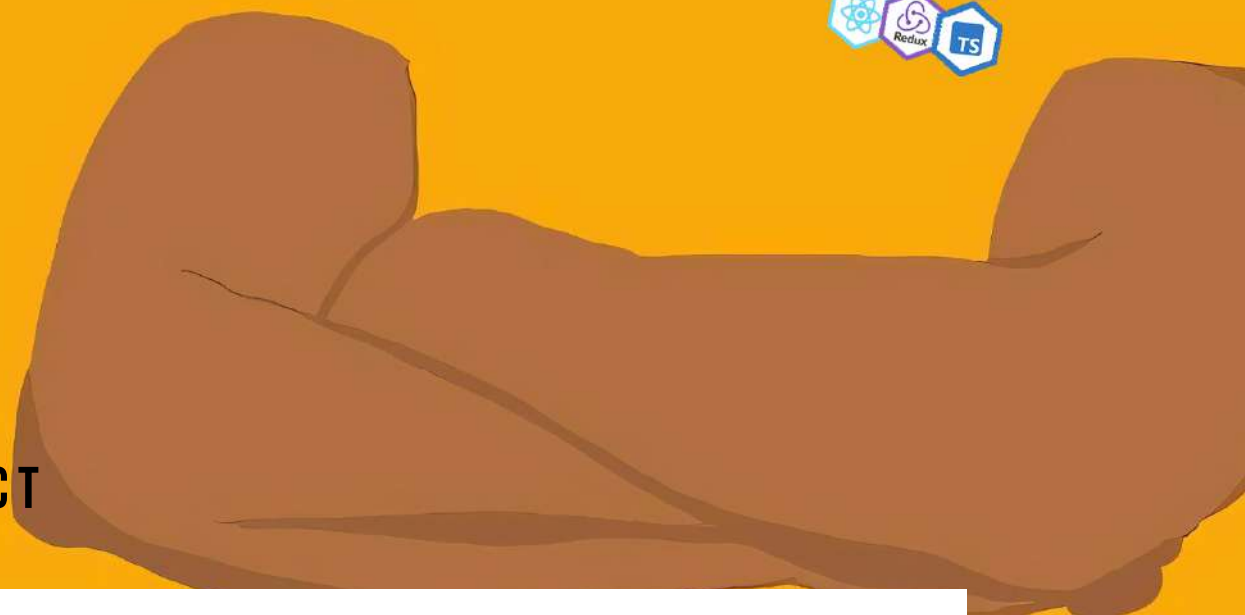
COMPLETE JAVASCRIPT

CHEATSHEET



BY

PAPA REACT

MEET SONNY

ALSO KNOWN AS PAPA REACT

> ***To truly succeed,***
> ***we must get Comfortable with the Uncomfortable***

Sonny Sangha

ssssangha

TABLE OF CONTENTS

STRINGS

STRING METHODS

CHARAT()

The charAt() method returns the character at the specified index in a string. The index of the first character is 0, the second character is 1, and so on.

```
1 var str = "HELLO WORLD";
2 var res = str.charAt(0);
3
4 console.log(res)
5 // output H
```

CHARCODEAT()

The charCodeAt() method returns the Unicode of the character at the specified index in a string.

```
1 var str = "HELLO WORLD";
2 var res = str.charCodeAt(0);
3
4 console.log(res)
5 // output 72
```

CONCAT()

The concat() method is used to join two or more strings. This method does not change the existing strings, but returns a new string containing the text of the joined strings.

```
1 var str1 = "Hello ";
2 var str2 = "world!";
3 var res = str1.concat(str2);
4
5 console.log(res)
6 // output Hello world!
```

ENDSWITH()

The endsWith() method determines whether a string ends with the characters of a specified string. This method returns true if the string ends with the characters, and false if not.

```
1 var str = "Hello world, welcome to the
  universe.";
2 var res = str.endsWith("universe.");
3
4 console.log(res)
5 // output true
```

FROMCHARCODE()

The fromCharCode() method converts Unicode values into characters. This is a static method of the String object, and the syntax is always String. fromCharCode()

```
var res = String.fromCharCode(65);

console.log(res)
// output A
```

INCLUDES()

The includes() method determines whether a string contains the characters of a specified string.

```
var str = "Hello world, welcome to the
universe.";
var res = str.includes("world");

console.log(res)
// output true
```

INDEXOF()

The indexOf() method returns the position of the first occurrence of a specified value in a string. This method returns -1 if the value to search for never occurs.

```
var str = "Hello world, welcome to the
universe.";
var res = str.indexOf("welcome");

console.log(res)
// output 13
```

LASTINDEXOF()

The lastIndexOf() method returns the position of the last occurrence of a specified value in a string. The string is searched from the end to the beginning, but returns the index starting at the beginning, at position 0.

```
var str = "Hello planet earth, you are a great
planet.";
var res = str.lastIndexOf("planet");

console.log(res)
// output 36
```

LOCALECOMPARE()

The localeCompare() method compares two strings in the current locale. The locale is based on the language settings of the browser.
Returns -1 if str1 is sorted before str2
Returns 0 if the two strings are equal
Returns 1 if str1 is sorted after str2

```
var str1 = "ab";
var str2 = "cd";
var res = str1.localeCompare(str2);

console.log(res)
// output -1
```

MATCH()

The match() method searches a string for a match against a regular expression, and returns the matches, as an Array object.

```
var str = "The rain in SPAIN stays mainly in
the plain";
var res = str.match(/ain/g);

console.log(res)
// output ["ain", "ain" , "ain"]
```

REPEAT()

The repeat() method returns a new string with a specified number of copies of the string it was called on.

```
var str = "Hello world!";
str.repeat(2);

console.log(str)
// output Hello world!Hello world!
```

REPLACE()

The replace() method searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced.

```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft",
"W3Schools");

console.log(res)
// output Visit W3Schools!
```

SEARCH()

The search() method searches a string for a specified value, and returns the position of the match.

```
var str = "Visit W3Schools!";
var res = str.search("W3Schools");

console.log(res)
// output 6
```

SLICE()

The slice() method extracts parts of a string and returns the extracted parts in a new string.

```
var str = "Hello world!";
var res = str.slice(0, 5);

console.log(res)
// output Hello
```

SPLIT()

The split() method is used to split a string into an array of substrings, and returns the new array.

```
var str = "How are you doing today?";
var res = str.split(" ");

console.log(res)
// output ["How", "are", "you", "doing",
"today?"]
```

STARTSWITH()

The startsWith() method determines whether a string begins with the characters of a specified string.

```
var str = "Hello world, welcome to the
universe.";
var res = str.startsWith("Hello");

console.log(res)
// output true
```

SUBSTR()

The substr() method extracts parts of a string, beginning at the character at the specified position, and returns the specified number of characters.

```
var str = "Hello world!";
var res = str.substr(1, 4);

console.log(res)
// output ello
```

SUBSTRING()

The substring() method extracts the characters from a string, between two specified indices, and returns the new sub string.

```
var str = "Hello world!";
var res = str.substring(1, 4);

console.log(res)
// output ell
```

TOLOCALELOWERCASE()

The toLocaleLowerCase() method converts a string to lowercase letters, according to the host's current locale.

```
var str = "Hello World!";
var res = str.toLocaleLowerCase();

console.log(res)
// hello world!
```

TOLOCALEUPPERCASE()

The toLocaleUpperCase() method converts a string to uppercase letters, according to the host's current locale.

```
var str = "Hello World!";
var res = str.toLocaleUpperCase();

console.log(res)
// HELLO WORLD!
```

TOLOWERCASE()

The toLowerCase() method converts a string to lowercase letters. It does not change the original string.

```
var str = "Hello World!";
var res = str.toLowerCase();

console.log(res)
// Output hello world!
```

TOUPPERCASE()

The toUpperCase() method converts a string to uppercase letters. It does not change the original string.

```
var str = "Hello World!";
var res = str.toUpperCase();

console.log(res)
// Output HELLO WORLD!
```

TRIM()

The trim() method removes whitespace from both sides of a string. It does not change the original string.

```
let str = "   Hello World   ";
console.log(str);
// Output: "   Hello world   "

let res = str.trim();
console.log(res);
// Output: Hello world
```

ARRAYS

ARRAY PROPERTIES

CONSTRUCTOR

In JavaScript, the constructor property returns the constructor function for an object. For JavaScript arrays the constructor property returns function Array() { [native code] }

```
1 var fruits = ['Banana', 'Orange', 'Apple'];
2 fruits.constructor;
3
4 // Returns
5 // function Array() { [native code] }
```

LENGTH

The length property sets or returns the number of elements in an array.

```
1 var fruits = ["Banana", "Orange", "Apple"];
2 fruits.length;
3 // Return 3
4
5 var vegetables = [];
6 vegetables.length = 3;
7 // sets array length to 3
```

PROTOTYPE

The prototype constructor allows you to add new properties and methods to the Array() object.

```
1 // Make a new array method
2 Array.prototype.myUcase = function() {
3   for (i = 0; i < this.length; i++) {
4     this[i] = this[i].toUpperCase();
5   }
6 };
7 // Make an array, then call the myUcase method:
8 var fruits = ["Banana", "Orange", "Apple"];
9 fruits.myUcase();
```

ARRAY METHODS

CONCAT()

The concat() method is used to join two or more arrays. This method does not change the existing arrays, but returns a new array, containing the values of the joined arrays.

```
var hege = ["Cecilie", "Lone"];
var stale = ["Emil", "Tobias", "Linus"];
var children = hege.concat(stale);

// Returns a new array
```

COPYWITHIN()

copies array elements to another position in the array, overwriting the existing values. This method will never add more items to the array. Note: this method overwrites the original array.

```
var fruits = ["Banana", "Orange", "Apple"];
fruits.copyWithin(2, 0);

// overwrites the original array
```

ENTRIES()

The entries() method returns an Array Iterator object with key/value pairs..

```
var fruits = ["Banana", "Orange", "Apple"];
var f = fruits.entries();

/* returns new object
  {
    [0, "Banana"],
    [1, "Orange"],
    [2, "Apple"]
  }
*/
```

