JavaScript Cheat Sheet + PDF

May 29, 2017 | No Comments
JavaScript. We've all heard of it, but understanding anything beyond the name is not a simple task.

That's why you're here:

> Start your blog today with a special OnBlastBlog discount from Bluehost. Only $2.95/mo with a free domain name and email address. You can't beat this offer!

It's time to brush up on your JavaScript.

Since you're here to learn how to create a website or make a blog from scratch, I'm going to show you what this Javascript programming language is all about, but more than that, I'm going to give you complete and unrestricted access to a cheat sheet that will give you the edge you've been looking for.

Ready? Here's what we're covering:

- ✔ JavaScript: What The Heck is That?
- ✔ The Greatest JavaScript Cheat Sheet This Side of The Internet
- ✔ Frequently Asked Questions About JavaScript
- ✔ The Power (and Limitations) Of JavaScript

JavaScript: What The Heck is That?

JavaScript is a programming language that's used to add interactivity to web pages. It runs on the computer of your visitors and doesn't require anything annoying like constant downloads to work properly. Examples are things like polls and quizzes.

Here are some quick facts about JavaScript:

- ✔ It's a lightweight, interpreted programming language
- ✔ Designed for creating network-centric applications
- ✔ Complementary to and integrated with Java and HTML
- ✔ Open and cross-platform
- ✔ Originally known as LiveScript, but Netscape changed it to JavaScript

RECENT BLOGGING HE


The Lady Blo
Guide to Get
Traffic from
2020
What would happen to your blog if
down forever? What if they suspen
account and there's no …


UK SEO: Unit
Kingdom Se
Engines Mar
Should Know
Did you know that Google is the mo
search engine in the UK, with 93.1%
it …


How to Start
Photography
Step by Step
There are already
million blogs online. If you're ready
ranks with a photography blog, now

The Greatest JavaScript Cheat Sheet This Side of The Internet

MORE RECENT BLOGG

What Does a Marketer Do: Important Th Understand

The way people do business has cha dramatically in the last decade. Havi presence is a must to …

How to Set U Store in Just Steps

Global ecommerc almost $3.46 trillion in 2019, and it it's slowing down anytime soon. …

How to Start Blog in 2020 Strategies fo

With millions of b you might not think there is room to mark in the blogging world, …

