

75 Useful JavaScript Code Snippets for Your Next Project

SWIPE



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



Sajan Kota

Generate a random string of a given length



```
1 const randomString = (length = 10) => {
2     let result = '';
3     while (result.length < length) {
4         result += Math.random().toString(36).slice(2);
5     }
6     return result.slice(0, length);
7 }
8
```

SWIPE



Sajan Kota

Copy content to the clipboard



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

SWIPE



Clear all cookies



```
1 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



Sajan Kota

Scroll to the top of the page



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

SWIPE



Sajan Kota

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



Sajan Kota

Find out whether the current tab is active



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

SWIPE



Sajan Kota

Redirect the user to a URL



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

SWIPE



Open the browser print box



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

SWIPE



Generate a random boolean value



```
1 // Returns a random boolean value (true or false)  
2  
3 const randomBoolean = () => Math.random() >= 0.5; randomBoolean();
```

SWIPE



Sajan Kota

Generate a random number between two values



```
1 // Returns a random number between the given minimum and maximum values.  
2  
3 const randomNumber = Math.random() * (max - min) + min  
4  
5
```

SWIPE



Sajan Kota

Check if a given number is an integer



```
1 // Determines if a number is an integer.  
2  
3  
4 const isInteger = (num) => num % 1 === 0
```

SWIPE



Remove duplicate values in an array



```
1 // Removes duplicate values from an array.  
2  
3 const removeDuplicates = (arr) => [...new Set(arr)]  
4  
5 console.log(removeDuplicates([1, 2, 2, 3, 3, 4, 4, 5, 5, 6])) // [1, 2, 3, 4, 5, 6]
```

SWIPE



Sajan Kota

Get the unique values in an array



```
1 // Get the unique values in an array
2
3 const uniqueArr = (arr) => [...new Set(arr)];
4
5 console.log(uniqueArr([1, 2, 3, 1, 2, 3, 4, 5]));
6 // [1, 2, 3, 4, 5]
```

SWIPE



Sajan Kota

Check if a variable is an array



```
1 // Check if a variable is an array
2
3 const isArray = (arr) => Array.isArray(arr);
4
5 console.log(isArray([1, 2, 3]));
6 // true
7 console.log(isArray({ name: 'Ovi' }));
8 // false
9 console.log(isArray('Hello World'));
10 // false
```

SWIPE



Sajan Kota

Check if the date is Weekend



```
1 // Check if the date is Weekend.  
2  
3 const isWeekend = (date) => [0, 6].indexOf(date.getDay()) !== -1;  
4  
5 console.log(isWeekend(new Date(2021, 4, 14)));  
6 // false (Friday)  
7  
8 console.log(isWeekend(new Date(2021, 4, 15)));  
9 // true (Saturday)
```

SWIPE



Sajan Kota

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 // 139
```

SWIPE



Capitalize a String



```
1 // Capitalizing a string
2
3
4 const capitalize = (str) => str.charAt(0).toUpperCase() + str.slice(1);
5
6 console.log(capitalize('hello world'));
7 // Hello world
```

SWIPE



Get the day of the year from a date



```
1 // Get the day of the year from a date
2
3 const dayOfYear = (date) => Math.floor((date - new Date(date.getFullYear(), 0, 0)) / (1000 * 60 * 60 * 24))
4
5 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



Get the first n elements of an array



```
1 // Get the first n elements of an array
2
3 const take = (arr, n) => arr.slice(0, n);
4
5 console.log(take([1, 2, 3, 4, 5], 3)); // [1, 2, 3]
```

SWIPE



Get the last n elements of an array



```
1 // Get the last n elements of an array
2
3 const takeRight = (arr, n) => arr.slice(-n);
4
5 console.log(takeRight([1, 2, 3, 4, 5], 3)); // [3, 4, 5]
```

SWIPE



Remove all vowels from a string



```
1 // Remove all vowels from a string
2
3 const removeVowels = (str) => str.replace(/[aeiou]/gi, '')
4
5 removeVowels('hello world') // 'hll wrld'
```

SWIPE



Check if a string contains a substring



