Javascript CheatSheet Books

Javascript cheatsheet contains useful code syntax with examples which is handy while coding.

▼ **Table of Contents**

Data Types

Javascript is a dynamically typed language and hence though there are data types, variables are not bound to them.

Data Type	Description
number	Represents numbers like integers, floating values, etc.
string	Represents one or more characters
bigint	Represents integers of arbitrary length
null	Represents unknown values
undefined	Represents undefined values
object	Represents complex data structures
boolean	Represents either true or false

Variables

Keyword	Description	Scope
var	var is used to declare variables (old way of declaring variables)	Function or global scope
let	let is also used to declare variables (new way). The value of these variables can change after assignment.	Global or block Scope
const	const is used to declare constant values. The value of these variables cannot be changed after assignment.	Global or block Scope

```
let variable-name; // Just declaration
let variable-name = value; // declaring variable and assigning it with some value
let var1 = value1, var2 = value2, var3 = value3; // multiple variables declaration with thei
```

Escape Sequences in String

Code	Output
`\'`	Single Quote
`\"`	Double Quote
`\\`	Backslash
`\n`	New Line
`\t`	Tab Space
`\r`	Carriage Return
`\b`	Word Boundary
`\f`	Form Feed

Basics

On page script

```
<script type="text/javascript">  ...
</script>
```

To include external javascript file

```
<script src="filename.js"></script>
```

Comments

- Single line comments - //
- Multi line comments - /* */

Output

```
console.log("hello");       // to display message to the browser console
document.write(x);     // to write to HTML
prompt("Your name?","friend");     // input dialog. Second argument is the initial value
alert(x); // writes in an alert box
```

Strict mode

```
"use strict";  // throw errors for some of javascript silent errors and tells browser to be
```

Conditional Statements

If

```
let age = 20;
if (age > 18) {
  console.log('Adult');
}
```

If-else

```
let age=7;
if(age<0) {
console.log('invalid age');
} else if (age<5 && age >0) {
console.log('Infant');
} else if (age>=5 && age<=18) {
  console.log('child');
} else if (age>18) {
  console.log("Adult");
}
```

Switch

```
// what is hello in different languages
let language = "italian";
switch(language){
  case "french" :
    console.log('BONJOUR');
    break;

  case "spanish":
    console.log('Hola');
    break;

  case "hindi" :
    console.log("Namaste");
    break;

  default:
    console.log('Hello');
}
```

Loops

for

```
console.log('simple for loop');
for(let i=1;i<=10;i++) {
  console.log(i);
}
```

for..in

```
let info = {
  name: "foo",
  id: 123
}
for (let x in info) {
  console.log(x); // prints keys of info which are name and id
}
```

for..of

```
let mobiles = [ "iPhone", "Samsung", "OnePlus", "Pixel"];
for(let mbl of mobiles) {
    console.log(mbl);
}
```

while

```
let i=1;
while(i<=10) {
  console.log(i);
  i++;
}
```

do-while

```
let i=1;
do {
  console.log(i);
  i++;
} while(i<=10);
```

Operators

Type	Operators
Arithmetic Operators	+ - * / % ++ –
Comparision Operators	== === != !== > >= < <=

Type	Operators
Bitwise Operators	& ^ | ^ ~ « » »>
Logical Operators	&& || !
Assignment Operators	= += -= *= /= %=
Special Operators	`? :` , `,` , new, typeof, void, yield, delete, in, instanceof

Arrays

```
let arrayName = [value1, value2,..etc];
// or
let arrayName = new Array("value1","value2",..etc);
```

Array Methods

Method name	Syntax
forEach()	arrayname.forEach(function(err, doc){ //code });
map()	arrayName.map(function(err, doc)){ //code });
filter()	arrayName.filter(function(err, doc)) { //code });
reduce()	arrayName.reduce(function(err, doc)) { //code });
find()	arrayName.find(function(err, doc)) { //code });
indexOf()	arrayName.indexOf(element);
from()	Array.from(arrayLike[, mapFn[, thisArg]]);
every()	Array.every(callback(element[, index[, array]])[, thisArg])
some()	Array.some(callback(element[, index[, array]])[, thisArg])
includes()	arrayName.includes(value-to-be-checked[, starting-search-index])

Destructuring Arrays

You can use destructuring to easily store the elements of an array into variables:

```
let arr = [3, 4, 5];
let [a,b,c] = arr;
console.log(a,b,c);
//This will print 3, 4 and 5
let [x,,z] = arr;
console.log(x,z);
//This will only print 3 and 5
//Hence you can skip elements as well
```

You can also make a function return multiple values by making it return an array and immediately destructuring the result:

```
function calc(x, y)
  {
    let sum =x+y;
    let product = x*y;
    return [sum, product];
  }
let [sum, product] = calc(10, 20);
console.log(sum, product);
//This prints 30 and 200
```

Functions

```
// declaring a function
function function-name(parameters){ // here parameters are optional
    //code
}
function-name(parameters); // calling a function
```

Arrow function

```
(argument-list) => expression
```

Example

```
let sum= (a,b,c) => {
    return a+b+c;
}
console.log(sum(10,20,30));
```

