Additionnal Questions for Hands-on NodeJs + JavaScript + TypeScript … ‘require()’, ‘export / import’

Esilv 2023

arnaud.nauwynck@gmail.com

This document:
https://github.com/Arnaud-Nauwynck/presentations/tree/main/web/tp1-questions-nodejs.pdf

Objectives

This Hands-On contains many small questions,
It is a Step-by-Step discovery for nodejs modules & imports

Questions are optionnal and may not be finished during the 3h session.

The main objectives of the session is to get familiar with your IDE (WebStorm) environment, and be able to browse nodeJs modules.

If you are already familiar with FrontEnd development,
do the questions, you might learn things on nodejs import.

Question 1 : ChatGPT …

This is a up to You to ask ChatGPT things about NodeJs
to learn things you do not understand

Ask him about « require() », « import », « named import »,
« import * » , « import as », « default export »,
« CommonJs », « ESM module », etc…

Except for explanations, please do NOT use ChatGPT
(or GithubCopilot) during the Hands-On to write Code !

Answer 1… Example

A

can you use "import" in nodejs like in typescript

Yes, you can use the `import` statement in Node.js just like in TypeScript if you are using ECMAScript modules (ESM). Starting from Node.js version 13.2.0, ESM support has been added to Node.js, allowing you to use `import` and `export` for module loading and exporting. Here's how you can use it:

1. **Enable ECMAScript Modules**:
   To use `import` and `export` in Node.js, you need to use the `.mjs` file extension for your modules or specify `"type": "module"` in your `package.json`:

A

peut on avoir dans le meme projet des require() et des imports? voire dans le meme fichier ?

Oui, il est possible d'avoir à la fois des `require()` (CommonJS) et des `import` (ECMAScript modules - ESM) dans le même projet, voire dans le même fichier, lorsque vous utilisez Node.js. Cette flexibilité est offerte par Node.js pour faciliter la transition entre les deux systèmes de modules, et elle peut être utile dans certaines situations, notamment lorsque vous travaillez avec des modules tiers qui utilisent encore CommonJS (`require()`).

MAIS … 20 lignes plus loin ??!!!! ….
ChatGPT fabule le contraire …

3. **Compatibilité** :
   - Gardez à l'esprit que les modules utilisant `require()` et ceux utilisant `import` ne peuvent pas être mélangés dans le même fichier, mais ils peuvent coexister dans le même projet. Assurez-vous de comprendre quel système de module est utilisé dans chaque fichier afin de l'utiliser correctement.

Question 2: use export & require()

Create 2 files

- File « q2/A.ts »
  to export a class named « A », that has a constructor logging a message

- File « q2/B.ts »
  use « require » syntax, like « constant moduleA = require(….); »
  then use class A from moduleA, using « new » to create an object

Question 3: Run the code in « node »

Run either from command-line,

C:> tsc q2/a.ts

C:> node q2/a.js

or Click « Run » in Webstorm IDE

Notice that « *.ts » may not be executed directly, but « *.js » below

Do not forget to create a default tsconfig.json, for { "sourceMap": true }

… It should Run correctly

Question 3 (next)…

Add

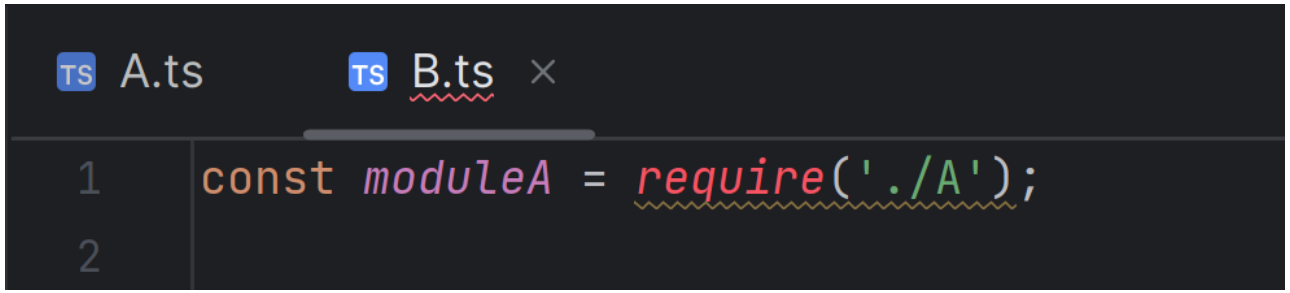
```
console.log('moduleA', moduleA);
```

Do you see details of moduleA content in console ?

