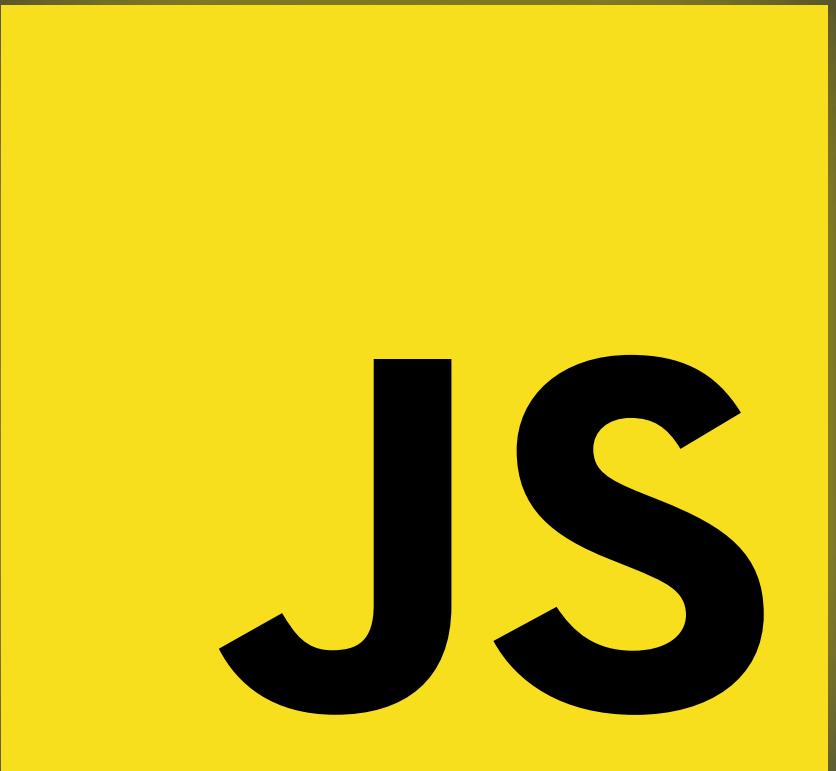




# The Ultimate JavaScript CheatSheet

A large, bold, black 'JS' logo centered on a solid yellow square background, which is itself centered on a dark gray gradient background.

JS

# Table of Content

## JavaScript Basics

- Variables
- Operators
- Functions
- Conditional Statements
- Truthy / Falsy Values
- String Escape Characters

## String & Array Methods

- String Methods
- Array Methods
- Looping
- Array methods for looping

- Value vs Reference

## Math & Date Methods

- Math Object Methods
- Date Object Methods

## Window Object

- Window Object Methods
- Window Object Properties

## DOM Events & Properties

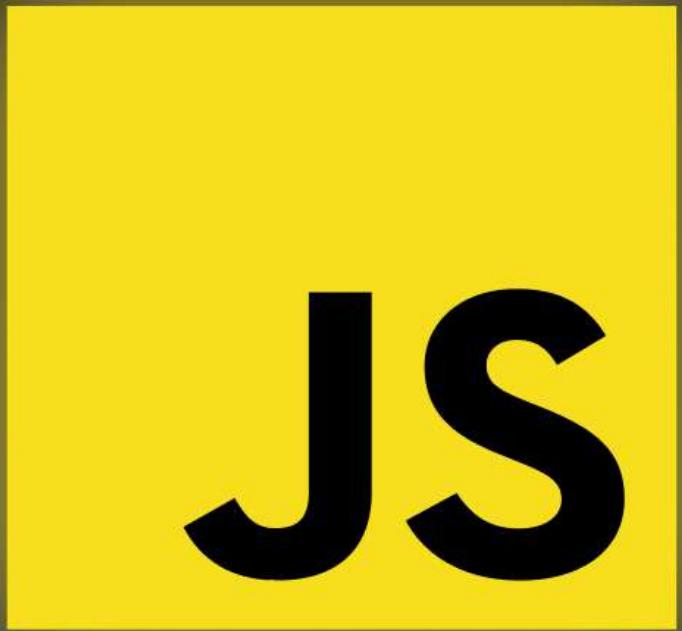
- Input Events
- Mouse Events
- HTML DOM Events
- HTML DOM Event Properties & Methods



# JavaScript

# CheatSheet

## Basic



# Variables

## **var variableName = value**

Can be reassigned and are only available inside the function they're created in. Its function scoped.

## **const variableName = value**

Cannot be reassigned and not accessible before they appear within the code. Its block scoped.

## **let variableName = value**

It can be reassigned but its similar to const i.e. block scoped.

If variables are not created inside a function or block they are globally scoped.

## **What is the block?**

A block is a set of opening and closing curly brackets.

# Variables

## Primitive Data Types:

<b>Number</b>	5, 6.5, 7 etc
<b>String</b>	"Hello everyone" etc
<b>Boolean</b>	true or false
<b>Null</b>	represents null i.e. no value at all
<b>Undefined</b>	A variable that has not been assigned a value is undefined.
<b>Symbol</b>	used as an identifier for object properties.