EVERY()

The every() method checks if all elements in an array pass a test (provided as a function).

```
var ages = [32, 33, 16, 40];

function checkAdult(age) {
  return age >= 18;
}

ages.every(checkAdult);

/* checks every element returns ->
false */
```

FILL()

The fill() method fills the specified elements in an array with a static value. You can specify the position of where to start and end the filling. If not specified, all elements will be filled.

```
var fruits = ["Banana", "Orange", "Apple"];
fruits.fill("Kiwi");

/* returns new array ->
[Kiwi,Kiwi,Kiwi,Kiwi] */
```

FILTER()

The filter() method creates an array filled with all array elements that pass a test (provided as a function).

```
var ages = [32, 33, 16, 40];

function checkAdult(age) {
  return age >= 18;
}

ages.filter(checkAdult)

/* returns new array ->
[32,33,40] */
```

FIND()

The find() method returns the value of the first element in an array that pass a test (provided as a function).

```
var ages = [3, 10, 18, 20];

function checkAdult(age) {
  return age >= 18;
}

ages.find(checkAdult)
/* returns fist accurance of value -> 18 */
```

FINDINDEX()

The findIndex() method returns the index of the first element in an array that pass a test (provided as a function).

```
var ages = [3, 10, 18, 20];

function checkAdult(age) {
  return age >= 18;
}

ages.findIndex(checkAdult)
/* returns fist index of value -> 2 */
```

FOREACH()

The forEach() method calls a function once for each element in an array, in order.

```
let numbers = [65, 44, 12, 4];
numbers.forEach(myFunction)

function myFunction(item, index, arr) {
  arr[index] = item * 10;
}

console.log(numbers);

// output => [650,440,120,40]
```

INCLUDES()

The includes() method determines whether an array contains a specified element..

```
//Check if an array includes "Banana", starting
the search at position 3
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var n = fruits.includes("Banana", 3);

console.log(n);

// Output false
```

INDEXOF()

The indexOf() method searches the array for the specified item, and returns its position.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var a = fruits.indexOf("Apple");

console.log(a);

// Output 2
```

ISARRAY()

The isArray() method determines whether an object is an array.Thinction returns true if the object is an array, and false if not.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];

var result = Array.isArray(fruits);

console.log(result)
// Output true
```

JOIN()

Convert the elements of an array into a string. The join() method returns the array as a string.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];

var energy = fruits.join(" and ");

console.log(energy)

// Output Banana and Orange and Apple and Mango
```

LASTINDEXOF()

The lastIndexOf() method searches the array for the specified item, and returns its position.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var a = fruits.lastIndexOf("Apple");

console.log(a)

// Output 2
```

MAP()

The map() method creates a new array with the results of calling a function for every array element.

```
var numbers = [4, 9, 16, 25];
var x = numbers.map(Math.sqrt)

console.log(x)

// Output [2,3,4,5]
```

POP()

The pop() method removes the last element of an array, and returns that element.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.pop();

console.log(fruits)

// Output ["Banana", "Orange", "Apple"]
```

PUSH()

The push() method adds new items to the end of an array, and returns the new length.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var length = fruits.push("Kiwi");

console.log(fruits)
console.log(length)

// Output
// ["Banana", "Orange", "Apple", "Mango",
"Kiwi"]
// 5
```

REDUCE()

The reduce() method executes a provided function for each value of the array (from left-to-right) and reduces the array to a single value.

```
var numbers = [15.5, 2.3, 1.1, 4.7];

function getSum(total, num) {
  return total + Math.round(num);
}

var result = numbers.reduce(getSum, 0);
console.log(result)

// Output 24
```

REDUCERIGHT()

The reduceRight() method executes a provided function for each value of the array (from right-to-left) and reduces the array to a single value.

```
var numbers = [175, 50, 25];
function myFunc(total, num) {
  return total - num;
}

var result = numbers.reduceRight(myFunc);
console.log(result)

// Output -200
```

REVERSE()

The reverse() method reverses the order of the elements in an array.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.reverse();

console.log(fruits)

// Output
["Mango","Apple","Orange","Banana"]
```

SOME()

The some() method checks if any of the elements in an array pass a test (provided as a function). It executes the function once for each element present in the array

```
var ages = [3, 10, 18, 20];

function checkAdult(age) {
  return age >= 18;
}

console.log(ages.some(checkAdult))
// Output true;
```

SLICE()

The slice() method selects the elements starting at the given start argument, and ends at, but does not include, the given end argument. It returns the selected elements in an array, as a new array object.

```
var fruits = ["Banana", "Orange", "Lemon",
"Apple", "Mango"];
var citrus = fruits.slice(1, 3);

console.log(citrus)

// Output ["Orange", "Lemon"];
```

SHIFT()

The shift() method removes the first item of an array.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.shift();

console.log(fruits)

// Output ["Orange", "Apple", "Mango"];
```

SORT()

The sort() method sorts the items of an array.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.sort();

console.log(fruits)
// Output
["Apple","Banana","Mango","Orange"];
```

SPLICE()

The splice() method adds/removes items to/from an array, and returns the removed item(s).

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");

console.log(fruits)
// Output
["Banana","Orange","Lemon","Kiwi","Apple","M
ango"];
```

TOSTRING()

The toString() method returns a string with all the array values, separated by commas.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var x = fruits.toString();

console.log(fruits)
// Output Banana,Orange,Apple,Mango
```

UNSHIFT()

The unshift() method adds new items to the beginning of an array, and returns the new length.

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
fruits.unshift("Lemon","Pineapple");

console.log(fruits)
// output
// ["Lemon", "Pineapple", "Banana", "Orange",
"Apple", "Mango"]
```

VALUEOF()

The valueOf() method returns the array. This method is the default method of the array object. Array.valueOf() will return the same as Array

```
var fruits = ["Banana", "Orange", "Apple",
"Mango"];
var v = fruits.valueOf();

console.log(v)
// output
// ["Banana", "Orange", "Apple", "Mango"]
```

CONSOLE METHODS

CONSOLE.ASSERT()

The console.assert() method writes a message to the console, but only if an expression evaluates to false.

```
1 console.assert(document.getElementById("demo"),
  "You have no element with ID 'demo'");
2
3 // Output
4 // Assertion failed: You have no element with
  ID 'demo'
```

CONSOLE.CLEAR()

The console.clear() method clears the console. It will also write a message in the console: “Console was cleared”.

```
1 console.clear();
2
3 // Output
4 // Console was cleared
```

CONSOLE.COUNT()

Writes to the console the number of times that particular console.count() is called. You can add a label that will be included in the console view.

```
1 console.count("myLabel");
2 console.count("myLabel");
3
4 // Output
5 // myLabel: 1
6 // myLabel: 2
```

CONSOLE.ERROR()

he console.error() method writes an error message to the console.The console is useful for testing purposes.

```
1 console.error("You made a mistake");
2
3 // Output
4 // You made a mistake
```

CONSOLE.GROUP()

The console.group() method indicates the start of a message group.All messages will from now on be written inside this group.

```
1 console.log("Hello world!");
2 console.group();
3 console.log("Hello again, this time inside a
  group!");
4
5 // Output
6   Hello world!                              VM25:3
7 ▼ console.group                             VM25:4
8     Hello again, this time inside a group!  VM25:5
```

CONSOLE.GROUPCOLLAPSED()

The console.groupCollapsed() method indicates the start of a collapsed message group. Click the expand button to open the message group.

```
1 console.groupCollapsed();
2 console.log("Hello again, this time inside a
  collapsed group!");
3
4 // Output
5
6   Hello world!                              VM24:3
7 ▶ console.groupCollapsed                    VM24:4
```

CONSOLE.GROUPEND()

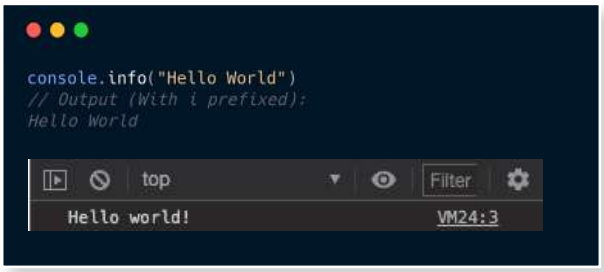
The console.groupEnd() method indicates the end of a message group.

CONSOLE.TABLE()

The console.table() method writes a table in the console view. The first parameter is required, and must be either an object, or an array, containing data to fill the table.

CONSOLE.INFO()

The console.info() method writes a message to the console.



CONSOLE.LOG()

The console.log() method writes a message to the console. The console is useful for testing purposes.

CONSOLE.TIME()

Use the console.time() method to start the timer. The console.timeEnd() method ends a timer, and writes the result in the console view.

CONSOLE.WARN()

The console.warn() method writes a warning to the console.