Social Media Sheet [Infogr

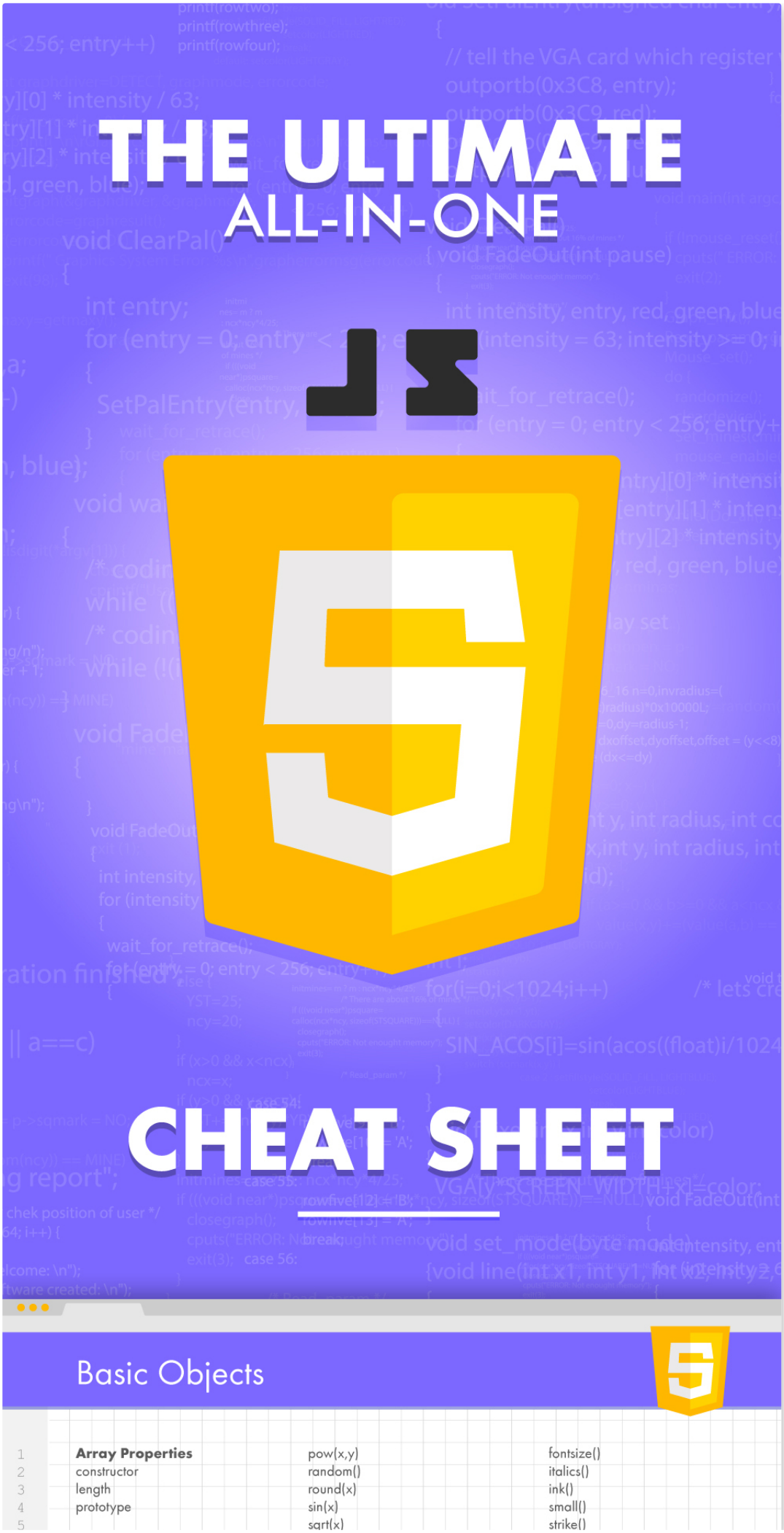
When I began cre around 2003, my with social media Myspace, man can I …

Link Building 2020 & Beyo

Websites don't jus show up and rank page. The majority that do rank are …

HELPFUL TIPS FOR BLO

- How to Create a Webs Free
- Free Blog Sites
- Blog Name Genarator



- Blog Post Ideas
- Working from Home
- Finding Blogging Jobs

BLOGGING GUIDES

- How to Start a Blog
- Best Free Blogging Sit
- How to Make a Websi
- Why Blog? 50+ Reaso
- What to Know Before Blog
- Blogging Best Practice
- Blog Post Ideas
- How to Make Money E
- How to Design the Per Website
- How to Promote Your
- How to Write the Perf Post
- Top WordPress SEO T Your Blog
- Why Free Websites ar Good

POPULAR BLOGGING T

Link Building Guide for 20 Beyond

List of Best Free Blogging Platforms

How to Make a Website

150+ Blog Post Ideas

The 10 Greatest Blog & Do Name Generators

Make An Impression With Gut-Bustingly Funny Busi Ideas

Living the Dream: A No No Guide on How to Work Fro

How to Start a Tech Blog: Haven't Heard Before

Array Methods
concat()
indexOf()
join()
lastIndexOf()
pop()
push()
reverse()
shift()
slice()
sort()
splice()
toString()
unshift()
valueOf()

Boolean Properties
constructor
Protoype

Boolean Methods
toString()
valueOf()

Math Properties
E
LN2
LN10
LOG2E
LOG10E
PI
SQRT1_2
SQRT2

Math Methods
abs(x)
acos(x)
asin(x)
atan(x)
atan2(y,x)
ceil(x)
cos(x)
exp(x)
floor(x)
log(x)
max(x,y,z...,n)
min(x,y,z...,n)
tan(x)

Number Properties
constructor
MAX_VALUE
MIN_VALUE
NEGATIVE_INFINITY
POSITIVE_INFINITY
prototype