# Variables

## Non-Primitive Data Types:

<b>Object</b>	instance through which we can access members
<b>Array</b>	group of similar values
<b>RegExp</b>	represents regular expression

# Operators

## Basic Operators

- +** Addition
- Subtraction
- \*** Multiplication
- /** Division
- ()** Grouping operator
- %** Modulus (remainder)
- ++** Increment numbers
- Decrement numbers

# Operators

## Comparison Operators

**==** Equal to

**====** Equal value and equal type

**!=** Not equal

**!==** Not equal value or not equal type

**>** Greater than

**<** Less than

**>=** Greater than or equal to

**<=** Less than or equal to

# Operators

## Logical Operators

**&&** Logical and

**||** Logical or

**!** Logical not

## Bitwise Operators

Bitwise operators in Javascript are mostly used for numerical conversions/computations, because sometimes they're much faster than their Math or parseInt equivalents

# Operators

## Bitwise Operators

**&** AND statement

**|** OR statement

**~** NOT

**^** XOR

**<<** Zero fill left shift

**>>** Signed right shift

**>>>** Zero Fill right shift

# Functions

## Normal Function Declaration

```
function name (parameter) {  
    // statements  
}
```

## Function stored in a variable

```
let name = function (parameter) {  
    // statements  
}
```

## Arrow Function

```
const name = (parameter) => {  
    // statements  
}
```

# Conditional Statements

- Use **if** to specify a block of code to be executed, if a specified condition is true
- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed

## If - Else Statements

```
if (condition) {  
    // code to be executed if the  
    // condition is true  
}  
else {  
    // code to be executed if the  
    // condition is false  
}
```

# Conditional Statements

## If – Else If – Else Statements

```
if (condition1) {  
    // code to be executed if the  
    condition is true  
  
} else if (condition2) {  
    // code to be executed if the  
    condition1 is false and  
    condition2 is true  
  
} else {  
    // code to be executed if  
    condition1 is false and  
    condition2 is false  
}
```

# Conditional Statements

## Switch Statement

```
switch(expression) {  
    case x:  
        // code block  
        break;  
  
    case y:  
        // code block  
        break;  
  
    default:  
        // code block  
}
```

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- If there is no match, the default code block is executed.

# Conditional Statements

## Ternary Operator

**condition ? exprIfTrue : exprIfFalse**

### **condition**

An expression whose value is used as a condition.

### **exprIfTrue**

An expression which is executed if the condition is truthy.

### **exprIfFalse**

An expression which is executed if the condition is falsy.

# Truthy / Falsy Values

## FALSY Values

- false
- 0 (zero)
- "", "", `` (empty strings)
- null
- undefined
- NaN (not a number)

**Note** : Empty array ([]) is not falsy

## TRUTHY Values

- Everything that is not FALSY

# Strings

```
let variableName = "Hello world"
```

## Escape Characters

**\'** Single quote

**\\"** Double quote

**\\"** Backslash

**\b** Backspace

**\f** Form feed

**\n** New line

**\r** Carriage return

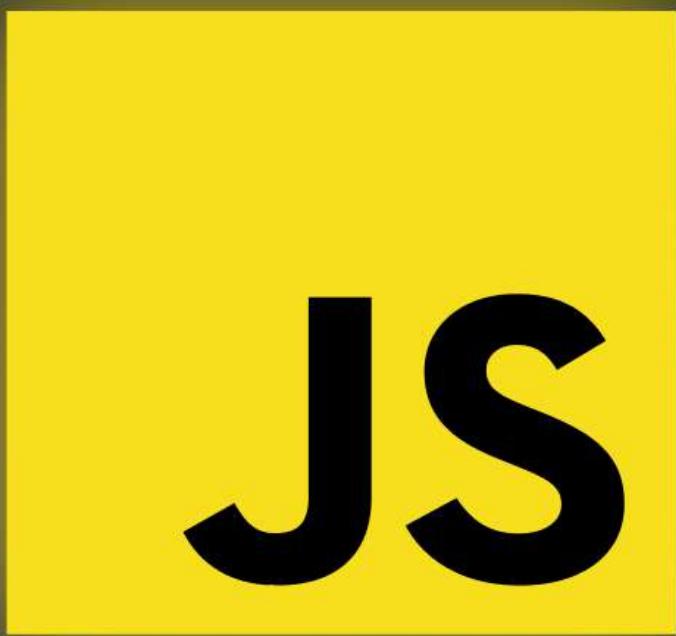
**\t** Horizontal tabulator

**\v** Vertical Tabulator



# JavaScript CheatSheet

- String Methods
- Array Methods



# String Methods

## **string[index]**

get a certain character of a string

## **string.length**

return the number of characters in a string

## **string.split(' ')**

returns an array of words of a string

## **string.split('')**

returns an array of characters of a string

## **string.toLowerCase()**

returns a lowercased string

## **string.toUpperCase()**

returns an uppercased string

# String Methods

## **string.charAt(index)**

returns a new string consisting of the single character located at the specified offset into the string.

