Typescript ReferenceError: exports is not defined

Asked 3 years, 11 months ago Active 1 month ago Viewed 224k times

Ask Question

125

Trying to implement a module following the official handbook, I get this error message:

> Uncaught ReferenceError: exports is not defined
>
> at app.js:2

But nowhere in my code do I ever use the name `exports`.

How can I fix this?

Files

app.ts

```
let a = 2;
let b:number = 3;

import Person = require ('./mods/module-1');
```


module-1.ts

```
 export class Person {
  constructor(){
    console.log('Person Class');
  }
}
export default Person;
```

tsconfig.json

```
{
   "compilerOptions": {
        "module": "commonjs",
        "target": "es5",
        "noImplicitAny": false,
        "sourceMap": true,
        "outDir": "scripts/"
    },
    "exclude": [
        "node_modules"
    ]
}
```

typescript module

Share Improve this question Follow — edited May 16 '17 at 10:43 iFreilicht 9,558 5 32 63 — asked Mar 27 '17 at 9:39 George C. 4,487 9 37 69

Are you sure you did not type `exports` with an **s** at the end instead of `export`? That would explain the error message as with **s** is wrong. – Igor Mar 27 '17 at 10:18

i type **export** not *exports* – George C. Mar 27 '17 at 10:29

any example from repositorie that gona work 10000% – George C. Mar 27 '17 at 10:41

Where is this being run? On a web page? On a node.js server? You will need a module loader in the run-time environment that the javascript finally runs in. From the compiler flags you are using commonjs. I am not that familiar with commonjs, but you will need to get commonjs set up before Typescript modules will work or you will need to change to another module loader (like require.js) and get that one set up. – Mike Wodarczyk Apr 19 '17 at 19:23

Check stackoverflow.com/a/57509175/11127383 – Daniel Danielecki Aug 15 '19 at 14:30

Add a comment

19 Answers

Active Oldest Votes

52

EDIT:

This answer might not work depending if you're not targeting `es5` anymore, I'll try to make the answer more complete.

Original Answer

If CommonJS isn't installed (which defines `exports`), you have to remove this line from your `tsconfig.json`:

```
"module": "commonjs",
```

As per the comments, this alone may not work with later versions of `tsc`. If that is the case, you can install a module loader like CommonJS, SystemJS or RequireJS and then specify that.

Note:

Look at your `main.js` file that `tsc` generated. You will find this at the very top:

```
Object.defineProperty(exports, "__esModule", { value: true });
```

It is the root of the error message, and after removing `"module": "commonjs",`, it will vanish.

Share Improve this answer Follow — edited Feb 10 at 18:46 Lucas Bustamante 10.5k 5 61 72 — answered May 16 '17 at 10:34 iFreilicht 9,558 5 32 63

11 Yeah I recompiled my .ts file and the error still exists after commenting out `"module": "commonJs",` :(– Acidic9 Sep 22 '17 at 9:17

4 If installing CommonJS will make this error go away, then how does one actually install CommonJS? I have spent the better part of an hour Googling, and I cannot find *any* instructions on how to do so. Can someone *please* explain? – Sturm Oct 8 '18 at 14:45

1 @Sturm My answer may be worded confusingly. CommonJS is only a specification. You can find a (not necessarily exhaustive) list of implementations here: en.wikipedia.org/wiki/CommonJS – iFreilicht Oct 13 '18 at 22:00

2 I have module: amd, not module: commonjs, and I have this line in my transpiled .js file. – pabrams Apr 5 '19 at 16:45

3 If you have target as ES3 or ES5 in your tsconfig.json then typescript will automatically set your module to CommonJS if it's not defined. Removing module isn't enough, you need to set something else instead - see typescriptlang.org/docs/handbook/compiler-options.html – Jake Jul 2 '19 at 6:17

Show **14** more comments

63

Few other Solutions for this issue

- Add the following line before other references to Javascript. This is a nice little hack.

```
<script>var exports = {};</script>
```

- This issue occurs with the latest version of TypeScript, this error can be eliminated by referring to typescript version 2.1.6

Share Improve this answer Follow — edited Jan 21 '18 at 9:42 outoftime 635 6 20 — answered Oct 23 '17 at 10:47 Venkatesh Muniyandi 3,827 1 26 34

3 @ChuckLeButt Hope this helps stackoverflow.com/questions/19059580/... – Venkatesh Muniyandi Apr 25 '18 at 23:43

7 I hate that this works! And it definitely works. Isn't there a way to get TypeScript to transpile into something that Node expects? Is it even Node that's complaining? Or is it the browser? I can't stand not understanding what the problem actually is. – skillit zimberg Feb 14 '20 at 18:08

Add a comment

23

I solved the issue by removing `"type": "module"` field from `package.json`.

