75 Useful JavaScript Code Snippets for Your Next Project

SWIPE

Sajan Kota

Generate a random string

We can use **Math.random()** to generate a random alphanumeric string, it's very convenient when we need a unique ID.

```
1   const randomString = () => Math.random().toString(36).slice(2)
2
```

SWIPE

Generate a random string of a given length

```
const randomString = (length = 10) => {
    let result = '';
    while (result.length < length) {
        result += Math.random().toString(36).slice(2);
    }
    return result.slice(0, length);
}
```

SWIPE

Copy content to the clipboard

```
1  const copyToClipboard = (text) => navigator.clipboard.writeText(text);
2  copyToClipboard("Hello World");
3
4
```

SWIPE

Sajan Kota

Clear all cookies

```
const clearCookies = document.cookie.split(';').forEach(cookie => document.cookie = cookie.replace(/^ +/, '').replace(/=.*/, `=;expires=${new Date(0).toUTCString()};path=/`));
```

SWIPE

Get the selected text

```
1  const getSelectedText = () => window.getSelection().toString();
2  getSelectedText();
```

SWIPE

Scroll to the top of the page

```
1   const goToTop = () => window.scrollTo(0, 0);
2   goToTop();
```

SWIPE

Check whether the user has scrolled to the bottom of a page

```
1  const scrolledToBottom = () => document.documentElement.clientHeight + window.scrollY >= document.documentElement.scrollHeight;
2
```

SWIPE

Find out whether the current tab is active

```
1   const isTabInView = () => !document.hidden;
```

SWIPE

Sajan Kota

Redirect the user to a URL

```
const redirect = url => location.href = url
redirect("https://www.google.com/")
```

SWIPE

Open the browser print box

```
1   const showPrintDialog = () => window.print()
```

SWIPE

Generate a random boolean value

```
// Returns a random boolean value (true or false)

const randomBoolean = () => Math.random() >= 0.5; randomBoolean();
```

SWIPE

Generate a random number between two values

```
/// Returns a random number between the given minimum and maximum values.

const randomNumber = Math.random() * (max - min) + min

```

SWIPE

Sajan Kota

Check if a given number is an integer

```
// Determines if a number is an integer.


const isInteger = (num) => num % 1 === 0
```

SWIPE

Sajan Kota

Remove duplicate values in an array

```
// Removes duplicate values from an array.

const removeDuplicates = (arr) => [...new Set(arr)]

console.log(removeDuplicates([1, 2, 2, 3, 3, 4, 4, 5, 5, 6])) // [1, 2, 3, 4, 5, 6]
```

SWIPE

Get the unique values in an array

```
// Get the unique values in an array

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

console.log(uniqueArr([1, 2, 3, 1, 2, 3, 4, 5]));
// [1, 2, 3, 4, 5]
```

SWIPE

Check if a variable is an array

```
// Check if a variable is an array

const isArray = (arr) => Array.isArray(arr);

console.log(isArray([1, 2, 3]));
// true
console.log(isArray({ name: 'Ovi' }));
// false
console.log(isArray('Hello World'));
// false
```

SWIPE

Check if the date is Weekend

```
// Check if the date is Weekend.

const isWeekend = (date) => [0, 6].indexOf(date.getDay()) !== -1;

console.log(isWeekend(new Date(2021, 4, 14)));
// false (Friday)

console.log(isWeekend(new Date(2021, 4, 15)));
// true (Saturday)
```

SWIPE

Calculate number of days between two dates

```
1  // Calculate number of days between two dates
2
3  const daysDiff = (date, date2) => Math.ceil(Math.abs(date - date2) / 86400000);
4
5  console.log(daysDiff(new Date('2021-05-10'), new Date('2021-11-25')));
6  // 199
```

SWIPE

Sajan Kota

Capitalize a String

```
// Capitalizing a string


const capitalize = (str) => str.charAt(0).toUpperCase() + str.slice(1);

console.log(capitalize('hello world'));
// Hello world
```

SWIPE

Get the day of the year from a date

```
// Get the day of the year from a date

const dayOfYear = (date) => Math.floor((date - new Date(date.getFullYear(), 0, 0)) / (1000 * 60 * 60 * 24))

dayOfYear(new Date()) // 74
```

