Published in CodeX

Kesk -*- Follow

Oct 23 · 7 min read · Member-only · Listen

# 30 Super Useful JavaScript One-Liners

Save time in your day-to-day work and have fun with them

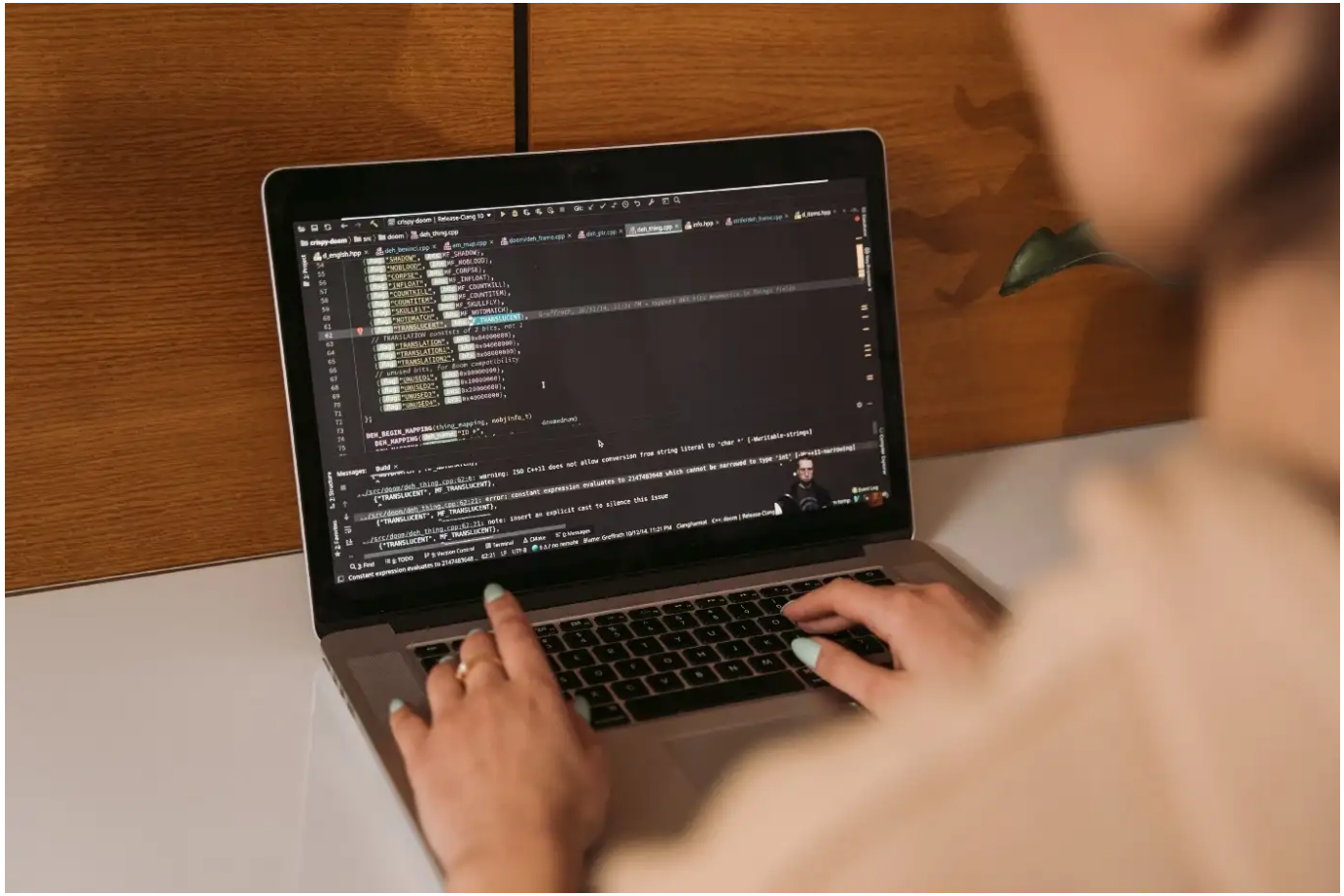


Photo by MART PRODUCTION from Pexels: https://www.pexels.com/photo/close-up-shot-of-a-person-using-a-laptop-7172094/

In this post, I list a series of 30 one-liners in JavaScript that are very useful when developing using vanilla js ( ≥ ES6). They are also an elegant way to solve problems using all the power the language offers us in the latest versions.

