4 Ways to Convert String to Character Array in JavaScript

Here are 4 ways to split a word into an array of characters. "Split" is the most common and more robust way. But with the addition of ES6, there are more tools in the JS arsenal to play with

I always like to see all the possible ways to solve something because then you can choose the best way for your use case. Also, when you see it pop up in someone's codebase, you will understand it with ease 👍

```
const string = 'word';

// Option 1
string.split('');

// Option 2
[...string];

// Option 3
Array.from(string);

// Option 4
Object.assign([], string);

// Result:
// ['w', 'o', 'r', 'd']
```


Scenarios

Instead of going through the pros and cons of each different way. Let me show you the different scenarios where one is preferred over the other.

Array of Characters

If all you're doing is wanting separate the string by each string character, all of the ways are good and will give you the same result

```
const string = 'hi there';

const usingSplit = string.split('');
const usingSpread = [...string];
const usingArrayFrom = Array.from(string);
const usingObjectAssign = Object.assign([], string);

// Result
// [ 'h', 'i', ' ', 't', 'h', 'e', 'r', 'e' ]
```

Specific Separators

If you want to split your string by a specific character, then `split` is the way to go.

```
const string = 'split-by-dash';

const usingSplit = string.split('-');
// [ 'split', 'by', 'dash' ]
```

The other ways are limited by each string character only

```
const string = 'split-by-dash';

const usingSpread = [...string];
const usingArrayFrom = Array.from(string);
const usingObjectAssign = Object.assign([], string);

// Result:
// [ 's', 'p', 'l', 'i', 't', '-', 'b', 'y', '-', 'd', 'a', 's', 'h' ]
```

Strings containing Emojis

If your strings contain emojis, then `split` or `Object.assign` might not be the best choice. Let's see what happens:

```
const string = 'cake😋';

const usingSplit = string.split('');
const usingObjectAssign = Object.assign([], string);

// Result
// [ 'c', 'a', 'k', 'e', '', '' ]
```

However, if we use the other ways, it works :

```
const usingSpread = [...string];
const usingArrayFrom = Array.from(string);

// Result
// [ 'c', 'a', 'k', 'e', '😋' ]
```

This is because `split` separates characters by UTF-16 code units which are problematic because emoji characters are UTF-8. If we look at our yum emoji `'😋'` it's actually made up of 2 characters NOT 1 as we perceive.

```
'😋'.length; // 2
```

This is what's called grapheme clusters - where the user perceives it as 1 single unit, but under the hood, it's in fact made up of multiple units. The newer methods `spread` and `Array.from` are better equipped to handle these and will split your string by **grapheme clusters** 👍

A caveat about `Object.assign` ⚠️

One thing to note `Object.assign` is that it doesn't actually produce a pure array. Let's start with its definition

> The Object.assign() method copies all enumerable own properties from one or more source objects to a target object

The key there is "copies all enumerable own properties". So what we're doing here `Object.assign([], string)` it copying ALL of our string properties over to our new array. Which means we have an Array PLUS some string methods.

TypeScript Test: Result array is not a `string[]` type 😱

This is more evident if we use the TypeScript Playground. Feel free to copy the code and paste in the playground, where you can hover on the variable to view the types. Since this is just an article, I'll paste the result here so you can follow along.

```
const word = 'word';

const usingSplit = string.split('');
const usingSpread = [...string];
const usingArrayFrom = Array.from(string);

// Result:
// string[] 👈 Which means it's an Array of strings
```

However, if we look at the result type of `Object.assign`. It doesn't give us an Array of strings.

```
const usingObjectAssign = Object.assign([], string);

// Result:
// never[] & "string" 👈 which means NOT Array of strings 😱
```

TypeScript Test: Result array can access string properties 😱

We can do further check this by accessing a property that should only be available to a `String`.

```
const string = 'string';
const array = [];

string.bold; // ✅(method) String.bold(): string
array.bold; //  ❌Property 'bold' does not exist on type
```

So that means if I call `bold` on our Array, it should tell us this property doesn't exist. This is what we expect to see:

```
Array.from('string').bold;
// Property 'bold' does not exist on type
```

BUT, if we call `bold` on our supposedly Array created by `Object.assign`, it works 😱

```
Object.assign([], 'string').bold;
// (method) String.bold(): string
```

☝ And this is because `Object.assign` copies over ALL the properties over from the original String. Here's how I'd explain it in non-dev terms. You go to a store to buy a dog. But then store `Object.assign` sells you a dog that has dragon wings. This sounds super cool, but this isn't really a rental friendly pet. Hmm...I don't think this is my best example. But I think you get my point 😂

Conversion seems okay in browser 🙂

Now I don't think this is a major deal-breaker, cause:

> It seems that browsers have some kind of mechanism in place to "safely" do Object.assign([], "string") and avoid adding the methods of that string to the array.

