Angular 2 Architecture

Course Map: Day 1

1. ASP.NET Core Architecture
2. Getting Started with ASP.NET Core
3. Design Patterns, Unit Testing
4. Entity Framework Core

Course Map: Day 2

5. Introduction to TypeScript

6. Using VS Code for TypeScript

7. **Angular 2 Architecture**

8. Using Angular CLI for Client Apps

Agenda

- Evolution of Web App Architecture
- Model-View-ViewModel Pattern
- Modules, templates, components
- Services, metadata
- Root module and component
- Platform bootstrapper

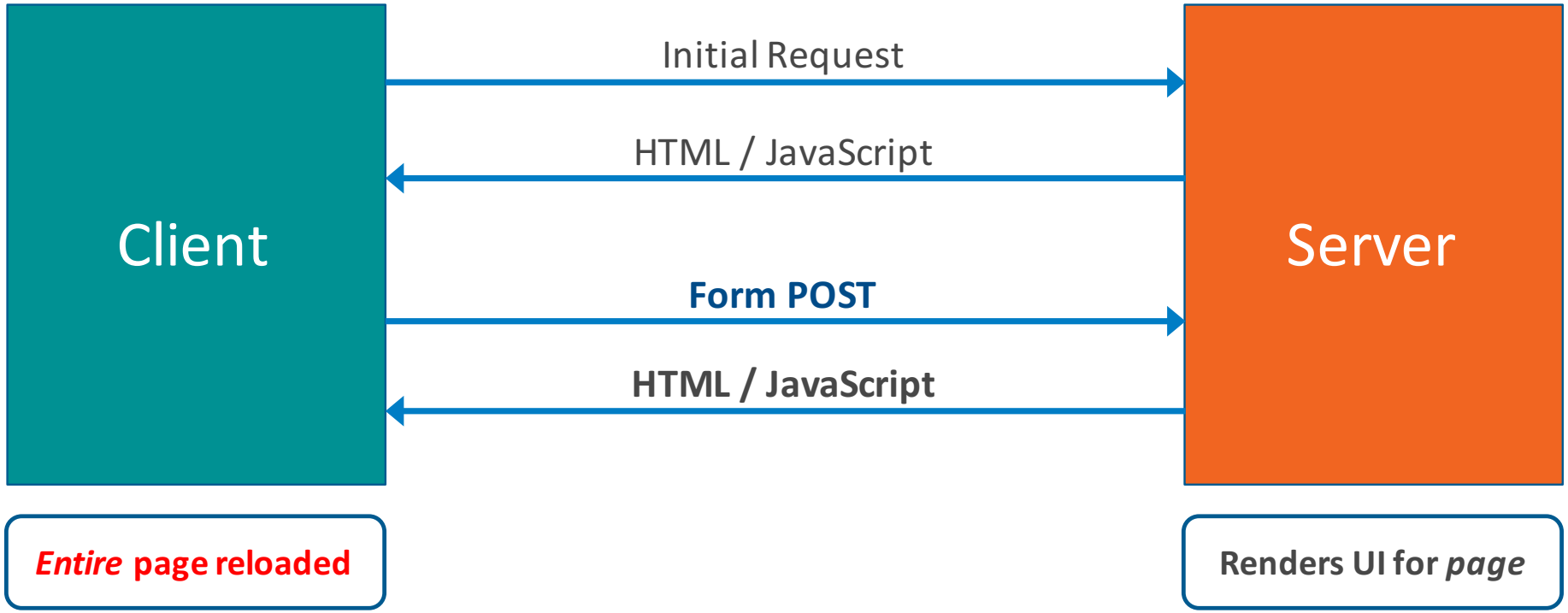
Get the Bits

github.com / tonysneed /

Kliant.AspNetCore-Angular

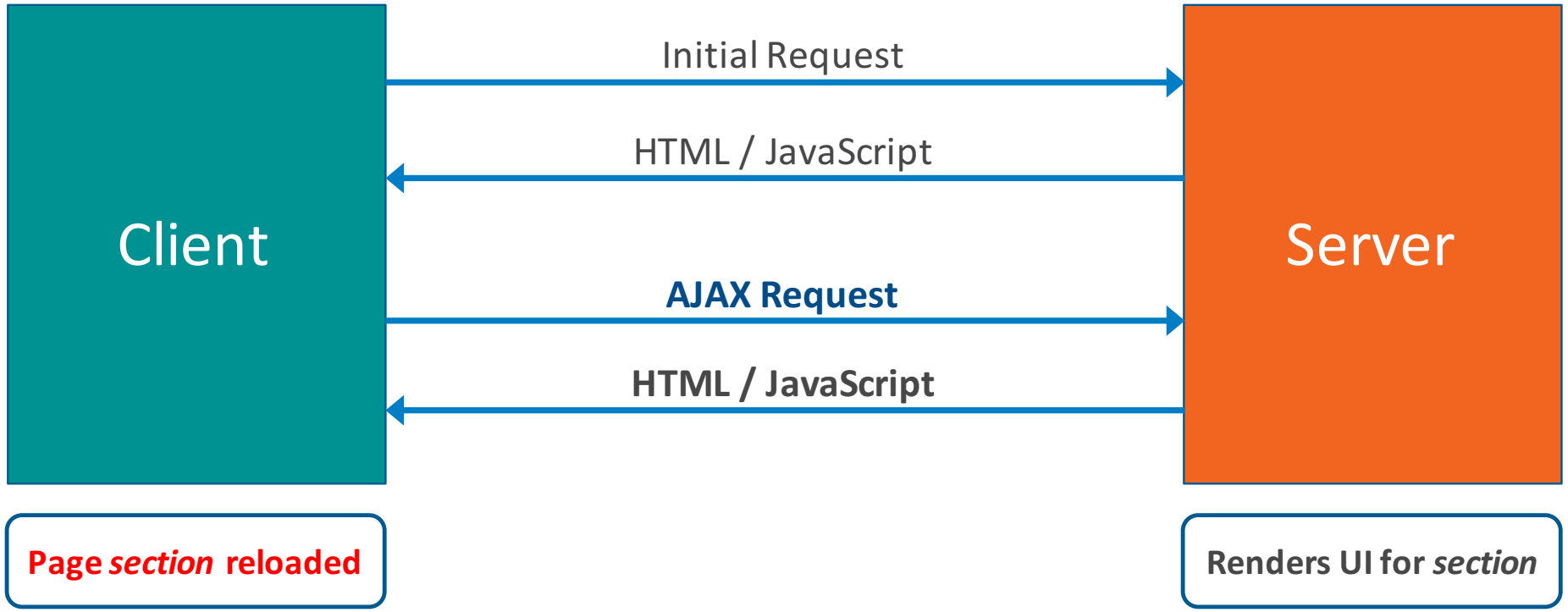