## **string.replace(substr, newSubstr)**

returns a new string with a substring (substr) replaced by a new one (newSubstr).

## **string.includes(searchString)**

performs a case-sensitive search to determine whether one string may be found within another string, returns true or false.

## **string.substr(start, length)**

returns a portion of the string, starting at the specified index and extending for a given number.

# String Methods

## **string.includes('substring')**

checks whether a substring exists inside of a string  
[check the character case]

## **string.indexOf(searchValue)**

returns the index of the first occurrence of the specified value, starting the search at fromIndex.  
Returns -1 if the value is not found.

## **string.lastIndexOf(searchValue)**

returns the index of the last occurrence of the specified value, searching backwards from fromIndex. Returns -1 if the value is not found.

## **string.slice(beginIndex, endIndex)**

extracts a section of a string and returns it as a new string, without modifying the original string.

# Array Methods

## **array[index]**

returns a certain value from an array

## **push(value)**

adds the value to the end of the array

## **pop()**

removes the value from the end of the array

## **shift()**

removes the value from the start of the array

## **unshift(value)**

adds the value to the start of the array

## **splice(fromIndex, no\_of\_elements)**

removes the number\_of\_elements, starting from fromIndex from the array

# Array Methods

## **slice(fromIndex, toIndex)**

copies a certain part of the array

## **concat()**

Join several arrays into one

## **join('')**

returns a string of array values

## **array.length**

returns the number of elements in the array

## **reverse()**

reverse the order of the elements in an array

## **toString()**

returns a string representing the specified array and its elements.

# Array Methods

## **toString()**

returns a string representing the specified array and its elements.

## **includes(searchElement)**

determines whether an array includes a certain value among its entries, returning true or false as appropriate.

## **sort()**

It sorts the elements of an array in place and returns the sorted array. It sorts an array alphabetically.

## **index0f(searchElement)**

returns the index of the first occurrence of that value

## **lastIndex0f(searchElement)**

returns the index of the last occurrence of that value

# Looping

## For Loop

```
for (st 1; st 2; st 3) {
    // code block to be executed
}
```

**st 1** is executed (one time) before the execution of the code block.

**st 2** defines the condition for executing the code block.

**st 3** is executed (every time) after the code block has been executed.

---

## While Loop

```
while (condition) {
    // code block to be executed
}
```

# Looping

## Do While Loop

```
do {  
    // code block to be executed  
}  
while (condition);
```

---

## For In Loop

```
for (key in object) {  
    // code block to be executed  
}
```

---

## For Of Loop

```
for (variable of iterable) {  
    // code block to be executed  
}
```

# Array methods for looping

## array.forEach()

It executes a provided function once for each array element.

```
array.forEach((element, index) => {  
    // code block to be executed  
})
```

---

## array.map()

It creates a new array populated with the results of calling a provided function on every element in the calling array.

```
array.map((element, index) => {  
    // code block to be executed  
})
```

# Array method for looping

## array.filter()

It creates a new array with all elements that pass the test implemented by the provided function.

```
array.filter((element, index) => {  
    // code block to be executed  
})
```

---

## array.findIndex()

It returns the index of the first element in the array that satisfies the provided testing function

```
array.findIndex((el, idx, arr) => {  
    // code block to be executed  
})
```

# Array method for looping

## array.some()

It tests whether at least one element in the array passes the test implemented by the provided function

```
array.some((el, index, array)) => {  
    // code block to be executed  
}
```

---

## array.every()

It tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

```
array.every((element, index) => {  
    // code block to be executed  
})
```

# Array method for looping

## array.reduce()

It runs a function on each array element to produce (reduce it to) a single value. It works from left-to-right.

```
array.reduce((prevValue, currentValue,  
currentIndex, array)) => {  
    // code block to be executed  
}
```

---

## array.reduceRight()

It runs a function on each array element to produce (reduce it to) a single value. It works from right-to-left.

```
array.reduceRight((accumulator,  
currentValue, index, array)) => {  
    // code block to be executed  
}
```

# VALUE VS REFERENCE

## Arrays



```
const numbers = [1, 2, 3, 4]
const anotherNumbers = numbers
anotherNumbers.push(5);

console.log(numbers === anotherNumbers); // true
```

## Objects



```
const person = { firstName: 'John', lastName: 'Doe' };
const anotherPerson = person;
anotherPerson.lastName = 'Smith';

console.log(person === anotherPerson); // true
```

# VALUE VS REFERENCE

## Cloning Array

### One Level Deep



```
// Shallow cloning - One level Deep

const original = [1, 2, 3, 4];

const new0g = [...original];

new0g.push(5);

console.log(original === new0g) // false
```