DOM EVENT PROPERTIES AND METHODS

MOUSEEVENT ALTKEY

The altKey property returns a Boolean value that indicates whether or not the “ALT” key was pressed when a mouse event was triggered.

```
if (event.altKey) {
  alert("The ALT key was pressed!");
} else {
  alert("The ALT key was NOT pressed!");
}
```

KEYBOARDEVENT ALTKEY

The altKey property returns a Boolean value that indicates whether or not the “ALT” key was pressed when a key event was triggered.

```
var x = document.getElementById("demo");
if (event.altKey) {
  x.innerHTML = "The ALT key was pressed!";
} else {
  x.innerHTML = "The ALT key was NOT pressed!";
}
```

ANIMATIONEVENT ANIMATIONNAME

The animationName property returns the name of the animation, when an animation event occurs.

```
var x = document.getElementById("myDIV");
x.addEventListener("animationstart",
 myStartFunction);

function myStartFunction(event) {
  this.innerHTML = "Animation-name is: " +
 event.animationName;
}
```

BUBBLES

The bubbles event property returns a Boolean value that indicates whether or not an event is a bubbling event.

```
// Find out if a specific event can bubble or
 not:
var x = event.bubbles;
```

MOUSEEVENT BUTTON

The button property returns a number that indicates which mouse button was pressed when a mouse event was triggered.

```
// Find out which mouse button that was pressed
 when a mouse event was triggered:
alert("You pressed button: " + event.button)
```

MOUSEEVENT BUTTONS

The buttons property returns a number that indicates which mouse button or mouse buttons were pressed when a mouse event was triggered.

```
// Find out which mouse button(s) that was
 pressed when a mouse event was triggered:
var x = event.buttons;
```

CANCELABLE

The cancelable event property returns a Boolean value indicating whether or not an event is a cancelable event.

```
// Find out if a specific event is cancelable:
var x = event.cancelable;
```

KEYBOARDEVENT CHARCODE

The charCode property returns the Unicode character code of the key that triggered the onkeypress event.

```
// The charCode property returns the Unicode
character code of the key that triggered the
onkeypress event.
var x = event.charCode;
```

MOUSEEVENT CLIENTX

The clientX property returns the horizontal coordinate (according to the client area) of the mouse pointer when a mouse event was triggered.

```
// Get the horizontal coordinate
var x = event.clientX;
// Get the vertical coordinate
var y = event.clientY;
var coor = "X coords: " + x + ", Y coords: " +
y;
```

MOUSEEVENT CLIENTY

The clientY property returns the vertical coordinate (according to the client area) of the mouse pointer when a mouse event was triggered.

```
// Get the horizontal coordinate
var x = event.clientX;
// Get the vertical coordinate
var y = event.clientY;
var coor = "X coords: " + x + ", Y coords: " +
y;

// output
// X coords: 142, Y coords: 99
```

KEYBOARDEVENT CODE

The code property returns the key that triggered the event.

```
// Return the key that was pressed:
var x = event.code;
```

CREATEEVENT()

The createEvent() method creates an event object.

```
var x = document.createEvent("MouseEvent");
x.initMouseEvent("mouseover", true, true,
window, 0, 0, 0, 0, 0, false, false, false,
false, 0, null);

document.getElementById("myDiv").dispatchEvent(
x);
```

MOUSEEVENT CTRLKEY

The ctrlKey property returns a Boolean value that indicates whether or not the "CTRL" key was pressed when a mouse event was triggered.

```
// Find out whether or not the "CTRL" key was
pressed when a mouse button is clicked:
if (event.ctrlKey) {
  alert("The CTRL key was pressed!");
} else {
  alert("The CTRL key was NOT pressed!");
}
```

CURRENTTARGET

The currentTarget event property returns the element whose event listeners triggered the event.

```
// Get the element whose event listeners
triggered a specific event:
alert(event.currentTarget);
```

INPUTEVENT DATA

The data property returns the character that was inserted with the event.

```
// Return the input data from a text field:
function myFunction(event) {
  var x = event.data;
}
```

DEFAULTPREVENTED

The defaultPrevented event property checks whether the preventDefault() method was called for the event.

```
document.getElementById("myAnchor").addEventLis
tener("click", function(event){
  event.preventDefault()
  alert("Was preventDefault() called: " +
event.defaultPrevented);
});
```

WHEELEVENT DELTAX

The deltaX property returns a positive value when scrolling to the right, and a negative value when scrolling to the left, otherwise 0.

```
// Return whether the user scrolls left or
right:
function myFunction(event) {
  var x = event.deltaX;
}
```

WHEELEVENT DELTAY

The deltaY property returns a positive value when scrolling down, and a negative value when scrolling up, otherwise 0.

```
// Return whether the user scrolls up or down:
function myFunction(event) {
  var y = event.deltaY;
}
```

WHEELEVENT DELTAZ

The deltaZ property returns a positive value when scrolling in, and a negative value when scrolling out, otherwise 0.

```
// Return whether the user scrolls in or out (along the z-axis):
function myFunction(event) {
  var z = event.deltaZ;
}
```

WHEELEVENT DELTAMODE

The deltaMode property returns a number representing the length unit of the scrolling values (deltaX, deltaY, and deltaZ).

```
function myFunction(event) {
  let z = event.deltaMode
}
```

UIEVENT DETAIL

The detail property returns a number with details about the event.

```
// Find out how many times the mouse was clicked in the same area:
var x = event.detail;
```

ANIMATIONEVENT ELAPSEDTIME

The elapsedTime property returns the number of seconds an animation has been running, when an animation event occurs.

```
var x = document.getElementById("myDIV");
x.addEventListener("animationiteration", myRepeatFunction);

function myRepeatFunction(event) {
  this.innerHTML = "Elapsed time: " + event.elapsedTime;
}
```

TRANSITIONEVENT ELAPSEDTIME

The elapsedTime property returns the number of seconds a transition has been running, when a transitionend event occurs.

```
document.getElementById("myDIV").addEventListener("transitionend", myFunction);

function myFunction(event) {
  this.innerHTML = "Transition lasted: " + event.elapsedTime + " seconds";
}
```

EVENTPHASE

The eventPhase event property returns a number that indicates which phase of the event flow is currently being evaluated.

```
The number is represented by 4 constants:

0. NONE
1. CAPTURING_PHASE - The event flow is in capturing phase
2. AT_TARGET - The event flow is in target phase, i.e. it is being evaluated at the event target
3. BUBBLING_PHASE - The event flow is in bubbling phase
```

MOUSEEVENT GETMODIFIERSTATE()

The getModifierState() method returns true if the specified modifier key was pressed, or activated.

```
// Is the Caps Lock key activated?
var x = event.getModifierState("CapsLock");
```

INPUTEVENT INPUTTYPE

The inputType property returns the type of change that was done by the event.

```
// Return the type of input the event was about:
function myFunction(event) {
  var x = event.inputType;
}
```

ISTRUSTED

The isTrusted event property returns a Boolean value indicating whether the event is trusted or not.

```
function myFunction(event) {
  if ("isTrusted" in event) {
    if (event.isTrusted) {
      alert ("The " + event.type + " event is trusted.");
    } else {
      alert ("The " + event.type + " event is not trusted.");
    }
  } else {
    alert ("The isTrusted property is not supported by your browser");
  }
}
```

KEYBOARDEVENT KEY

The key property returns the identifier of the key that was pressed when a key event occured.

```
// Get the keyboard button that was pressed when a key event occured:
var x = event.key;
```

KEYBOARDEVENT KEYCODE

The keyCode property returns the Unicode character code of the key that triggered the onkeypress event

```
// Get the Unicode value of the pressed keyboard key:
var x = event.keyCode;
```

KEYBOARDEVENT KEY

The key property returns the identifier of the key that was pressed when a key event occured.

```
// Get the keyboard button that was pressed when a key event occured:
var x = event.key;
```

KEYBOARDEVENT METAKEY

The metaKey property returns a Boolean value that indicates whether or not the "META" key was pressed when a key event was triggered.

```
if (event.metaKey) {
  alert("The META key was pressed!");
} else {
  alert("The META key was NOT pressed!");
}
```

MOUSEEVENT METAKEY

The metaKey property returns a Boolean value that indicates whether or not the "META" key was pressed when a mouse event was triggered.

```
var x = document.getElementById("demo");
if (event.metaKey) {
  x.innerHTML = "The META key was pressed!";
} else {
  x.innerHTML = "The META key was NOT
 pressed!";
}
```

KEYBOARDEVENT LOCATION

The location property returns a number that indicates the location of a key on the keyboard or device.

```
// Get the location of the key:
var x = event.location;
```

HASHCHANGEEVENT NEWURL

The newURL property returns the URL of the document, after the hash (anchor part) has been changed.

```
// When the hash has been changed, get the URL
 we are navigating to:
event.newURL;
```

MOUSEEVENT PAGEY

The pageY property returns the vertical coordinate (according to the document) of the mouse pointer when a mouse event was triggered.

```
// Output the coordinates of the mouse pointer
 when the mouse button is clicked on an element:

// Get the horizontal coordinate
var x = event.pageX;
// Get the vertical coordinate
var y = event.pageY;

var coor = "X coords: " + x + ", Y coords: " +
 y;
```

MOUSEEVENT PAGEX

The pageX property returns the horizontal coordinate (according to the document) of the mouse pointer when a mouse event was triggered.

```
// Output the coordinates of the mouse pointer
 when the mouse button is clicked on an element:

// Get the horizontal coordinate
var x = event.pageX;
// Get the vertical coordinate
var y = event.pageY;

var coor = "X coords: " + x + ", Y coords: " +
 y;
```

HASHCHANGEEVENT OLDURL

The oldURL property returns the URL of the document, before the hash (anchor part) was changed.

```
event.oldURL;
// output https://www.w3schools.com/
```

PAGETRANSITIONEVENT PERSISTED

The persisted property returns a Boolean value that indicates if the webpage is loaded directly from the server

```
function myFunction(event) {
  alert(event.persisted);
}
```

PREVENTDEFAULT()

The preventDefault() method cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur.

```
document.getElementById("myAnchor").addEventLis
tener("click", function(event){
  event.preventDefault()
});
```

TRANSITIONEVENT PROPERTYNAME

The propertyName property returns the name of the CSS property associated with the transition, when a transitionevent occurs.

```
document.getElementById("myDIV").addEventListen
er("transitionend", myFunction);

function myFunction(event) {
  this.innerHTML = "Property name is: " +
event.propertyName;
}
```

MOUSEEVENT RELATEDTARGET

The relatedTarget property returns the element related to the element that triggered the mouse event.

```
<p onmouseover="getRelatedElement(event)">Mouse
over this paragraph.</p>

<script>
function getRelatedElement(event) {
  alert("The cursor just exited the " +
event.relatedTarget.tagName + " element.");
}
</script>
```

MOUSEEVENT SHIFTKEY

The relatedTarget property returns the element related to the element that The shiftKey property returns a Boolean value that indicates whether or not the “SHIFT” key was pressed when a mouse event was triggered.triggered the focus/blur event.

```
if (event.shiftKey) {
  alert("The SHIFT key was pressed!");
} else {
  alert("The SHIFT key was NOT pressed!");
}
```