Share Improve this answer Follow — answered Apr 2 '20 at 18:33 Masih Jahangiri 3,667 1 16 25

but why does this help? – DauleDK Oct 8 '20 at 13:07

@DauleDK because we use Webpack to bundle app, this field has usage in node.js engine. in webpack config we ues common module system – Masih Jahangiri Oct 8 '20 at 14:26

Add a comment

13

`npm install @babel/plugin-transform-modules-commonjs`

and add to to .babelrc plugins resolved my question.

Share Improve this answer Follow — answered Dec 23 '18 at 2:23 Yi Zhang 139 1 3

This worked for me, Thanks! – Siddhartha Chowdhury Dec 28 '20 at 13:51

Add a comment

7

my solution is a sum up of everything above with little tricks I added, basically I added this to my html code

```
<script>var exports = {"__esModule": true};</script>
<script src="js/file.js"></script>
```

this even allows you to use `import` instead of `require` if you're using electron or something, and it works fine with typescript 3.5.1, target: es3 -> esnext.

Share Improve this answer Follow — answered Jun 6 '19 at 12:57 Hocine Abdellatif 451 5 15

4 some progress here, the error just changed to be `app.js:3 Uncaught ReferenceError: require is not defined` – Muhammed Moussa Feb 16 '20 at 18:37

this happens if nodeIntegration is set to false when creating the window, the error happens because you updated your electronJS or you're following an outdated source. nodeIntegration was true by default, now it's false, so you have to enable it manually if you want to access nodeJS in renderer process. See [Electron require() is not defined][1] [1]: stackoverflow.com/questions/44391448/... – Hocine Abdellatif Feb 16 '20 at 18:51

I'm not using electron, anyway I solved in my case by adding parcel to make bundle stuff and removed other tricky ways – Muhammed Moussa Feb 16 '20 at 18:56

Add a comment

7

This is fixed by setting the `module` compiler option to `es6`:

```
{
  "compilerOptions": {
    "module": "es6",
    "target": "es5",
  }
}
```

Share Improve this answer Follow — answered Nov 3 '19 at 13:06 Code Whisperer 20.5k 17 53 79

Add a comment

6

I had the same problem and solved it adding "**es5**" library to tsconfig.json like this:

```
{
    "compilerOptions": {
        "target": "es5", //defines what sort of code ts generates, es5 because it's what mo
        "module": "commonjs",
        "moduleResolution": "node",
        "sourceMap": true,
        "emitDecoratorMetadata": true, //for angular to be able to use metadata we specify
        "experimentalDecorators": true, //angular needs decorators like @Component, @Inject
        "removeComments": false,
        "noImplicitAny": false,
        "lib": [
            "es2016",
            "dom",
            "es5"
        ]
    }
}
```

Share Improve this answer Follow — answered Oct 16 '17 at 9:30 rubensa 860 11 8

Add a comment

5

For people still having this issue, if your compiler target is set to ES6 you need to tell babel to skip module transformation. To do so add this to your `.babelrc` file

```
{
  "presets": [ ["env", {"modules": false} ]]
}
```

Share Improve this answer Follow — answered Dec 12 '17 at 22:08 Cezar Augusto 6,169 4 24 34

1 Thanks man! I've been trying to work with older code base which dump all into global and scripts are included in browser with the `<script>` tag. I was hoping to use module along with older code and it seems like that is not possible. This at least allow me to use ES6 and let me deal with export later. – huggie Feb 9 '18 at 7:01

2 doing this I got the following error: **Using removed Babel 5 option: .modules - Use the corresponding module transform plugin in the `plugins` option. Check out babeljs.io/docs/plugins/#modules** – chharvey Apr 6 '19 at 15:01