SWIPE

Check if a string is a palindrome

```
1  // Check if a string is a palindrome
2
3  const isPalindrome = str => str === str.split('').reverse().join('');
4
5  console.log(isPalindrome('racecar')); // true
6  console.log(isPalindrome('hello')); // false
```

SWIPE

Sajan Kota

Get the first n elements of an array

```
// Get the first n elements of an array

const take = (arr, n) => arr.slice(0, n);

console.log(take([1, 2, 3, 4, 5], 3)); // [1, 2, 3]
```

SWIPE

Get the last n elements of an array

```
// Get the last n elements of an array

const takeRight = (arr, n) => arr.slice(-n);

console.log(takeRight([1, 2, 3, 4, 5], 3)); // [3, 4, 5]
```

SWIPE

Remove all vowels from a string

```
// Remove all vowels from a string

const removeVowels = (str) => str.replace(/[aeiou]/gi, '')

removeVowels('hello world') // 'hll wrld'
```

SWIPE

Check if a string contains a substring

```
// Check if a string contains a substring

const contains = (str, substr) => str.includes(substr);

console.log(contains('hello world', 'world')); // true
console.log(contains('hello', 'world')); // false

```

SWIPE

Get the current time in hh:mm:ss format

```
// Get the current time in hh:mm:ss format

const getTime = () => new Date().toLocaleTimeString();

console.log(getTime()); // "9:30:42 PM" (output will vary based on current time)
```

SWIPE

Check if an object is empty

```
// Check if an object is empty

const isEmpty = obj => Object.keys(obj).length === 0;

console.log(isEmpty({})); // true
console.log(isEmpty({ name: 'John' })); // false
```

SWIPE

Checks if the provided array is not empty

```
// Checks if the provided array is not empty.

const isArrayNotEmpty = (arr) => {
    // Check if 'arr' is an array and has more than 0 elements.
    return Array.isArray(arr) && arr.length > 0;
};

// Examples for demonstration.
console.log(isArrayNotEmpty([])); // Should print false
console.log(isArrayNotEmpty([1, 2, 3])); // Should print true
```

SWIPE

Find the max value in an array

```
// Find the max value in an array

Math.max(...array)
```

SWIPE

Get the current date and time

```
1   // Get the Current date and time
2   new Date().toString();
3
```

SWIPE

Reverse a String

```
// Reverse a string

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

reverse('hello world');
// Result: 'dlrow olleh'
```

SWIPE

Sajan Kota

Sort Arrays

```
1  //Ascending
2  const sortAscending = (array) => array.sort((a, b) => a - b);
3
4  //Descending
5  const sortDescending = (array) => array.sort((a, b) => b - a);
```

SWIPE

Extract the Domain name from an email

```
// Extract Domain name from an email

const email = "selemondev19@gmail.com";
const extractDomain = (email) => email.split('@')[1];
console.log(extractDomain(email)); // "gmail.com"
```

SWIPE

Flatten an Nested Array

```
// Flatten an nested array

const flat = (arr) => arr.flat(1);
flat(['cat', ['lion', 'tiger']]); // ['cat', 'lion', 'tiger']
```

SWIPE

Sajan Kota

Generates a Random Color in Hexadecimal Format

```
// Generates a Random Color in Hexadecimal Format

const randomColor = () => `#${Math.random().toString(16).slice(2, 8).padEnd(6, '0')}`

randomColor(); // #9dae4f
randomColor(); // #6ef10e

```

Checks if a given value is a valid Hexadecimal Color Code

```
// Checks if a given value is a valid Hexadecimal Color Code

const isHexColor = (hex) => {
    const regex = /#?([0-9A-Fa-f]{6}|[0-9A-Fa-f]{3})/;
    return regex.test(hex);
}
```

SWIPE

Sajan Kota

Get the current time in a specific timezone

```
// Get the current time in a specific timezone

const currentTimeInTimezone = (timezone) => new Date().toLocaleTimeString('en-US', { timeZone: timezone })
currentTimeInTimezone('Europe/London') // "8:28:00 AM" (for the current time in London)
```

SWIPE

Convert a String to kebab-case

```
// Convert a string to kebab-case

const toKebabCase = (str) => str.toLowerCase().replace(/\s+/g, '-')

toKebabCase('Hello World') // 'hello-world'
toKebabCase('This is a Test') // 'this-is-a-test'
```