```
1 // Check if a string contains a substring
2
3 const contains = (str, substr) => str.includes(substr);
4
5 console.log(contains('hello world', 'world')) // true
6 console.log(contains('hello', 'world')) // false
7
```

SWIPE



Sajan Kota

Get the current time in hh:mm:ss format



```
1 // Get the current time in hh:mm:ss format
2
3 const getTime = () => new Date().toLocaleTimeString();
4
5 console.log(getTime()); // "9:30:42 PM" (output will vary based on current time)
```

SWIPE



Sajan Kota

Check if an object is empty



```
1 // Check if an object is empty
2
3 const isEmpty = obj => Object.keys(obj).length === 0;
4
5 console.log(isEmpty({})); // true
6 console.log(isEmpty({ name: 'John' })); // false
```

SWIPE



Checks if the provided array is not empty



```
1 // Checks if the provided array is not empty.  
2  
3 const isArrayNotEmpty = (arr) => {  
4     // Check if 'arr' is an array and has more than 0 elements.  
5     return Array.isArray(arr) && arr.length > 0;  
6 };  
7  
8 // Examples for demonstration.  
9 console.log(isArrayNotEmpty([])); // Should print false  
10 console.log(isArrayNotEmpty([1, 2, 3])); // Should print true
```

SWIPE



Find the max value in an array



```
1 // Find the max value in an array  
2  
3 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



```
1 // Reverse a string
2
3 const reverse = str => str.split('').reverse().join('');
4
5 reverse('hello world');
6 // 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



```
1 // Extract Domain name from an email  
2  
3 const email = "selemondev19@gmail.com";  
4 const extractDomain = (email) => email.split('@')[1];  
5 console.log(extractDomain(email)); // "gmail.com"
```

SWIPE



Sajan Kota

Flatten an Nested Array



```
1 // Flatten an nested array
2
3 const flat = (arr) => arr.flat(1);
4 flat(['cat', ['lion', 'tiger']])); // ['cat', 'lion', 'tiger']
5
```

SWIPE



Generates a Random Color in Hexadecimal Format



```
1 // Generates a Random Color in Hexadecimal Format
2
3 const randomColor = () => `#${Math.random().toString(16).slice(2, 8).padEnd(6, '0')}`
4
5 randomColor(); // #9dae4f
6 randomColor(); // #6ef10e
7
```

SWIPE



Sajan Kota

Checks if a given value is a valid Hexadecimal Color Code



```
1 // Checks if a given value is a valid Hexadecimal Color Code  
2  
3 const isHexColor = (hex) => {  
4     const regex = /#?([0-9A-Fa-f]{6}|[0-9A-Fa-f]{3})/;  
5     return regex.test(hex);  
6 }
```

SWIPE



Get the current time in a specific timezone



```
1 // Get the current time in a specific timezone
2
3 const currentTimeInTimezone = (timezone) => new Date().toLocaleTimeString('en-US', { timeZone: timezone })
4 currentTimeInTimezone('Europe/London') // "8:28:00 AM" (for the current time in London)
```

SWIPE



Sajan Kota

Convert a String to kebab-case



```
1 // Convert a string to kebab-case
2
3 const toKebabCase = (str) => str.toLowerCase().replace(/\s+/g, '-')
4
5 toKebabCase('Hello World') // 'hello-world'
6 toKebabCase('This is a Test') // 'this-is-a-test'
```

SWIPE



Sajan Kota

Shuffle an Array



```
1 // Shuffle an array
2
3 const shuffle = arr => arr.sort(() => Math.random() - 0.5);
4
5 console.log(shuffle([1, 2, 3, 4, 5])); // [2, 5, 1, 4, 3] (output will vary)
```

SWIPE



Sajan Kota

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



