Cheatography

JavaScript Survival Kit Cheat Sheet

by pseud via cheatography.com/26472/cs/7447/

About this document

This cheat-sheet explains the most important parts of the JavaScript language, defines some key terms and shows the syntax through small examples.
However, it's no substitute for proper studying - you can't learn to program off of a cheat sheet (sorry!).

Variables - Explained

What is a variable?

A variable is a storage location, a "box", which we associate with a name (an identifier). The variable can hold a single value and its value may be changed

What is an identifier?

It's the "name" affixed the variable. Later on, whether updating or retrieving its value, we'll use refer to the variable by its identifier.

What can a variable hold?

Any `string`, `number`, `boolean`, `array`, `object` or `Function`.

Variables - Explained (cont)

Why use variables?

Use them to "remember" things in the program. Sometimes, the collection of all variables (everything the program remembers) is called *the state* of the program.

Where to read more

Read the section "Variables" at:
eloquentjavascript.net/02_program_structure.html

Variables - Examples

Define a variable

```
var name = "Adalina";
```

NB - subsequent examples assume we have defined this variable.

Retrieve the variable's value

Simply refer to the variable's *identifier*:

```
console.log(name);
```

is (in this case) the same as:

```
console.log("Adalina");
```

Update the variable's value

```
name = "Emma";
```

NB - The syntax is the same as defining the variable, *sans* the `var` keyword!

Objects - Explained

What is an object?

If a variable is a "box" which can hold a value, then an object is a box of boxes, holding many values - each of which is a *property*.

What is a property?

A property is some small part of an object which holds some data (e.g. `string`) or a `Function`. Each property has an *identifier*, just like variables.

Where to read more

eloquentjavascript.net/04_data.html - The introduction and the paragraphs "Properties" and "Objects"

Objects - Examples

Define an object

Define an object with two properties whose identifiers are "name" and "species":

```
var my_pet = {
   name: "spot",
   species: "dog"
}
```

NB Subsequent examples will assume we start with this object.
NB It isn't necessary to define a variable to hold the array (but you almost always will).

By **pseud**
cheatography.com/pseud/

Published 9th March, 2016.
Last updated 11th March, 2016.

</> Objects - Examples (cont)

Retrieve a property

Get the value of the `name` property:

```
my_pet["name"]
```

or

```
my_pet.name
```

Update a property

To change the value of the `name` property (i.e. rename our pet):

```
my_pet["name"] = "sparky";
```

or

```
my_pet.name = "sparky";
```

Add a property

```
my_pet["breed"] = 'bulldog';
```

or

```
my_pet.breed = 'bulldog';
```

NB adding/updating a property uses the same syntax - if the property didn't exist, it is added.

Remove a property

To remove the `species` property:

```
delete my_pet["species"];
```

or

```
delete my_pet.species;
```

Comparisons

`x === y`	`true` if `x` is equal to `y`
`x !== y`	`true` if `x` is different from `y`
`x >= y`	`true` if `x` is greater than, or equal to `y`
`x <= y`	`true` if `x` is less than, or equal to `y`

Comparisons (cont)

`x > y`	`true` if `x is greater than y``
`x < y`	`true` if `x` is less than `y`
`!x`	`true` if `x` is `false`
`x && y`	`true` if *both* `x` *and* `y` are `true`
`x || y`	`true` if *either* (or both) `x` or `y` are `true`

Conditions - False & True

What's a condition?

A condition is really just an expression. When we use an expression as a condition, we're not interested in its value, but whether or not that value is *truthy*.

What's a truthy value?

In JavaScript, all but 6 values are `truthy`, that is, unless your *condition* evaluates to one of those 6 values, the code guarded by the if-block will be run.

What are the falsy values?

These 6 values will cause the condition to fail and the code it guards to be skipped:

- `false`
- `0` - (the number zero)
- `""` - (the empty string)
- `null`
- `undefined`
- `NaN` - not a number

Conditions - False & True (cont)

Where are conditions used?

Conditions determine which code block to evaluate in if-statements and when to terminate a loop.

Functions - Explained

What is a function?

Functions group code together into a block which is given a name (an *identifier*). Functions often accept arguments to modify their behaviour.