SWIPE

Shuffle an Array

```
// Shuffle an array

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

console.log(shuffle([1, 2, 3, 4, 5])); // [2, 5, 1, 4, 3] (output will vary)
```

SWIPE

Shuffle an array using the Fisher-Yates (Knuth) Shuffle algorithm

```javascript
// Shuffle an array using Fisher-Yates (Knuth) Shuffle algorithm

const fisherYatesShuffle = (arr) => {
    for (let i = arr.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1)); // Generate a random index
      [arr[i], arr[j]] = [arr[j], arr[i]]; // Swap elements at i and j
    }
    return arr;
  };

  // Example usage
  const originalArray = [1, 2, 3, 4, 5];
  const shuffledArray = fisherYatesShuffle([...originalArray]);
  console.log(shuffledArray); // Output will be a shuffled version of the original array
```

SWIPE

Sajan Kota

Convert RGB color code to valid Hexadecimal color code

```
// Convert RGB colour Code to valid Hexadecimal Color Code
const rgbToHex = (r, g, b) => "#" + ((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1)

rgbToHex(255, 255, 255)  // '#ffffff'

```

Truncate a number to a fixed decimal point

```
// Truncate a number to a fixed decimal point

const round = (n, d) => Number(Math.round(n + "e" + d) + "e-" + d)

round(1.005, 2) //1.01
round(1.555, 2) //1.55
```

SWIPE

Sajan Kota

Remove falsy values from Array

```
// Remove falsy values from array

const removeFalsy = (arr) => arr.filter(Boolean);

removeFalsy([0, 'a string', '', NaN, true, 5, undefined, 'another string', false]);
// ['a string', true, 5, 'another string']
```

SWIPE

A function that toggles a boolean value

```
// function that toggles a boolean value

const toggleBool = () => (bool = !bool);
```

SWIPE

Sajan Kota

Swapping Two Variables

A concise way to swap the values of two variables using array

```
// Swapping Two Variables

[foo, bar] = [bar, foo];

// a concise way to swap the values of two variables using array destructuring.
```

SWIPE

Sajan Kota

Merging multiple arrays using Concatenation

The concat() method returns a new array containing elements from both arr1 and arr2.

```
// Merging multiple arrays using Concatenation (concat() method):

// The concat() method returns a new array containing elements from both arr1 and arr2.

const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const mergedArray = arr1.concat(arr2);

// mergedArray: [1, 2, 3, 4, 5, 6]
```

SWIPE

Merging multiple arrays using Spread Operator

The spread operator is concise and visually clear. It takes the elements from both arrays and creates a new array.

```
// Merging multiple arrays using Spread Operator(...):

// The spread operator is concise and visually clear. It takes the elements from both arrays and creates a new array.

const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const mergedArray = [...arr1, ...arr2];

// mergedArray: [1, 2, 3, 4, 5, 6]
```

SWIPE

Sajan Kota

Merging multiple arrays using push() and apply()

This approach modifies arr1 in place by using push() along with apply() to add elements from arr2.

```
// Merging multiple arrays using push() and apply() (for merging arrays with variable lengths):

// This approach modifies arr1 in place by using push() along with apply() to add elements from arr2.

const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
arr1.push.apply(arr1, arr2);

// arr1: [1, 2, 3, 4, 5, 6]
```

SWIPE

Sajan Kota

Merging multiple arrays using push and Spread Operator

The spread operator is used to expand the elements of arr2 into individual arguments for the push() method.

```
// Merging multiple arrays using push() and Spread Operator:

// The spread operator is used to expand the elements of arr2 into individual arguments for the push() method.

const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
arr1.push(...arr2);

// arr1: [1, 2, 3, 4, 5, 6]
```

SWIPE

Sajan Kota

Merging multiple arrays using concat() with Spread Operator

The spread operator is used to expand the elements of arr2 as arguments to the concat() method.

```
// Merging multiple arrays using concat() with Spread Operator:

// The spread operator is used to expand the elements of arr2 as arguments to the concat() method.

const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const mergedArray = arr1.concat(...arr2);

// mergedArray: [1, 2, 3, 4, 5, 6]
```