KEYBOARDEVENT SHIFTKEY

The shiftKey property returns a Boolean value that indicates whether or not the “SHIFT” key was pressed when a key event was triggered.

```
if (event.shiftKey) {
  x.innerHTML = "The SHIFT key was pressed!";
} else {
  x.innerHTML = "The SHIFT key was NOT
pressed!";
}
```

STOPIMMEDIATEPROPAGATION()

The stopImmediatePropagation() method prevents other listeners of the same event from being called.

```
var x = document.getElementById("myBtn");
x.addEventListener("click", myFunction);
x.addEventListener("click", someOtherFunction);

function myFunction(event) {
  alert ("Hello World!");
  event.stopImmediatePropagation();
}

// This function will not be executed
function someOtherFunction() {
  alert ("I will not get to say Hello World");
}
```

STOPPROPAGATION()

The stopPropagation() method prevents propagation of the same event from being called.

```
function func1(event) {
  alert("DIV 1");
  event.stopPropagation();
}
```

TARGET

The target event property returns the element that triggered the event.

```
// Get the element that triggered a specific
event:
alert(event.target);
```

TOUCHEVENT TARGETTOUCHES

The targetTouches property returns an array of Touch objects, one for each finger that is touching the current target element.

```
// Find out how many fingers that touches an element:
function countTouches(event) {
  var x = event.targetTouches.length;
}
```

TOUCHEVENT TOUCHES

The touches property returns an array of Touch objects, one for each finger that is currently touching the surface.

```
// Find out how many fingers that touches the surface:
function countTouches(event) {
  var x = event.touches.length;
}
```

TYPE

The type event property returns the type of the triggered event.

```
// Return the type of event that was triggered:
var x = event.type;
```

KEYBOARDEVENT WHICH

The which property returns the Unicode character code of the key that triggered the onkeypress event,

```
// Get the Unicode value of the pressed keyboard key:
alert("You pressed button: " + event.which);
```

TIMESTAMP

The timeStamp event property returns the number of milliseconds from the document was finished loading until the specific event was created.

```
// Get the number of milliseconds since this document was loaded:
var n = event.timeStamp;
```

TRANSITIONEND EVENT

The transitionend event occurs when a CSS transition has completed.

```
// Code for Safari 3.1 to 6.0
document.getElementById("myDIV").addEventListener("webkitTransitionEnd", myFunction);

// Standard syntax
document.getElementById("myDIV").addEventListener("transitionend", myFunction);
```

MOUSEEVENT WHICH

The which property returns a number that indicates which mouse button was pressed when a mouse event was triggered.

```
// Find out which mouse button that was pressed when a mouse event was triggered:
alert("You pressed button: " + event.which);
```

VIEW

The view event property returns a reference to the Window object where the event occured.

```
// Get the event view:
var x = event.view;
```

SCREEN OBJECT

AVAILHEIGHT

The availHeight property returns the height of the user's screen, in pixels, minus interface features like the Windows Taskbar.

```
var x = "Available Height: " +
screen.availHeight;

console.log(x);

// output
// Available Height: 768
```

AVAILWIDTH

The availWidth property returns the width of the user's screen, in pixels, minus interface features like the Windows Taskbar.

```
var x = "Available Width: " +
screen.availWidth;

console.log(x);

// output
// Available Width: 1024
```

COLORDEPTH

The colorDepth property returns the bit depth of the color palette for displaying images (in bits per pixel).

```
var x = "Color Depth: " + screen.colorDepth +
  " bits per pixel";

console.log(x);
// output
// Color Depth: 24 bits per pixel
```

HEIGHT

The height property returns the total height of the user's screen, in pixels.

```
var x = "Total Height: " + screen.height;

console.log(x);
// output
// Total Height: 1050px
```

PIXELDEPTH

The pixelDepth property returns the color resolution (in bits per pixel) of the visitor's screen.

```
var x = "Color resolution: " +
screen.pixelDepth + " bits per pixel";

console.log(x);
// output
// Color resolution: 24 bits per pixel
```

WIDTH

The width property returns the total width of the user's screen, in pixels.

```
var x = "Total Width: " + screen.width + "px";

console.log(x);
// output
// Total Width: 1680px
```

BROWSER WINDOW

WINDOW PROPERTIES

CLOSED

The closed property returns a Boolean value indicating whether a window has been closed or not.The closed property returns a Boolean value indicating whether a window has been closed or not.

```
var myWindow;

// Open the window
myWindow = window.open("", "myWindow",
"width=400, height=200");

// close the window
myWindow.close();

console.log(myWindow.closed)
// Output true
```

FRAMEELEMENT

The frameElement property returns the <iframe> element in which the current window is inserted. If the document window is not placed within an <iframe>, the return value of this property is null.

```
var frame = window.frameElement;  // Get the
<iframe> element of the window

if (frame) {   // If the window is in an
<iframe>, change its source
  frame.src = "https://www.w3schools.com/";
}

console.log(frame)
// Output <iframe> ...</iframe>
```

FRAMES

The frames property returns an array-like object, which represents all <iframe> elements in the current window. The <iframe> elements can be accessed by index numbers. The index starts at 0.

```
<iframe src="https://www.w3schools.com">
</iframe>
<iframe src="https://www.w3schools.com">
</iframe>

<script>
window.frames[0].location =
"https://www.w3schools.com/jsref/";
</script>

// Output -> This will chnage the first iframe
to the source https://www.w3schools.com/jsref/"
```

INNERWIDTH AND INNERHEIGHT

The innerWidth property returns the width of a window's content area.
The innerHeight property returns the height of a window's content area.
These properties are read-only.

```
var w = window.innerWidth;
var h = window.innerHeight;

console.log("Width: " + w);
console.log("Height: " + h);

// Output
// Width: 825
// Height: 765
```

LENGTH

The length property returns the number of <iframe> elements in the current window.

```
<iframe src="https://www.cnn.com"></iframe>
<iframe src="https://www.bbc.com"></iframe>
<iframe src="https://www.nytimes.com"></iframe>

var res = window.length;
console.log(res);

// Output 3
```

LOCALSTORAGE

The localStorage and sessionStorage properties allow to save key/value pairs in a web browser.

```
localStorage.setItem("lastname", "Smith");

var res = localStorage.getItem("lastname");

console.log(res);

// Output Smith
```

NAME

The name property sets or returns the name of the window. This property is often used to modify the name of a window,

```
var myWindow = window.open("", "MsgWindow",
"width = 200, height = 100");

console.log("This window's name is: " +
myWindow.name);

// Output This window's name is: MsgWindow
```

PARENT

The parent property returns the parent window of the current window.

```
//Change the background-color of an <iframe>
element's parent document:

parent.document.body.style.backgroundColor =
"red";
```

OUTERHEIGHT AND OUTERWIDTH

The outerWidth property returns the outer width of the browser window, including all interface elements (like toolbars/scrollbars).

```
var w = window.outerWidth;
var h = window.outerHeight;

console.log("Width: " + w);
console.log("Height: " + h);

// Output
// Width: 1680
// Height: 1050
```

OPENER

The opener property returns a reference to the window that created the window.

```
// Open a new window
var myWindow = window.open("", "myWindow",
"width=200, height=100");

// Write some text in the new window
myWindow.document.write("<p>This is
'myWindow'</p>");

// Write some text in the window that created
the new window
myWindow.opener.document.write("<p>This is the
source window!</p>");
```

PAGEXOFFSET AND PAGEYOFFSET

The pageXOffset and pageYOffset properties returns the pixels the current document has been scrolled from the upper left corner of the window, horizontally and vertically.

```
window.scrollBy(100, 100);
alert(window.pageXOffset + window.pageYOffset);

// Output
// pageXOffset: 100, pageYOffset: 100
```

SCREENX AND SCREENY

The screenX and screenY properties returns the x (horizontal) and y (vertical) coordinates of the window relative to the screen.

```
var myWindow = window.open("", "myWin");
myWindow.document.write("<p>This is 'myWin'");

console.log("ScreenLeft: " + myWindow.screenX);
console.log("ScreenTop: " + myWindow.screenY);

// Output
// This is 'myWin'
// ScreenX: 0
//ScreenY: 0
```

SCREENLEFT AND SCREENTOP

The screenLeft and screenTop properties returns the x (horizontal) and y (vertical) coordinates of the window relative to the screen.

```
var myWindow = window.open("", "myWin");

myWindow.document.write("<p>This is 'myWin'");

console.log("ScreenLeft: " +
  myWindow.screenLeft);

console.log("ScreenTop: " +
  myWindow.screenTop);

// Output
// This is 'myWin'
// ScreenLeft: 0
// ScreenTop: 0
```

WINDOW PROPERTIES

ALERT()

The alert() method displays an alert box with a specified message and an OK button.

```
alert("Hello! I am an alert box!!");

// The alert() method displays an alert box
  with a specified message and an OK button.
```

ATOB()

The atob() method decodes a base-64 encoded string. This method decodes a string of data which has been encoded by the btoa() method.

```
var str = "Hello World!";
var enc = window.btoa(str);
var dec = window.atob(enc);

var res = "Encoded String: " + enc + "<br>" +
  "Decoded String: " + dec;

// output
// Encoded String: SGVsbG8gV29ybGQh
// Decoded String: Hello World!
```

BLUR()

The blur() method removes focus from the current window.

```
// Opens a new window
var myWindow = window.open("", "", "width=200,
  height=100");

// Some text in the new window
myWindow.document.write("<p>A new window!
  </p>");

// Assures that the new window does NOT get
  focus
myWindow.blur();
```

BTOA()

The btoa() method encodes a string in base-64.

```
var str = "Hello World!";
var enc = window.btoa(str);

var res = "Encoded String: " + enc;

// output
// The original string: Hello World!
// Encoded String: SGVsbG8gV29ybGQh
```

CLEARINTERVAL()

The clearInterval() method clears a timer set with the setInterval() method. The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

```
<button onclick="myStopFunction()">Stop time</button>

<script>
var myVar = setInterval(myTimer, 1000);

function myTimer() {
  var d = new Date();
  var t = d.toLocaleTimeString();
  document.getElementById("demo").innerHTML = t;
}

function myStopFunction() {
  clearInterval(myVar);
}
</script>
```