What is an argument?

Think of function arguments as variables which are defined & available to the code inside the function. The value of an argument is determined by the point the function is called and the argument(s) is supplied.

Why use functions?

Functions are the primary way of defining more complex or specific actions than is built into JavaScript and to organise code. *In other words - functions are handy when we wish to use a piece of code more than once.*

Where to read more

eloquentjavascript.net/03_functions.html

By **pseud**
cheatography.com/pseud/

Published 9th March, 2016.
Last updated 11th March, 2016.

</> Functions - Syntax

Define a function

Define a function called `takeFive`, which returns the number 5 when called:

```
function takeFive() {
   return 5;
}
```

NB - we will be using this function in some of the examples below.

Call a function

Call `takeFive`, which takes no arguments:

```
takeFive();
```

NB - Note the parentheses' that follow the function's identifier - *that's* what tells JavaScript to call the function rather than just returning it as a (`Function`) value.

Define a function (with arguments)

```
function add5(num) {
   console.log("I got num=" +
num);
   return num + 5;
}
```

NB - To have more arguments than just `num`, type out additional identifiers (names) of arguments and add a comma (`,`) between each.

</> Functions - Syntax (cont)

Call a function (with arguments)

```
var x = add5(10);
var y = add5(-5);
```

NB - *This amounts to manually typing:*

```
var num1 = 5;
console.log("I got num=" +
num1);
var x = num1 + 5;
var num2 = -5;
console.log("I got num=" +
num2);
var y = num2 + 5;
```

🎓 if-statement - Explained

What's an if-statement?

If-statements are used to group code together into a block which is *only* evaluated if the *condition* evaluates to `true`.

NB - see "Conditions - Falsy & Truthy" for an explanation of conditions.

What does an if-statement look like?

```
if (CONDITION) {
   //evaluate this code if
CONDITION
   //is true
} else if (OTHER-CONDITION) {
   //evaluate this code if
CONDITION
   //is false, but OTHER-
CONDITION
   //is true
} else {
   //evaluate this code if no
condition
   //evaluated to true.
}
```

🎓 if-statement - Explained (cont)

Which parts are needed?

Only the `if`-part is needed. `else if` and `else` blocks are optional.

Also, you can have as many `else if` blocks as you'd like.

</> if-statements - Examples

if-statement

```
if (pet_type === "dog") {
   //done if var 'pet_type' is
"dog"
}
```

if/else statement

```
if (pet_type === "dog") {
   //if var 'pet_type' is "dog"
} else {
   //if var 'pet_type' is
something else
}
```

if/else if/else statement

```
if (pet_type === "dog") {
   //if var 'pet_type' is "dog"
} else if (pet_type === "cat") {
   //if var 'pet_type' is "cat"
} else {
   //if var 'pet_type' is
something else
}
```

🎓 (while) Loops - Explained

What is a (while) loop?

Loops allow repeating a block of code for as long as a *condition* remains true.

By **pseud**
cheatography.com/pseud/

Published 9th March, 2016.
Last updated 11th March, 2016.

(while) Loops - Explained (cont)

Real world (tm) loop example

Think of this exchange:
Passenger: *Are we there yet?*
Driver: *No, not yet*
...
Passenger: *Are we there yet?*

If the passenger is really obnoxious and keeps repeating the question, and the driver patiently answers each time - they are essentially in a conversational *loop*!

Syntax Example

```
while (CONDITION) {
   //evaluate code in this block
}
```

Where to read more

Mid-way through the page linked below, look for the heading "while and do loops":
eloquentjavascript.net/02_program_structure.html

(while) Loops - Examples

How do I loop forever

```
while (true) {
   //keep doing this until time ends
}
```

(while) Loops - Examples (cont)

How do I loop X times?

To loop X times (say 3), we ensure the condition evaluates to `false` at the start of the fourth loop:

```
var count = 0;
while (count < 4) {
   //increase count by 1
   count = count + 1;
   //evaluate this code until count
   //is 4 or more
}
```

NB - if we don't ensure our condition eventually becomes invalid, we will loop forever.