Add a comment

3

I had this same error too. In my case it was because we had an old-fashioned import statement in our TypeScript AngularJS project like this:

```
import { IAttributes, IScope } from "angular";
```

which was compiled to JavaScript like this:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
```

This was needed back in the old days because we then used `IAttributes` in the code and TypeScript wouldn't have known what to do with it otherwise. But after removing the import statement, and converting `IAttributes` to `ng.IAttributes` those two JavaScript lines disappeared - and so did the error message.

Share Improve this answer Follow — answered Jun 27 '19 at 14:42 MatX 90 5

1 How would that work for a Typescript type? I have to import it, so Typescript compiles it. – BluE Jul 8 '19 at 9:39

What do you mean by "Typescript type"? If it's an integeral type known by TypeScript - such as number or string - then you can use that right away. If you define your own types in a module, check this: typescriptlang.org/docs/handbook/modules.html – MatX Jul 8 '19 at 11:18

I meant the type definitions of an external library like @types/jquery from npmjs.com/package/@types/jquery . To make the $ variable usable from within my Typescript code, I put this in my code: import * as $ from "jquery"; – BluE Jul 8 '19 at 11:25

I am targeting es5, do I need another library like requirejs to make this work? – BluE Jul 8 '19 at 11:26

Add a comment

2

Simply add `libraryTarget: 'umd'`, like so

```
const webpackConfig = {
  output: {
    libraryTarget: 'umd' // Fix: "Uncaught ReferenceError: exports is not defined".
  }
};

module.exports = webpackConfig; // Export all custom Webpack configs.
```

Share Improve this answer Follow — answered Aug 15 '19 at 9:16 Daniel Danielecki 3,085 2 25 39

Add a comment

2

for me, removing `"esModuleInterop": true` from tsconfig.json did the trick.

Share Improve this answer Follow — edited Oct 25 '19 at 16:34 Dharman 20.9k 18 53 104 — answered Oct 25 '19 at 16:20 user3033599 29 1

How does removing it work since `exports` can ONLY be used if `esModuleInterop` is set to true ? – Ananta K Roy Jan 11 at 12:48

This worked for me (removed the above error). I'm using ESM with typescript (with jest), so I don't want exports. – crimbo Jan 20 at 23:41

Add a comment

1

For some ASP.NET projects `import` and `export` may not be used at all in your Typescripts.

The question's error showed up when I attempted to do so and I only discovered later that I just needed to add the generated JS script to the *View* like so:

```
<script src="~/scripts/js/[GENERATED_FILE].Index.js" asp-append-version="true"></script>
```

Share Improve this answer Follow — answered Jul 17 '19 at 16:09 CPHPython 6,179 2 33 55

Add a comment

1

Note: This might not be applicable for OP's answer, but I was getting this error, and this how I solved it.

So the problem that I was facing was that I was getting this error when I retrieved a 'js' library from a particular CDN.

The only wrong thing that I was doing was importing from the CDN's `cjs` directory like so: `https://cdn.jsdelivr.net/npm/@popperjs/core@2.4.0/dist/cjs/popper.min.js`

Notice the `dist/cjs` part? That's where the problem was.

I went back to the CDN (jsdelivr) in my case and navigated to find the `umd` folder. And I could find another set of `popper.min.js` which was the correct file to import: `https://cdn.jsdelivr.net/npm/@popperjs/core@2.4.0/dist/umd/popper.min.js`.

Share Improve this answer Follow — answered May 27 '20 at 23:38 Arpan Srivastava 169 2 8

Add a comment

0

Try what @iFreilicht suggested above. If that didn't work after you've installed webpack and all, you may have just copied a webpack configuration from somewhere online and configured there that you want the output to support CommonJS by mistake. Make sure this isn't the case in `webpack.config.js`:

```
module.exports = {
  mode: process.env.NODE_ENV || "development",
  entry: {
    index: "./src/js/index.ts"
  },
  ...
  ...
  output: {
    libraryTarget: 'commonjs',         <==== DELETE THIS LINE
    path: path.join(__dirname, 'build'),
    filename: "[name].bundle.js"
  }
};
```

Share Improve this answer Follow — answered Aug 14 '19 at 18:42 duliba 181 1 8

Add a comment

0

Had the same issue and fixed it by changing the JS packages loading order.

Check the order in which the packages you need are being called and load them in an appropriate order.

In my specific case (not using module bundler) I needed to load `Redux`, then `Redux Thunk`, then `React Redux`. Loading `React Redux` before `Redux Thunk` would give me `exports is not defined`.

Share Improve this answer Follow — answered Nov 5 '19 at 17:15 Pat 203 1 9

Add a comment

0

I had the same issue, but my setup required a different solution.

I'm using the `create-react-app-rewired` package with a `config-overrides.js` file. Previously, I was using the `addBabelPresets` import (in the `override()` method) from `customize-cra` and decided to abstract those presets to a separate file. Coincidentally, this solved my problem.

I added `useBabelRc()` to the `override()` method in `config-overrides.js` and created a `babel.config.js` file with the following:

```
module.exports = {
    presets: [
        '@babel/preset-react',
        '@babel/preset-env'
    ],
}
```

Share Improve this answer Follow — answered Aug 7 '20 at 20:34 Doug Wilhelm 556 4 23

Add a comment

0

To solve this issue, put these two lines in your *index.html* page.