Number Methods
toExponential(x)
toFixed(x)
toPrecision(x)
toString()
valueOf()

String Properties
constructor
length
prototype

String Methods
charAt()
charCodeAt()
concat()
fromCharCode()
indexOf()
lastIndexOf()
match()
replace()
search()
slice()
split()
substr()
substring()
toLowerCase()
toUpperCase()
valueOf()

String HTML Wrapper Methods
anchor()
big()
blink()
bold()
fixed()
fontcolor()
sub()
sup()

Global Properties
Infinity
NaN
undefined

Global Methods
decodeURI()
decodeURIComponent()
encodeURI()
encodeURIComponent()
escape()
eval()
isFinite()
isNaN()
Number()
parseFloat()
parseInt()
String()
unescape()

Date Objects

Date Properties

constructor
prototype

Date Methods

getDate()
getDay()
getFullYear()
getHours()
getMilliseconds()
getMinutes()
getMonth()
getSeconds()
getTime()
getTimezoneOffset()
getUTCDate()
getUTCDay()
getUTCFullYear()
getUTCHours()
getUTCMilliseconds()
getUTCMinutes()
getUTCMonth()
getUTCSeconds()
parse()
setDate()
setFullYear()
setHours()
setMilliseconds()
setMinutes()
setMonth()
setSeconds()
setTime()
setUTCDate()
setUTCFullYear()
setUTCHours()
setUTCMilliseconds()
setUTCMinutes()
setUTCMonth()
setUTCSeconds()
toDateString()
toISOString()
toJSQN()
toLocaleDateString()
toLocaleTimeString()
toLocaleString()
toString()
toTimeString()
toUTCString()
UTC()
valueOf()

Browser

Window Properties
closed
defaultStatus
document
frames
history
innerHeight
innerWidth
length
location
name
navigator
opener
outerHeight
outerWidth
pageXOffset
pageYOffset
parent
screen
screenLeft
screenTop
screenX
screenY
self
status
top

Window Methods
alert()
blur()
clearInterval()
clearTimeout()
close()
confirm()
focus()
moveBy()
moveTo()
open()
print()
prompt()
resizeBy()
resizeTo()
scrollBy()
scrollTo()
setInterval()
setTimeout()

Navigator Properties
appCodeName
appName
appVersion
cookieEnabled
platform
userAgent

Navigator Methods
javaEnabled()
registerContentHandler()
registerProtocolHandler()

Screen Properties
availHeight
availWidth
colorDepth
height
pixelDepth
width

History Properties
length

History Methods
back()
forward()
go()

Location Properties
hash
host
hostname
href
pathname
port
protocol
search

Location Methods
assign()
reload()
replace()

DOM Events

Mouse Events
click
dblclick
mousedown
mousemove
mouseover
mouseout
mouseup

Keyboard Events
keydown
keypress
keyup

Frame Events
abort
error
load
resize
scroll
unload

Form Events
blur

Event Object Properties
bubbles
cancelable
currentTarget
eventPhase
target
timeStamp
type

Event Object Methods
initEvent()
preventDefault()
stopPropagation()

EventTarget Object
addEventListener()
dispatchEvent()
removeEventListener()

EventListener Object
handleEvent()

MouseEvent/KeyboardEvent Object
screenY
shiftKey

MouseEvent/KeyboardEvent Methods
initMouseEvent()
initKeyboardEvent()

change
focus
reset
select
onsubmit

Event Object Constant
AT_TARGET
BUBBLING_PHASE
CAPTURING_PHASE

altKey
button
clientX
clientY
ctrlKey
keyIdentifier
keyLocation
metaKey
relatedTarget
screenX

DOM Node

Node Properties
attributes
baseURI
childNodes
firstChild
lastChild
localName
namespaceURI
nextSibling
nodeName
nodeType
nodeValue
ownerDocument
parentNode
prefix
previousSibling
textContent

NodeMethods
appendChild()
cloneNode()
compareDocumentPosition()
getFeature(feature,version)
getUserData(key)
hasAttributes()
hasChildNodes()
insertBefore()
isDefaultNamespace()
isEqualNode()
isSameNode()
isSupported()
lookupNamespaceURI()
lookupPrefix()
normalize()
removeChild()
replaceChild()
setUserData(key,data,handler)

