Angular Fundamentals

GETTING STARTED WITH ANGULAR

Jim Cooper
SOFTWARE CRAFTSMAN

@jimthecoop

Required Prerequisites

Basic JavaScript
app.pluralsight.com/paths/skills/javascript

Basic HTML
app.pluralsight.com/paths/skills/html5

Helpful Prerequisites

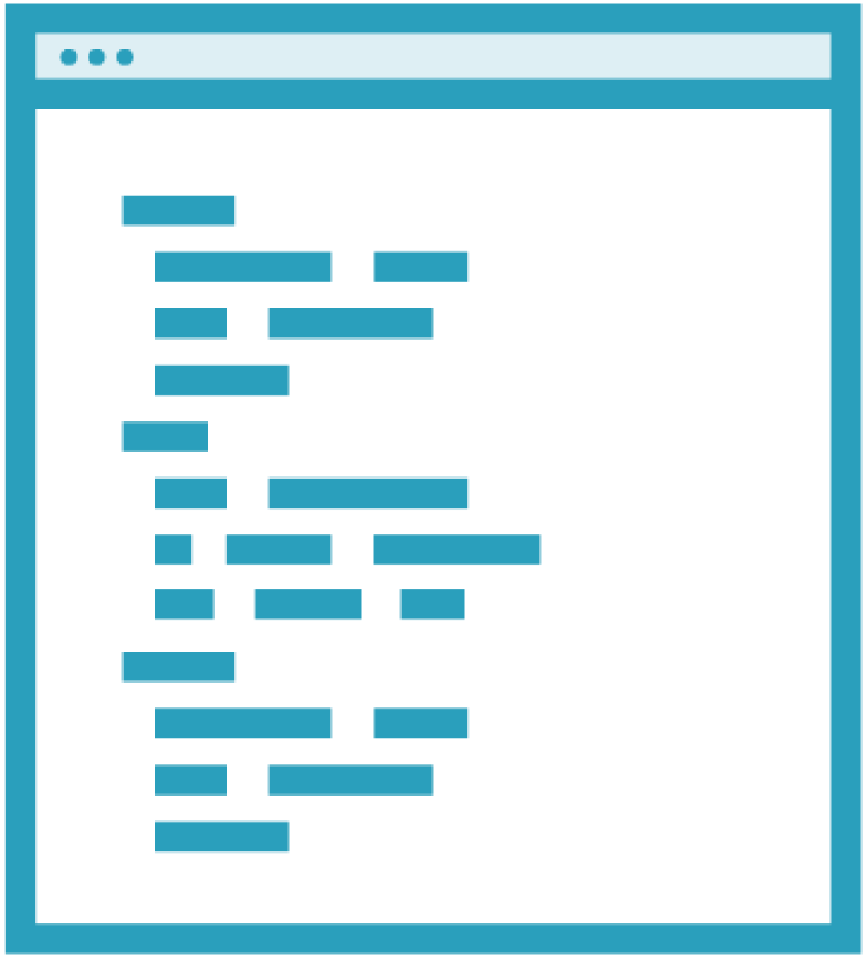
Basic Node and Npm
app.pluralsight.com/courses/npm-playbook

Modules and Module Loaders
app.pluralsight.com/courses/javascript-module-fundamentals

