

TypeScript Modules

Dan Wahlin

Twitter: @danwahlin

John Papa

Twitter: @john_papa



What's a Module?





Separated

Testable

JavaScript Modules

Maintainable

Reusable

You May Be a Module if ...

Explicitly declare a module

```
module dataservice {  
    // code  
};
```

dataservice Module

No module declaration, no exports, no imports

Global Module

Global Namespace

window

```
class TestClass implements ITest {  
    private a = 2;  
    public b = 4;  
};  
  
var t = new TestClass();
```

Module Flexibility

- **Extend modules**

- Custom modules or the global module

*Extend modules
within or across files*

- **Separation of concerns**

- Each module has a specific role

“Ravioli”

- **Open**

- Import other modules
 - Export features

*Choose what to
expose*

Internal Modules



Internal – Named Module

Named Module

```
namespace Shapes {  
    interface IRectangle {  
        height: number;  
        width: number;  
    }  
  
    class Rectangle implements IRectangle {  
        constructor (public height: number, public width: number) {  
        }  
    }  
  
    var rect: IRectangle = new Rectangle(10, 4);  
}
```

```
var myRectangle = Shapes._____
```

Inaccessible,
nothing was exported

Exporting Internal Modules

```
namespace Shapes {  
    export class Rectangle {  
        constructor (public height: number, public width: number) {  
        }  
    }  
}  
  
var myRectangle = new Shapes.Rectangle(2,4);
```

*Accessible,
Because it was exported*

Extending Internal Modules

```
namespace Shapes {
```

Export

```
  export class Rectangle {  
    constructor (public height: number, public width: number) {  
    }  
  }  
}
```

```
var rect = new Shapes.Rectangle(2,4);
```

```
namespace Shapes {
```

```
  export class Circle {  
    constructor (public radius: number) {  
    }  
  }  
}
```

Extending the
Shapes module

```
var circle = new Shapes.Circle(20);
```

Immediately-Invoked Function Expression

(Pronounced “iffy”)

```
(function () {  
    console.log("hi there");  
})();
```

IIFE

outer () disambiguates function expression from statement

can “lock in” values and save state

minimize global scope pollution and create privacy

Emitting IIFE

TypeScript

```
namespace Shapes {  
  export class Rectangle {  
    constructor (  
      public height: number,  
      public width: number) {  
    }  
  }  
}
```

```
var rect =  
  new Shapes.Rectangle(2,4);
```

Rectangle IIFE

JavaScript

```
var Shapes;  
(function (Shapes) {  
  var Rectangle = (function () {  
    function Rectangle(height, width) {  
      this.height = height;  
      this.width = width;  
    }  
    return Rectangle;  
  })();  
  Shapes.Rectangle = Rectangle;  
})(Shapes || (Shapes = {}));  
  
var rect =  
  new Shapes.Rectangle(2, 4);
```

Shapes
IIFE

Referencing Internal Modules



Separating Internal Modules

- **Modules separated across files**

*Separation is ideal
for larger projects*

- **Must load them in the proper sequence**
 - Script tags

*Can get difficult to
maintain in larger
projects*

- **Reference them**
 - `/// <reference path="shapes.ts" />`

Separation

shapes.ts

export

```
namespace Shapes {  
  export class Rectangle {  
    constructor (  
      public height: number, public width: number) {  
    }  
  }  
}
```

reference

shapemaker.ts

```
/// <reference path="shapes.ts" />  
  
namespace ShapeMaker {  
  var rect = new Shapes.Rectangle(2,4);  
}
```

Importing External Modules and Managing Large Applications



Internal and External Modules

Internal

- Namespace-like modules
- For grouping code
- No need to “import” them

External

- Separately loadable modules
- Exported entities can be imported into other modules

`import` viewmodels = `require`('./viewmodels');