Node Types
1 - Element
2 - Attr
3 - Text
4 - CDATASection
5 - EntityReference
6 - Entity
7 - ProcessingInstruction
8 - Comment
9 - Document
10 - DocumentType
11- DocumentFragment
12 - Notation

nodeName Returns
Element
Element name

Attr
attribute name

Text
#text

CDATASection
#cdata-section

EntityReference
null

Entity
null

ProcessingInstruction
content of node

Comment
comment text

Document
null

DocumentType
null

DocumentFragment
null

Notation
null

RegExp

Modifiers
i
Perform case-insensitive matching

g
Perform a global match (find all matches, instead of stopping after the first match)

m
Perform multi line matching

\W
Find a non-word character

\d
Find a digit

\D
Find a non-digit character

\s
Find a whitespace character

\S
Matches any string that contains at least one n

n*
Matches any string that contains zero or more occurrences of n

n?
Matches any string that contains zero or one occurrences of n

n{X}
Matches any string that contains

Brackets

[abc]
Find any character between the brackets

[^abc]
Find any character not between the brackets

[0-9]
Find any digit from 0 to 9

[A-Z]
Find any character from uppercase A to uppercase Z

[a-z]
Find any character from lowercase a to lowercase z

[A-z]
Find any character from uppercase A to lowercase z

[adgk]
Find any character in the given set

[^adgk]
Find any character outside the given set

(red|blue|green)
Find any of the alternatives specified

Metacharacters

Find a single character, except newline or line terminator

\w
Find a word character

Find a non-whitespace character

\b
Find a match at the beginning/end of a word

\B
Find a match not at the beginning/end of a word

\0
Find a NUL character

\n
Find a new line character

\f
Find a form feed character

\r
Find a carriage return character

\t
Find a tab character

\v
Find a vertical tab character

\xxx
Find the character specified by an octal number xxx

\xdd
Find the character specified by a hexadecimal number dd

\uxxxx
Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

n+
a sequence of X n's

n{X,Y}
Matches any string that contains a sequence of X to Y n's

n{X,}
Matches any string that contains a sequence of at least X n's

n$
Matches any string with n at the end of it

^n
Matches any string with n at the beginning of it

?=n
Matches any string that is followed by a specific string n

?!n
Matches any string that is not followed by a specific string n

RegExp Properties

global
ignoreCase
lastIndex
multiline
source

RegExp Methods

exec()
test()

Core DOM

Nodelist Properties

length

Nodelist Methods

item()

NamedNodeMap Properties

length

NamedNodeMap Methods

getNamedItem()
getNamedItemNS()
item()
removeNamedItem()
removeNamedItemNS()
setNamedItem()
setNamedItemNS()

Document Properties

doctype
documentElement
documentURI
domConfig
implementation
inputEncoding
xmlEncoding

Document Methods

adoptNode(node)
createAttribute()
createAttributeNS(URI,name)
createCDATASection()
createComment()
createDocumentFragment()
createElement
create ElementNS()
createEntityReference()
createProcessingInstruction()
createTextNode()
getElementById()
getElementsByTagName()
getElementsbyName()
querySelectorAll()
querySelector()
getElementsByTagNameNS()
importNode()
normalizeDocument()

Element Properties

schemaTypeInfo
tagName

Element Methods

getAttributeNS()
getAttributeNode()
getAttributeNodeNS()
getElementsByTagName()
getElementsByTagNameNS()
hasAttribute()
hasAttributeNS()
removeAttribute()
removeAttribute(NS)
removeAttributeNode()
setAttribute()
setAttributeNS()
setAttributeNode()
setAttributeNodeNS()
setIdAttribute()
setIdAttributeNS()
setIdAttributeNode()

Attr Properties

isId
name
ownerElement
schemaTypeInfo
specified
value

```
xmlVersion                    getAttribute()
```