I have grouped them into the following categories:

- Date
- Strings
- Numbers
- Arrays
- Utils

Without further ado, I'll get to it. I hope you find them helpful!

## Dates

### 1. Knowing if a date corresponds to the current date

As simple as converting the two dates to the same format and comparing them.

<date> is a Date instance.

```
const isCurrentDay = (date) =>  new Date().toISOString().slice(0, 10) === date.toISOString().slice(0, 10);
```

isCurrentDay function.

```
1  const isCurrentDay = (date) =>  new Date().toISOString().slice(0, 10) === date.toISOStrir
```

examples is hosted with ❤️ by GitHub view raw

isCurrentDay function.

```
> const isCurrentDay = (date) =>  new Date().toISOString().slice(0, 10) === date.toISOString().slice(0, 10);
< undefined
> isCurrentDay(new Date());
< true
> isCurrentDay(new Date('2022-10-20'));
< false
```

isCurrentDay example

## 2. How to know if a date is between two dates

We check if the past date is within the min-max range.

<min>, <max>, and <date> are Date instances.

```
const isBetweenTwoDates = ( min, max, date) => date.getTime() >= min.getTime() && date.getTime() <= max.getTime();
```

isBetweenTwoDates function.

```
1  const isBetweenTwoDates = ( min, max, date) => date.getTime() >= min.getTime() && date.ge
```

examples is hosted with ❤️ by GitHub isBetweenTwoDates function. view raw

```
> const isBetweenTwoDates = ( min, max, date) => date.getTime() >= min.getTime() && date.getTime() <= max.getTime();
< undefined
> const min = new Date('2022-10-01');
< undefined
> const max = new Date('2022-10-15');
< undefined
> const date1 = new Date('2022-10-10');
< undefined
> isBetweenTwoDates(min, max, date1);
< true
> const date2 = new Date('2022-10-20');
< undefined
> isBetweenTwoDates(min, max, date2);
< false
```

isBetweenTwoDates example.

### 3. How to know if a date falls on a weekend

The getDay method returns a number between 0 and 6, representing the day of the week for the given date.

<date> is a Date instance.

```
const isWeekend = ( date ) => date.getDay() === 6 || date.getDay() === 0;
```

isWeekend function.

```
1  const isWeekend = ( date ) => date.getDay() === 6 || date.getDay() === 0;
```

isWeekend.js hosted with ❤️ by GitHub view raw

isWeekend function.

```
> const isWeekend = ( date ) => date.getDay() === 6 || date.getDay() === 0;
< undefined
> const saturday = new Date('2022-10-22');
< undefined
> isWeekend(saturday)
< true
> const monday = new Date('2022-10-24');
< undefined
> isWeekend(monday);
< false
```

isWeekend example.

## 4. Check if a date is in a year

Similar to when we used to check if a date corresponded to the current day. In this case, we get the year and compare it.

<date> and <year> are two Date instances.

```
const isInAYear = (date, year) => date.getUTCFullYear() === new Date(`${year}`).getUTCFullYear();
```

isInAYear function.

1 const isInAYear = (date, year) => date.getUTCFullYear() === new Date(`${year}`).getUTCFul

examples is hosted with ❤ by GitHub view raw

isInAYear function.

```
> const isInAYear = (date, year) => date.getUTCFullYear() === new Date(`${year}`).getUTCFullYear();
< undefined
> isInAYear(new Date(), '2022')
< true
> isInAYear(new Date(), '2021')
< false
```

isInAYear example.

## 5. Convert an hour to AM-PM format

We can use mathematical expressions to determine whether the elapsed time is less than 13 hours or more and thus determine whether it is “AM” or “PM.”

```
const toAMPMFormat= (h) => `${h % 12 === 0 ? 12 : h % 12}${h < 12 ? ' am.' : ' pm.'}`;
```

toAMPMFormat function.


```
1  const toAMPMFormat= (h) => `${h % 12 === 0 ? 12 : h % 12}${h < 12 ? ' am.' : ' pm.'}`;
```


examples.js hosted with ❤️ by GitHub view raw

toAMPMFormat function.

```
> const toAMPMFormat= (h) => `${h % 12 === 0 ? 12 : h % 12}${h < 12 ? ' am.' : ' pm.'}`;
< undefined
> toAMPMFormat(12);
< '12 pm.'
> toAMPMFormat(13);
< '1 pm.'
> toAMPMFormat(23);
< '11 pm.'
> toAMPMFormat(8);
< '8 am.'
```

toAMPMFormat example.

