About “*.d.ts” in TypeScript

210

47

I'm feeling curious about `*.d.ts` because of a newbie of TypeScript. And I was told by someone that this kind of file is something like "head file" in C++ but for JS only. But I cannot convert a pure JS file to `*.d.ts` file unless I forcely change the `*.js` to `*.ts`. So I have three files: a JS file, a TS file and a `*.d.ts` file.

1. What's the relationship between them?
2. How can I use the `*.d.ts` file? Does it mean I can delete the `*.ts` file permanently?
3. If so, how can the `*.d.ts` file know which JS file is mapping to itself?

Many thanks! It would be very nice if someone can give me an example.

typescript .d.ts

edited Mar 28 at 2:00

Rich

3,890 6 28 52

asked Jan 21 '14 at 1:05

user3221822

5 Answers

224

The "d.ts" file is used to provide typescript type information about an API that's written in JavaScript. The idea is that you're using something like jQuery or underscore, an existing javascript library. You want to consume those from your typescript code.

Rather than rewriting jquery or underscore or whatever in typescript, you can instead write the d.ts file, which contains only the type annotations. Then from your typescript code you get the typescript benefits of static type checking while still using a pure JS library.

13 Many thanks! But how to map a *.d.ts file to a js file? How does the js file know which d.ts file is mapping to itself? Can you give me an example? – user3221822 Jan 21 '14 at 1:13

2 But the d.ts file is generated from the js file, and if the js file knows nothing about d.ts. How to call the functions from d.ts in other ts files without the js file? I'm puzzled..... – user3221822 Jan 21 '14 at 1:51

5 See stackoverflow.com/questions/18091724/… . You need to add a ///<reference line to the top of the consuming ts file. You'll need to have both the d.ts and the .js file available. – Chris Tavares Jan 21 '14 at 1:53

2 the d.ts file is generally handwritten from js file documentation. A large number of these are available for popular javascript libraries : github.com/borisyankov/DefinitelyTyped – basarat Jan 21 '14 at 2:22

9 Where do you put custom d.ts files if you're making custom ones for your project? – Jay Sep 19 '16 at 19:54

45

`d` stands for declaration files:

> When a TypeScript script gets compiled there is an option to generate a declaration file (with the extension .d.ts) that functions as an interface to the components in the compiled JavaScript. In the process the compiler strips away all function and method bodies and preserves only the signatures of the types that are exported. The resulting declaration file can then be used to describe the exported virtual TypeScript types of a JavaScript library or module when a third-party developer consumes it from TypeScript.
>
> The concept of declaration files is analogous to the concept of header file found in C/C++.

```
declare module arithmetics {
    add(left: number, right: number): number;
    subtract(left: number, right: number): number;
    multiply(left: number, right: number): number;
    divide(left: number, right: number): number;
}
```

> Type declaration files can be written by hand for existing JavaScript libraries, as has been done for jQuery and Node.js.
>
> Large collections of declaration files for popular JavaScript libraries are hosted on GitHub in DefinitelyTyped and the Typings Registry. A command-line utility called typings is provided to help search and install declaration files from the repositories.

3 Note: the `typings` command line tool isn't really needed since TypeScript 2.0. The more up-to-date approach is to use typing wrappers via the npm repository under the `@types` namespace. For further detail see github.com/typings/typings/blob/master/README.md – Burt_Harris Dec 20 '17 at 5:37

30

I could not comment and thus adding this as an answer. We had some pain trying to map existing types to a javascript library.

To map a d.ts file to its javascript file you need to give the d.ts file the same name as the javascript file, keep them in the same folder, and point the code that needs it to the d.ts file.

eg: test.js and test.d.ts are in the testdir then you import it like this in a react component:

```
import * as Test from "./testdir/test";
```

The d.ts file was exported as a namespace like this:

```
export as namespace Test;

export interface TestInterface1{}
export class TestClass1{}
```

answered Aug 29 '18 at 14:31

ConstantinM
406 5 7

21 Nobody gave an answer as to how to connect d.ts to js, so i thought this is the right place. – ConstantinM Aug 31 '18 at 13:37

Note that if you're creating a declaration file for some JS that you didn't create (a package from npm for example) that the `.d.ts` file must also be named the same as the package to be imported. – electrovir Jun 5 at 17:03

Like @takeshin said .d stands for declaration file for typescript (.ts).

3. "Type definition" and "Type checking" are major add-on functionalities that typescript provides over javascript. (check difference between type script and javascript)

If you are thinking if typescript is just syntactic superset, what benefits does it offer - https://basarat.gitbooks.io/typescript/docs/why-typescript.html#the-typescript-type-system

To Answer this post -

As we discussed, typescript is superset of javascript and needs to be transpiled into javascript. So if a library or third party code is written in typescript, it eventually gets converted to javascript which can be used by javascript project but vice versa does not hold true.

For ex -

If you install javascript library -

```
npm install --save mylib
```

and try importing it in typescript code -

```
import * from "mylib";
```

you will get error.

> "Cannot find module 'mylib'."

As mentioned by @Chris, many libraries like underscore, Jquery are already written in javascript. Rather than re-writing those libraries for typescript projects, an alternate solution was needed.

In order to do this, you can provide type declaration file in javascript library named as *.d.ts, like in above case mylib.d.ts. Declaration file only provides type declarations of functions and variables defined in respective javascript file.

Now when you try -

```
import * from "mylib";
```


Michal
97 3 17


Keshav
641 8 14

4

Worked example for a *specific case*:

Let's say you have *my-module* that you're sharing via *npm*.

You install it with `npm install my-module`

You use it thus:

```
import * as lol from 'my-module';

const a = lol('abc', 'def');
```

The module's logic is all in `index.js` :

```
module.exports = function(firstString, secondString) {

  // your code

  return result
}
```

To add typings, create a file `index.d.ts` :

```
declare module 'my-module' {
  export default function anyName(arg1: string, arg2: string): MyResponse;
}

interface MyResponse {
  something: number;
  anything: number;
}
```