Evolution of Web App Architecture

Traditional Web Apps



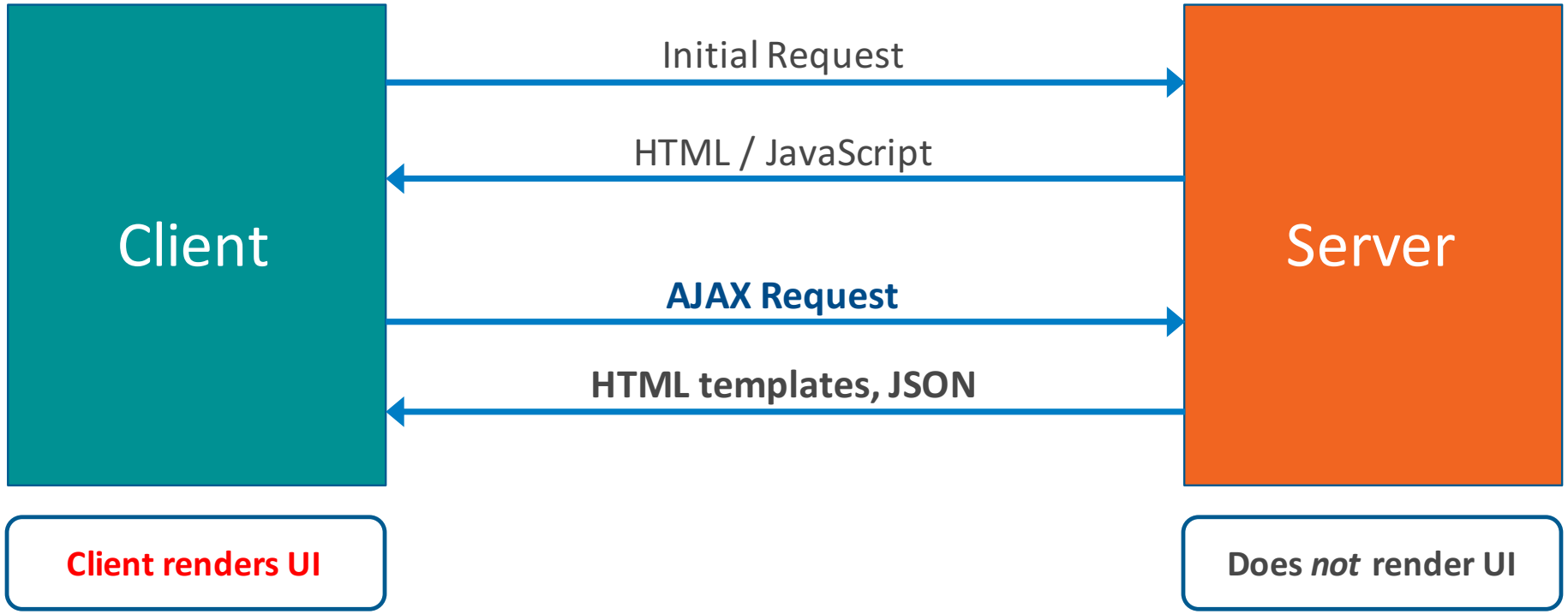
Evolution of Web App Architecture

AJAX Web Apps



Evolution of Web App Architecture

SPA Web Apps



Building Modern Web Apps Is Hard

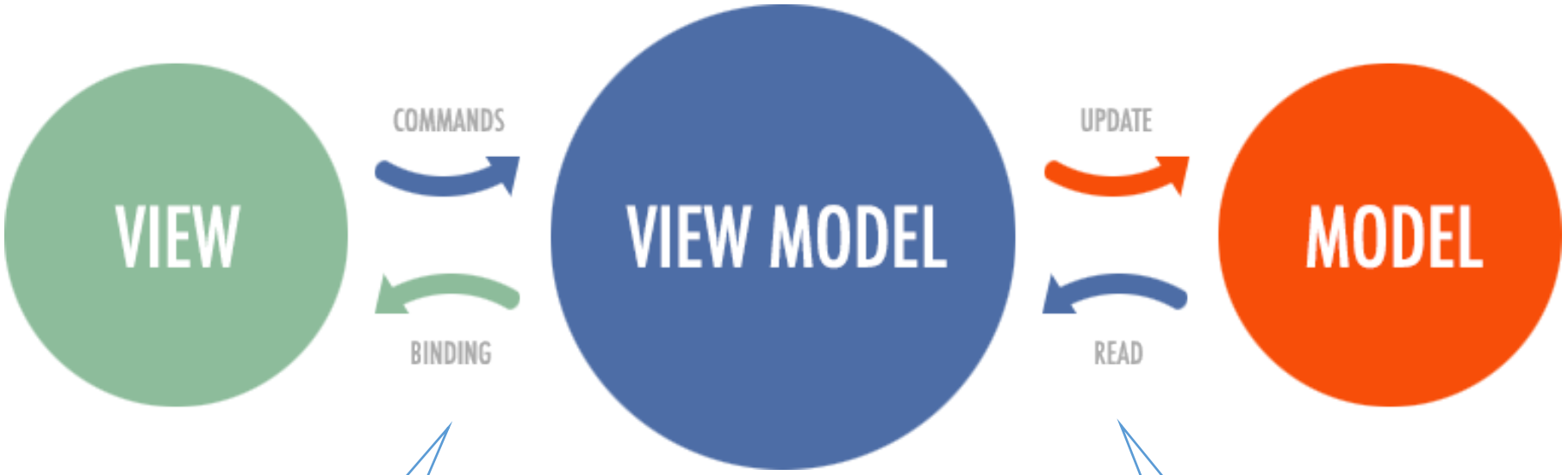
Do I have to reinvent the wheel again?

