# Understanding Module Resolution

**Dan Wellman**
LEAD UI DEVELOPER

Module resolution is the process of locating and loading modules

# Overview

In this module we will learn:

- How relative and non-relative imports are handled
- How to enable module tracing for diagnostic information
- How to use the baseUrl configuration
- How to use the paths configuration
- How to use virtual directories

# Module Resolution Strategies

## Module resolution strategies

- Classic
- Node

## Module resolution strategies

- Classic
- Node (default)

## Module resolution strategies

- Classic (backwards-compatibility)
- Node (default)

TypeScript's node strategy mirrors Node.js

With some additional behaviour on top

```
const myModule = require('./path');
```

Node.js uses the require function to load modules

```
const myModule = require('./path');
```

The path may be relative or non-relative

```
const myModule = require('./path');
```

Relative paths start
    '/'

```
const myModule = require('./path');
```

Relative paths start

- '/'
- './'

```
const myModule = require('./path');
```

Relative paths start

- '/'
- './'
- '../'

```
const myModule = require('path'); // non-relative
```

Anything else is non-relative

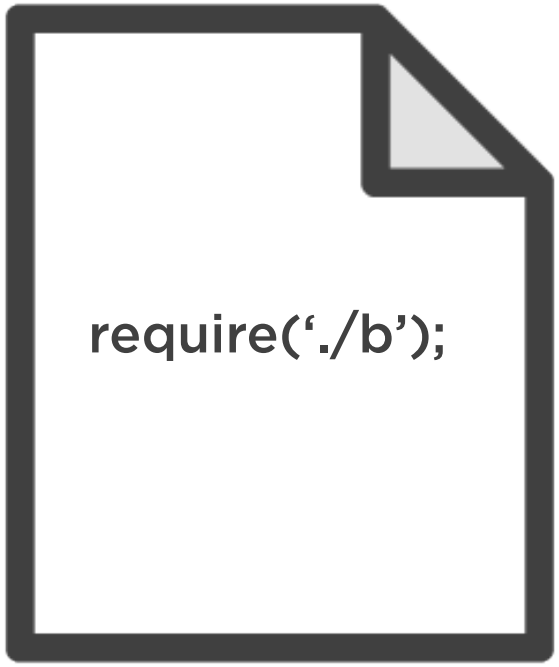
```
const myModule = require('./path');
```

Relative paths start

- '/'
- './'
- '../'

# Node.js Relative Imports

/src/a.js

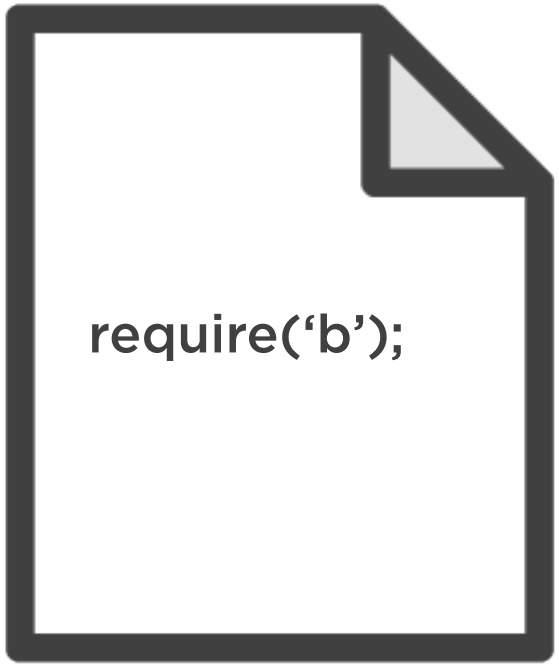



**Node will look for module b in:**

- /src/b.js
- /src/b/package.json
  - If the main property of package.json points to a file, look for that file
- /src/b/index.js