Graphics

CanvasRenderingContext 2D Methods

arc()
arcTo()
beginPath()
bezierCurveTo()
clearRect()
clip()
closePath()
createImageData()
createLinearGradient()
createPattern()
createRadialGradient()
drawCustomFocusRing()
drawImage()
drawSystemFocusRing()
fill()
fillRect()
fillText()
getImadeData()
getLineDash()
isPointInPath()
lineTo()
measureText()
moveTo()
putImageData()
quadracticCurveTo()
rect()
restore()
rotate()
save()
scale()
scrollPathIntoView()
setLineDash()
setTransform()
stroke()
strokeRect()
strokeText()
transform()
translate()

Quick Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <title>Odd or Even?</title>
  </head>
  <body>
    <input type="number" id="inputNumber"><button id="runBtn">Odd or Even?</button>
    <p class="result">Enter a number and click the button to find out if the number is odd or even.</p>
    <script type="text/javascript">
      // Use a wrapping function to prevent global variables
      (function () {
        // Enable the strict mode (in short, it tells us some potential errors)
        "use strict";

        // You can select elements using document.querySelector
        var $result = document.querySelector(".result");

        // Another way is using getElementById(<id>)
        var $input = document.getElementById("inputNumber");
        var $btn = document.getElementById("runBtn");

        // Append a click handler on the button
        $btn.addEventListener("click", function (event) {

          // Validate the input
          if (!$input.value) {
            return alert("Please provide a number.");
          }

          // When clicking on the button, take the value from input
          // Also convert it into a number
          var x = +$input.value;

          // Create  a new variable that will contain the result
          // ("odd" or "event")
          var result;
```

```
                // Check if the number is even
                if (x % 2 === 0) {
                  result = "even";
                } else {
                  result = "odd";
                }

                // Tell the user if the number is odd or even
                $result.textContent = x.toString() + " is " + result;
              });
            })();
          </script>
        </body>
      </html>
```

Accessing Dom Elements

```
//Returns a reference to the element by its ID.
document.getElementById(id);

//Returns an array-like object of all child elements which have all of the given class names.
document.getElementsByClassName(names);

//Returns an HTMLCollection of elements with the given tag name.
document.getElementsByTagName(name);

//Returns the first element within the document that matches the specified group of selectors.
document.querySelector(selectors);

//Returns a list of the elements within the document (using depth-first preorder traversal of the document's nodes)
//that match the specified group of selectors.
document.querySelectorAll(selectors);
```

Grab Children/Parent Node(s)

```
//Get child nodes
var stored = document.getElementById('heading');
var children = stored.childNodes;
console.log(children);

//Get parent node
var parental = children.parentNode;
```

Create New DOM Elements

```
//create new elments
```

```
var newHeading = document.createElement('h1');
var newParagraph = document.createElement('p');

//create text nodes for new elements
var h1Text= document.createTextNode("This is the fucking header text!");
var paraText= document.createTextNode("This is the fucking Paragraph text!");

//attach new text nodes to new elements
newHeading.appendChild(h1Text);
newParagraph.appendChild(paraText);

//elements are now created and ready to be added to the DOM.
```

Add Elements to the DOM

```
//grab element on page you want to add stuff to
var firstHeading = document.getElementById('firstHeading');

//add both new elements to the page as children to the element we stored in firstHeading.
firstHeading.appendChild(newHeading);
firstHeading.appendChild(newParagraph);

//can also insert before like so

//get parent node of firstHeading
var parent = firstHeading.parentNode;

//insert newHeading before FirstHeading
parent.insertBefore(newHeading, firstHeading);

//Suppose you have the following HTML
<div id="box1">
    <p>Some example text</p>
</div>
<div id="box2">
    <p>Some example text</p>
</div>

//you can insert another snippet of HTML between #box1 and #box2
var box2 = document.getElementById("box2");
box2.insertAdjacentHTML('beforebegin', '<div><p>This gets inserted.</p></div>');

//beforebegin - The HTML would be placed immediately before the element, as a sibling.
//afterbegin - The HTML would be placed inside the element, before its first child.
//beforeend - The HTML would be placed inside the element, after its last child.
//afterend - The HTML would be placed immediately after the element, as a sibling.
```

Add/Remove/Toggle/Check Classes