Modern SPA Frameworks

- Promote established **patterns**
  - Data Binding, Dependency Injection, MV-*
- Modularity with **web components**
- Support for **multiple platforms**
  - Web, mobile, desktop

Modern SPA Frameworks

Model-View-ViewModel



Events trigger commands, elements bind to properties

ViewModel uses a service to retrieve and update models

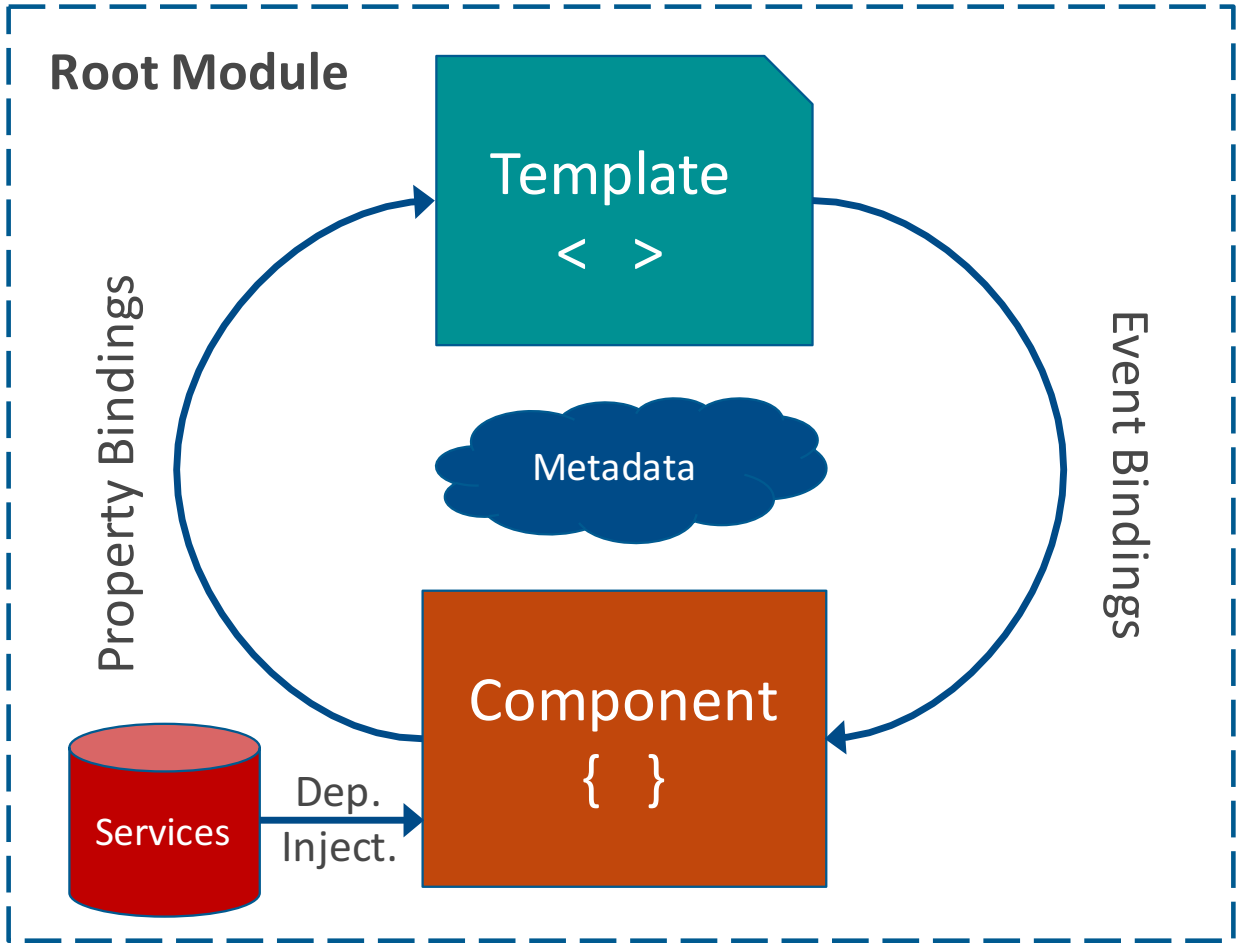
Introducing Angular

- Open source
- Second generation
- Authored by Google
- Faster than v1
- Rich set of features
- Command-line tools
- Leverages TypeScript