Thank you *@lukeshiru:* for sharing this knowledge for me 👋 He also created a TypeScript playground code so you can see > link

Community Input

@CaptainOrion_: Turn string into char Array but using map function 🤣

```
Array.prototype.map.call('word', eachLetter => eachLetter);

// ['w', 'o', 'r', 'd']
```

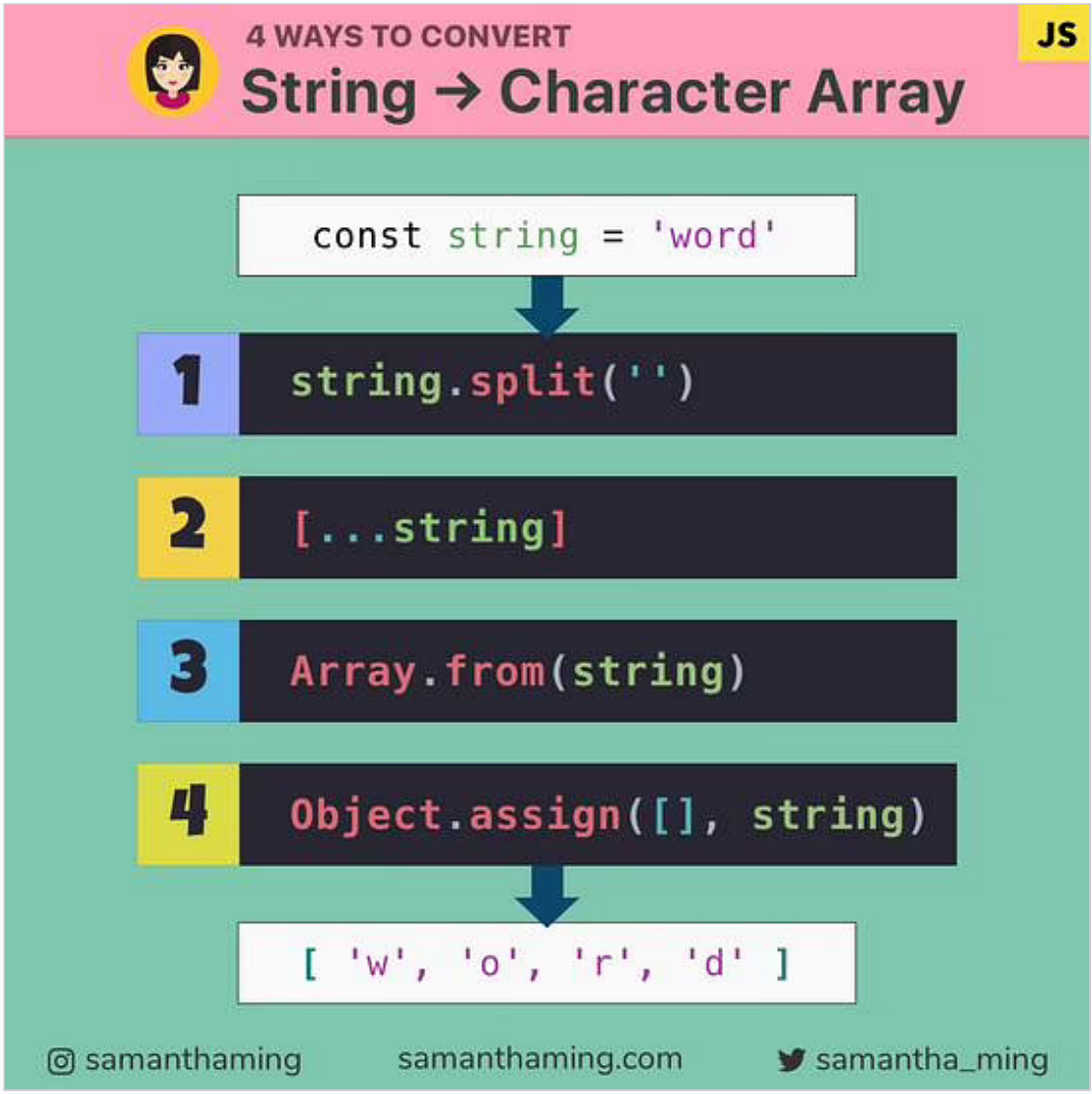
@HiUmesh2: `Array.prototype.slice.call('string')` wil do the trick too

@inside.code: Extra info: it's safer to use the spread operator (second method) rather than `String.prototype.split('')` (first method), because `split()` doesn't work with some uncommon characters.

@faerberrr: I had a string that contained special characters like `åæãă` etc. When I split them using the `.split('')` method and ran `.length`, it returned twice the expected value! Switching to the spread operator fixed the problem.

Resources

- MDN Web Docs: split
- MDN Web Docs: spread
- MDN Web Docs: Array.from
- MDN Web Docs: Object.assign
- Stack Overflow: How do I split a string, breaking at a particular character?
- Stack Overflow: How do you get a string to a character array in JavaScript?
- Stack Overflow: How do I split a string into an array of characters?
- Stack Overflow: Convert utf-8 to Unicode to find emoji in string in java



Download HD Image