```
//grab element on page you want to add stuff to
var firstHeading = document.getElementById('firstHeading');

//add both new elements to the page as children to the element we stored in firstHeading.
firstHeading.appendChild(newHeading);
firstHeading.appendChild(newParagraph);

//can also insert before like so

//get parent node of firstHeading
```

```
var parent = firstHeading.parentNode;

//insert newHeading before FirstHeading
parent.insertBefore(newHeading, firstHeading);

//Suppose you have the following HTML
<div id="box1">
    <p>Some example text</p>
</div>
<div id="box2">
    <p>Some example text</p>
</div>

//you can insert another snippet of HTML between #box1 and #box2
var box2 = document.getElementById("box2");
box2.insertAdjacentHTML('beforebegin', '<div><p>This gets inserted.</p></div>');

//beforebegin - The HTML would be placed immediately before the element, as a sibling.
//afterbegin - The HTML would be placed inside the element, before its first child.
//beforeend - The HTML would be placed inside the element, after its last child.
//afterend - The HTML would be placed immediately after the element, as a sibling.
```

Add/Remove Array Item

```
//create an empty array
var myArray = [];

//create array with items. Can store any type
var myOtherArray = [myArray, true, "A random string"];

//call specific value in an array
myOtherArray[0];
//will return myArray

//change value for this item
myOtherArray[0] = false;
//will now return false


//add to end of array
myOtherArray[myOtherArray.length] = "new stuff";
//will return the new item "new stuff"

//or you can use push()
myOtherArray.push("new stuff");
//will return new length of array


//you can remove this last item by using pop()
myOtherArray.pop();
//will return the last item of the array and will have removed it from myOtherArray


//shift and unshift will do the same for the begging of the Array
myOtherArray.shift();
//will remove and return first item of array

myOtherArray.unshift(1,2);
//this will add 1 and 2 to beginning of array and return new length

//you can use delete keyword but turn value to undefine and not shorten length. so we use splice()
myOtherArray.splice(2, 1);
//this will remove and return  the third item only.
//first arg is where to start and second is how many things to splice. this example is 1.
```

Adding/Removing Properties in Object

```
//create an object
var newObject = {};

//add a property to object
newObject.newPropName = "super fly";

//or other syntax
newObject['other new prop name'] = "mildly fly";

//Now newObject.newPropName will return super fly
newObject.newPropName;

//now to delete
delete newObject.newPropName;
```

Conditionals

```
//If Else statements
var a = 1;
varb b = 2;

if( a < b ) {
  console.log('the if is true!');
} else {
  console.log('the if is false!');
}


//Multi If Else statements
var a = 1;
varb b = 2;
varb c = 3;

if( a > b ) {
  console.log('the if is true!');
} else if(a > c) {
  console.log('OK, THIS if is Ture!');
} else {
  console.log('None of these were true');
}


//Ternary operators. same as if else
var a = 1;
varb b = 2;

a === b ? console.log('The statement is true) : console.log('The statement is false');


//switch statements.
var a = 4;
switch (a) {
  case "Oranges":
    console.log("Orange? really?");
    break;
  case 1:
    console.log("a is equal to 1.");
    break;
  case 2:
    console.log("a is equal to 2.");
    break;
  case 3:
    console.log("a is equal to 3.");
    break;
```

```
    break;
  case 4:
    console.log("a is equal to 4.");
    break;
  default:
    console.log("I run if no one else is true.");
}
```

Loops

```
//while loop
var i = 0;
while( i < 10 ) {
  console.log(i);
  i += 1
}

//do while loop
var i = 0;
do {
  console.log(i);
  i += 1
} while( i < 10 )

//for loop
for ( var i = 0; i < 10; i++ ) {
    console.log(i);
}

//for in statments
var obj = {a:1, b:2, c:3};

for ( var prop in obj ) {
  //check if property is inherited or not
  if(obj.hasOwnProperty(prop)) {
    console.log("obj." + prop + " = " + obj[prop]);
  }

