Creating Basic TypeScript Modules

Dan Wellman
LEAD UI DEVELOPER

Overview

Import and export are fundamental to TypeScript modules

Export variations in detail

Import variations in detail

Additional techniques like conditional imports and JSON imports

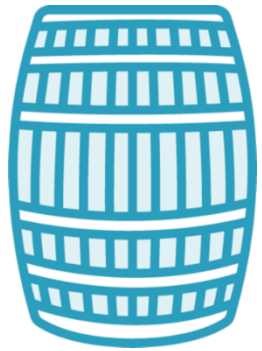
Exporting Code from a Module

Typically, modules will export values or functionality

Prefix any declaration with the export keyword

- var/let/const
- function()
- class

Barrel Files

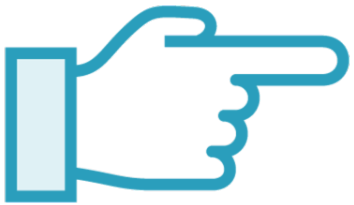


Useful for simplifying imports

Barrel Files

Useful for simplifying imports

Watch out for circular imports!



Next up, default exports

Default Exports

Default exports differ from named exports in several ways, and use a different import syntax

Default Exports

Can exist next to other exports

Default Exports

Can exist next to other exports

Supported by all module loaders

Default Exports

Can exist next to other exports

Supported by all module loaders

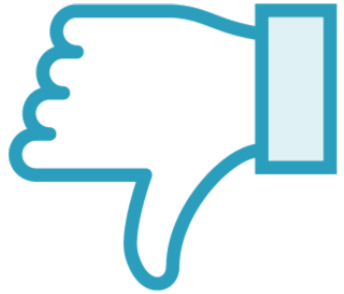
Recommended by TypeScript

Default Exports

Can exist next to other exports

Supported by all module loaders

Recommended by TypeScript



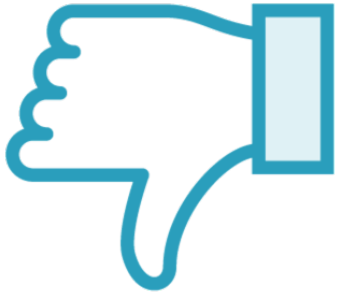
Mainly useful for libraries/frameworks

Default Exports

Can exist next to other exports

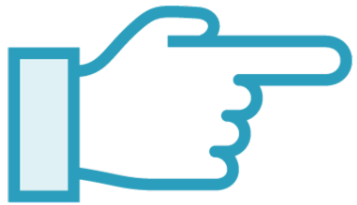
Supported by all module loaders

Recommended by TypeScript



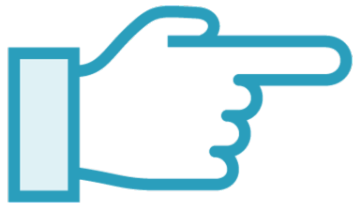
Mainly useful for libraries/frameworks

Poor editor support/discoverability



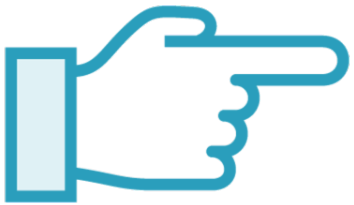
Next up, imports

Importing Code into a Module



Next up, default imports

Default Imports



Next up, optionally loading modules

Optionally Loading Modules

```
import { something } from ‘./somewhere’;
```

Imports must be at the top level of a file

```
if (condition) {
  import { something } from ‘./somewhere’; // error
}
```

Imports must be at the top level of a file

Cannot be nested!

```
import('./somewhere');
```

Import expressions can also be used

```
if (condition) {
  import('./somewhere'); // works!
}
```

Imports must be at the top level of a file

Can be nested!

```
if (condition) {
  import('./somewhere'); // works!
}
```

Imports must be at the top level of a file

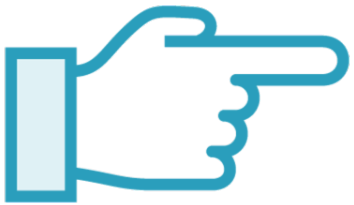
Can be nested!

Must be used with the ESNext as the module target

ESNext is a placeholder for future ECMAScript features

Doesn’t support import()

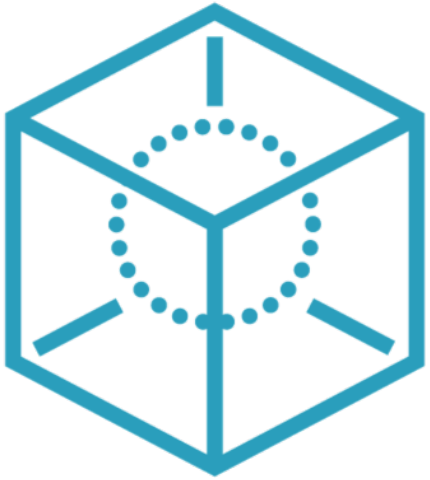
Does support import()



Next up, loading other file types

Loading JSON Files

Only JavaScript files can be modules in ES Modules!



Some 3rd party module loaders can allow JSON files to be loaded as modules

Summary

Creating basic modules with import and export

Any declaration can be exported

- Named exports
- Renamed exports
- Wildcard exports
- Export statements
- Barrel files
- Default exports

Summary

Creating basic modules with import and export

Different types of import

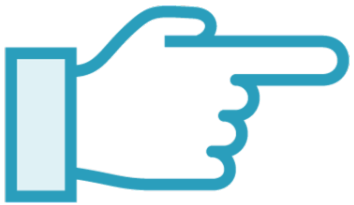
- Import/from keywords
- Renamed imports
- Wildcard/barrel imports
- Default imports
- require()

Summary

Creating basic modules with import and export

Optional imports using import()

JSON imports



Next up, ambient modules