Try to dump the object properties in code
Or else, you need to run the Debugger ! .. See next question 5

Question 4:

Using WebStorm IDE, the « require()» from Q2 is in RED
… showing a WARNING, ERROR, or missing Node configuration ?



Fix the WARNING as suggested by WebStorm

Question 5: Debug and inspect value

Re-launch in Debug mode and Put Breakpoint

Inspect content of variable « moduleA »

Question 6 : can you modify the « const » object content ?

Add code

```
moduleA.dummyField = 1;
```

Any compile Error in TypeScript?
What about Generated JavaScript ?
Runtime Error ?

Question 7: meaning of « const »

What is the meaning of « const » ?

Can you make object immutable ?

Question 8: meaning of « let » and « var »

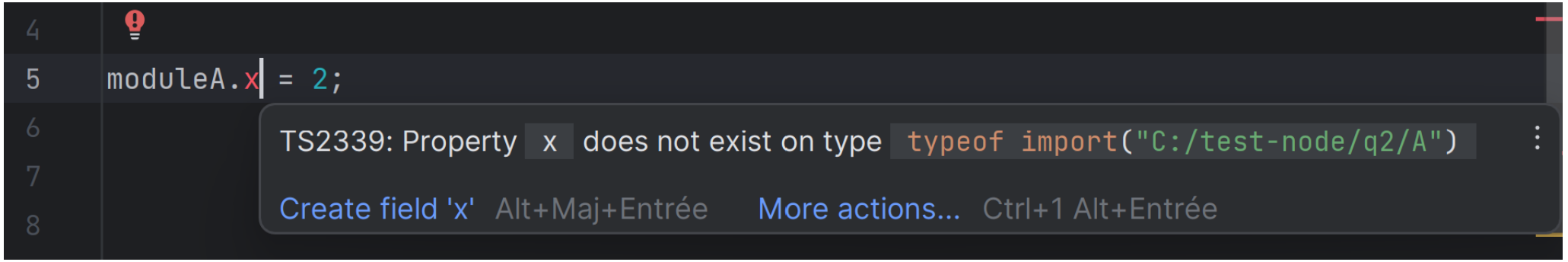
Does JavaScript accept accessing unknown(undeclared) variable ?

What is the meaning of « let » and « var » ?
What difference(s) between « let » and « var » ?

Knowing that « var » can be re-declared several times but not « let », what would you generally recommend to use ?

Question 9: property does not exist… but Runtime OK ?

WebStorm shows an ERROR for « property does not exist »