Get the actual type of JavaScript primitives

```
// Get the actual type of javascript primitives
const trueTypeOf = (obj) => {
    return Object.prototype.toString.call(obj).match(/\s([a-zA-Z]+)/)[1].toLowerCase();
};

console.log(trueTypeOf(''));
// string
console.log(trueTypeOf(0));
// number
console.log(trueTypeOf());
// undefined
console.log(trueTypeOf(null));
// null
console.log(trueTypeOf({}));
// object
console.log(trueTypeOf([]));
// array
console.log(trueTypeOf(0));
// number
console.log(trueTypeOf(() => {}));
// function

```

SWIPE

Sajan Kota

Truncate string at the end

The function will truncate the input string to the specified length and add an ellipsis (...) at the end if the string is longer than the specified length.

```
// Truncate string at the end

const truncateString = (string, length) => {
    return string.length < length ? string : `${string.slice(0, length - 3)}...`;
};

console.log(
    truncateString('Hi, I should be truncated because I am too loooong!', 36),
);

// Hi, I should be truncated because...
```

SWIPE

Truncate string from the middle

The function is to truncate the input string while keeping the specified number of characters from the start and end, and adding an ellipsis (...) in the middle.

```
// Truncate string at the end

const truncateString = (string, length) => {
    return string.length < length ? string : `${string.slice(0, length - 3)}...`;
};

console.log(
    truncateString('Hi, I should be truncated because I am too loooong!', 36),
);

// Hi, I should be truncated because...
```

SWIPE

Sajan Kota

Get the value of a browser cookie by its name


```
// Get the value of a browser cookie by its name.

const getCookieValue = (name) => {

    const cookies = `; ${document.cookie}`;
    const cookieArray = cookies.split(`; ${name}=`);
    const cookieValue = cookieArray.pop().split(';').shift();
    return cookieValue;

  };

// Usage
    const gaValue = getCookieValue('_ga');
    console.log(gaValue);

// Result could be something like: "GA1.2.1929736587.1601974046"
```

SWIPE

Get the time (HH:MM:SS) from a JavaScript Date object.

```
// Get the time (HH:MM:SS) from a JavaScript Date object.

const getTimeFromDate = (date) => {
    return date.toTimeString().slice(0, 8);
  };

  // Usage example:
  const exampleDate = new Date(2021, 0, 10, 17, 30, 0);  // Jan 10, 2021, 17:30:00
  const timeString = getTimeFromDate(exampleDate);
  console.log(timeString);  // Output should be: "17:30:00"
```

SWIPE

Sajan Kota

Check if a number is even or odd

```
// Check if a number is even or odd
const isEven = num => num % 2 === 0;

console.log(isEven(2));
```

SWIPE

Converts the first character of a given string to lowercase

```
// Converts the first character of a given string to lowercase.

const lowercaseFirst = (str) => {
    const firstCharLowercase = str.charAt(0).toLowerCase();
    const restOfString = str.slice(1);
    return `${firstCharLowercase}${restOfString}`;
};

// Example usage
console.log(lowercaseFirst('Hello World')); // Outputs 'hello World'
```

SWIPE

Sajan Kota

Repeats a given string a specific number of times

```
// Repeats a given string a specific number of times.

const repeat = (str, numberOfTimes) => {
    return str.repeat(numberOfTimes);
};
```

SWIPE

Check if the code is running in Node.js

```
1  // Check if the code is running in Node.js
2  const isNode = typeof process !== 'undefined' && process.versions !== null && process.versions.node !== null;
```

Check if the code is running in the browser

```
1  // Check if the code is running in the browser
2  const isBrowser = typeof window === 'object' && typeof document === 'object';
```

SWIPE

Get all siblings of a given element

```
// Get all siblings of a given element.

const siblings = ele => Array.from(ele.parentNode.children).filter(child => child !== ele);
```

SWIPE

Go back to the previous page using the history object

```
// Go back to the previous page using the history object.

history.back();
// Or
history.go(-1);
```

SWIPE

Function to get the largest element in an array

```
// Function to get the largest element in an array
const getLargest = (arr) => {
return arr.reduce((largest, num) => Math.max(largest, num), -Infinity);
};

// Define an array for testing
const arr = [13, 7, 11, 3, 9, 15, 17];

// Output the smallest element in the array
console.log(getSmallest(arr));
 // Output should be 3
```