# Strings

## 6. Capitalize the first letter of a sentence

We convert the first letter to capital letters and then append the rest of the letters of the sentence using <join.>

```
const capitalize = ([first, ...rest]) => `${first.toUpperCase()}${rest.join('')}`;
```

capitalize function.

```
1  const capitalize = ([first, ...rest]) => `${first.toUpperCase()}${rest.join('')}`;
```

examples.js hosted with ❤️ by GitHub view raw

capitalize function.

```
> const capitalize = ([first, ...rest]) => `${first.toUpperCase()}${rest.join('')}`;
<- undefined
> capitalize('hello');
<- 'Hello'
> capitalize('hello world');
<- 'Hello world'
>
```

capitalize example.

## 7. Convert a letter to his associate emoji

```
const letterToEmoji = c => String.fromCodePoint(c.toLowerCase().charCodeAt() + 127365);
```

letterToEmaji function.

```
1  const letterToEmoji = c => String.fromCodePoint(c.toLowerCase().charCodeAt() + 127365);
```

examples.js hosted with ❤️ by GitHub view raw

letterToEmaji function.

```
> const letterToEmoji = c => String.fromCodePoint(c.toLowerCase().charCodeAt() + 127365);
<- undefined
> letterToEmoji('a');
<- 'A'
> letterToEmoji('x');
<- 'x'
```

letterToEmaji example.

## 8. How to know if a string is a palindrome

```
const isPalindrome = (str) => str.toLowerCase() === str.toLowerCase().split('').reverse().join('');
```

isPalindrome function.

```
1  const isPalindrome = (str) => str.toLowerCase() === str.toLowerCase().split('').reverse()
```

examples is hosted with ❤ by GitHub view raw

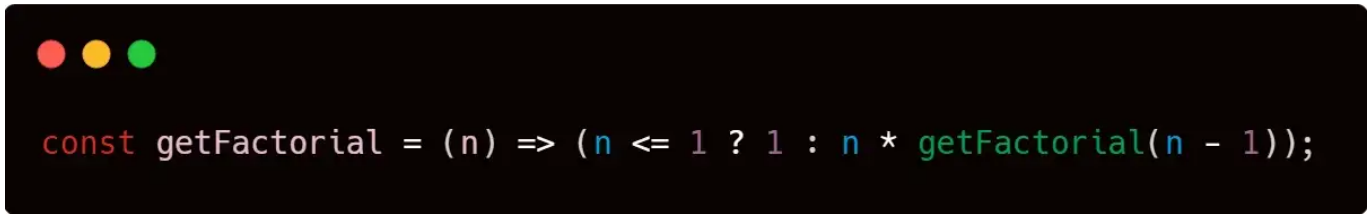
isPalindrome function.

```
> const isPalindrome = (str) => str.toLowerCase() === str.toLowerCase().split('').reverse().join('');
<- undefined
> isPalindrome('Rotator')
<- true
> isPalindrome('Hello')
<- false
```

isPalindrome example.

## Numbers

### 9. How to calculate the factorial of a number




getFactorial function.

```
1  const getFactorial = (n) => (n <= 1 ? 1 : n * getFactorial(n - 1));
```

examples.js hosted with ❤️ by GitHub view raw

```
> const getFactorial = (n) => (n <= 1 ? 1 : n * getFactorial(n - 1));
<- undefined
> getFactorial(1);
<- 1
> getFactorial(2);
<- 2
> getFactorial(3);
<- 6
> getFactorial(4);
<- 24
```

getFactorial example.