WebStorm suggest adding « // @ts-ignore » .. Try it

Suggest another way in Javascript to assign a property, not causing an error

Question 10 : call require() several times… check same object is returned

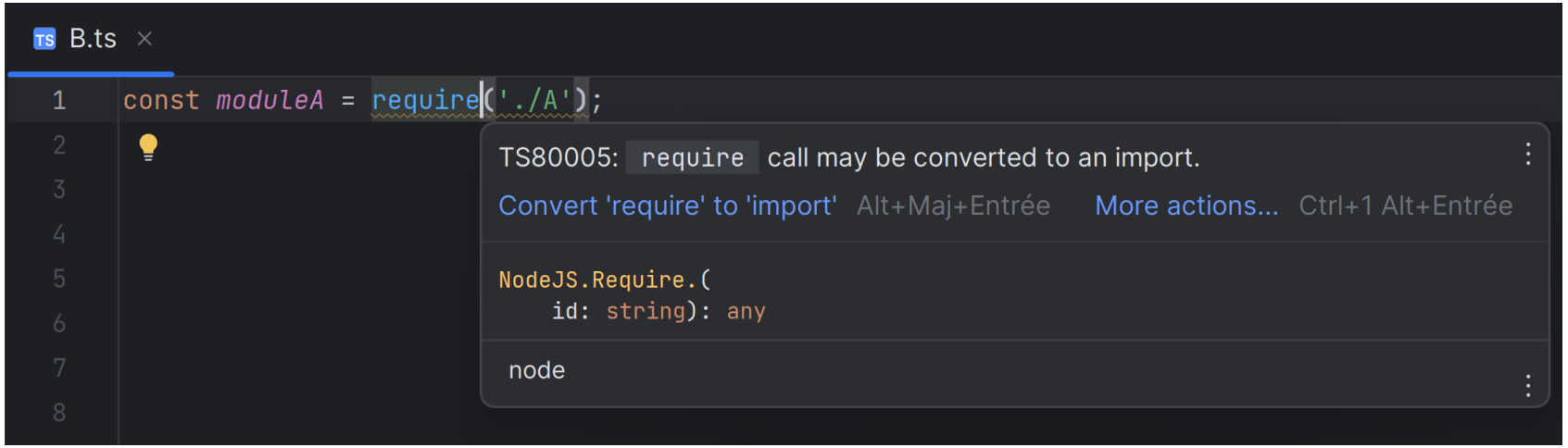
const moduleA = require('./A');
const moduleA2 = require('./A')

Are they equals… Same content? Or same pointer ?

From Debugger, modify « moduleA » object to add a dummy property

Also modify from moduleA2
.. See if moduleA and moduleA2 are in-sync

Question 11 : perform WebStorm suggested change convert ‘require’ to ‘import’

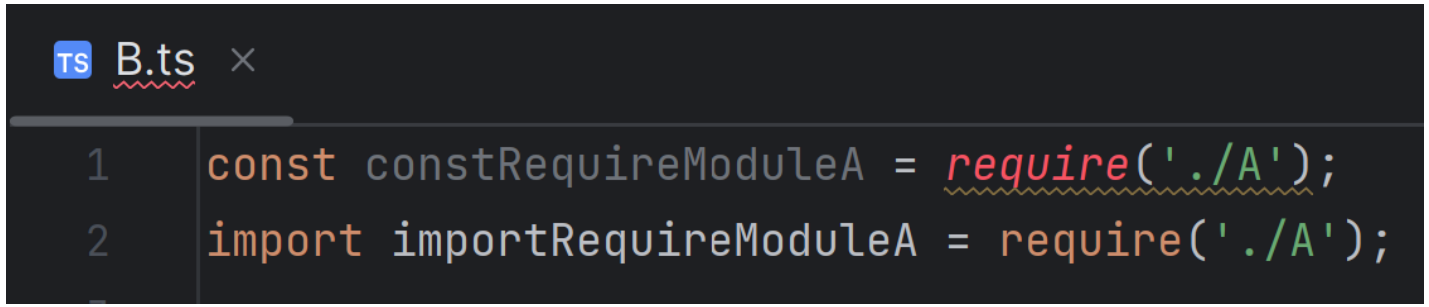


What do you get ?

Is is a standard named import like in TypeScript ?

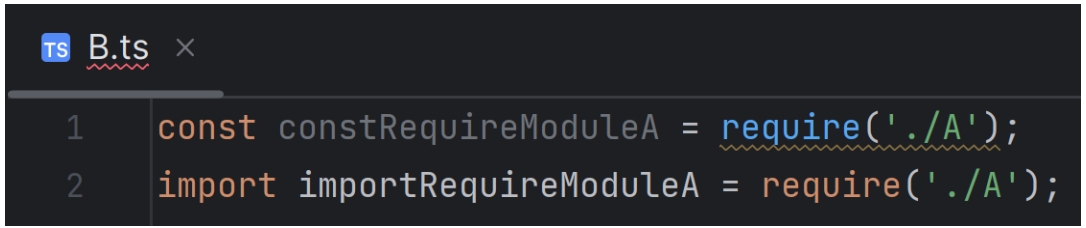
Question 12: Difference in generated JavaScript code ?

