

What is “export default” in javascript?

Asked 5 years, 6 months ago Active 9 months ago Viewed 278k times



File: [SafeString.js](#)

453



78

```
// Build out our basic SafeString type  
function SafeString(string) {  
  this.string = string;  
}
```

```
SafeString.prototype.toString = function() {  
  return "" + this.string;  
};
```

```
export default SafeString;
```

I have never seen `export default` before. Are there any equivalent stuff for `export default` that can be easier to understand?

javascript

node.js

ecmascript-6

edited Mar 11 '17 at 13:04



giannis christofakis

5,577 4 43 61

asked Jan 14 '14 at 15:21



damphat

9,074 4 37 51

18 This is a very clear explanation on this 24ways.org/2014/javascript-modules-the-es6-way. – nish1013 Oct 7 '15 at 14:23

`export` keyword details [here](#). Currently it is *not* supported natively by any of the web browsers. – RBT May 1 '17 at 6:47

It's now supported in all browsers but IE. – Brian Di Palma May 30 '18 at 14:02

Very good answer stackoverflow.com/a/36426988/5473170 – Suraj Jain Jan 27 at 10:18

4 Answers



It's part of the ES6 module system, [described here](#). There is a helpful example in that documentation, also:

365

If a module defines a default export:

```
export default function() { console.log("hello!") }
```

then you can import that default export by omitting the curly braces:

```
import foo from "foo";  
foo(); // hello!
```

Update: As of June 2015, the module system is defined in [§15.2](#) and the `export` syntax in particular is defined in [§15.2.3](#) of the ECMAScript 2015 specification.

edited Nov 2 '18 at 21:18

answered Jan 14 '14 at 15:25



p.s.w.g

124k

19

221

262

58 Just a note for future readers, the inline module syntax has been removed from ES6 and now only one module is permitted in a file (just remove `module "foo" {` and the ending `}`). Everything else in this answer is still correct. – [Qantas 94 Heavy](#) Apr 27 '14 at 13:47 ✎

3 Link says "**This page is no longer current**" – [Richard de Wit](#) Feb 11 '15 at 7:37 ✎

1 @GeenHenk I suppose that's to be expected since ES6 is still a draft. I've provided an updated link and a disclaimer. – [p.s.w.g](#) Feb 11 '15 at 17:04

1 ES6 has been approved: infoq.com/news/2015/06/ecmascript-2015-es6 – [Tal Weiss](#) Jun 23 '15 at 13:02 ✎

4 I do not see how `export default function(){}` is any different from `export = function(){}...` – [Alexander Mills](#) Jan 20 '17 at 10:10 ✎

`export default` is used to export a single class, function or primitive from a script file.

120

The export can also be written as

```
export default function SafeString(string) {  
  this.string = string;  
}  
  
SafeString.prototype.toString = function() {
```

```
    return "" + this.string;
};
```

This is used to import this function in another script file

Say in *app.js*, you can

```
import SafeString from './handlebars/safe-string';
```

A little about export

As the name says, it's used to export functions, objects, classes or expressions from script files or modules

Utilities.js

```
export function cube(x) {
    return x * x * x;
}
export const foo = Math.PI + Math.SQRT2;
```

This can be imported and used as

App.js

```
import { cube, foo } from 'Utilities';
console.log(cube(3)); // 27
console.log(foo);    // 4.555806215962888
```

Or

```
import * as utilities from 'Utilities';
console.log(utilities.cube(3)); // 27
console.log(utilities.foo);    // 4.555806215962888
```

When export default is used, this is much simpler. Script files just exports one thing. *cube.js*

```
export default function cube(x) {
    return x * x * x;
};
```

and used as *App.js*


```
import Cube from 'cube';
console.log(Cube(3)); // 27
```

answered May 13 '17 at 8:47



[sudo bangbang](#)

11.4k 6 45 57

 `export default function(){} can be used when the function has no name. There can only be one default export in a file. The alternative is a named export.`

68 This [page](#) describes `export default` in detail as well as other details about modules that I found very helpful.

edited Jul 25 '18 at 20:08

answered Jul 13 '15 at 13:47



[Greg Gum](#)

12.9k 19 96 145

12 You can use default and named exports together if you want to. – [Bergi](#) Jul 13 '15 at 19:15

@Greg gum the page is outdated. It is redirecting to exploringjs.com/es6/ch_modules.html – [rajakvk](#) Nov 8 '15 at 17:30

7 This answer is better than accepted one because it explains what `default` mean and for me the question was about this word. – [Dariusz Sikorski](#) Jun 4 '16 at 6:16

1 @DariuszSikorski the accepted answer explains what `default` means, being that the default export can be imported without using braces. This answer is actually pretty wrong as it says you can only use `default` when there is only one export in a file, which is not true at all. You can have several exports in the same file, but of course only 1 of them can be set as the `default` one. – [realUser404](#) Sep 12 '17 at 21:28

1 I have updated this answer @realUser404 – [Greg Gum](#) Jul 25 '18 at 20:09

 As explained on this [MDN page](#)

8

There are two different types of export, named and default. You can have multiple named exports per module but only one default export[...]Named exports are useful to export several values. During the import, it is mandatory to use the same name of the

corresponding object. But a default export can be imported with any name

For example:

```
let myVar; export default myVar = 123; // in file my-module.js

import myExportedVar from './my-module' // we have the freedom to use 'import
myExportedVar' instead of 'import myVar' because myVar was defined as default export

console.log(myExportedVar);           // will log 123
```

answered Oct 28 '18 at 17:42



manfall19

139 1 3