JavaScript Modules

What is a module?

Any of a number of distinct but interrelated units from which a program may be built up or into which a complex activity may be analysed.

– Oxford English Dictionary

tl;dr: **A unit of code.**

Why would I use modules?

Modules help us to:

- write maintainable and testable code.
- write reusable code.

Modules

Like a good author will divide a book into chapters; good programmers split a program into modules. -- Somebody, probably

Modules in JavaScript

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION: THERE ARE 14 COMPETING STANDARDS.

14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.

YEAH!

SOON:

SITUATION: THERE ARE 15 COMPETING STANDARDS.

CommonJS Modules

This is the most widely used method of defining modules, used in node and `npm` packages, but it doesn't work well in the browser.

Asynchronous Module Definition (AMD)

More complicated to use, but designed to work well in the browser with a loading library, e.g. `require.js`.

ECMAScript 6 Modules

An attempt at supporting the best of both worlds:

- A compact, simple syntax
- Asynchronous loading for browsers

Can I use ES6 Modules today?

The short answer? Kind of.

What does a module look like?

```typescript
// lib/randomInteger.ts
function randomInteger(min: number, max: number) {
  min = Math.ceil(min)
  max = Math.floor(max)

  return Math.floor(Math.random() * (max - min)) + min
}

export default randomInteger
```

```typescript
// main.ts
import randomInteger from './lib/randomInteger'

const role = randomInteger(0, 6) + 1
console.log(`You just rolled a ${role}!`)
```

Modules can export more than one thing

```
// util.ts
export function squareRoot(number: number) {
  return Math.sqrt(number)
}

export function square(x: number) {
  return x * x
}

export function diagonalLength(x: number, y: number) {
  return squareRoot(square(x) + square(y))
}
```

Importing from a module by name

```
// main.ts
import { diagonalLength } from './util'

console.log(diagonalLength(4, 3)) // -> 5
```

Renaming a named export!

```
// main.ts
import { diagonalLength as pythagoreanLength } from './util'

console.log(pythagoreanLength(4, 3)) // -> 5
```

Importing the default *and* something by name

```
import React, { useState } from 'react'
```

- Imports the default export and names it React
- Imports a named export named useState

See, modules are easy!

😉