CLEARTIMEOUT()

The clearTimeout() method clears a timer set with the setTimeout() method.

```
var myVar;

function myFunction() {
  myVar = setTimeout(function(){ alert("Hello"); }, 3000);
}

function myStopFunction() {
  clearTimeout(myVar);
}
```

CLOSE()

The close() method closes the current window.

```
// Opens a new window
var myWindow = window.open("", "myWindow", "width=200, height=100");

// Closes the new window
myWindow.close();
```

CONFIRM()

The confirm() method displays a dialog box with a specified message, along with an OK and a Cancel button.

```
confirm("Press a button!");
// The confirm() method displays a dialog box with a specified message, along with an OK and a Cancel button.
```

FOCUS()

The focus() method sets focus to the current window.

```
// Opens a new window
var myWindow = window.open("", "", "width=200, height=100");
// Some text in the new window
myWindow.document.write("<p>A new window!</p>");
// Assures that the new window gets focus
myWindow.focus();
```

GETSELECTION()

Returns a Selection object representing the range of text selected by the user

```
getSelection()
// Returns a Selection object representing the range of text selected by the user
```

GETCOMPUTEDSTYLE()

The getComputedStyle() method gets all the actual (computed) CSS property and values of the specified element.

```
var elem = document.getElementById("test");

// Get the background color of the element
var theCSSprop = window.getComputedStyle(elem,
null).getPropertyValue("background-color");

console.log(theCSSprop);
// Output rgb(173, 216, 230)
```

MATCHMEDIA()

The window.matchMedia() method returns a MediaQueryList object representing the results of the specified CSS media query string.

```
if (window.matchMedia("(max-width:
700px)").matches) {
  /* The viewport is less than, or equal to,
700 pixels wide */
} else {
  /* The viewport is greater than 700 pixels
wide */
}
```

MOVEBY()

The moveBy() method moves a window a specified number of pixels relative to its current coordinates.

```
// Opens a new window
function openWin() {
  myWindow = window.open('', '', 'width=200,
height=100');
}

function moveWin() {
  // Moves the new window
  myWindow.moveBy(250, 250);
  // Sets focus to the new window
  myWindow.focus();
}
```

MOVETO()

The moveTo() method moves a window's left and top edge to the specified coordinates.

```
// Opens a new window
myWindow = window.open('', '', 'width=200,
height=100');

// Moves the new window
myWindow.moveTo(500, 100);

// Sets focus to the new window
myWindow.focus();
```

OPEN()

The open() method opens a new browser window, or a new tab, depending on your browser settings and the parameter values.

```
window.open("https://www.w3schools.com");
// The open() method opens a new browser
window, or a new tab, depending on your browser
settings and the parameter values.
```

PRINT()

The print() method opens the Print Dialog Box, which lets the user to select preferred printing options.

```
window.print();
// The print() method opens the Print Dialog
Box, which lets the user to select preferred
printing options.
```

PROMPT()

The prompt() method displays a dialog box that prompts the visitor for input.

```
var person = prompt("Please enter your name",
"Harry Potter");

if (person != null) {
  document.getElementById("demo").innerHTML =
  "Hello " + person + "! How are you today?";
}
```

RESIZEBY()

The resizeBy() method resizes a window by the specified amount, relative to its current size.

```
// Opens a new window
myWindow = window.open("", "", "width=100,
height=100");

// Resizes the new window
myWindow.resizeBy(250, 250);

// Sets focus to the new window
myWindow.focus();
```

RESIZETO()

The resizeTo() method resizes a window to the specified width and height.

```
// Opens a new window
myWindow = window.open("", "", "width=100,
height=100");

// Resizes the new window
myWindow.resizeTo(250, 250);

// Sets focus to the new window
myWindow.focus();
```

SCROLLBY()

The scrollBy() method scrolls the document by the specified number of pixels.

```
// Scroll 100px to the right
window.scrollBy(100, 0);
```

SCROLLTO()

The scrollTo() method scrolls the document to the specified coordinates.

```
window.scrollTo(500, 0);
// The scrollTo() method scrolls the document
to the specified coordinates.
```

SETINTERVAL()

The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).

```
setInterval(function(){ alert("Hello"); },
3000);
// The setInterval() method calls a function or
evaluates an expression at specified intervals
(in milliseconds).
```

SETTIMEOUT()

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.

```
setTimeout(function(){ alert("Hello"); },
3000);
// The setTimeout() method calls a function or
evaluates an expression after a specified
number of milliseconds.
```

STOP()

The stop() method stops window loading.

```
window.stop();
// The stop() method stops window loading.
```

THE DOCUMENT OBJECT

ADOPTNODE()

The adoptNode() method adopts a node from another document.

```
<iframe src="https://www.w3schools.com/"
style="height:380px;width:520px;"></iframe>
<p>Test paragraph</p>

<script>
var frame =
document.getElementsByTagName("IFRAME")[0]
var p = document.getElementsByTagName("P")[0];
var x = document.adoptNode(p);
console.log(x);

</script>
// Output

    <p>Test paragraph</p>          VM1550:8
```

ANCHORS

The anchors collection returns a collection of all <a> elements in the document that have a name attribute.

```
<a name="html">HTML Tutorial</a><br>
<a name="css">CSS Tutorial</a><br>
<a name="xml">XML Tutorial</a><br>

<script>
  var x = document.anchors;
  console.log(x);
</script>
// Output
                                        VM102:4
HTMLCollection(3) [a, a, a, html: a, css:
a, xml: a]
  0: a
  1: a
  2: a
  length: 3
  css: a
  html: a
  xml: a
  __proto__: HTMLCollection
```

BASEURI

The baseURI property returns the base URI of the HTML document. This property is read-only.

```
var x = document.baseURI;

// Output
// https://www.w3schools.com/
```

BODY

The body property sets or returns the document's body.

```
document.body.style.backgroundColor = "yellow";

// Sets the background color of body to yellow
```

CLOSE()

The close() method closes the output stream previously opened with the document.open() method

```
document.open();
document.write("<h1>Hello World</h1>");
document.close();

// displays the collected data in this process.
// Hello World
```

COOKIE

The cookie property sets or returns all name/value pairs of cookies in the current document.

```
var x = document.cookie;

// returns all name/value pairs of cookies in
the current document.
```

CHARACTERSET

The characterSet property returns the character encoding for the document.

```
var x = document.characterSet;

// UTF-8
```

CREATECOMMENT()

The createComment() method creates a Comment node with the specified text.

```
var c = document.createComment("My personal
comments");
document.body.appendChild(c);
```

CREATEATTRIBUTE()

The createAttribute() method creates an attribute with the specified name, and returns the attribute as an Attr object.

```
// Get the first <h1> element in the document
var h1 = document.getElementsByTagName("H1")
[0];
// Create a "class" attribute
var att = document.createAttribute("class");
// Set the value of the class attribute
att.value = "democlass";
// Add the class attribute to <h1>
h1.setAttributeNode(att);
```

CREATEDOCUMENTFRAGMENT()

The createDocumentFragment() method creates an imaginary Node object, with all the properties and methods of the Node object.

```
var d = document.createDocumentFragment();
d.appendChild(document.getElementsByTagName("LI
")[0]);
d.childNodes[0].childNodes[0].nodeValue =
"Milk";
document.getElementsByTagName("UL")
[0].appendChild(d);
```

CREATEELEMENT()

The createElement() method creates an Element Node with the specified name.The createElement() method creates an Element Node with the specified name.

```
// Create a <button> element
var btn = document.createElement("BUTTON");
// Insert text
btn.innerHTML = "CLICK ME";
// Append <button> to <body>
document.body.appendChild(btn);
```

CREATEEVENT()

The createDocumentFragment() method creates an imaginary Node object, with all the properties and methods of the Node object.

```
// Creat a new event object
var x = document.createEvent("MouseEvent");
// Initialize the event object to simulate
mouseover event
x.initMouseEvent("mouseover", true, true,
window, 0, 0, 0, 0, 0, false, false, false,
false, 0, null);
// Fire the event on mouseover of "myDiv"
document.getElementById("myDiv").dispatchEvent(
x);
```

CREATETEXTNODE()

The createTextNode() method creates a Text Node with the specified text.

```
// Create a <h1> element
var h = document.createElement("H1")
// Create a text node
var t = document.createTextNode("Hello World");
// Append the text to <h1>
h.appendChild(t);
```

CREATECOMMENT()

The createComment() method creates a Comment node with the specified text.

```
var c = document.createComment("My personal
comments");
document.body.appendChild(c);
```

DEFAULTVIEW

The defaultView property returns the document's Window Object.

```
var x = document.defaultView;
// returns the document's Window Object.
```

DESIGNMODE

The designMode property sets or returns whether the document is editable or not.

```
document.designMode = "on";
// The designMode property sets or returns
whether the document is editable or not.
```

DOCUMENTELEMENT

The documentElement property returns the documentElement of the document, as an Element object. For HTML documents the returned object is the <html> element.

```
var x = document.documentElement.nodeName;

console.log(x);
// output HTML
```

DOCUMENTURI

The documentURI property sets or returns the location of a document. If the document was created by the DocumentImplementation object, or if it is undefined, the return value is null.

```
var x = document.documentURI;

console.log(x);
// output https://www.w3schools.com
```

DOMAIN

The domain property returns the domain name of the server that loaded the current document.

```
var x = document.domain;
console.log(x);
// output www.w3schools.com
```

EMBEDS

The embeds collection returns a collection of all <embeds> elements in the document.

```
var x = document.embeds.length;
console.log(x);
// output 2
```

EXECCOMMAND()

The execCommand() method executes the specified command for the selected part of an editable section.

```
document.execCommand("bold");

// The execCommand() method executes the specified command for the selected part of an editable section.
```

FORMS

The forms collection returns a collection of all <form> elements in the document.

```
var x = document.forms;

// The forms collection returns a collection of all <form> elements in the document.
```

