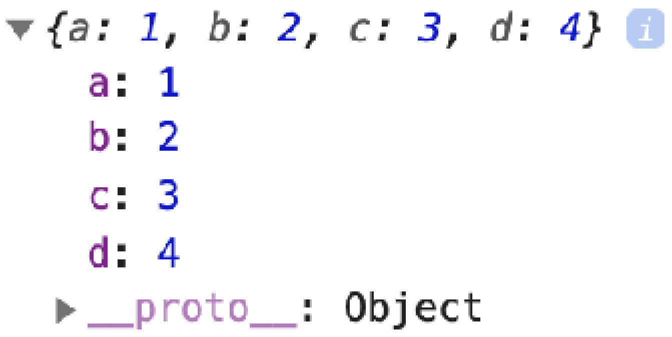
Some simple and amazing JavaScript tricks

JS

Some simple and amazing

JavaScript tricks



```
▼{a: 1, b: 2, c: 3, d: 4}
   a: 1
   b: 2
   c: 3
   d: 4
 ▶__proto__: Object
```

Trick 1: Combining Multiple Objects Consider you have three different objects: const obj1 = {'a': 1, 'b': 2}; const obj2 = {'c': 3};

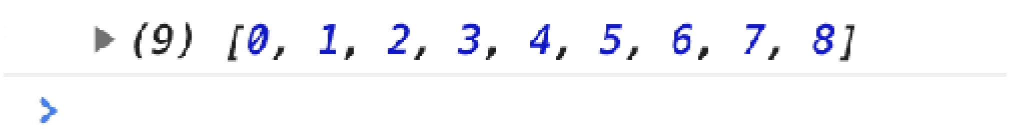
const obj3 = {'d': 4};

If we want an object to contain the combined properties of all these objects, we can do so with the simple code below: const objCombined = {...obj1, ...obj2, ...obj3}; Logging objCombined will print the following value in the console:

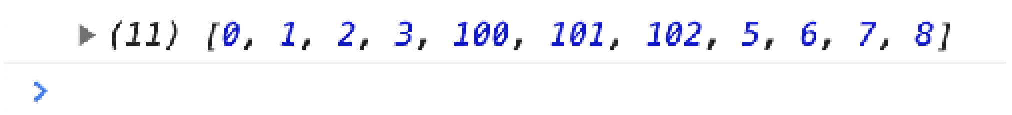
The object objCombined is a newly created object. Updating any value of obj1, obj2, or obj3 will not affect the values of objCombined.

For nested objects, the references of the inner objects will be copied and will not create new ones. The spread syntax will copy all the properties of the object, but will only create a new object at the top level.

You can also combined objects using the Object.assign() method.



```
▶ (9) [0, 1, 2, 3, 4, 5, 6, 7, 8]
>
```



```
▶ (11) [0, 1, 2, 3, 100, 101, 102, 5, 6, 7, 8]
>
```

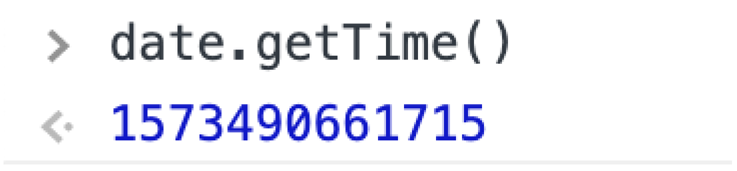
Trick 2: Inserting values in between an array

Consider you have the following array of integers: const arr = [0, 1, 2, 3, 5, 6, 7, 8]; What if we want to insert the integer 4 at index 4 of the array?

We can simply do so by using the splice function in Array's prototype. The syntax of the splice function is: arr.splice(index, itemsToDelete, item1ToAdd, item2ToAdd, ...) To insert the integer 4 at index 4, we write the code: arr.splice(4, 0, 4);

This will update the array to: To insert multiple integers at an index, we can also write: arr.splice(4, 0, 100, 101, 102); This will update the original array to: Using splice will mutate the original array, not create a new one.

Want to learn more about splice *function? Read more about it in the following article:*



```
> date.getTime()
<· 1573490661715
```

Trick 3: Get the current timestamp To get the current timestamp in JavaScript, you simply need to execute the following code:

var date = new Date();

date.getTime()

This will give the timestamp for the date.

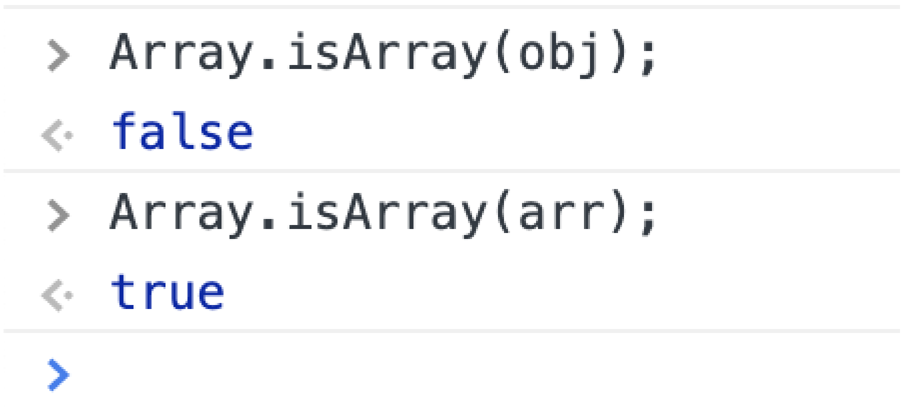
There is also a shortcut to get the current timestamp.

+new Date()

Or

Date.now()

```
> Array.isArray(obj);
<· false
> Array.isArray(arr);
<· true
>
```



```
> Array.isArray(obj);
<· false
> Array.isArray(arr);
<· true
>
```

Trick 4: Check if an object is an array To check if an object is an array, you can call the isArray() method of the Array object.

const obj = {data: 1};

const arr = [1, 2, 3];

To check for an array, use the following code snippet:
Array.isArray(obj); // false Array.isArray(arr); // true

Trick 5: Object destructuring Consider you have the following object in JavaScript: const user = {

name: 'Kunal',

age: 25,

profile: 'https://medium.com/@kunaltandon.kt',

followers: 1000,

blogs: 57

};

We can directly get the variables for the object's properties using the following syntax:

const { name, profile, blogs, followers } =

user;console.log(name); // Kunal console.log(profile); //

https://medium.com/@kunaltandon.kt

console.log(blogs); // 57

console.log(followers); // 1000

After running the above statement, we will now have 4 different variables containing the object's properties.

For this to work, the variable names on the left-hand side must match the key names of the object exactly.

The syntax is called **Object Destructuring**.

Trick 6: Rest parameter syntax Did you know that you can create a function that accepts any number of arguments?

There is a special syntax called the **rest parameter syntax** to create such a function.

```
function sum(...values) {

console.log(values);

}sum(1);

sum(1, 2);

sum(1, 2, 3);

sum(1, 2, 3, 4);
```

Calling the sum functions will collect the values as an array of the parameters passed to the function.

This will print the following output.

```
[1]

[1, 2]

[1, 2, 3]

[1, 2, 3, 4]
```

We can also complete the sum function and make it calculate the sum of all the parameters passed to the function.

```
function sum(...values) {

let sum = 0;

for (let i = 0; i < values.length; i++) {

sum += values[i];

}

return sum;

}console.log(sum(1));

console.log(sum(1, 2));

console.log(sum(1, 2, 3));

console.log(sum(1, 2, 3, 4));
```

This will print the output as:

```
1

3

6

10
```