```
1 // Shuffle an array using Fisher-Yates (Knuth) Shuffle algorithm
2
3 const fisherYatesShuffle = (arr) => {
4     for (let i = arr.length - 1; i > 0; i--) {
5         const j = Math.floor(Math.random() * (i + 1)); // Generate a random index
6         [arr[i], arr[j]] = [arr[j], arr[i]]; // Swap elements at i and j
7     }
8     return arr;
9 }
10
11 // Example usage
12 const originalArray = [1, 2, 3, 4, 5];
13 const shuffledArray = fisherYatesShuffle([...originalArray]);
14 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



```
1 // Convert RGB colour Code to valid Hexadecimal Color Code
2 const rgbToHex = (r, g, b) => "#" + ((1 << 24) + (r << 16) + (g << 8) + b).toString(16).slice(1)
3
4 rgbToHex(255, 255, 255) // '#ffffff'
5
```

SWIPE



Sajan Kota

Truncate a number to a fixed decimal point



```
1 // Truncate a number to a fixed decimal point
2
3 const round = (n, d) => Number(Math.round(n + "e" + d) + "e-" + d)
4
5 round(1.005, 2) //1.01
6 round(1.555, 2) //1.55
7
```

SWIPE



Remove falsy values from Array



```
1 // Remove falsy values from array
2
3 const removeFalsy = (arr) => arr.filter(Boolean);
4
5 removeFalsy([0, 'a string', '', NaN, true, 5, undefined, 'another string', false]);
6 // ['a string', true, 5, 'another string']
```

SWIPE



A function that toggles a boolean value



```
1 // function that toggles a boolean value  
2  
3 const toggleBool = () => (bool = !bool);
```

SWIPE



Swapping Two Variables

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



```
1 // Swapping Two Variables
2
3 [foo, bar] = [bar, foo];
4
5 // 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.



```
1 // Merging multiple arrays using Concatenation (concat() method):  
2  
3 // The concat() method returns a new array containing elements from both arr1 and arr2.  
4  
5 const arr1 = [1, 2, 3];  
6 const arr2 = [4, 5, 6];  
7 const mergedArray = arr1.concat(arr2);  
8  
9 // mergedArray: [1, 2, 3, 4, 5, 6]
```

SWIPE



Sajan Kota

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.



```
1 // Merging multiple arrays using Spread Operator(...):
2
3 // The spread operator is concise and visually clear. It takes the elements from both arrays and creates a new array.
4
5 const arr1 = [1, 2, 3];
6 const arr2 = [4, 5, 6];
7 const mergedArray = [...arr1, ...arr2];
8
9 // 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.



```
1 // Merging multiple arrays using push() and apply() (for merging arrays with variable lengths):
2
3 // This approach modifies arr1 in place by using push() along with apply() to add elements from arr2.
4
5 const arr1 = [1, 2, 3];
6 const arr2 = [4, 5, 6];
7 arr1.push.apply(arr1, arr2);
8
9 // 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.



```
1 // Merging multiple arrays using push() and Spread Operator:  
2  
3 // The spread operator is used to expand the elements of arr2 into individual arguments for the push() method.  
4  
5 const arr1 = [1, 2, 3];  
6 const arr2 = [4, 5, 6];  
7 arr1.push(...arr2);  
8  
9 // 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.



```
1 // Merging multiple arrays using concat() with Spread Operator:  
2  
3 // The spread operator is used to expand the elements of arr2 as arguments to the concat() method.  
4  
5 const arr1 = [1, 2, 3];  
6 const arr2 = [4, 5, 6];  
7 const mergedArray = arr1.concat(...arr2);  
8  
9 // mergedArray: [1, 2, 3, 4, 5, 6]
```

SWIPE



Sajan Kota

Get the actual type of JavaScript primitives



```
1 // Get the actual type of javascript primitives
2 const trueTypeOf = (obj) => {
3     return Object.prototype.toString.call(obj).match(/\s([a-zA-Z]+)\)/)[1].toLowerCase();
4 };
5
6 console.log(trueTypeOf(''));
7 // string
8 console.log(trueTypeOf(0));
9 // number
10 console.log(trueTypeOf());
11 // undefined
12 console.log(trueTypeOf(null));
13 // null
14 console.log(trueTypeOf({}));
15 // object
16 console.log(trueTypeOf([]));
17 // array
18 console.log(trueTypeOf(0));
19 // number
20 console.log(trueTypeOf(() => {}));
21 // function
22
```