FULLSCREENELEMENT

The fullscreenElement property returns the current element that is displayed in fullscreen mode, or null when not in fullscreen.

```
var elem = document.fullscreenElement;

// The fullscreenElement property returns the current element that is displayed in fullscreen mode, or null when not in fullscreen.
```

FULLSCREENENABLED()

The fullscreenEnabled() method returns a Boolean value indicating whether the document can be viewed in fullscreen mode.

```
/* Get the element you want displayed in fullscreen */
var elem = document.getElementById("myvideo");

/* Function to open fullscreen mode */
function openFullscreen() {
  /* If fullscreen mode is available, show the element in fullscreen */
  if (document.fullscreenEnabled) {
    /* Show the element in fullscreen */
    elem.requestFullscreen();
  }
}
```

GETELEMENTBYID()

The getElementById() method returns the element that has the ID attribute with the specified value.

```
document.getElementById("demo");

// Return the element in the DOM with ID "demo"
```

GETELEMENTSBYCLASSNAME()

The getElementsByClassName() method returns a collection of all elements in the document with the specified class name

```
document.getElementsByClassName('product');

// Returns an array of elements which have the classname 'product'
```

GETELEMENTSBYNAME()

The getElementsByName() method returns a collection of all elements in the document with the specified name (the value of the name attribute)

```
var x = document.getElementsByName("fname");

// The getElementsByName() method returns a
collection of all elements in the document with
the specified name (the value of the name
attribute)
```

GETELEMENTSBYTAGNAME()

The getElementsByTagName() method returns a collection of all elements in the document with the specified tag name, as an HTMLCollection object.

```
var x = document.getElementsByTagName("LI");

// The getElementsByTagName() method returns a
collection of all elements in the document with
the specified tag name, as an HTMLCollection
object.
```

HASFOCUS()

The hasFocus() method returns a Boolean value indicating whether the document (or any element inside the document) has focus.

```
var x = document.getElementById("demo");

if (document.hasFocus()) {
  x.innerHTML = "The document has focus.";
} else {
  x.innerHTML = "The document DOES NOT have
focus.";
}
```

HEAD

The head property returns the <head> element of the current document.

```
var x = document.head;

// The head property returns the <head> element
of the current document.
```

IMAGES

The images collection returns a collection of all <img> elements in the document.

```
var x = document.images.length;

// The images collection returns a collection
of all <img> elements in the document.
```

IMPLEMENTATION

The implementation property returns the DOMimplementation object that handles this document, as a DocumentImplementation object.

```
var imp = document.implementation;
var res = imp.hasFeature("HTML", "1.0");

console.log(res);
// Output true
```

INPUTENCODING

The inputEncoding property returns the character encoding for the document.

```
var x = document.inputEncoding;
console.log(x);
// Output UTF-8
```

LASTMODIFIED

The lastModified property returns the date and time the current document was last modified.

```
var x = document.lastModified;
console.log(x);
// Output 02/28/2021 19:09:55
```

LINKS

The links collection returns a collection of all links in the document.

```
var x = document.links;

// The links collection returns a collection of
  all links in the document.
```

NORMALIZE()

The normalize() method removes empty Text nodes, and joins adjacent Text nodes.

```
document.normalize();

// The normalize() method removes empty Text
  nodes, and joins adjacent Text nodes.
```

NORMALIZEDOCUMENT()

The normalizeDocument() method removes empty Text nodes, and joins adjacent Text nodes.

```
document.normalizeDocument();

// The normalize() method removes empty Text
  nodes, and joins adjacent Text nodes.
```

OPEN()

The open() method opens an output stream to collect the output from any document.write() or document.writeln() methods.

```
document.open();
document.write("<h1>Hello World</h1>");
document.close();
// Once all the writes are performed, the
  document.close() method causes any output
  written to the output stream to be displayed.
```

QUERYSELECTOR

The normalizeDocument() method removes empty Text nodes, and joins adjacent Text nodes.

```
var x = document.querySelector(".example");
// selects the first paragraph with
  class="example".
```

QUERYSELECTORALL()

The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object.

```
var x = document.querySelectorAll(".example");
// returns all elements in the document that
  matches a specified CSS selector(s), as a
  static NodeList object.
```

READYSTATE

The readyState property returns the (loading) status of the current document.

```
var x = document.readyState;
console.log(x);
// output complete
// when the document is loaded completely
```

REFERRER

The referrer property returns the URL of the document that loaded the current document.

```
var x = document.referrer;

// https://www.w3schools.com/jsref/dom_obj_document.asp
```

REMOVEEVENTLISTENER()

The document.removeEventListener() method removes an event handler that has been attached with the document.addEventListener() method.

```
// Attach an event handler to the document
document.addEventListener("mousemove",
myFunction);

// Remove the event handler from the document
document.removeEventListener("mousemove",
myFunction);
```

SCRIPTS

The scripts collection returns a collection of all <script> elements in the document.

```
var x = document.scripts;
console.log(x);

// Output
                                                    VM83:4
▾HTMLCollection(2) [script, script]
  ▸0: script
  ▸1: script
   length: 2
  ▸__proto__: HTMLCollection
```

TITLE

The title property sets or returns the title of the current document (the text inside the HTML title element).

```
var x = document.title;

console.log(x);
// Output
// The title of the document
```

URL

The URL property returns the full URL of the current HTML document.

```
var x = document.URL;

console.log(x);
// Output
// https://www.w3schools.com/
```

WRITE()

The write() method writes HTML expressions or JavaScript code to a document.

```
document.write("Hello World!");

// Appends "Hello World!" to HTML document
```

WRITELN()

The writeln() method is identical to the document. write() method, with the addition of writing a newline character after each statement.

```
document.writeIn("Hello World!");

// Appends "Hello World! \n" in script tags
```

HISTORY OBJECT

LENGTH

The length property returns the number of URLs in the history list of the current browser window.

```
var x = history.length;

// returns the number of urls in history ex. 4
```

BACK()

The back() method loads the previous URL in the history list.

```
<button onclick="goBack()">Go Back</button>

<script>
function goBack() {
  window.history.back();
}
</script>

//  Takes you to the previous url in history
```

FORWARD()

The forward() method loads the next URL in the history list.

```
<button onclick="goForward()">Go
Forward</button>

<script>
function goForward() {
  window.history.forward();
}
</script>

//  Takes user to the next url in history
```

GO()

The go() method loads a specific URL from the history list.

```
<button onclick="goBack()">Go Back 2
Pages</button>

<script>
function goBack() {
  window.history.go(-2);
}
</script>
```

NAVIGATOR OBJECT

APPCODENAME

The appCodeName property returns the code name of the browser.

```
var x = "Browser CodeName: " +
navigator.appCodeName;

console.log(x);
// output
// Browser CodeName: Mozilla
```

APPNAME

The appName property returns the name of the browser.

```
var x = "Browser Name: " + navigator.appName;

console.log(x);
// output
// Browser Name: Netscape
```

APPVERSION

The appVersion property returns the version information of the browser.

```
var x = "Version Info: " +
navigator.appVersion;

console.log(x);
// output
// Version info: 5.0 (Macintosh; Intel Mac OS X
11_2_2) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/88.0.4324.192 Safari/537.36
```

COOKIEENABLED

The cookieEnabled property returns a Boolean value that specifies whether cookies are enabled in the browser.

```
var x = "Cookies enabled: " +
navigator.cookieEnabled;

console.log(x);
// output
// Cookies enabled: true
```

GEOLOCATION

The geolocation property returns a Geolocation object that can be used to locate the user's position.

```
navigator.geolocation.getCurrentPosition(showPo
sition);
function showPosition(position) {
 console.log(position);
}
// output
                                        VM186:13
  GeolocationPosition {coords: Geolocatio
▼ nCoordinates, timestamp: 1614674770687}
  ▶ coords: GeolocationCoordinates {lati…
    timestamp: 1614674770687
  ▶ __proto__: GeolocationPosition
```

LANGUAGE

The language property returns the language version of the browser.

```
var x = "Language of the browser: " +
navigator.language;

// output
// Language of the browser: en-US
```

ONLINE

The onLine property returns a Boolean value that specifies whether the browser is in online or offline mode.

```
1 var x = "Is the browser online? " +
  navigator.onLine;
2
3 console.log(x)
4 // output
5 // Is the browser online? true
```

PRODUCT

The product property returns the engine (product) name of the browser.

```
1 var x = "Browser's Engine Name: " +
  navigator.product;
2
3 console.log(x)
4 // output
5 // Browser's Engine Name: Gecko
```

USERAGENT

The userAgent property returns the value of the user-agent header sent by the browser to the server.

```
1 var x = "User-agent header sent: " +
  navigator.userAgent;
2
3 console.log(x)
4 // output
5 // User-agent header sent: Mozilla/5.0
  (Macintosh; Intel Mac OS X 11_2_2)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/88.0.4324.192 Safari/537.36
```

PLATFORM

The platform property returns for which platform the browser is compiled.

```
1 var x = "Platform: " + navigator.platform;
2
3 console.log(x)
4 // output
5 // Platform: MacIntel
```

GEOLOCATION OBJECT

COORDINATES

The coordinates property returns the position and altitude of the device on Earth.

```
function showPosition(position) {
  console.log(position.coords)
}

navigator.geolocation.getCurrentPosition(showPosition);

// output
                                                    VM148:13
GeolocationCoordinates {latitude: 28.399148699999998, longitude: 77.0970584, altitude: null, accuracy: 20, altitudeAccuracy: null, …}
```

POSITION

The position property returns the position and altitude of the device on Earth.

```
var x = "Browser's Engine Name: " +
navigator.product;

console.log(x)
// output
// Browser's Engine Name: Gecko
```

POSITIONERROR

Returns the reason of an error occurring when using the geolocating device

POSITIONOPTIONS

Describes an object containing option properties to pass as a parameter of Geolocation. getCurrentPosition() and Geolocation.watchPosition()

CLEARWATCH()

Unregister location/error monitoring handlers previously installed using Geolocation. watchPosition()