# VALUE VS REFERENCE

## Cloning Array

### Two Level Deep



```
// Deep cloning - Two level Deep

const users = [
  { name: 'John', age: 24 },
  { name: 'Victor', age: 31 },
];

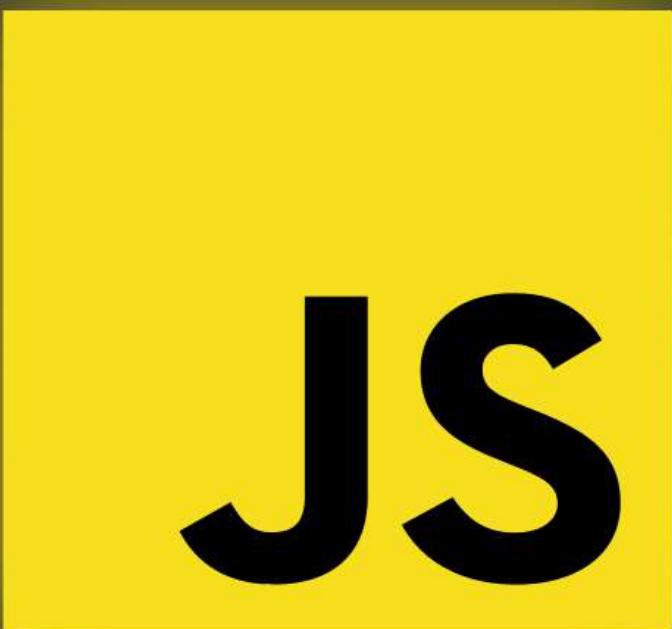
const newUser = JSON.parse(JSON.stringify(users))

console.log(users === newUser) // false
```



# JavaScript CheatSheet

- Date Methods
- Math Methods

A large, bold, black 'JS' logo centered on a yellow rectangular background. The yellow rectangle has a thin black border and is positioned in the lower right quadrant of the page, overlapping the footer area.

# Math Object

## **abs(x)**

Returns the absolute value of x

## **acos(x)**

Returns the arccosine of x, in radians

## **acosh(x)**

Returns the hyperbolic arccosine of x

## **asin(x)**

Returns the arcsine of x, in radians

## **asinh(x)**

Returns the hyperbolic arcsine of x

## **atan(x)**

Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians

# Math Object

## **atan2(y, x)**

Returns the arctangent of the quotient of its arguments

## **atanh(x)**

Returns the hyperbolic arctangent of x

## **cbrt(x)**

Returns the cubic root of x

## **ceil(x)**

Returns x, rounded upwards to the nearest integer

## **cos(x)**

Returns the cosine of x (x is in radians)

## **cosh(x)**

Returns the hyperbolic cosine of x

# Math Object

## **exp(x)**

Returns the value of Ex

## **floor(x)**

Returns x, rounded downwards to the nearest integer

## **log(x)**

Returns the natural logarithm (base E) of x

## **max(x, y, z, ..., n)**

Returns the number with the highest value

## **min(x, y, z, ..., n)**

Returns the number with the lowest value

## **pow(x, y)**

Returns the value of x to the power of y

# Math Object

## **random()**

Returns a random number between 0 and 1

## **round(x)**

Rounds x to the nearest integer

## **sign(x)**

Returns if x is negative, null or positive (-1, 0, 1)

## **sin(x)**

Returns the sine of x (x is in radians)

## **sinh(x)**

Returns the square root of x

## **sqrt(x)**

Returns the square root of x

# Math Object

## **tan(x)**

Returns the tangent of an angle

## **tanh(x)**

Returns the hyperbolic tangent of a number

## **trunc(x)**

Returns the integer part of a number (x)

# Date Objects

## Get Date Methods

### **getFullYear()**

Get the year as a four digit number (yyyy)

### **getMonth()**

Get the month as a number (0-11)

### **getDate()**

Get the day as a number (1-31)

### **getHours()**

Get the hour (0-23)

### **getMinutes()**

Get the minute (0-59)

# Date Objects

## **getTime()**

Get the time (milliseconds since January 1, 1970)

## **getDay()**

Get the weekday as a number (0-6)

## **Date.now()**

Get the time. ECMAScript 5.

## **getSeconds()**

Get the second (0-59)

## **getMilliseconds()**

Get the millisecond (0-999)

# Date Objects

## UTC Date Methods

### **getUTCDate()**

Same as getDate(), but returns the UTC date

### **getUTCDay()**

Same as getDay(), but returns the UTC day

### **getUTCFullYear()**

Same as getFullYear(), but returns the UTC year

### **getUTCMonth()**

Same as getMonth(), but returns the UTC month

### **getUTCHours()**

Same as getHours(), but returns the UTC hour

# Date Objects

## **getUTCMilliseconds()**

Same as getMilliseconds(), but returns the UTC milliseconds

## **getUTCSeconds()**

Same as getSeconds(), but returns the UTC seconds

## **getUTCMilliseconds()**

Same as getMilliseconds(), but returns the UTC milliseconds

# Date Objects

## Set Date Methods

### **setDate()**

Set the day as a number (1-31)

