponent(enc);         // decode a URI component
encodeURIComponent(uri);         // encode a URI component
parseFloat();                    // returns floating point number of string
parseInt();                      // parses a string and returns an integer
isFinite();                      // is variable a finite, legal number
isNaN();                         // is variable an illegal number
```

## Events:

```
<button onclick="myFunction();"> Click here </button>
```

- **Mouse Events:**
  - onclick
  - oncontextmenu
  - ondblclick
  - onmousedown
  - onmouseenter
  - onmouseleave
  - onmousemove
  - onmouseover
  - onmouseout
  - onmouseup

- **Keyboard Events:**
  - onkeydown
  - onkeypress
  - onkeyup
- **Frame Events:**
  - onabort
  - onbeforeunload
  - onerror
  - onhashchange
  - onload
  - onpageshow
  - onpagehide
  - onresize,
  - onscroll
  - onunload
- **Form Events:**
  - onblur
  - onchange
  - onfocus
  - onfocusin
  - onfocusout
  - oninput
  - oninvalid
  - onreset
  - onsearch
  - onselect
  - onsubmit

- **Drag Events:**
  - ondrag
  - ondragend
  - ondragenter
  - ondragleave
  - ondragover
  - ondragstart
  - ondrop
- **Clipboard Events:**
  - oncopy
  - oncut
  - onpaste
- **Media Events:**
  - onabort
  - oncanplay
  - oncanplaythrough
  - ondurationchange
  - onended
  - onerror
  - onloadeddata
  - onloadedmetadata
  - onloadstart
  - onpause
  - onplay
  - onplaying
  - onprogress
  - onratechange
  - onseeked
  - onseeking
  - onstalled
  - onsuspend

- ontimeupdate
- onvolumechange
- onwaiting

- **Animation Events:**
  - Animationend
  - Animationiteration
  - Animationstart

- **Miscellaneous Events:**
  - transitioned
  - onmessage
  - onmousewheel
  - ononline
  - onoffline
  - onpopstate
  - onshow
  - onstorage
  - ontoggle
  - onwheel
  - ontouchcancel
  - ontouchmove

## Errors:

```
try {                                   // try block of code
        undefinedFunction();
}
catch(err) {                            // block to handle errors
        console.log(err.message);
}
```

## Throw error:

```
throw "My error message";                    // throws a text
```

## Input validation:

```
const input = document.getElementById("num");      // get input element
try {
    const x = input.value;                         // get input value
    if (x == "") throw "empty";                    // error cases
    if (isNaN(x)) throw "not a number";
    x = Number(x);
    if (x > 20) throw "too high";
} catch (err) {                                    // if there's an error
    console.log(`Input is ${err}`);                // output error
    console.error(err);                            // write the error in console
} finally {
    console.log("Done");              // executed regardless of the try catch block
}
```

## Error Names:

- **RangeError:** A number is "out of range".
- **ReferenceError:** An illegal reference has occurred.
- **SyntaxError:** A syntax error has occurred.
- **TypeError:** A type error has occurred.
- **URIError:** An encodeURI() error has occurred.

## **JSON:** JavaScript Object Notation

```
let str = '{"names":[' +                                    // create JSON object
'{"first":"Koushik","lastN":"Sadhu" },' +
'{"first":"Pranay","lastN":"Gupta" },' +
'{"first":"Shuvam","last":"Chodhury" }]}';
obj = JSON.parse(str);
console.log(obj.names[1].first);

// Send:

let myObj = { "name":"Nidhi", "age":20, "city":"Kolkata" };      // create object
let myJSON = JSON.stringify(myObj);                              // stringify
window.location = `demo.html?x=${myJSON}`;                       // sending to php

// Storing and retrieving:

let myObj = { "name":"Nidhi", "age":20, "city":"Kolkata" };
let myJSON = JSON.stringify(myObj);                              // storing data
localStorage.setItem("testJSON", myJSON);
let text = localStorage.getItem("testJSON");                     // retrieving data
let obj = JSON.parse(text);
document.write(obj.name);
```

## Promises:

```
function sum(a, b) {
   return new Promise((resolve, reject) => {
      setTimeout(() => {                                  // send the response after 1 second
         if (typeof a !== "number" || typeof b !== "number") {     // testing input types
         return reject(new TypeError("Inputs must be numbers"));
        }
   resolve(a + b);
   }, 1000);
 });
}
let myPromise = sum(10, 5);
myPromise.then((result) => {
console.log("10 + 5: ", result);
   return sum(null, "foo");                    // Invalid data and return another promise
 })
 .then(() => {                                 // Won't be called because of the error
 })
 .catch((err) => {
   console.error(err); // => Inputs must be numbers
});
```