GETCURRENTPOSITION()

Returns the current position of the device

WATCHPOSITION()

Returns a watch ID value that then can be used to unregister the handler by passing it to the Geolocation.- clearWatch() method

LOCATION OBJECT

HASH

The hash property sets or returns the anchor part of a URL, including the hash sign (#).

```
// Return the anchor part of a URL. Assume that
  the current URL is
  http://www.example.com/test.htm#part2:

var x = location.hash;
console.log(x);

// output
// #part2
```

HOST

The host property sets or returns the hostname and port of a URL.

```
var x = location.host;
console.log(x);

// output
// www.w3schools.com
```

HOSTNAME

The hostname property sets or returns the hostname of a URL.

```
var x = location.hostname;
console.log(x);

// output
// www.w3schools.com
```

HREF

The href property sets or returns the entire URL of the current page.

```
var x = location.href;
console.log(x);

// output
// https://www.w3schools.com/
```

ORIGIN

The origin property returns the protocol, hostname and port number of a URL.

```
// Return the protocol, hostname and port
  number of a URL. Assume that the current URL is
  https://www.w3schools.com:4097/test.htm#part2:

var x = location.origin;

// output
// https://www.w3schools.com:4097
```

PATHNAME

The pathname property sets or returns the pathname of a URL.

```
// assuming url is
  https://www.w3schools.com/jsref/tryit.asp
var x = location.pathname;
console.log(x);

// output
// /jsref/tryit.asp
```

PORT

The port property sets or returns the port number the server uses for a URL.

```
var x = "Port: " + location.port;

console.log(x);
// output
// Port: 443
```

SEARCH

The search property sets or returns the querystring part of a URL, including the question mark (?).

```
// Return the querystring part of a URL. Assume
that the current URL is
https://www.w3schools.com/submit.htm?
email=someone@example.com:
var x = location.search;

console.log(x);
// output
// ?email=someone@example.com
```

ASSIGN()

The assign() method loads a new document.

```
location.assign("https://www.w3schools.com");

// output
// opens https://www.w3schools.com
```

RELOAD()

The reload() method is used to reload the current document.

```
location.reload();

// The reload() method is used to reload the
current document.
```

REPLACE()

The replace() method replaces the current document with a new one.

The difference between this method and assign(), is that replace() removes the URL of the current document from the document history, meaning that it is not possible to use the “back” button to navigate back to the original document.

```
location.replace("https://www.w3schools.com");

// The replace() method replaces the current
document with a new one.
```

HTML DOM EVENTS

ONABORT

The abort event occurs when the loading of an audio/video has been aborted, and not because of an error.

```
<video onabort="myFunction()">
// The onabort event occurs when the loading of
  an audio/video has been aborted, and not
  because of an error.
```

AFTERPRINT

The afterprint event occurs when a page has started printing, or if the print dialogue box has been closed.

```
<body onafterprint="myFunction()">
// The afterprint event occurs when a page has
  started printing, or if the print dialogue box
  has been closed.
```

ANIMATIONEND

The animationend event occurs when a CSS animation has completed.

```
var x = document.getElementById("myDIV");

// Code for Chrome, Safari and Opera
x.addEventListener("webkitAnimationEnd",
  myEndFunction);

// Standard syntax
x.addEventListener("animationend",
  myEndFunction);
```

ANIMATIONITERATION

The animationiteration event occurs when a CSS animation is repeated.

```
var x = document.getElementById("myDIV");

// Code for Chrome, Safari and Opera
x.addEventListener("webkitAnimationIteration",
  myRepeatFunction);

// Standard syntax
x.addEventListener("animationiteration",
  myRepeatFunction);
```

ANIMATIONSTART

The animationstart event occurs when a CSS animation has started to play.

```
var x = document.getElementById("myDIV");

// Code for Chrome, Safari and Opera
x.addEventListener("webkitAnimationStart",
  myStartFunction);

// Standard syntax
x.addEventListener("animationstart",
  myStartFunction);
```

BEFOREPRINT

The beforeprint event occurs when a page is about to be printed (before the print dialogue box appears).

```
<body onbeforeprint="myFunction()">
// The beforeprint event occurs when a page is
  about to be printed (before the print dialogue
  box appears).
```

BEFOREUNLOAD

The onbeforeunload event occurs when the document is about to be unloaded.

```
<body onbeforeunload="return myFunction()">
// The onbeforeunload event occurs when the document is about to be unloaded.
```

BLUR

The onblur event occurs when an object loses focus. The onblur event is most often used with form validation code (e.g. when the user leaves a form field).

```
<input type="text" onblur="myFunction()">
// TThe onblur event occurs when an object loses focus. The onblur event is most often used with form validation code (e.g. when the user leaves a form field).
```

CANPLAY

The oncanplay event occurs when the browser can start playing the specified audio/video (when it has buffered enough to begin).

```
<video oncanplay="myFunction()">
// The oncanplay event occurs when the browser can start playing the specified audio/video (when it has buffered enough to begin).
```

CANPLAYTHROUGH

The oncanplaythrough event occurs when the browser estimates it can play through the specified media without having to stop for buffering.

```
<video oncanplaythrough="myFunction()">
// The oncanplaythrough event occurs when the browser estimates it can play through the specified media without having to stop for buffering.
```

CHANGE

The onchange event occurs when the value of an element has been changed.

```
<select onchange="myFunction()">
// The onchange event occurs when the value of an element has been changed.
```

CLICK

The onclick event occurs when the user clicks on an element.

```
<button onclick="myFunction()">Click me</button>
// The onclick event occurs when the user clicks on an element.
```

CONTEXTMENU

The oncontextmenu event occurs when the user right-clicks on an element to open the context menu.

```
<div oncontextmenu="myFunction()" contextmenu="mymenu">
// Execute a JavaScript when the user right-clicks on a <div> element with a context menu:
```

COPY

The oncopy event occurs when the user copies the content of an element.

```
// Execute a JavaScript when copying some text of an <input> element:
<input type="text" oncopy="myFunction()" value="Try to copy this text">
```

CUT

The oncut event occurs when the user cuts the content of an element.

```
// Execute a JavaScript when cutting some text in an <input> element:
<input type="text" oncut="myFunction()" value="Try to cut this text">
```

DBLCLICK

The ondblclick event occurs when the user double-clicks on an element.

```
// The ondblclick event occurs when the user double-clicks on an element.
<p ondblclick="myFunction()">Double-click me</p>
```

DRAG

The ondrag event occurs when an element or text selection is being dragged.

```
// Execute a JavaScript when a <p> element is being dragged:
<p draggable="true" ondrag="myFunction(event)">Drag me!</p>
```

DRAGEND

The ondragend event occurs when the user has finished dragging an element or text selection.

```
// Execute a JavaScript when the user has finished dragging a <p> element:
<p draggable="true" ondragend="myFunction(event)">Drag me!</p>
```

DRAGENTER

The ondragenter event occurs when a draggable element or text selection enters a valid drop target.

```
// Execute a JavaScript when a draggable element enters a drop target:
<div ondragenter="myFunction(event)"></div>
```

DRAGLEAVE

The ondragleave event occurs when a draggable element or text selection leaves a valid drop target.

```
// Execute a JavaScript when a draggable element is moved out of a drop target:
<div ondragleave="myFunction(event)"></div>
```

DRAGOVER

The ondragover event occurs when a draggable element or text selection is being dragged over a valid drop target.

```
// Execute a JavaScript when an element is being dragged over a drop target:
<div ondragover="myFunction(event)"></div>
```

DRAGSTART

The ondragstart event occurs when the user starts to drag an element or text selection.

```
// Execute a JavaScript when the user starts to drag a <p> element:
<p draggable="true" ondragstart="myFunction(event)">Drag me!</p>
```

DROP

The ondrop event occurs when a draggable element or text selection is dropped on a valid drop target.

```
// Execute a JavaScript when a draggable element is dropped in a <div> element:
<div ondrop="myFunction(event)"></div>
```

DURATIONCHANGE

The ondurationchange event occurs when the duration of the audio/video is changed.

```
// Execute a JavaScript when the duration of a video has changed:
<video ondurationchange="myFunction()">
```

ENDED

The onended event occurs when the audio/video has reached the end.

```
// Execute a JavaScript when an audio has ended:
<audio onended="myFunction()">
```

ERROR

The onerror event is triggered if an error occurs while loading an external file (e.g. a document or an image).

```
// Execute a JavaScript if an error occurs when loading an image:
<img src="image.gif" onerror="myFunction()">
```

FOCUS

The onfocus event occurs when an element gets focus. The onfocus event is most often used with <input>, <select>, and <a>.

```
// Execute a JavaScript when an input field gets focus:
<input type="text" onfocus="myFunction()">
```

FOCUSIN

The onfocusin event occurs when an element is about to get focus.

```
// Execute a JavaScript when an input field is about to get focus:
<input type="text" onfocusin="myFunction()">
```

FOCUSOUT

The onfocusout event occurs when an element is about to lose focus.

```
// Execute a JavaScript when an input field is about to lose focus:
<input type="text" onfocusout="myFunction()">
```

FULLSCREENCHANGE

The fullscreenchange event occurs when an element is viewed in fullscreen mode.

```
document.addEventListener("fullscreenchange", function() {
  output.innerHTML = "fullscreenchange event fired!";
});
```

FULLSCREENERROR

The fullscreenerror event occurs when an element can not be viewed in fullscreen mode, even if it has been requested.

```
// Alert some text if an element can not be viewed in fullscreen mode:
document.addEventListener("fullscreenerror", function() {
  alert("Fullscreen denied")
});
```

HASHCHANGE

The onhashchange event occurs when there has been changes to the anchor part (begins with a '#' symbol) of the current URL.

```
// Execute a JavaScript when the anchor part has been changed:

<body onhashchange="myFunction()">
```

INPUT

The oninput event occurs when an element gets user input. This event occurs when the value of an <input> or <textarea> element is changed.

```
// Execute a JavaScript when a user writes something in an <input> field:
<input type="text" oninput="myFunction()">
```