### **setFullYear()**

Set the year (optionally month and day)

### **setMonth()**

Set the month (0-11)

### **setHours()**

Set the hour (0-23)

### **setMinutes()**

Set the minutes (0-59)

# Date Objects

## **setSeconds()**

Set the seconds (0-59)

## **setMilliseconds()**

Set the milliseconds (0-999)

## **setTime()**

Set the time (milliseconds since January 1, 1970)

# Date Objects

## **new Date()**

Creates a new date object with the current date and time

## **new Date(year, month, ...)**

creates a new date object with a specified date and time

## **new Date(dateString)**

creates a new date object from a date string

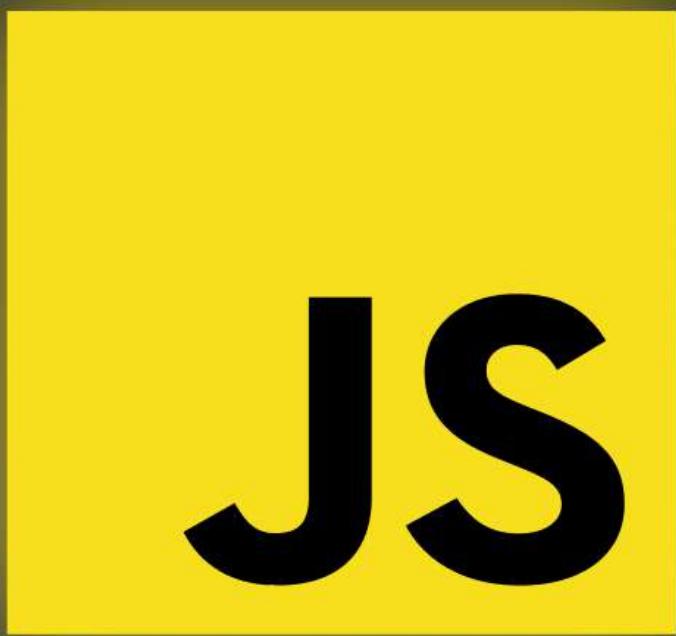
## **new Date(milliseconds)**

creates a new date object as zero time plus milliseconds



# JavaScript CheatSheet

- Window Object
- Window Methods



# Window Object Methods

## **alert()**

Displays an alert box with a message & an OK button

## **atob()**

Decodes a base-64 encoded string

## **blur()**

Removes focus from the current window

## **btoa()**

Encodes a string in base-64

## **clearInterval()**

Clears a timer set with setInterval()

## **clearTimeout()**

Clears a timer set with setTimeout()

# Window Object Methods

## **close()**

Closes the current window

## **confirm()**

Displays a dialog box with a message and an OK and a Cancel button

## **getComputedStyle()**

Gets the current computed CSS styles applied to an element

## **getSelection()**

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

## **matchMedia()**

Returns a MediaQueryList object representing the specified CSS media query string

# Window Object Methods

## **focus()**

Sets focus to the current window

## **moveBy()**

Moves a window relative to its current position

## **moveTo()**

Moves a window to the specified position

## **open()**

Opens a new browser window

## **print()**

Prints the content of the current window

## **prompt()**

Displays a dialog box that prompts the visitor for input

# Window Object Methods

## **requestAnimationFrame()**

Requests the browser to call a function to update an animation before the next repaint

## **resizeBy()**

Resizes the window by the specified pixels

## **resizeTo()**

Resizes the window to the specified width and height

## **scrollBy()**

Scrolls the document by the number of pixels

## **scrollTo()**

Scrolls the document to the specified coordinates

## **stop()**

Stops the window from loading

# Window Object Methods

## **setInterval()**

Calls a function or evaluates an expression at specified intervals (in milliseconds)

## **setTimeout()**

Calls a function or evaluates an expression after a specified number of milliseconds

# Window Object Properties

## **closed**

Returns a Boolean value indicating whether a window has been closed or not

## **console**

Returns a reference to the Console object, which provides methods for logging information to the browser's console

## **defaultStatus**

Sets or returns the default text in the statusbar of a window

## **document**

Returns the Document object for the window (See Document object)

## **frames**

Returns all <iframe> elements in the current window

# Window Object Properties

## history

Returns the History object for the window

## innerHeight

Returns the height of the window's content area (viewport) including scrollbars

## innerWidth

Returns the width of a window's content area (viewport) including scrollbars

## length

Returns the number of <iframe> elements in the current window

## localStorage

Allows to save key/value pairs in a web browser.  
Stores the data with no expiration date