SWIPE



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.



```
1 // Truncate string at the end
2
3 const truncateString = (string, length) => {
4     return string.length < length ? string : `${string.slice(0, length - 3)}...`;
5 }
6
7 console.log(
8     truncateString('Hi, I should be truncated because I am too loooong!', 36),
9 );
10
11 // Hi, I should be truncated because...
```

SWIPE



Sajan Kota

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.



```
1 // Truncate string at the end
2
3 const truncateString = (string, length) => {
4     return string.length < length ? string : `${string.slice(0, length - 3)}...`;
5 }
6
7 console.log(
8     truncateString('Hi, I should be truncated because I am too loooong!', 36),
9 );
10
11 // Hi, I should be truncated because...
```

SWIPE



Sajan Kota

Get the value of a browser cookie by its name



```
1 // Get the value of a browser cookie by its name.  
2  
3 const getCookieValue = (name) => {  
4  
5     const cookies = `; ${document.cookie}`;  
6     const cookieArray = cookies.split(`; ${name}=`);  
7     const cookieValue = cookieArray.pop().split(';').shift();  
8     return cookieValue;  
9  
10 };  
11  
12 // Usage  
13 const gaValue = getCookieValue('_ga');  
14 console.log(gaValue);  
15  
16 // Result could be something like: "GA1.2.1929736587.1601974046"  
17
```

SWIPE



Sajan Kota

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



```
1 // Get the time (HH:MM:SS) from a JavaScript Date object.  
2  
3 const getTimeFromDate = (date) => {  
4     return date.toTimeString().slice(0, 8);  
5 };  
6  
7 // Usage example:  
8 const exampleDate = new Date(2021, 0, 10, 17, 30, 0); // Jan 10, 2021, 17:30:00  
9 const timeString = getTimeFromDate(exampleDate);  
10 console.log(timeString); // Output should be: "17:30:00"
```

SWIPE



Sajan Kota

Check if a number is even or odd



```
1 // Check if a number is even or odd
2 const isEven = num => num % 2 === 0;
3
4 console.log(isEven(2));
```

SWIPE

Sajan Kota

Converts the first character of a given string to lowercase



```
1 // Converts the first character of a given string to lowercase.  
2  
3 const lowercaseFirst = (str) => {  
4     const firstCharLowercase = str.charAt(0).toLowerCase();  
5     const restOfString = str.slice(1);  
6     return `${firstCharLowercase}${restOfString}`;  
7 };  
8  
9 // Example usage  
10 console.log(lowercaseFirst('Hello World')) // Outputs 'hello World'
```

SWIPE



Sajan Kota

Repeats a given string a specific number of times



```
1 // Repeats a given string a specific number of times.  
2  
3 const repeat = (str, numberoftimes) => {  
4     return str.repeat(numberoftimes);  
5 };
```

SWIPE

Sajan Kota

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 === true;
```

SWIPE

Sajan Kota

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

Sajan Kota

Get all siblings of a given element



```
1 // Get all siblings of a given element.  
2  
3 const siblings = ele => Array.from(ele.parentNode.children).filter(child => child !== ele);
```

SWIPE

Sajan Kota

Go back to the previous page using the history object



```
1 // Go back to the previous page using the history object.  
2  
3 history.back();  
4 // Or  
5 history.go(-1);
```

SWIPE



Sajan Kota

Function to get the largest element in an array



```
1 // Function to get the largest element in an array
2 const getLargest = (arr) => {
3   return arr.reduce((largest, num) => Math.max(largest, num), -Infinity);
4 };
5
6 // Define an array for testing
7 const arr = [13, 7, 11, 3, 9, 15, 17];
8
9 // Output the smallest element in the array
10 console.log(getSmallest(arr));
11 // Output should be 3
```

SWIPE



Sajan Kota

Function to get the smallest element in an array