SWIPE

Function to get the smallest element in an array

```
// Function to get the smallest element in an array
const getSmallest = (arr) => {
    return arr.reduce((smallest, num) => Math.min(smallest, num), Infinity);
};

// Define an array for testing
const arr = [13, 7, 11, 3, 9, 15, 17];

// largest elements in the array
console.log(getLargest(arr));
// Output should be 17
```

SWIPE

Function to toggle the visibility of an HTML element

```
// Function to toggle the visibility of an HTML element.

const toggle = element => {
    // Validate that the argument is an HTMLElement
    if (!(element instanceof HTMLElement)) {
        throw new Error("The argument must be an HTML element.");
    }

    // Check the current display property of the element
    // Set it to "block" if it is "none" or to "none" otherwise
    element.style.display = (element.style.display === "none" ? "block" : "none");
};

```

SWIPE

Sajan Kota

Function to remove HTML tags from a given String

```
// Function to remove HTML tags from a given string.

const stripHtml = html => {
    // Validate the input to be a string
    if (typeof html !== 'string') {
      throw new Error("The argument must be a string containing HTML.");
    }

    // Create a new DOMParser instance
    // Parse the HTML string using the DOMParser into a DOM object
    // Navigate to the body of the DOM object
    // Extract and return the text content (stripping the HTML tags)
    return (new DOMParser().parseFromString(html, 'text/html')).body.textContent || '';
};
```

SWIPE

Define a function to check if a given value is a DOM Element

```
// Define a function to check if a given value is a DOM Element.
const isDOMElement = (value) => {
    // Check if value is an object to begin with
    if (typeof value !== 'object') {
      return false;
    }

    // Check if nodeType property is equal to 1 (indicating an Element node)
    if (value.nodeType !== 1) {
      return false;
    }

    // Check if the style property exists and is an object
    if (typeof value.style !== 'object') {
      return false;
    }

    // Check if ownerDocument property exists and is an object
    if (typeof value.ownerDocument !== 'object') {
      return false;
    }

    // If all conditions are met, return true
    return true;
```

SWIPE

Sajan Kota

Define a function to check if a given string is a valid URL

```
// Define a function to check if a given string is a valid URL
const isURL = (url) => {
    // Define a regex pattern for a URL
    // This pattern checks for the protocol (http or https), the domain, and then any optional parameters, paths, or queries
    const regex = /^(https?:\/\/)?[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!\$&'\(\)\*\+,;=.]+$/;

    // Use the test method to check if the URL matches the regex pattern
    return regex.test(url);
};

// Test the function
console.log(isURL("http://example.com"));  // Output: true
console.log(isURL("invalid-string"));     // Output: false
```

SWIPE

Sajan Kota

Check if a string starts with a given prefix

```
1  // Check if a string starts with a given prefix.
2  const startsWith = (str, prefix) => str.startsWith(prefix)
```

Check if a string ends with a given suffix

```
1  // Check if a string ends with a given suffix.
2  const endsWith = (str, suffix) => str.endsWith(suffix)
```

SWIPE

Check if a value is an object

```
1  // Check if a value is an object.
2  // Returns true if the value is an object, otherwise returns false.
3  const isObject = (obj) => obj === Object(obj);
4
```

SWIPE

Check if a value is a function

```
// Check if a value is a function.
// Returns true if the value is a function, otherwise returns false
const isFunction = (fn) => typeof fn === 'function';
```

SWIPE

Check if a value is a Promise

```
1  // Check if a value is a Promise.
2  // Returns true if the value is a Promise object, otherwise returns false.
3  const isPromise = (promise) => promise instanceof Promise;
```

Check if an HTML element is entirely within the viewport

```
// Check if an HTML element is entirely within the viewport.

function isInViewport(element) {
  // Destructure the bounding rectangle object to get individual properties
  const { top, left, bottom, right } = element.getBoundingClientRect();

  // Cache viewport dimensions for better performance
  const viewportHeight = window.innerHeight || document.documentElement.clientHeight;
  const viewportWidth = window.innerWidth || document.documentElement.clientWidth;

  // Check if the element is entirely within the viewport
  return (
      top >= 0 &&  // Top edge is inside the viewport
      left >= 0 &&  // Left edge is inside the viewport
      bottom <= viewportHeight &&  // Bottom edge is inside the viewport
      right <= viewportWidth  // Right edge is inside the viewport
  );
}
```

Which of these JavaScript code snippets do you use in your projects?

Sajan Kota