# Window Object Properties

## **location**

Returns the Location object for the window

## **name**

Sets or returns the name of a window

## **navigator**

Returns the Navigator object for the window

## **opener**

Returns a reference to the window that created the window

## **parent**

Returns the parent window of the current window

## **self**

Returns the current window

# Window Object Properties

## **outerHeight**

Returns the height of the browser window, including toolbars/scrollbars

## **outerWidth**

Returns the width of the browser window, including toolbars/scrollbars

## **pageXOffset**

Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window

## **pageYOffset**

Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window

# Window Object Properties

## **screen**

Returns the Screen object for the window

## **screenLeft**

Returns the horizontal coordinate of the window relative to the screen

## **screenTop**

Returns the vertical coordinate of the window relative to the screen

## **screenX**

Returns the horizontal coordinate of the window relative to the screen

## **screenY**

Returns the vertical coordinate of the window relative to the screen

# Window Object Properties

## **sessionStorage**

Allows to save key/value pairs in a web browser.  
Stores the data for one session

## **scrollX**

An alias of pageXOffset

## **scrollY**

An alias of pageYOffset

## **status**

Sets or returns the text in the statusbar of a window

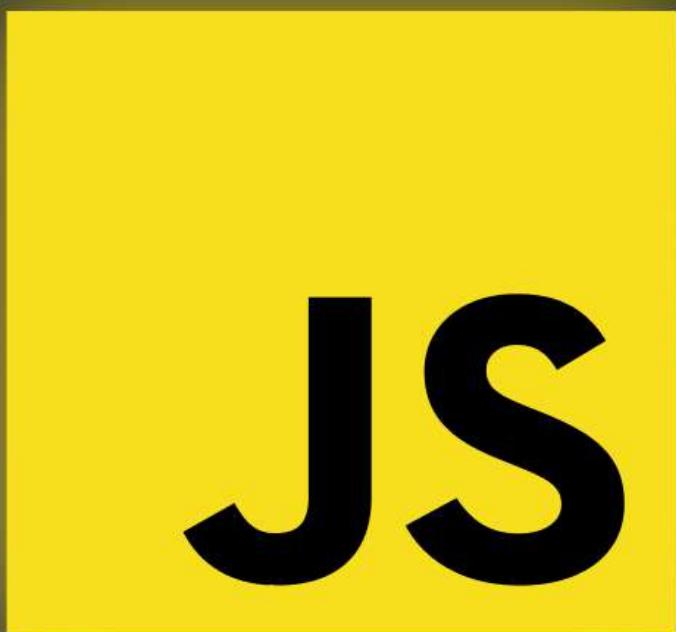
## **top**

Returns the topmost browser window



# JavaScript CheatSheet

- DOM Events
- DOM Properties



# **Input Events**

## **onblur**

When a user leaves an input field

## **onchange**

When a user changes the content of an input field

## **onfocus**

When an input field gets focus

## **onselect**

When input text is selected

## **onsubmit**

When a user clicks the submit button

## **onreset**

When a user clicks the reset button

# Input Events

## **onkeydown**

When a user is pressing/holding down a key

## **onkeypress**

When a user is pressing/holding down a key

## **onkeyup**

When the user releases a key

## **onkeydown**

When the user is pressing a key

# Mouse Events

## **onclick**

When the user clicks on an element

## **oncontextmenu**

When the user right-clicks on an element to open a context menu

## **ondblclick**

When the user double-clicks on an element

## **onmouseenter**

When the pointer is moved onto an element

## **onmouseleave**

When the pointer is moved out of an element

## **onmousemove**

when the pointer is moving while it is over an element

# Mouse Events

## **onmousedown**

When the user presses a mouse button over an element

## **onmouseout**

When a user moves the mouse pointer out of an element, or out of one of its children

## **onmouseover**

When the pointer is moved onto an element, or onto one of its children

## **onmouseup**

when a user releases a mouse button over an element

# HTML DOM Events

## **abort**

The event occurs when the loading of a media is aborted

## **afterprint**

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

## **animationend**

The event occurs when a CSS animation has completed

## **animationiteration**

The event occurs when a CSS animation is repeated

## **animationstart**

The event occurs when a CSS animation has started

# HTML DOM Events

## **beforeprint**

The event occurs when a page is about to be printed

## **beforeunload**

The event occurs before the document is about to be unloaded

## **blur**

The event occurs when an element loses focus

## **canplay**

The event occurs when the browser can start playing the media (when it has buffered enough to begin)

## **canplaythrough**

The event occurs when the browser can play through the media without stopping for buffering

# HTML DOM Events

## change

The event occurs when the content of a form element, or the checked state have changed

## click

The event occurs when the user clicks on an element

## contextmenu

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

## copy

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

## cut

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

# HTML DOM Events

## **dblclick**

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

## **drag**

The event occurs when an element is being dragged

## **dragend**

The event occurs when the user has finished dragging an element

## **dragenter**

The event occurs when the dragged element enters the drop target

## **dragleave**

The event occurs when the dragged element leaves the drop target

# HTML DOM Events

## **dragover**

The event occurs when the dragged element is over the drop target

## **dragstart**

The event occurs when the user starts to drag an element

## **drop**

The event occurs when the dragged element is dropped on the drop target

## **durationchange**

The event occurs when the duration of the media is changed

## **ended**

The event occurs when the media has reach the end (useful for messages like "thanks for listening")