## 10. How to obtain the Fibonacci of a number

```
const getFibonacci = (n, memo = {}) => memo[n] || (n <= 2 ? 1 : (memo[n] = getFibonacci(n - 1, memo) +
getFibonacci(n - 2, memo)));
```

getFibonnaci function.

```
1  const getFibonacci = (n, memo = {}) => memo[n] || (n <= 2 ? 1 : (memo[n] = getFibonacci(r
```

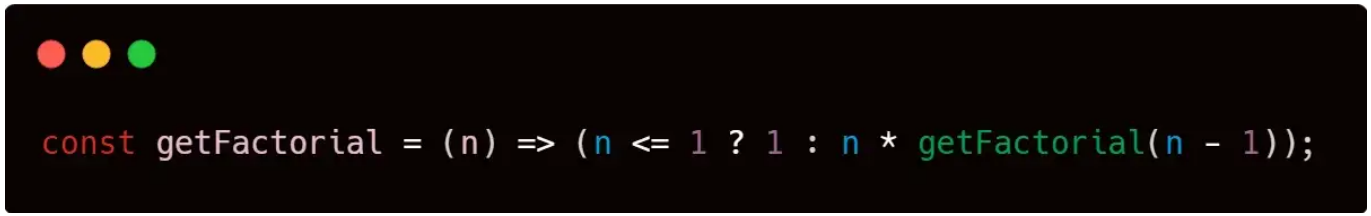
examples.js hosted with ❤ by GitHub view raw

getFibonnaci function.

```
> const getFibonacci = (n, memo = {}) => memo[n] || (n <= 2 ? 1 : (memo[n] = getFibonacci(n - 1, memo) +
  getFibonacci(n - 2, memo)));
<- undefined
> getFibonacci(1);
<- 1
> getFibonacci(2);
<- 1
> getFibonacci(3);
<- 2
> getFibonacci(4);
<- 3
> getFibonacci(10);
<- 55
```

getFibonnaci example.

## 11. How to obtain the factorial of a number






```
1  const getFactorial = (n) => (n <= 1 ? 1 : n * getFactorial(n - 1));
```

examples.js hosted with ❤ by GitHub view raw

getFibonnaci function.

```
> const getFactorial = (n) => (n <= 1 ? 1 : n * getFactorial(n - 1));
< undefined
> getFactorial(1);
< 1
> getFactorial(10);
< 3628800
> getFactorial(100);
< 9.33262154439441e+157
```

getFibonnaci example.

## Arrays

### 12. Copy an array to another array

copyToArray function.

```
1  const copyToArray = (arr) => [...arr];
```

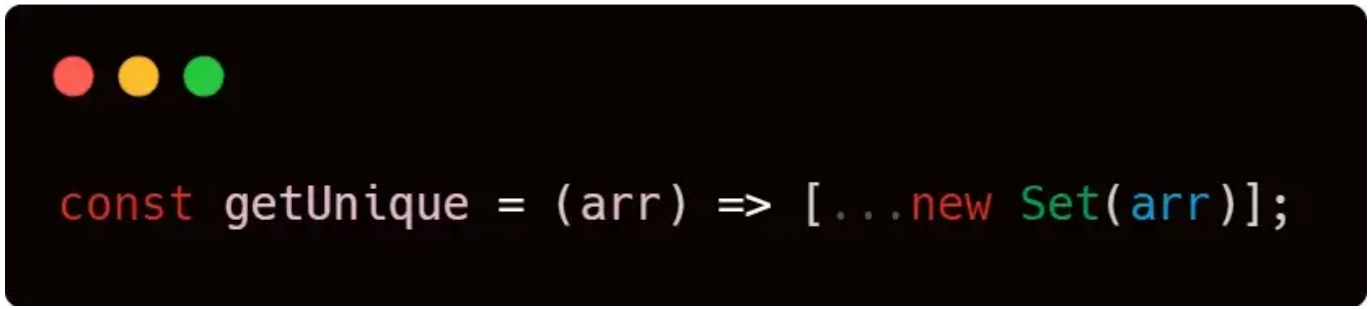
examples.js hosted with ❤️ by GitHub view raw

copyToArray function.

```
> const copyToArray = (arr) => [...arr];
<- undefined
> const array1 = [1,2,3,4];
<- undefined
> const array2 = copyToArray(array1);
<- undefined
> array2;
<- ▶ (4) [1, 2, 3, 4]
> array1 === array2
<- false
```

copyToArray example.