- CommonJS or AMD Conventions
 - <http://requirejs.org/>

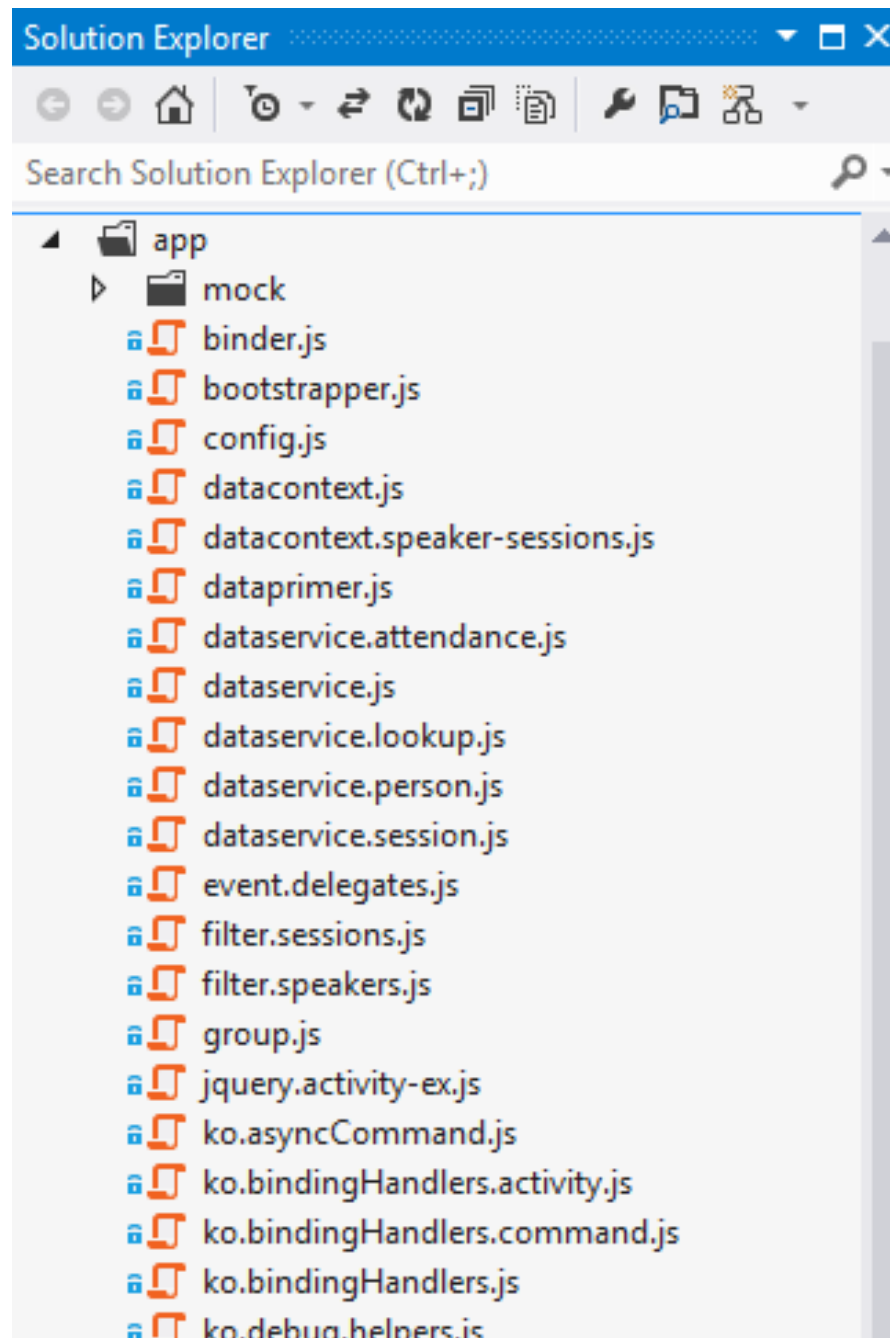
When you see
“import”, think
external module

A photograph of a cluttered child's room. In the foreground, a wooden bed frame is visible on the left. A white teddy bear sits on one of the bed's steps. To the right, a wooden desk is covered with books and other items. In the background, a bookshelf is filled with books. The floor is covered with various toys, including a blue basket, a red bag, and a small table with chairs. The room appears to be a child's bedroom or playroom.

Why?

**Sequencing script
dependencies is hard**

Many Modules



How do we
Manage
Dependencies
and Order?

AMD

- **Asynchronous Module Definition**

- Manage Dependencies
- Loads them asynchronously

- **Loads modules in sequence**

- Based on defined dependencies
- Who requires who ?

- **require.js**

*Learn More about
Require.js in my course
Single Page Apps*

*SPA Basics:
Separating the
Ravioli*

Loading Module Dependencies with Require.js

```
require(['bootstrapper'],  
        (bootstrapper) => {  
            bootstrapper.run();  
        });
```

main.ts

```
import gt = require('./greeter');  
  
export function run() {  
    var el = document.getElementById('content');  
    var greeter = new gt.Greeter(el);  
    greeter.start();  
}
```

bootstrapper.ts

```
export class Greeter {  
    start() {  
        this.timerToken = setInterval(() =>  
  
this.span.innerText = new Date().toUTCString(), 500);  
    }  
}
```

greeter.ts

Recap



TypeScript Modules

■ Modules

- Why? More maintainable and re-usable for large projects
- Extendable
- Control accessibility
- Organize your code across multiple files
- More maintainable for large projects

■ Internal Modules

- Development time references for the tools and type checking
- Must sequence `<script>` tags properly

■ External Modules

- Modules that use the CommonJS or AMD conventions
- Dependency resolution using `require.js` (<http://requirejs.org>)