Cheatography

Javascript Cheat Sheet

by krabat1 (krabat1) via cheatography.com/2778/cs/31328/

Variable types

	var	let	const
block scoped*	✖	✔	✔
tdz**	✖	✔	✔
creates global property*	✔	✖	✖
reassignable**	✔	✔	✖
redeclarable*	✔	✖	✖

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode
* https://medium.com/@allansendagi/block-scope-in-javascript-8fd2f909e848
** https://discuss.codecademy.com/t/what-is-the-difference-between-let-and-var-when-declaring-and-what-about-reassigning/492581/2

https://www.valentinog.com/blog/var/
https://www.freecodecamp.org/news/var-let-and-const-whats-the-difference/
https://fontawesome.com/v3/icons/

Data types

```
var age = 18;                          // number
var name = "Jane";                     // string
var truth = false;                     // boolean
var sheets = ["HTML","-CSS","JS"];     // array
var a; typeof a;                       // undefined
var a = null;                          // value null
```

Objects:

```
var student = {                        // object name
firstName:"Jane",
lastName:"Doe",
age:18,
height:170,
fullName : function() {
return this.firstName + " " + this.lastName;
}
};
```

Data types (cont)

```
// list of properties and values
// object function

// object end
student.age = 19;                      // setting value
student[age]++;                        // incrementing
name = student.full-Name();            // call object function
```

Operators

Comparison Operators

Comparison operators **compare** two values and return a boolean value, either `true` or `false`.

==	Equal to	`5==5; //true`
!=	Not equal to	`5!=5; //false`
>	Greater than	`3>2; //true`
>=	Greater than or equal to	`3>=3; //true`
<	Less than	`3<2; //false`
<=	Less than or equal to	`2<=2; //true`
===	Strict equal to	`5==='5'; //false`
!==	Strict not equal to	`5!=='5'; //true`

Logical Operators

Logical operators perform logical operations and return a boolean value, either `true` or `false`.

&&	Logical AND	`true && false; // false`
||	Logical OR	`true || false; // true`
!	Logical NOT	`!true; // false`

Operators (cont)

? : Ternary operator

Returns value based on the condition

```
(5 > 3) ? 'success' : 'error'; //
condition ? expres sio nIfTrue : e
False
```

Assignment operators

Assignment operators are used to assign valu

=	Assignment operator	`a = 7;`
+=	Addition assignment	`a += 5;`
*=	Multiplication Assignment	`a = 3;`
-=	Subtraction Assignment	`a -= 2;`
/=	Division Assignment	`a /= 2;`
%=	Remainder Assignment	`a %= 2;`
**=	Exponentiation Assignment	`a = 2;`

Arithmetic Operators

Arithmetic operators are used to perform arithr

+	Addition	`x + y`
-	Subtraction	`x - y`
*	Multiplication	`x * y`
/	Division	`x / y`
%	Remainder	`x % y`

By **krabat1** (krabat1)
cheatography.com/krabat1/

Not published yet.
Last updated 5th January, 2024.

Operators (cont)

++	Increment (increments by 1)	`++x or x++`
--	Decrement (decrements by 1)	`--x or x--`
**	Exponentiation (Power)	`x ** y`

Bitwise Operators

Bitwise operators perform operations on binary representations of numbers.

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Left shift
>>	Sign-propagating right shift
>>>	Zero-fill right shift

String Operators

In JavaScript, you can also use the + operator to concatenate (join) two or more strings.

+	Concatenation operator

Other Operators

Operators (cont)

, evaluates each of its operands (from left to right) and returns the value of the last operand.	`let x = 1; x = (x++, x); consol e.l og(x); // 2`
delete deletes an object's property, or an element of an array	`delete x`
typeof returns a string indicating the data type	`typeof 3; // " num be r "`
void discards the expression's return value	`void(x)`
in returns true if the specified property is in the object	`prop in object`

Operators (cont)

instanceof returns true if the specified object is of of the specified object type	`object instanceof object e`

Loopz

https://www.w3schools.com/js/js_loop_for.asp	https://www.programiz.com/javascript/for-loop
https://www.w3schools.com/jsref/jsref_foreach.asp	https://www.programiz.com/javascript/forEach
https://www.w3schools.com/js/js_loop_forof.asp	https://www.programiz.com/javascript/for-of
https://www.w3schools.com/js/js_loop_forin.asp	https://www.programiz.com/javascript/for-in
https://www.w3schools.com/js/js_loop_while.asp	https://www.programiz.com/javascript/while-loop
https://www.w3schools.com/js/js_break.asp	https://www.programiz.com/javascript/break-statement
	https://www.programiz.com/javascript/continue-statement
https://www.w3schools.com/js/js_iterables.asp	
https://www.w3schools.com/js/js_switch.asp	https://www.programiz.com/javascript/switch-statement

Global Methods

Math functions III. - Number Methods

`isFinite()`	a value is a finite number?
`isInte ger()`	a value is an integer?
`isNaN()`	a value is Number.NaN?
`isSafe Int eger()`	a value is a safe integer?
`toExpo nen tial(x)`	Converts a num. into an exponential notation
`toFixed(x)`	Formats a num. with x numb.s of digits after the decimal point
`toLoca leS tring()`	Converts a num. into a string, based on the locale settings
`toPrec isi on(x)`	Formats a num. to x length
`toString()`	Converts a num. to a string
`valueOf()`	Returns the primitive value of a num.