## 13. Get the unique elements from an Array




getUnique function.

getUnique function.

```
> const getUnique = (arr) => [...new Set(arr)];

  const arr = [1, 1, 1,2,2,3,4,4,4];
  getUnique(arr);
<· ▶ (4) [1, 2, 3, 4]
```

getUnique example.

## 14. Shuffle Array

The following snippet shuffles an array in a very efficient way.

```
const shuffle = (arr) => arr.sort(() => Math.random() - 0.5);
```

shuffle function.

shuffle function.

```
> const shuffle = (arr) => arr.sort(() => Math.random() - 0.5);
< undefined
> const arr = [1, 2, 3, 4, 5, 6, 7, 8];
< undefined
> shuffle(arr);
< ▶ (8) [6, 4, 1, 3, 8, 2, 7, 5]
```

shuffle example.

## 15. Group an Array by a property

```
const groupBy = (arr, groupFn) =>   arr.reduce( (grouped, obj) => ({...grouped, [groupFn(obj)]: [...
(grouped[groupFn(obj)] || []), obj], }),{});
```

groupBy function.

groupBy function.

```
const cars = [
  { name: 'Ford' },
  { name: 'Bmw' },
  { name: 'Audi' },
  { name: 'Mercedes' },
];
undefined
const groupedByCarNameLength = groupBy(cars, (car) => car.name.length);
undefined
groupedByCarNameLength
▼{3: Array(1), 4: Array(2), 8: Array(1)}
  ▼3: Array(1)
    ▶0: {name: 'Bmw'}
      length: 1
    ▶[[Prototype]]: Array(0)
  ▼4: Array(2)
    ▶0: {name: 'Ford'}
    ▶1: {name: 'Audi'}
      length: 2
    ▶[[Prototype]]: Array(0)
  ▼8: Array(1)
    ▶0: {name: 'Mercedes'}
      length: 1
    ▶[[Prototype]]: Array(0)
  ▶[[Prototype]]: Object
```

groupBy example.

## 16. Reverse a String

We can utilize the built-in Array methods like reverse() and join() to create a one-liner that does the same thing.


reverseString function.

reverseString function.

```
> const reverse = (str) => str.split('').reverse().join('');
< undefined
> reverse("Hello!")
< '!olleH'
```

reverseString example.

### 17. Check if two arrays contain the same values

We can use Array.sort() and Array.join() methods to check if two arrays contain the same values.

```
const containSameValues= (arr1, arr2) =>   arr1.sort().join(',') === arr2.sort().join(',');
```

containgSameValues function.

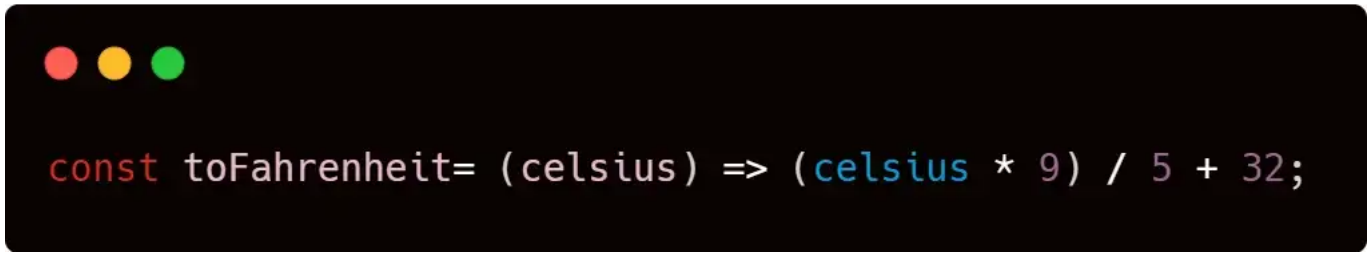
containgSameValues function.

```
> const containSameValues = (arr1, arr2) =>
    arr1.sort().join(',') === arr2.sort().join(',');
< undefined
> const array1 = [1, 2, 3, 4,5];
< undefined
> const array2 = [1, 2, 5, 3, 4];
< undefined
> const array3 = [1, 2 ,3, 6, 7]
< undefined
> containSameValues(array1, array2);
< true
> containSameValues(array1, array3);
< false
```

containgSameValues example.