# HTML DOM Events

## **error**

The event occurs when an error occurs while loading an external file

## **focus**

The event occurs when an element gets focus

## **focusin**

The event occurs when an element is about to get focus

## **focusout**

The event occurs when an element is about to lose focus

## **fullscreenchange**

The event occurs when an element is displayed in fullscreen mode

# HTML DOM Events

## **fullscreenerror**

The event occurs when an element can not be displayed in fullscreen mode

## **input**

The event occurs when an element gets user input

## **invalid**

The event occurs when an element is invalid

## **keydown**

The event occurs when the user is pressing a key

## **keypress**

The event occurs when the user presses a key

## **keyup**

The event occurs when the user releases a key

# HTML DOM Events

## **load**

The event occurs when an object has loaded

## **loadeddata**

The event occurs when media data is loaded

## **loadedmetadata**

The event occurs when meta data (like dimensions and duration) are loaded

## **loadstart**

The event occurs when the browser starts looking for the specified media

## **message**

The event occurs when a message is received through the event source

# HTML DOM Events

## **mousedown**

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

## **mouseenter**

The event occurs when the pointer is moved onto an element

## **mouseleave**

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

## **mousemove**

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

## **mouseover**

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

# HTML DOM Events

## **mouseout**

The event occurs when a user moves the mouse pointer out of an element, or out of one of its children

## **mouseup**

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

## **offline**

The event occurs when the browser starts to work offline

## **online**

The event occurs when the browser starts to work online

## **open**

The event occurs when a connection with the event source is opened

# HTML DOM Events

## **pagehide**

The event occurs when the user navigates away from a webpage

## **pageshow**

The event occurs when the user navigates to a webpage

## **paste**

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

## **pause**

The event occurs when the media is paused either by the user or programmatically

## **play**

The event occurs when the media has been started or is no longer paused

# HTML DOM Events

## **playing**

The event occurs when the media is playing after having been paused or stopped for buffering

## **popstate**

The event occurs when the window's history changes

## **progress**

The event occurs when the browser is in the process of getting the media data (downloading the media)

## **ratechange**

The event occurs when the playing speed of the media is changed

## **resize**

The event occurs when the document view is resized

# HTML DOM Events

## **reset**

The event occurs when a form is reset

## **scroll**

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

## **search**

The event occurs when the user writes something in a search field (for <input="search">)

## **seeked**

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

## **seeking**

The event occurs when the user starts moving/ skipping to a new position in the media

# HTML DOM Events

## **select**

The event occurs after the user selects some text (for <input> and <textarea>)

## **show**

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

## **stalled**

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

## **storage**

The event occurs when a Web Storage area is updated

## **submit**

The event occurs when a form is submitted

# HTML DOM Events

## **suspend**

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

## **timeupdate**

The event occurs when the playing position has changed (like when the user fast forwards to a different point in the media)

## **toggle**

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

## **touchcancel**

The event occurs when the touch is interrupted

## **touchend**

The event occurs when a finger is removed from a touch screen

# HTML DOM Events

## **touchmove**

The event occurs when a finger is dragged across the screen

## **touchstart**

The event occurs when a finger is placed on a touch screen

## **transitionend**

The event occurs when a CSS transition has completed

## **unload**

The event occurs once a page has unloaded (for <body>)

## **volumechange**

The event occurs when the volume of the media has changed (includes setting the volume to "mute")

# HTML DOM Events

## **waiting**

The event occurs when the media has paused but is expected to resume (like when the media pauses to buffer more data)

## **wheel**

The event occurs when the mouse wheel rolls up or down over an element

## **hashchange**

The event occurs when there has been changes to the anchor part of a URL

# HTML DOM Event Properties and Methods

## **altKey**

Returns whether the "ALT" key was pressed when the mouse event was triggered

## **animationName**

Returns the name of the animation

## **bubbles**

Returns whether or not a specific event is a bubbling event

## **button**

Returns which mouse button was pressed when the mouse event was triggered

# HTML DOM Event Properties and Methods

## **cancelable**

Returns whether or not an event can have its default action prevented

## **charCode**

Returns the Unicode character code of the key that triggered the onkeypress event

## **changeTouches**

Returns a list of all the touch objects whose state changed between the previous touch and this touch

## **clientX**

Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered

# HTML DOM Event Properties and Methods

## `clientY`

Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered

## `clipboardData`

Returns an object containing the data affected by the clipboard operation

## `code`

Returns the code of the key that triggered the event

## `composed`

Returns whether the event is composed or not

## `createEvent()`

Creates a new event

# HTML DOM Event Properties and Methods

## **ctrlKey**

Returns whether the "CTRL" key was pressed when the mouse event was triggered

## **currentTarget**

Returns the element whose event listeners triggered the event

## **dataTransfer**

Returns an object containing the data being dragged/dropped, or inserted/deleted

## **defaultPrevented**

Returns whether or not the preventDefault() method was called for the event

# HTML DOM Event Properties and Methods

## **deltaX**

Returns the horizontal scroll amount of a mouse wheel (x-axis)