INVALID

The oninvalid event occurs when a submittable <input> element is invalid.

```
// Alert some text if an input field is invalid:
<input type="text" oninvalid="alert('You must fill out the form!');" required>
```

KEYDOWN

The onkeydown event occurs when the user is pressing a key (on the keyboard).

```
// Execute a JavaScript when a user is pressing a key:
<input type="text" onkeydown="myFunction()">
```

KEYPRESS

The onkeypress event occurs when the user presses a key (on the keyboard).

```
// Execute a JavaScript when a user presses a key:
<input type="text" onkeypress="myFunction()">
```

KEYUP

The onkeyup event occurs when the user releases a key (on the keyboard).

```
// Execute a JavaScript when a user releases a key:
<input type="text" onkeyup="myFunction()">
```

LOAD

The onload event occurs when an object has been loaded.

```
// Execute a JavaScript immediately after a page has been loaded:
<body onload="myFunction()">
```

LOADEDDATA

The onloadeddata event occurs when data for the current frame is loaded, but not enough data to play next frame of the specified audio/video.

```
// Execute a JavaScript when data for the current frame is available (for <video>):
<video onloadeddata="myFunction()">
```

LOADEDMETADATA

The onloadedmetadata event occurs when meta data for the specified audio/video has been loaded.

```
// Execute a JavaScript when meta data for a video is loaded:
<video onloadedmetadata="myFunction()">
```

LOADSTART

The onloadstart event occurs when the browser starts looking for the specified audio/video. This is when the loading process starts.

```
// Execute a JavaScript when the video is starting to load:
<video onloadstart="myFunction()">
```

MESSAGE

The onmessage event occurs when a message is received through an event source.

```
var source = new EventSource("demo_sse.php");
// Append to document
source.onmessage = function(event) {
  document.getElementById("myDIV").innerHTML += event.data + "<br>";
};
```

MOUSEDOWN

The onmousedown event occurs when a user presses a mouse button over an element.

```
// Execute a JavaScript when pressing a mouse button over a paragraph:
<p onmousedown="myFunction()">Click the text!</p>
```

MOUSEENTER

The onmouseenter event occurs when the mouse pointer is moved onto an element.

```
// Execute a JavaScript when moving the mouse pointer onto an image:
<img onmouseenter="bigImg(this)" src="smiley.gif" alt="Smiley">
```

MOUSELEAVE

The onmouseleave event occurs when the mouse pointer is moved out of an element.

```
// Execute a JavaScript when moving the mouse pointer out of an image:
<img onmouseleave="normalImg(this)" src="smiley.gif" alt="Smiley">
```

MOUSEMOVE

The onmousemove event occurs when the pointer is moving while it is over an element.

```
// Execute a JavaScript when moving the mouse pointer over a <div> element:
<div onmousemove="myFunction()">Move the cursor over me</div>
```

MOUSEOVER

The onmouseover event occurs when the mouse pointer is moved onto an element, or onto one of its children.

```
// Execute a JavaScript when moving the mouse pointer onto an image:
<img onmouseover="bigImg(this)" src="smiley.gif" alt="Smiley">
```

MOUSEOUT

The onmouseout event occurs when the mouse pointer is moved out of an element, or out of one of its children.

```
// Execute a JavaScript when moving the mouse pointer out of an image:
<img onmouseout="normalImg(this)" src="smiley.gif" alt="Smiley">
```

MOUSEUP

The onmouseup event occurs when a user releases a mouse button over an element.

```
// Execute a JavaScript when releasing a mouse button over a paragraph:
<p onmouseup="mouseUp()">Click the text!</p>
```

OFFLINE

The onoffline event occurs when the browser starts to work offline.

```
// Execute a JavaScript when the browser starts to work offline:
<body onoffline="myFunction()">
```

ONLINE

The ononline event occurs when the browser starts to work online.

```
// Execute a JavaScript when the browser starts to work online:
<body ononline="myFunction()">
```

OPEN

The onopen event occurs when a connection with an event source is opened.

```
var source = new EventSource("demo_sse.php");
source.onopen = function() {
  document.getElementById("myH1").innerHTML = "Getting server updates";
};
```

PAGEHIDE

The onpagehide event occurs when the user is navigating away from a webpage.

```
// Execute a JavaScript when the user is navigating away from a webpage:
<body onpagehide="myFunction()">
```

PAGESHOW

The onpageshow event occurs when a user navigates to a webpage.

```
// Execute a JavaScript when a user navigates to a webpage:
<body onpageshow="myFunction()">
```

PASTE

The onpaste event occurs when the user pastes some content in an element.

```
// Execute a JavaScript when pasting some text in an <input> element:
<input type="text" onpaste="myFunction()" value="Paste something in here">
```

PAUSE

The onpause event occurs when the audio/video is paused either by the user or programmatically.

```
// Execute a JavaScript when a video has been paused:
<video onpause="myFunction()">
```

PLAY

The onplay event occurs when the audio/video has been started or is no longer paused.

```
// Execute a JavaScript when a video has started to play:
<video onplay="myFunction()">
```

PLAYING

The onplaying event occurs when the audio/video is playing after having been paused or stopped for buffering.

```
// Execute a JavaScript when a video is ready to start after having been paused:
<video onplaying="myFunction()">
```

PROGRESS

The onprogress event occurs when the browser is downloading the specified audio/video.

```
// Execute a JavaScript when the video is downloading:
<video onprogress="myFunction()">
```

RATECHANGE

The onratechange event occurs when the playing speed of the audio/video is changed

```
// Execute a JavaScript when the playing speed of the video is changed:
<video onratechange="myFunction()">
```

RESIZE

The onresize event occurs when the browser window has been resized.

```
// Execute a JavaScript when the browser window is resized:
<body onresize="myFunction()">
```

RESET

The onreset event occurs when a form is reset.

```
// Execute a JavaScript when a form is reset:
<form onreset="myFunction()">
  Enter name: <input type="text">
  <input type="reset">
</form>
```

SCROLL

The onscroll event occurs when an element's scrollbar is being scrolled.

```
// Execute a JavaScript when a <div> element is being scrolled:
<div onscroll="myFunction()">
```

SEEKED

The onseeked event occurs when the user is finished moving/skipping to a new position in the audio/video

```
// Execute a JavaScript when the user is finished moving/skipping to a new position in the video:
<video onseeked="myFunction()">
```

SEEKING

The onseeking event occurs when the user starts moving/skipping to a new position in the audio/video.

```
// Execute a JavaScript when the user starts moving/skipping to a new position in the video:
<video onseeking="myFunction()">
```

SELECT

The onselect event occurs after some text has been selected in an element.

```
// Execute a JavaScript when some text has been selected:
<input type="text" onselect="myFunction()">
```

SHOW

The onshow event occurs when a <menu> element is shown as a context menu.

```
<div contextmenu="mymenu">
  <menu type="context" id="mymenu" onshow="myFunction()">
    <menuitem label="Refresh" onclick="window.location.reload();"></menuitem>
  </menu>
</div>
```

STALLED

The onstalled event occurs when the browser is trying to get media data, but data is not available.

```
// Execute a JavaScript when the browser is trying to get media data, but data is not available:
<video onstalled="myFunction()">
```

SUBMIT

The onsubmit event occurs when a form is submitted.

```
<form onsubmit="myFunction()">
  Enter name: <input type="text">
  <input type="submit">
</form>
```

SUSPEND

The onsuspend event occurs when the browser is intentionally not getting media data.

```
// Execute a JavaScript when the browser is intentionally not getting media data:
<video onsuspend="myFunction()">
```

TIMEUPDATE

The ontimeupdate event occurs when the playing position of an audio/video has changed.

```
// Execute a JavaScript when the current playback position has changed:
<video ontimeupdate="myFunction()">
```

TOGGLE

The ontoggle event occurs when the user opens or closes the <details> element.

```
// Execute a JavaScript when a <details> element is opened or closed:
<details ontoggle="myFunction()">
```

TOUCHCANCEL

The touchcancel event occurs when the touch event gets interrupted.

```
// Execute a JavaScript when a touch is interrupted (for touch screens only):
<p ontouchcancel="myFunction(event)">Touch me!</p>
```

TOUCHEND

The touchend event occurs when the user removes the finger from an element.

```
// Execute a JavaScript when the user releases the touch (for touch screens only):
<p ontouchend="myFunction(event)">Touch me!</p>
```

TOUCHMOVE

The touchmove event occurs when the user moves the finger across the screen.

```
// Execute a JavaScript when the user moves the finger over a P element (for touch screens only):
<p ontouchmove="myFunction(event)">Touch me!</p>
```

TOUCHSTART

The touchstart event occurs when the user touches an element.

```
// Execute a JavaScript when the user touches a P element (for touch screens only):
<p ontouchstart="myFunction(event)">Touch me!</p>
```

TRANSITIONEND

The transitionend event occurs when a CSS transition has completed.

```
// Code for Safari 3.1 to 6.0
document.getElementById("myDIV").addEventListener("webkitTransitionEnd", myFunction);

// Standard syntax
document.getElementById("myDIV").addEventListener("transitionend", myFunction);
```

UNLOAD

The onunload event occurs once a page has unloaded (or the browser window has been closed).

```
// Execute a JavaScript when a user unloads the document:
<body onunload="myFunction()">
```

VOLUMECHANGE

The onvolumechange event occurs each time the volume of a video/audio has been changed.

```
// Execute a JavaScript when the volume of a video has been changed:
<video onvolumechange="myFunction()">
```

WAITING

The onwaiting event occurs when the video stops because it needs to buffer the next frame.

```
// Execute a JavaScript when the video stops because it needs to buffer the next frame:
<video onwaiting="myFunction()">
```

WHEEL

The onwheel event occurs when the mouse wheel is rolled up or down over an element.

```
document.getElementById("myDIV").addEventListener("wheel", myFunction);

function myFunction() {
  this.style.fontSize = "35px";
}
```

JOIN ZERO TO FULL STACK HERO TO LEARN MORE,
VISIT WWW.PAPAREACT.COM

Sonny Sangha

ssssangha