}
```

Events

```
var newElement = document.getElementsByTagName('h1');

newElement.onclick = function() {
  console.log('clicked');
};

newElement.addEventListener("focus", function(event) {
    console.log('focused');
}, false);

newElement.removeEventListener("focus", function(event) {
    console.log('focused');
}, false);
```

```
window.onload = function() {
  console.log("Im loaded");
};
```

Timers

```
function simpleMessage() {
  alert("This is just a simple alert");
}

//set time out
window.setTimeout(simpleMessage, 5000);

//if you wanted to clear the timer.
var timer = window.setTimeout(simpleMessage, 5000);
window.clearTimeout(timer);

//set interval. will repeat every  5000ms
window.setInterval(simpleMessage, 5000);

//if you wanted to clear the intervals.

var intervalHandler = window.setInterval(simpleMessage, 5000);
window.clearInterval(intervalHandle);
```

Type Checking

```
var myNumber = 1;
var myString = "some Text";
var bools = true;
var myArray = [];
var myObj = {};
var notNumber = NaN;
var nullified = null;

typeof myNumber;
//returns "number"

typeof myString;
//returns "string"

typeof bools;
//returns "boolean"

typeof myArray;
//returns "object".

//Not super helpful so must check if it has length property to see if it is an array.
typeof myArray === 'object' && myArray.hasOwnProperty('length');
//returns true

typeof myObj;
//returns "object". Must do the same test as above but expect false back from check.

typeof notNumber;
```

```
typeof notNumber;
//returns "number". this is confusing but returns this as NaN is part of the global Number object.

//must check if isNaN()
typeof notNumber === 'number' && isNaN(notNumber);
//returns true if type of is "number" and is still NaN
```

Add Default Arguments for Function

```
//you provide defaults inside your function

var myFunction = function myFunction(arg1, arg2) {
  var arg1 = (typeof arg1 !== 'undefined') ? arg1 : "default argument one";
  var arg2 = (typeof arg2 !== 'undefined') ? arg2 : "default argument two";
  console.log(arg1 + " & " + arg2);
};
```

Throttle Functions on Resize

```
var optimizedResize = (function() {

  var callbacks = [];
  var running = false;

  // fired on resize event
  function resize() {

    if (!running) {
      running = true;

      if (window.requestAnimationFrame) {
        window.requestAnimationFrame(runCallbacks);
      } else {
          setTimeout(runCallbacks, 66);
      }
    }

  }

  // run the actual callbacks
  function runCallbacks() {
    callbacks.forEach(function(callback) {
      callback();
    });
    running = false;
  }

  // adds callback to loop
  function addCallback(callback) {
    if (callback) {
      callbacks.push(callback);
    }
  }

  return {
    // initalize resize event listener
    init: function(callback) {
      window.addEventListener('resize', resize);
```

```
      addCallback(callback);
    },
    // public method to add additional callback
    add: function(callback) {
      addCallback(callback);
    }
  }

}());

// start process
optimizedResize.init(function() {
    console.log('Resource conscious resize callback!')
});
```

Useful Math Functions and Constants

```
// Get the PI number
> Math.PI
3.141592653589793

// Get the E number
> Math.E
2.718281828459045

// Maximum between two numbers
> Math.max(7, 42)
42

// Minimum between numbers (arguments)
> Math.min(7, 42, 255, 264)
7

// Power
> Math.pow(2, 4)
16

// Random number between 0 (inclusive) and 1 (exclusive)
> Math.random()
0.2114640628617317

// Round
> Math.round(41.8)
42
```

SOURCES:

http://overapi.com/javascript
https://www.cheatography.com/davechild/cheat-sheets/javascript/
https://www.codementor.io/johnnyb/javascript-cheatsheet-fb54lz08k
https://codepen.io/davidicus/details/trhme
https://firstsiteguide.com/javascript-cheat-sheet/

Want to embed this cheat sheet?

Copy the snippet below & paste right into your text editor to share the love ❤