## **deltaY**

Returns the vertical scroll amount of a mouse wheel (y-axis)

## **deltaZ**

Returns the scroll amount of a mouse wheel for the z-axis

## **deltaMode**

Returns a number that represents the unit of measurements for delta values (pixels, lines or pages)

# HTML DOM Event Properties and Methods

## **detail**

Returns a number that indicates how many times the mouse was clicked

## **elapsedTime**

Returns the number of seconds an animation or transition has been running

## **eventPhase**

Returns which phase of the event flow is currently being evaluated

## **getTargetRanges()**

Returns an array containing target ranges that will be affected by the insertion/deletion

# HTML DOM Event Properties and Methods

## **getModifierState()**

Returns an array containing target ranges that will be affected by the insertion/deletion

## **inputType**

Returns the type of the change (i.e "inserting" or "deleting")

## **isComposing**

Returns whether the state of the event is composing or not

## **isTrusted**

Returns whether or not an event is trusted

# HTML DOM Event Properties and Methods

## **data**

Returns the inserted characters

## **key**

Returns the key value of the key represented by the event or returns the key of the changed storage item

## **keyCode**

Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event

## **location**

Returns the location of a key on the keyboard or device

# HTML DOM Event Properties and Methods

## **lengthComputable**

Returns whether the length of the progress can be computable or not

## **loaded**

Returns how much work has been loaded

## **metaKey**

Returns whether the "META" key was pressed when an event was triggered

## **MovementX**

Returns the horizontal coordinate of the mouse pointer relative to the position of the last mousemove event

# HTML DOM Event Properties and Methods

## **MovementY**

Returns the vertical coordinate of the mouse pointer relative to the position of the last mousemove event

## **newValue**

Returns the new value of the changed storage item

## **newURL**

Returns the URL of the document, after the hash has been changed

## **offsetX**

Returns the horizontal coordinate of the mouse pointer relative to the position of the edge of the target element

# HTML DOM Event Properties and Methods

## **offsetY**

Returns the vertical coordinate of the mouse pointer relative to the position of the edge of the target element

## **oldValue**

Returns the old value of the changed storage item

## **oldURL**

Returns the URL of the document, before the hash was changed

## **onemptied**

The event occurs when something bad happens and the media file is suddenly unavailable (like unexpectedly disconnects)

# HTML DOM Event Properties and Methods

## **pageX**

Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered

## **pageY**

Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered

## **persisted**

Returns whether the webpage was cached by the browser

## **propertyName**

Returns the name of the CSS property associated with the animation or transition

# HTML DOM Event Properties and Methods

## **preventDefault()**

Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur

## **pseudoElement**

Returns the name of the pseudo-element of the animation or transition

## **relatedTarget**

Returns the element related to the element that triggered the mouse event

## **repeat**

Returns whether a key is being hold down repeatedly, or not

# HTML DOM Event Properties and Methods

## **screenX**

Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered

## **screenY**

Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered

## **shiftKey**

Returns whether the "SHIFT" key was pressed when an event was triggered

## **state**

Returns an object containing a copy of the history entries

# HTML DOM Event Properties and Methods

## **stopImmediatePropagation()**

Prevents other listeners of the same event from being called

## **stopPropagation()**

Prevents further propagation of an event during event flow

## **storageArea**

Returns an object representing the affected storage object

## **target**

Returns the element that triggered the event

## **total**

Returns the total amount of work that will be loaded

# HTML DOM Event Properties and Methods

## **targetTouches**

Returns a list of all the touch objects that are in contact with the surface and where the touchstart event occurred on the same target element as the current target element

## **timeStamp**

Returns the time (in milliseconds relative to the epoch) at which the event was created

## **touches**

Returns a list of all the touch objects that are currently in contact with the surface

## **transitionend**

The event occurs when a CSS transition has completed

# HTML DOM Event Properties and Methods

## **type**

Returns the name of the event

## **url**

Returns the URL of the changed item's document

## **which**

Returns which mouse button was pressed when the mouse event was triggered

## **which**

Returns the Unicode character code of the key that triggered the onkeypress event, or onkeyup / down

## **view**

Returns a reference to the Window object where the event occurred

# Want to Become a JavaScript Master ?

If you're serious about learning JavaScript and getting a job as a web developer. I highly encourage you to enroll in my **Complete Path to JavaScript Mastery Course**. Don't waste your time following disconnected, outdated tutorials. My Course has everything you need in one place:

This course has everything you need in one place:

- 10 hours of HD video
- Unlimited access
- Self-paced learning – take your time if you prefer
- Watch it online or download and watch offline
- 30-day money-back guarantee
- Certificate of completion