https://angular.io

M-V-VM: The Angular Way

- **Templates**: HTML Views
- **Components**: ViewModels
- **Services**: Injected into VM
- **Metadata**: Glues things together



Angular From Scratch

1. Install **NPM packages** required by Angular
2. Set **TypeScript** compiler options
3. Configure **SystemJS** module loader
4. Main web page: **index.html**
5. Root app **component**
6. Root app **template**
7. Root app **module**
8. Platform **bootstrapper**

Angular Dependencies

```
npm install --save
        @angular/common
        @angular/compiler
        @angular/core
        @angular/forms
        @angular/http
        @angular/platform-browser
        @angular/platform-browser-dynamic
        @angular/router
        @angular/upgrade
        core-js reflect-metadata
        rxjs systemjs zone.js
```

TypeScript Compiler Options

```
{
    "compilerOptions": {
        "target": “es5", "module": "commonjs",
        "moduleResolution": "node",
        "emitDecoratorMetadata": true,
        "experimentalDecorators": true,
        "sourceMap": true,
        "removeComments": true,
        "noImplicitAny": true,
        "lib": ["es2015","es5",
            "dom", scripthost" ]    }
}
```

SystemJS Module Loading

```
(function (global) {  // systemjs.config.js
  System.config({
    map: { app: 'app', // Our app is in the app folder
      // Angular bundles (several omitted for clarity)
      '@angular/core': 'npm:@angular/core/bundles/core.umd.js',
      '@angular/common': 'npm:@angular/common/bundles/common.umd.js',
      '@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',
      'rxjs': 'npm:rxjs', // Other dependencies },
    packages: {  // Tell system loader how to load without extensions
      app: { main: './main.js', defaultExtension: 'js' },
    } });
})(this);
```

Host Web Page: Index.html

```
<head>
   <!- Polyfills for older browsers, dependencies, systemjs -->
   <script src="node_modules/core-js/client/shim.min.js"></script>
   <script src="node_modules/zone.js/dist/zone.js"></script>
   <script src="node_modules/reflect-metadata/Reflect.js"></script>
   <script src="node_modules/systemjs/dist/system.src.js"></script>
   <script src="systemjs.config.js"></script> <!- systemjs config -->
   <script>  <!-- Instruct systemjs to load app modules -->
     System.import('app').catch(function(err){ console.error(err); });
   </script>
</head>
<body>
   <my-app><h3>Loading ...</h3></my-app> <!-- app root component -->
</body>
```

Root App Component

```
import { Component } from “@angular/core";

@Component({
    selector: "my-app",                                  // Included on index.html
    templateUrl: “app.component.html"} )  // Path to HTML template
export class AppComponent {
    title = "Express-Angular Demo"; }          // Displayed on page
```

Root App Template

```
// app.component.html

<!-- Interpolation binding syntax -->
<h1>{{title}}</h1>
```

Root App Module

```
import { NgModule } from "@angular/core";
import { BrowserModule  } from "@angular/platform-browser";
import { HttpModule } from "@angular/http";
import { AppComponent } from "./app.component";
@NgModule({
    imports: [BrowserModule, HttpModule ], // Required Angular modules
    declarations: [AppComponent],
    providers: [/* TODO: Providers go here */],
    bootstrap: [AppComponent]})                    // Root app component
export class AppModule { }
```

Platform Bootstrapper

```
// main.ts

import { platformBrowserDynamic } from
    "@angular/platform-browser-dynamic";

import { AppModule } from "./app.module";

// Bootstrap app root module using the specified platform
platformBrowserDynamic().bootstrapModule(AppModule);
```

Demo: Angular From Scratch

Questions?