Math functions I.

Math.round(x.y); // x.5 ^ x.4 v	Math.trunc(x.y) // x
Math.ceil(); Math.floor(); // ^ v	Math.pow(x,y) // x^y
Math.sign() // -x, 0, x → -1, 0, 1	Math.abs(-x); // x
Math.sqrt(x); Math.cbrt(y); // √x ∛y	Math.exp(x) // value of Ex
Math.min(a,b,c,d); and Math.max(a,b,c,d);	Math.random();
Math.log(); Math.log2(); Math.log10();	

Math functions II. - Angle functions

`sin(x)`	// sine of x (x is in rad.)
`asin(x)`	// arcsine of x, in rad.
`asinh(x)`	// hyperb. arcsine of x
`sinh(x)`	// hyperb. sine of x
`cos(x)`	// cosine of x (x is in rad.)
`acos(x)`	// arccosine of x, in rad.
`acosh(x)`	// hyperb. arccosine of x
`cosh(x)`	// hyperb. cosine of x
`tan(x)`	// tang. of an angl
`atan(x)`	// arctang. of x as a numeric value btw. -PI/2 and PI/2 rad.
`atan2(y,x)`	// arctang. of the quotient of its arguments
`atanh(x)`	// hyperb. arctang. of x
`tanh(x)`	// hyperb. tang. of a num.

String functions

length	`str.length`

Returns the number of characters in a string

substr-ing()	`str.su bst rin g(i nde xStart, inde xEnd)`

Returns a specified part of the string

slice()	`str.sl ice (be gin Index, endIndex)`

Extracts and returns a section of the string

substr()	`str.su bst r(b egi nIndex, length)`

String functions (cont)

similar to `slice()`, but the second parameter the extracted part.

replace()	`str.re pla ce( pat te )`

replace a substring/pattern in the string

replaceAll()	`str.re pla ceA ll( pa ment)`

Returns string by replacing all matching patter

toUpperCase()	`str.toUpperCase()`
toLowerCase()	`str.toLowerCase()`

Returns uppercase/lowercase representation o

concat()	`str.co nca t(str1, ..`

Concatenates the arguments to the calling stri

~repeat()	`str.re pea t(c ount)`

Returns a string by repeating it given times

trim()	`str.trim()`

Removes whitespace from both ends of a strin

padStart()	`str.pa dSt art (ta rg tring)`
padEnd()	`str.padEnd(targetLeng`

Pads a string at the start/end to a given length

charAt()	`str.ch arA t(i ndex)`

Returns character at a specified index in string

charCodeAt()	`str.ch arC ode At( in`

Returns Unicode of the character at given inde

~fromCharC-ode()	`String.fr omC har Cod mN)`

Returns a string from the given UTF-16 code u

~codePointAt()	`str.co deP oin tAt (p`

Returns the Unicode point value at given index

~fromCodeP-oint()	`String.fr omC ode Poi numN)`

Returns a string using the given code points

String functions (cont)

split() `str.sp lit (se par ator, lmit)`

Returns the string divided into list of substring

- - Search Methods - -

indexOf() `str.in dex Of( sea rch Value, fromIndex)`

lastIndexOf() `str.lastIndexOf(searchValue, fromIndex)`

Returns the first index/last index of occurrence of a value

search() `str.se arc h(r egexp)`

Searches for specified value in the string

match() `str.ma tch (re gexp)`

Returns result of matching string with a regex

matchAll() `str.ma tch All (re gexp)`

Returns iterator of results matching with a regex

includes() `str.in clu des (se arc hSt ring, position)`

Checks if given string is found inside a string

startsWith() `str.st art sWi th( sea rchString, position)`

endsWith() `str.endsWith(searchString, length)`

Checks if a string begins/ends with a specified string

localeCompare() `str.lo cal eCo mpa re( com par eStr, locales, options)`

Compares two strings in the current locale

- - Template Literals - - `${...}`

Template literals provide an easy way to interpolate variables and expressions into strings.

Booleans

`Boole an(e xpr ession)`

The `Boolean()` function is used to convert various data types to boolean values.

You can use the `Boolean()` function to find out if an expression is `true`.

See **Comparison and Logical Operators**.

Array Functions

```
// const array_name = [item1, item2, ...];       // var firste lement =
```

push() and **unshift()**

adds an element at the end/beginning of the array.

pop() and **shift()**

remove the last/first element from an array.

length

```
// var last = array[ a
h-1]
```

returns the number of elements in an array

Arrays is objects...

```
typeof (array) // obje
```

...but arrays too.

```
Array.i sA rra y(a rra
```

Objects

Events

Event listener

Useful links

https://www.w3schools.com/jsref/default.asp

mdn web docs

educba.com - javascript-tutorial

https://www.javascripture.com/

Programiz - Getting Started

Codecademy - Cheatsheets / Learn JavaScript

htmlcheatsheet.com/js

Websitesetup - Javascript-Cheat-Sheet.pdf

javascript.info

Statements Reference

Date Reference

Regular expressions

Class Reference

Error Reference