Is there any difference in generated Javascript code for both lines ?



Noticed line 1 is in « RED » but line 2 is ok.

After Applying « npm i --save-dev @types/node », you have underline marker on line 1



Question 13: transform to standard {named} import

transform to standard « named import »
as in Angular – TypesScript

Adapt « new A() » code, now simpler

Check re-run code

See generated JavaScript code

Question 14: redo using « import * as » syntax

Redo using « import * as » syntax

See generated JavaScript code, Compare with « =require() »

<u>Optionnaly:</u>
Repeat several times the « import * as », using different names
then add dummy fields moduleA.x= .. and moduleB.y=.. in each.
Explain what should you get in each ?

Question 15 : show generated js code from ts

Compare generated JavaScript codes for corresponding TypeScript :

What is generated for line « import * as moduleA from ‘./A’ » ?

What is generated for line « import { A } from ‘./A’ » ?

What is generated for line « import { A as B } from ‘./A’ » ?

What is generated for lines « new moduleA.A() »? « new A() »? new B() » ?

Question 16: generated names for imports

See from Previous question:
generated code for « import {A} from ‘./A’ » => creates a constant declaration
« const A_1 = require(…); »

What if you add in your code a conflicting declaration ?
« const A_1 = ‘something else’; » ?

How is the conflict resolved ?

Question 17: export default … import named

Remember that file A.ts declares only 1 export, so
Create a similar new file « C.ts », which contains a class C … with « export **default** »

Now you can import with syntax « import C from ‘./C’ »

Notice NO more « { } » , as in « import { A } from ‘./A’ »

What is the generated JavaScript code for « import » and « new C() » ?

Question 18: export default… import with different name

Does it work ?

```
    import D from ‘./C’;
    new D();

import * as E from ‘./C’;
new E(); // <= compile error …
```

What is the correct syntax for creating the object?

```
import { C }  from ‘./C’;  // <= compile error .. Don’t use {}
```

Question 19: import from node_modules

Use import from ‘express’

Notice you do not specify relative path ‘./’
and in particular, do not use ‘../node_modules/express’

Question 20: Express … imported

What is the type of the imported object ?
… so how do you use it?

Check both in Debugger
and using suggested @types/express

Advanced Remark on Question 20 : exploring alternative syntaxes to import express

D.ts

```ts
import * as express1 from 'express';
const express2 = require('express');
import express3 = require('express');
import express4 from 'express' // <= need tsconfig.json "allowSyntheticDefaultImports": true
import * as express5 from 'express';
import { Express as express6 } from 'express'; // <= NOT same import .. ERROR !!

express1();
express2();
express3();
express4();
express5();
express6()
```

Question 21: re-export from import …

Create a file ‘q2/index.ts’ that re-export all from ‘q2/A’, ‘q2/C’

Check that you can now directly import {A,C} from ‘./q2’
(suffix ‘/index’ is optionnal)

Question 22: import from built-in modules

using built-in module ‘fs’, here is a simple program to list all files in a directory

```
import * as fs from 'fs';

fs.readdir('.', (err, files) => {
    files.forEach(f => {
        console.log('file: ' + f)
    })
});
```

1/ Execute it
2/ From your IDE, browse into ‘fs’ declaration
check you have source code « node_modules/@types/node/fs.d.ts »

Question 23: compare import NodeJs<->Angular… how is the generated JavaScript code in Angular ?

You started with ChatGPT, so Happily finish by ChatGPT !!
Ask him

What is generated javascript code for import in angular typescript ?
Differences between CommonJS, ES6, AMD, SystemJS ?
Whats is Webpack ?

What means EcmaScript compared to JavaScript?
ES6, EcmaScript2015 ?
ES7 … ES12 ? ES2021 ? Esnext ?

Next Steps … Hand-on 2 :

build a http Rest Api (Json) in nodejs, using express

define CRUD for a simple collection,
use Postman ,
add OpenApi, use swagger-ui