```
1 // Function to get the smallest element in an array
2 const getSmallest = (arr) => {
3     return arr.reduce((smallest, num) => Math.min(smallest, num), Infinity);
4 };
5
6 // Define an array for testing
7 const arr = [13, 7, 11, 3, 9, 15, 17];
8
9 // largest elements in the array
10 console.log(getLargest(arr));
11 // Output should be 17
```

SWIPE



Sajan Kota

Function to toggle the visibility of an HTML element



```
1 // Function to toggle the visibility of an HTML element.  
2  
3 const toggle = element => {  
4     // Validate that the argument is an HTMLElement  
5     if (!(element instanceof HTMLElement)) {  
6         throw new Error("The argument must be an HTML element.");  
7     }  
8  
9     // Check the current display property of the element  
10    // Set it to "block" if it is "none" or to "none" otherwise  
11    element.style.display = (element.style.display === "none" ? "block" : "none");  
12 };  
13
```

SWIPE



Sajan Kota

Function to remove HTML tags from a given String



```
1 // Function to remove HTML tags from a given string.  
2  
3 const stripHtml = html => {  
4     // Validate the input to be a string  
5     if (typeof html !== 'string') {  
6         throw new Error("The argument must be a string containing HTML.");  
7     }  
8  
9     // Create a new DOMParser instance  
10    // Parse the HTML string using the DOMParser into a DOM object  
11    // Navigate to the body of the DOM object  
12    // Extract and return the text content (stripping the HTML tags)  
13    return (new DOMParser().parseFromString(html, 'text/html')).body.textContent || '';  
14};
```

SWIPE



Sajan Kota

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

```
1 // Define a function to check if a given value is a DOM Element.  
2 const isDOMElement = (value) => {  
3     // Check if value is an object to begin with  
4     if (typeof value !== 'object') {  
5         return false;  
6     }  
7  
8     // Check if nodeType property is equal to 1 (indicating an Element node)  
9     if (value.nodeType !== 1) {  
10        return false;  
11    }  
12  
13    // Check if the style property exists and is an object  
14    if (typeof value.style !== 'object') {  
15        return false;  
16    }  
17  
18    // Check if ownerDocument property exists and is an object  
19    if (typeof value.ownerDocument !== 'object') {  
20        return false;  
21    }  
22  
23    // If all conditions are met, return true  
24    return true;
```

SWIPE

Sajan Kota

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



```
1 // Define a function to check if a given string is a valid URL
2 const isURL = (url) => {
3     // Define a regex pattern for a URL
4     // This pattern checks for the protocol (http or https), the domain, and then any optional parameters, paths, or queries
5     const regex = /^(https?:\/\/)?[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!\$&'\\()]*\+,\;=.]+$/;
6
7     // Use the test method to check if the URL matches the regex pattern
8     return regex.test(url);
9 };
10
11 // Test the function
12 console.log(isURL("http://example.com")); // Output: true
13 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)
```

SWIPE

Sajan Kota

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

Sajan Kota

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



Sajan Kota

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



Sajan Kota

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;
```

SWIPE



Sajan Kota

Check if an HTML element is entirely within the viewport



```
1 // Check if an HTML element is entirely within the viewport.  
2  
3 function isInViewport(element) {  
4     // Destructure the bounding rectangle object to get individual properties  
5     const { top, left, bottom, right } = element.getBoundingClientRect();  
6  
7     // Cache viewport dimensions for better performance  
8     const viewportHeight = window.innerHeight || document.documentElement.clientHeight;  
9     const viewportWidth = window.innerWidth || document.documentElement.clientWidth;  
10  
11    // Check if the element is entirely within the viewport  
12    return (  
13        top >= 0 && // Top edge is inside the viewport  
14        left >= 0 && // Left edge is inside the viewport  
15        bottom <= viewportHeight && // Bottom edge is inside the viewport  
16        right <= viewportWidth // Right edge is inside the viewport  
17    );  
18}
```

SWIPE



Sajan Kota

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