# Chapter 14: Importing external libraries

## Section 14.1: Finding definition files

for typescript 2.x:

definitions from DefinitelyTyped are available via @types npm package

```
npm i --save lodash
npm i --save-dev @types/lodash
```

but in case if you want use types from other repos then can be used old way:

for typescript 1.x:

Typings is an npm package that can automatically install type definition files into a local project. I recommend that you read the quickstart.

```
npm install -global typings
```

Now we have access to the typings cli.

1. The first step is to search for the package used by the project

```
typings search lodash
NAME              SOURCE HOMEPAGE                                        DESCRIPTION VERSIONS
UPDATED
lodash            dt     http://lodash.com/                                          2
 2016-07-20T00:13:09.000Z
lodash            global                                                             1
 2016-07-01T20:51:07.000Z
lodash            npm    https://www.npmjs.com/package/lodash                        1
 2016-07-01T20:51:07.000Z
```

2. Then decide which source you should install from. I use dt which stands for DefinitelyTyped a GitHub repo where the community can edit typings, it's also normally the most recently updated.
3. Install the typings files

```
typings install dt~lodash --global --save
```

Let's break down the last command. We are installing the DefinitelyTyped version of lodash as a global typings file in our project and saving it as a dependency in the `typings.json`. Now wherever we import lodash, typescript will load the lodash typings file.

4. If we want to install typings that will be used for development environment only, we can supply the `--save-dev` flag:

```
typings install chai --save-dev
```

## Section 14.2: Importing a module from npm

If you have a type definition file (d.ts) for the module, you can use an `import` statement.

```
import _ = require('lodash');
```

If you don't have a definition file for the module, TypeScript will throw an error on compilation because it cannot find the module you are trying to import.

In this case, you can import the module with the normal runtime `require` function. This returns it as the `any` type, however.

```
// The _ variable is of type any, so TypeScript will not perform any type checking.
const _: any = require('lodash');
```

As of TypeScript 2.0, you can also use a *shorthand ambient module declaration* in order to tell TypeScript that a module exists when you don't have a type definition file for the module. TypeScript won't be able to provide any meaningful typechecking in this case though.

```
declare module "lodash";

// you can now import from lodash in any way you wish:
import { flatten } from "lodash";
import * as _ from "lodash";
```

As of TypeScript 2.1, the rules have been relaxed even further. Now, as long as a module exists in your `node_modules` directory, TypeScript will allow you to import it, even with no module declaration anywhere. (Note that if using the `--noImplicitAny` compiler option, the below will still generate a warning.)

```
// Will work if `node_modules/someModule/index.js` exists, or if
`node_modules/someModule/package.json` has a valid "main" entry point
import { foo } from "someModule";
```

## Section 14.3: Using global external libraries without typings

Although modules are ideal, if the library you are using is referenced by a global variable (like $ or _), because it was loaded by a `script` tag, you can create an ambient declaration in order to refer to it:

```
declare const _: any;
```

## Section 14.4: Finding definition files with TypeScript 2.x

With the 2.x versions of TypeScript, typings are now available from the npm @types repository. These are automatically resolved by the TypeScript compiler and are much simpler to use.

To install a type definition you simply install it as a dev dependency in your projects package.json

e.g.

```
npm i -S lodash
npm i -D @types/lodash
```

after install you simply use the module as before

```
import * as _ from 'lodash'
```