```
<a href="https://www.onblastblog.com/javascript-cheat-sheet/"><img src="https://www.onblastblog.com/wp-content/uploads/2017/05/JS-Cheat-Sheet.jpg" alt="javascript cheat sheet" width="1000px" border="0" /></a>
Credit: <a href="https://www.onblastblog.com/javascript-cheat-sheet/" target="_blank">On Blast Blog</a>
```

Frequently Asked Questions About JavaScript

Here's one:

Are JavaScript and Java the same thing?

Something we need to get out of the way right now, is that **Java and JavaScript are not the same thing.** The names are similar, yes, but that's about the only similarity.

Now that we've covered that small, but important detail, we can dig deeper into JavaScript as a whole.

What do users need to utilize JavaScript on websites they visit?

This programming language is built into most major web browsers like Internet Explorer, Firefox, and Safari.

Users who visit your site need to have JavaScript support on their browsers, and it must be enabled (it is by default), for everything you've written to function properly.

Can I only use JavaScript if I know how to write it?

Here's an interesting fact:

You don't need an intimate knowledge of JavaScript to use it!

That's right, there are plenty of pre-written JavaScripts that people have made available for new users to just plug right into their web pages. You just have to know where to place the scripts, and you're good to go.

Is there a special program for writing JavaScript?

Now, if you're interested in writing JavaScript, you'll just need a plain text editor, like Notepad to get started. If you're using an editor to write your code, the job gets even easier, because these programs

will color-code the text to help you find mistakes quicker.

Can I use HTML instead of JavaScript?

No, JavaScript and HTML are two different beasts, but you can check out our HTML5 cheat sheet if you're in need of help with that programming language. You see, HTML is used as a markup language for defining elements of static web pages, while JavaScript is a programming language that's used for dynamic tasks.

Can I use PHP or some other server-side language instead?

This is a tough one to answer because it is possible. You can only use an alternate language if it runs before the page loads. Anything that runs after the page has loaded must be JavaScript. It's the only language supported by all web browsers that supports client-side scripting.

Does JavaScript go in the same file as HTML?

You can do it this way, but it's more easily used across multiple pages if you place it in separate files. You can use a .JS extension to help differentiate them. You can link your JavaScript to HTML by inserting a <script> tag.

You can add this same script to several pages by using the correct tags into each of the pages to establish a link.

What is Client-side JavaScript?

Client-side JavaScript is the most commonly used form of this language. The script should be added to, or referenced in HTML documents for the code to be properly understood and interpreted by the browser.

A web page can be made using static HTML but can include programs that interact with users and create dynamic HTML content. The client-side JavaScript mechanism allows you to do a lot more than you could with traditional CGI server-side scripts.

For example, JavaScript can be used to check if the user has entered a valid email address in a form field. Handy stuff!

The Advantages (And Limitations) Of JavaScript

JavaScript has a lot going for it, but like anything, it's not perfect. Let's take a look at what it does really well, and where you may need something else to get the job done. Understanding these limitations is key to properly utilizing the language.

The Advantages of JavaScript

Less Server Interaction – You can ensure that user input is valid before sending it to the server. This saves traffic and reduces the load on the server as a whole.

Immediate Feedback – Users can immediately find out if they've missed an entry, as opposed to waiting for a page to load.

More Interactivity – You can use JavaScript to create interfaces that react to users when they hover over them with a mouse or trigger them via a keyboard.

Richer User Interface – You can use JavaScript to include items like drag-and-drop components and sliders to improve the quality of your interfaces.

As you can see, JavaScript does a lot of things really well, but let's look at the limitations before we part. These are some of the features that some people may find lacking.

JavaScript's Limitation

Client-side JavaScript doesn't allow reading or writing of files. This was chosen for security reasons.

JavaScript cannot be used for networking applications because there's no support available for this.

JavaScript doesn't have support for multi-threading or multiprocessor capabilities.

Final Thoughts

JavaScript is a powerful tool for making your web pages more interactive, engaging, and better suited to the user experience. With this knowledge and cheat sheet, you'll have everything you need to create dynamic and exciting web pages.

How do you use JavaScript? Let us know in the comments!

Advanced Blogging, Technical Blogging

Post navigation

CMS Comparison: Joomla vs WordPress vs Drupal

Customizing WordPress