## Utils

### 18. Convert to Fahrenheit



```
const toFahrenheit= (celsius) => (celsius * 9) / 5 + 32;
```

toFahrenheit function.

toFahrenheit function.

```
> const toFahrenheit= (celsius) => (celsius * 9) / 5 + 32;
< undefined
> toFahrenheit(25);
< 77
```

toFahrenheit example.

## 19 . Convert to Celsius

```
const toCelsius=  (fahrenheit) => (fahrenheit- 32) * 5 / 9;
```

ToCelsius function.

ToCelsius function.

```
> const toFahrenheit= (celsius) => (celsius * 9) / 5 + 32;
<· undefined
> toFahrenheit(25);
<· 77
> const toCelsius=  (fahrenheit) => (fahrenheit- 32) * 5 / 9
<· undefined
> toCelsius(77);
<· 25
```

ToCelsius example.

## 20. How to clear all cookies from the browser

```
const clearAllCookies = () => document.cookie.split(';').forEach((c) => (document.cookie = c.replace(/^ +/,
'').replace(/=.*/, `=;expires=${new Date().toUTCString()};path=/`)));
```

clearAllCookies function.

clearAllCookies function.

## 21. How to convert from HEX to RGB

```
const toRGB= (hex) =>
    hex
        .replace(/^#?([a-f\d])([a-f\d])([a-f\d])$/i, (_, r, g, b) => `#${r}${r}${g}${g}${b}${b}`)
        .substring(1)
        .match(/.{2}/g)
        .map((x) => parseInt(x, 16));
```

toRGB function.

toRGB function.

```
> const toRGB= (hex) =>
    hex
      .replace(/^#?([a-f\d])([a-f\d])([a-f\d])$/i, (_, r, g, b) => `#${r}${r}${g}${g}${b}${b}`)
      .substring(1)
      .match(/.{2}/g)
      .map((x) => parseInt(x, 16));
<- undefined
> toRGB('#ffffff');
<- ▶ (3) [255, 255, 255]
```

toRGB example.

## 22. How to convert from RGB to HEX

```
const toHEX = (r,g,b) => "#" + ((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1);
```

toHEX function.

toHEX function.

```
> const toHEX = (r,g,b) => "#" + ((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1);
< undefined
> toHEX(255,255,255);
< '#ffffff'
```

toHEX example.

## 23. Check if a function is an Async function

```
const isAsyncFunction = (f) => Object.prototype.toString.call(f) === '[object AsyncFunction]';
```

isAsyncFunction function.

isAsyncFunction function.

```
> const isAsyncFunction = (f) => Object.prototype.toString.call(f) === '[object AsyncFunction]';
< undefined
> isAsyncFunction( () => {} )
< false
> isAsyncFunction( async () => {});
< true
> isAsyncFunction(function* () {});
< false
```

isAsyncFunction example.

## 24. How to know if a code is running in the browser

```
const runningInBrowser = typeof window === 'object' && typeof document === 'object';
```

runningInBrowser function.

runningInBrowser function.

```
> const runningInBrowser = typeof window === 'object' && typeof document === 'object';
  console.log(runningInBrowser);
  true
```

## 25. How to know if a code is running in node

```
const runningInNode= typeof process !== 'undefined' && process.versions != null && process.versions.node != null;
```

runningInNode function.

runningInNode function.

```
> const runningInNode= typeof process !== 'undefined' && process.versions != null && process.versions.node !=
  null;
  console.log(runningInNode);
  false                                                                                    VM3150:2
```

runningInNode example.

## 26. Detect Dark Mode

This a very handy way to check if the user has dark mode enabled on their browser.

```
const isDarkMode = () =>  window.matchMedia && window.matchMedia("(prefers-color-scheme: dark)").matches;
```

isDarkMode function.

isDarkMode function.

```
> const isDarkMode = () =>  window.matchMedia && window.matchMedia("(prefers-color-scheme: dark)").matches;

< undefined
> isDarkMode()
< false
```

isDarkMode example.