Data Types

Number	Any numeric value - `3`, `3.14`, `2e10`
String	Any sequence of characters inside quotation marks. `"d"`, `"dog"`, `"cute dog"`
Boolean	Two possible values, `true` or `false`. Used as conditions in if-statements & loops. Every expression can be boiled down into a boolean.
Array	A sequence of elements grouped together. E.g. `[1, 2, 3]` is an array of 3 numbers.
Object	An object which groups other values. `{ name: "Rachel", age: 22 }`

Syntax - (basic) data types

String

```
"d"
"To be or not to be"
"300" //in quotes, this is a string
'single quotes also work'
```

Number

```
300
3.1415
2e10
```

Boolean

```
true
false
```

Terminology

Syntax

The collection of rules about "what goes where" to form valid JavaScript code. **NB** - if you get a syntax error, you've written some code which isn't legal javascript.

Statement

A piece of code (usually a single line) which represents something we want done - some small task.

Expression

Some piece of code which, when evaluated, will yield a value back. E.g. `3 + 6`

Terminology (cont)

Evaluation

The thing which happens when the JavaScript interpreter analyses a piece of code and either does something in response (a statement) or yields a value (an expression).

JavaScript Interpreter

Some program which can understand, and act on JavaScript. Your browser (Firefox/Chrome) is a JavaScript interpreter.

(Code) Block

Blocks are delimited by `{ }` and used by if-statements, loops and functions to encapsulate some series of *statements* which should be executed.

Arrays - Explained

What is an array?

An array is a sequence of elements. Each element can be retrieved from the array by its index number.

What is an array element?

An element part of an array, it can be any data type (`string`, `number`, `boolean`, `array`, `object`) but it could also be a `Function`.

How can I get elements from the array?

The first element has index 0, the second has index 1 and so on.

Arrays - Explained (cont)

Where to read more

Read the introduction and the paragraph "data sets" at:
eloquentjavascript.net/04_data.html

Arrays - Examples

Defining an array

Define an array with 3 elements, the string `"one"`, the number `2` and the boolean `false`, in that order:

```
["one", 2, "three"]
```

Retrieve an element from the array

Get the second element of the array, `"b"`, by indexing into the array using the index number `1`:

```
["a", "b", "c"][1]
```

Updating an element

```
var pets = ['dog', 'cat',
'canary'];
pets[1] = 'lion';
```

Now the array would be:

```
['dog', 'lion', 'canary']
```

Add an element

Use the method `push`. **NB** push adds elements to the *end* of the array.

```
var pets = ['dog', 'cat',
'canary'];
pets.push('crocodile');
```

Now the array would be:

```
['dog', 'cat', 'canary',
'crocodile']
```

Arrays - Examples (cont)

Remove element(s)

Use `splice` - `splice` needs two arguments, the index of where to start and a number of elements to remove.

```
var pets = ['dog', 'cat',
'fish', 'bird'];
pets.splice(1,2);
```

Now the array would be:

```
['dog', 'bird']
```

Get number of elements in array

Use the `length` property on the array:

```
pets.length
```

Yes, arrays are actually a kind of object(!!) - which means it has some properties (like `length`) and methods attached to it.

Where to go for more?

http://www.codher.com

CodHer's official website :)
Learn about the organisation and upcoming events

asosio.com/72

CodHer's Asosio community.
Ask the mentors, get new JS assignments, download learning materials and (please!) discuss JavaScript with other attendees.

facebook.com/codhercph

Find event photos, keep current on upcoming events & find stories related to females in tech

By **pseud**
cheatography.com/pseud/

Published 9th March, 2016.
Last updated 11th March, 2016.

Helpful Sites

developer.mozilla.org/en-US/

> Huge site dedicated web developers. The "CSS" & "JavaScript" links under "Web Platform" are especially interesting to you.

learn.jquery.com

> Introduction/Guide to JQuery

api.jquery.com

> The JQuery API - go here to read more about a given JQuery function or to search for functionality.

eloquentjavascript.net

> Probably the best JavaScript textbook in existence - and it's free! An excellent and recommended read.

By **pseud**
cheatography.com/pseud/

Published 9th March, 2016.
Last updated 11th March, 2016.