# Node.js Non-relative Imports

/src/a.js

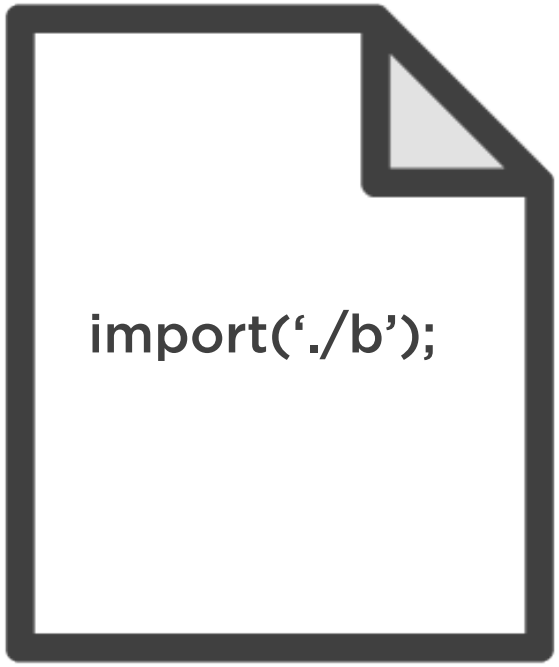



**Node will look for module b in:**

- /src/node_modules/b.js
- /src/node_modules/b/package.json
- /src/node_modules/b/index.js
- /node_modules/b.js
- /node_modules/b/package.json
- /node_modules/b/index.js

# TypeScript Relative Imports

/src/a.ts

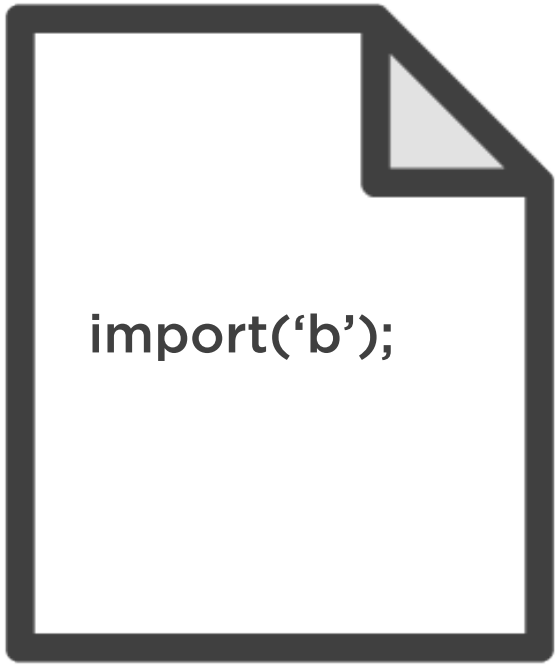



**TypeScript will look for module b in:**

- /src/b.ts(x)
- /src/b.d.ts
- /src/b/package.json
  - If the types property of package.json points to a file, look for that file
- /src/b/index.ts(x)
- /src/b/index.d.ts

# TypeScript Non-relative Imports

/src/a.ts

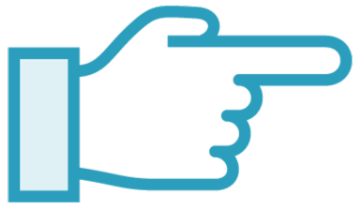



**TypeScript will look for module b in:**

- /src/node_modules/b.ts(x)
- /src/node_modules/b.d.ts
- /src/node_modules/b/package.json
  - If the types property of package.json points to a file, look for that file
- /src/node_modules/@types/b.d.ts
- /src/node_modules/b/index.ts(x)
- /src/node_modules/b/index.d.ts
- ...recursion

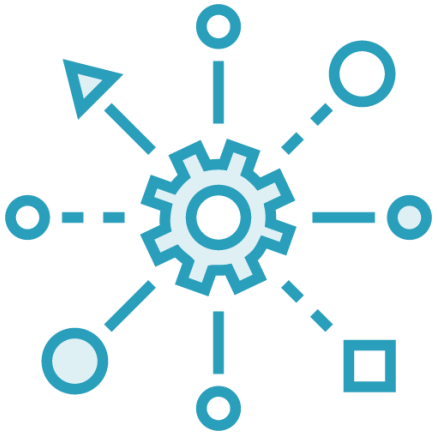
## TypeScript module resolution extras

- Looks for declaration files (.d.ts)
- Uses the types property in package.json
- Looks in node_modules/@types