## 27. Scroll to the Top

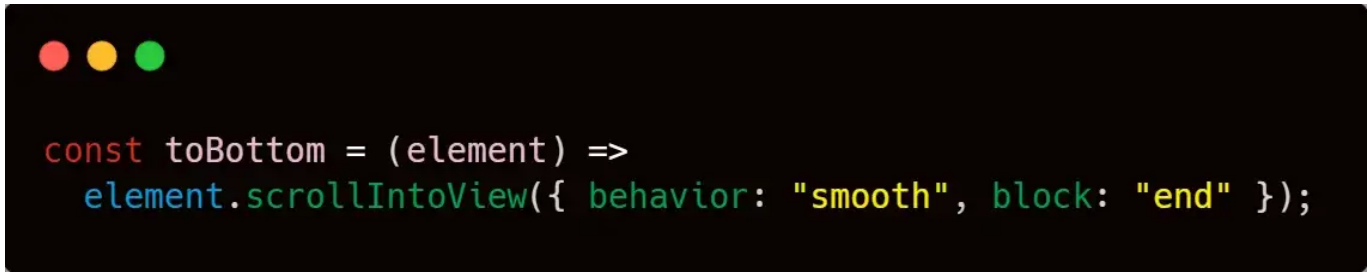
A one-line way to scroll elements is to use the <scrollIntoView> method.

```
const toTop = (element) =>
  element.scrollIntoView({ behavior: "smooth", block: "start" });
```

toTop function.

toTop function.

## 28. Scroll to Bottom



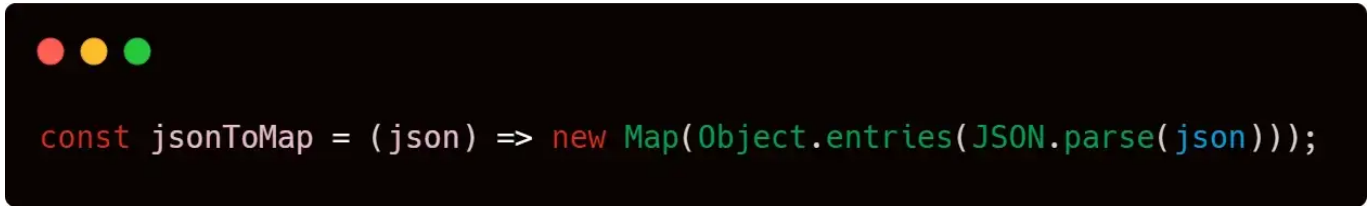
```
const toBottom = (element) =>
  element.scrollIntoView({ behavior: "smooth", block: "end" });
```

toBottom function.

toBottom function.

## 29. Convert a JSON to a Map

This function lets us convert a Map object to a JSON string in a simple way.




jsonToMap function.

jsonToMap function.

```
> const jsonToMap = (json) => new Map(Object.entries(JSON.parse(json)));
< undefined
> const json = '{"name":"Kate","age":"35","city":"NY"}';
  const map = jsonToMap(json);
< undefined
> map
< ▼Map(3) {'name' => 'Kate', 'age' => '35', 'city' => 'NY'}
    ▼[[Entries]]
      ▶0: {"name" => "Kate"}
      ▶1: {"age" => "35"}
      ▶2: {"city" => "NY"}
       size: 3
    ▶[[Prototype]]: Map
```

jsonToMap example

## 30. Generate a 128-bit UUID

This function allows us to generate a UUID with a 128-bit value for uniquely identifying an object or entity.

```
const generateUUID = (a) => a ? (a ^ ((Math.random() * 16) >> (a / 4))).toString(16) : ([1e7] + -1e3 + -4e3 + -8e3 +
-1e11).replace( /[018]/g, generateUUID);
```

generateUUID function.

generateUUID function.

```
> generateUUID()
<- 'e7428e5d-6a81-4412-9476-e058cad9d550'
> generateUUID()
<- 'eabdc1c6-b6ab-4263-8966-a845b86f0524'
```

generateUUID example.

## Final Throughs

If you have reached this point, you probably already have your own opinion about one-liners. For me, one-liners (JavaScript, Linux,...) are a compact and elegant way to solve problems, but you also have to be careful that the solution is not too difficult to read.

### Other useful one-liners

**The Art of Using the Command Line**

20 curated super bash One-liners commands for your day to day

medium.com

**18 Selected Super-Useful Linux One-Liners**

Functional, Practical, and some also extremely Dangerous

medium.com

https://betterprogramming.pub/25-awesome-linux-command-one-liners-9495f26f07fb

Java Script Javascript Tips Javascript Development One Liners

Web Development