```
<script>var exports = {"__esModule": true};</script>
<script type="text/javascript" src="/main.js">
```

Make sure to check your *main.js* file path.

Share Improve this answer Follow — edited Oct 6 '20 at 15:28 aboger 1,696 5 27 39 — answered Oct 6 '20 at 5:59 Rupesh Shitole 1

This only works if you don't have any `require()` statements in your code. – Daniel Tonon Oct 9 '20 at 11:02

Add a comment

-1

```
  {
    "compileOnSave": false,
    "compilerOptions": {
      "baseUrl": "./",
      "outDir": "./dist",
      "sourceMap": true,
      "declaration": false,
      "module": "esnext",
      "moduleResolution": "node",
      "emitDecoratorMetadata": true,
      "experimentalDecorators": true,
      "target": "es5",
      "typeRoots": ["node_modules/@types"],
      "lib": ["es2018", "dom"]
    }
  }
```

Share Improve this answer Follow — answered Jun 18 '20 at 0:49 Carlos Alves 1

7 Welcome to StackOverflow. Please add some context to your answer or some explanation so the user can understand what was done. Only posting a JSON may not be as helpful... – Bruno Monteiro Jun 18 '20 at 2:51

Add a comment

-1

If you are just using interfaces for types, leave out the `export` keyword and ts can pick up on the types without needing to import. They key is you cannot use `import` / `export` anywhere.

```
export interface Person {
   name: string;
   age: number;
}
```

into

```
interface Person {
   name: string;
   age: number;
}
```

Share Improve this answer Follow — answered Oct 7 '20 at 18:19 Matt 1,575 7 17

Add a comment

Your Answer

Sign up or log in

Sign up using Google

Sign up using Facebook

Sign up using Email and Password

Post as a guest

Name

Email

Required, but never shown

Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged typescript module or ask your own question.

The Overflow Blog
State of the Stack: a new quarterly update on community and product
Podcast 320: Covid vaccine websites are frustrating. This developer built a…
Featured on Meta
State of the Stack Q1 2021 Blog Post

Linked
6 visual studio typescript "Uncaught ReferenceError: exports is not defined at…"
444 Setting Environment Variables for Node to retrieve
389 Client on Node.js: Uncaught
6 Typescript & Electron exports is not defined
9 React 16.14.0: Error was not caught ReferenceError: exports is not defined
3 Typescript - Uncaught ReferenceError: exports is not defined
4 Exports is not defined ia tsc'ed (js) files
6 How to setup NativeScript Angular with Karma, karma-typescript, Jasmine, Mocha, Chai
3 Require… now exports is undefined, What am I missing here?
See more linked questions

Related
770 How do you explicitly set a new property on `window` in TypeScript?
756 get and set in TypeScript
1073 How to convert a string to number in TypeScript?
1 Using mysql types in typescript
3 Got "TS2300: Duplicate identifier 'Account'" error after upgraded to Typescript 2.9.1
17 LernaJS Typescript cannot find module

Hot Network Questions
Are questions on theory useful in interviews?
Why did Bobby Fischer resign his title?
What happens to the non-axial photons of a laser cavity?
Term to describe paradox where those with less subject matter expertise can sometimes make better teachers?
Self-Replicating Numbers
arXiv article says that code has been made available with the article, but I cannot find it
Am I allowed to use images from sites like Pixabay in my YouTube videos?
Can I use a MacBook as a server with the lid closed?
A circle touches two sides of a triangle and two of its medians
What tool do I need in order to remove these pedals?
Who is the true villain of Peter Pan: Peter, or Hook?
Who started the "-oid" suffix fashion in math?
Specify range of variable in a equation
How can I create a new environment and make the first paragraph after it withouth indentation?
Time complexity of min() and max()?
How do car leases work?
How can I tell if all USB-C ports on a MacBook Pro are the same speed, or if one or more will be faster than others?
How to select outermost vertices in a shape like this?
Why are nuclides with an even number of protons and neutrons more stable?
Understanding the behavior of C's preprocessor when a macro indirectly expands itself
What is this very long pipe instrument?
Recommendations for OR video channels (YouTube etc)
New DM on House Rules, concerning Nat20 & Rule of Cool
Compare the first 20 lines of two files
Question feed