Methods

String Methods

Method Name	Usage
search()	let sindex=str.search(“sub-string”);
slice(start,end)	let sub-str=str.slice(starting position,ending position);
substring(start,end	let sub-str=str.substring(starting position,ending position);
substr(start,length)	let sub-str=str.substr(starting position,length);
trim()	let str1= str.trim();
charAt()	let c=str.charAt(position);
charCodeAt()	let c=str.charCodAt(position);
split()	let array=str.split("");
length	let vln = str.length()
indexOf()	let index=str.indexOf(“sub-string”);
lastIndexOf()	let index=str.lastIndexOf(“sub-string”);
toUpperCase()	let str1=str.toUpperCase();
toLowerCase()	let str1=str.toLowerCase();

Method Name	Usage
replace()	let x=str.replace(“string to replace”,“replacement string”)
concat()	let str3=str1.concat(" “,str2);

Number related Methods

Method Name	Usage
Number()	Number(x);
parseInt()	parseInt(x);
parseFloat()	parseFloat(x);
toString()	let x = num.toString();
toFixed()	let x=num.toFixed(no of decimals)
toExponential()	let x=num.toExponential();
toPrecision()	let x=num.toPrecision(length)
valueOf()	let x= num.valueOf();

Date Methods

considering date = new Date() for the below methods.

Method	Usage	comments
getDay()	date.getDay()	returns day as number 0 to 6
Date.now()	let now = Date.now()	returns date and time
getFullYear()	date.getFullYear();	returns yyyy i.e., 2020
setFullYear()	let year = date.setFullYear(2020);	sets year as 2020

Method	Usage	comments
getMonth()	date.getMonth();	returns month as a number (0-11)
setMonth()	let month = date.setMonth(10);	sets month as 10 means november
getDate()	date.getDate();	retuns date as number 1 to 31
setDate()	let day = date.setDate(20);	sets date as 20
getHours()	date.getHours();	to get the hour (0-23)
setHours()	let hrs = date.setHours(20);	to set the hour (0-23)
getMinutes()	date.getMinutes();	to get the minute (0-59)
setMinutes()	let min = date.setMinutes(40);	to set the minutes (0-59)
getSeconds()	date.getSeconds();	to get the second (0-59)
setSeconds()	let sec = date.setSeconds(30);	to set the seconds (0-59)
getMilliseconds()	date.getMilliseconds();	to get the millisecond (0-999)
setMilliseconds()	let milli = date.setMilliseconds(500);	to set the milliseconds (0-999)
setTime()	let dateTime = date.setTime(1582268856705);	to set the time (milliseconds since January 1, 1970)
getTime()	date.getTime()	to get the time (milliseconds since January 1, 1970)

Math functions

Function	Comments
Math.PI;	returns pi value 3.141592653589793
Math.round(10.4);	returns 10

Function	Comments
Math.round(10.5);	returns 5
Math.pow(2,3);	returns 8 which is 2 to the power of 3
Math.sqrt(100);	returns 10
Math.abs(-5.3);	returns 5.3
Math.ceil(22.12);	returns 23 by rounding up
Math.floor(22.92);	returns 22 by rounding down
Math.min(2, 5, -7, 3);	returns the lowest value which is -7
Math.max(2, 5, -7, 3);	returns the highest value which is 5
Math.log(1);	returns log value as 0
Math.random();	returns a random number between 0 and 1
Math.sin(0);	returns 0
Math.cos(Math.PI);	to use tan, atan, asin, acos

Promises

```
let promise = new Promise(function(resolve, reject){
    //code
});
```

Async-Await

Syntax

```
async function functioname(parameters){
        //code
}
```

Example

```
async getTodos(userObj){
        const res = await fetch([url]);
        const data = await res.json()
        return data;
}
let data = await getTodos({fn: "foo"});
```

Error Handling

```
try {
    //code
} catch(err) {
    //code
}
```

throw error

```
throw "Error message";    // throw error text to user
```

Regular expressions

Syntax

```
/pattern/modifiers;
```

Modifiers	Description
g	Performs a global match and finds all
i	Performs case-insensitive matching
m	Performs multiline matching

Javascript DOM

The Javascript DOM (Document Object Model) is an interface that allows developers to manipulate the content, structure and style of a website.The browser creates a representation of the document known as Document Object Model (DOM). This document enables Javascript to access and manipulate the elements and styles of a website.

Some examples

Get element by ID

The getElementById() method is used to get a single element by its id. Let’s look at an example:

```
var title = document.getElementById(‘header-title’);
```

Get elements by class name

We can also get more than one object using the getElementsByClassName() method which returns an array of elements.

```
var items = document.getElementsByClassName(‘list-items’);
```

Get element by tag name

we can also get our elements by tag name using the getElementsByTagName() method.

```
var listItems = document.getElementsByTagName(‘li’);
```

Queryselector

The querySelector() method returns the first element that matches a specified CSS selector. That means that you can get elements by id, class, tag and all other valid CSS selectors. Here I just list a few of the most popular options.

```
var items = document.querySelector(‘list-items’);
var header = document.querySelector(‘#header’)
var listItems = document.querySelector(‘li’);
```

Queryselectorall

The querySelectorAll() method is completely the same as the querySelector() except that it returns all elements that fit the CSS Selector.

```
var heading = document.querySelectorAll(‘h1.heading’);
```