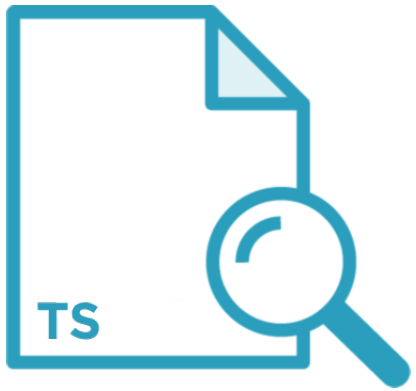
Next up, module resolution tracing

# Module Resolution Tracing

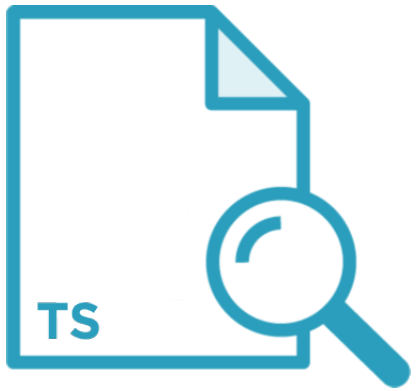


**TypeScript compilation is a multi-step process**

- Create list of modules to include
- Compile the files in the list

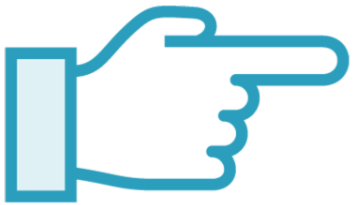



TypeScript tries to resolve modules from numerous locations



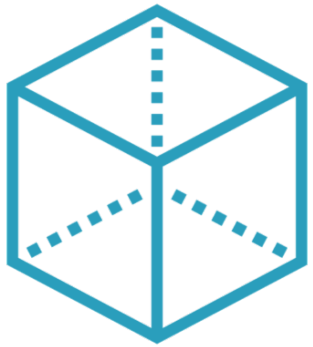

**A different approach is useful in different scenarios**

- Command-line option is quick and doesn't require a code-change
- Configuration option can be useful to understand application dependencies

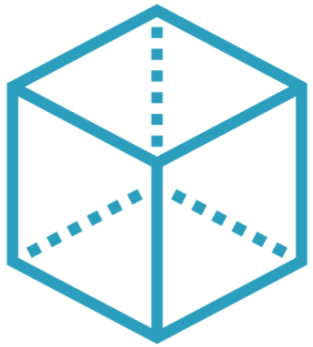


Next up, understanding the baseUrl

# Understanding baseUrl



The baseUrl configuration is only used for non-relative imports



The baseUrl configuration can be set in tsconfig or as a command-line flag

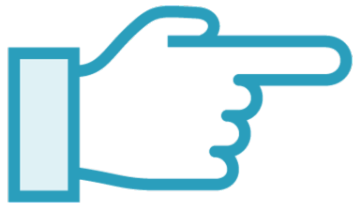
The behaviour of each does vary!

When we use the command line, the path for baseUrl is relative to the directory the command line is running in


When we use the tsconfig file, the path for baseUrl is relative to the tsconfig file itself

Real projects have more complex directory structures – usually using the command-line or the tsconfig file will behave differently

The command-line takes precedence!



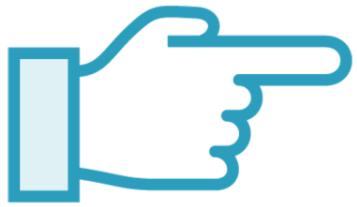
Next up, using the paths configuration

# Using the Paths Configuration

Path mapping can simplify imports, or reach modules outside of the baseUrl

baseUrl must be configured to use path mapping, and paths are always relative to the baseUrl

Path mapping is configured using the paths configuration option in tsconfig, never with a command-line flag



Next up, virtual directories

# Using Virtual Directories

Virtual directory – a directory that is the result of combining other source directories at run-time

The directories that make up a virtual directory are called root directories

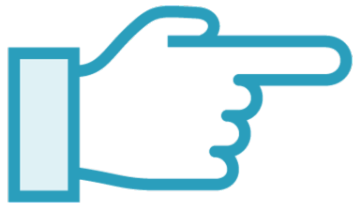
The rootDirs option is used to tell TypeScript about the root directories that we expect to be merged together

Real-world projects usually have much more capable build processes

# Summary

**In this module we:**

- Learned about the NodeJS module resolution strategy
- How to enable resolution tracing for extensive diagnostic information
- Resolution configuration options
  - baseUrl
  - paths
  - rootDirs



Next up, course summary