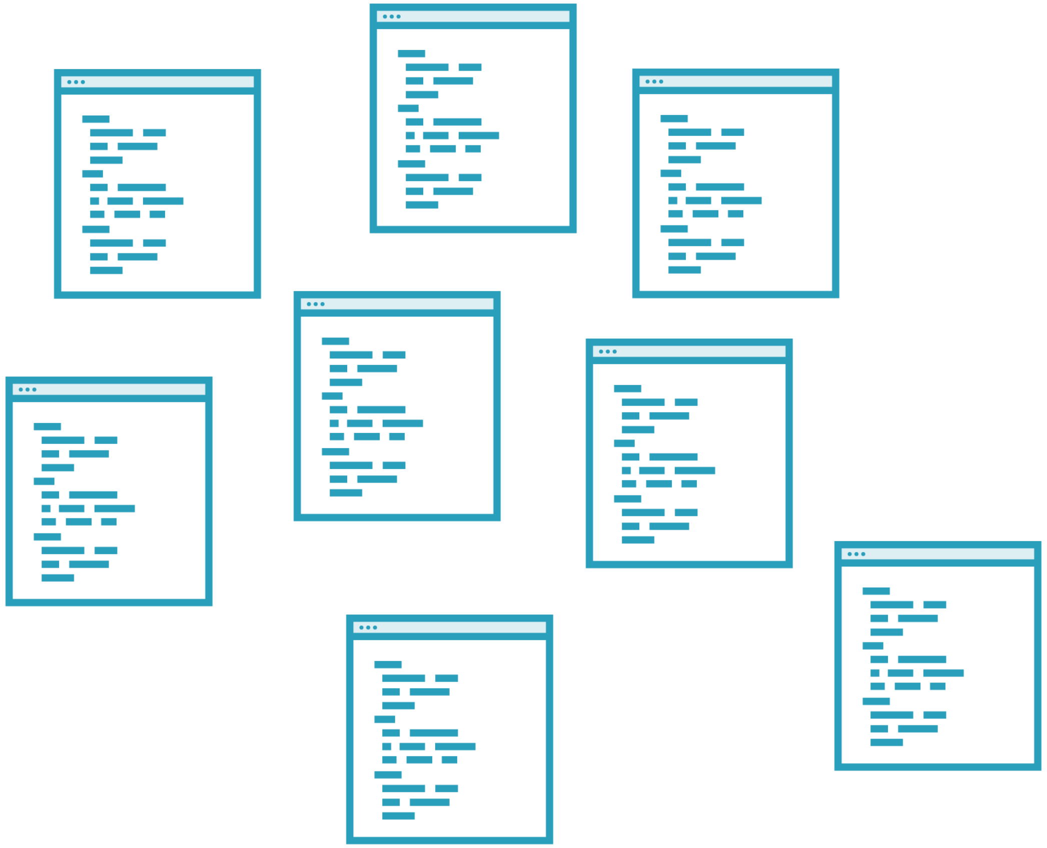
ES2015
app.pluralsight.com/courses/javascript-fundamentals-es6

TypeScript
app.pluralsight.com/courses/typescript

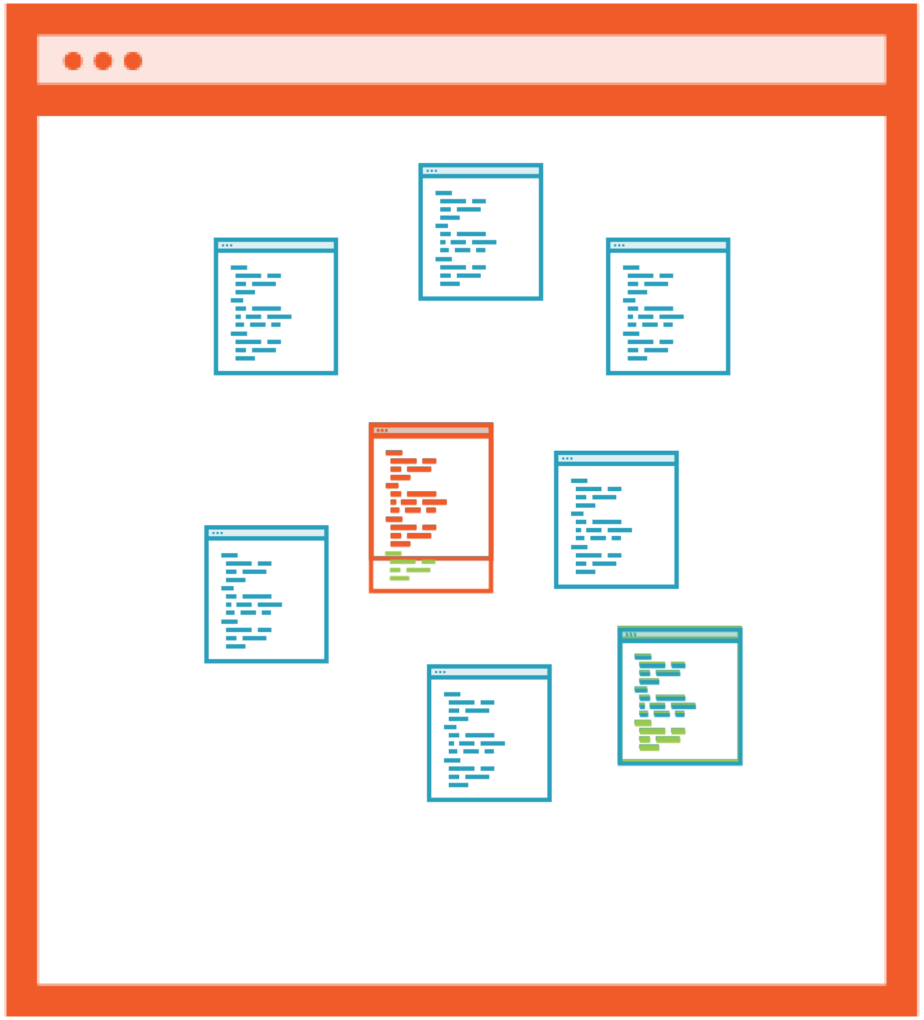
Why Modules are Important



Why Modules are Important

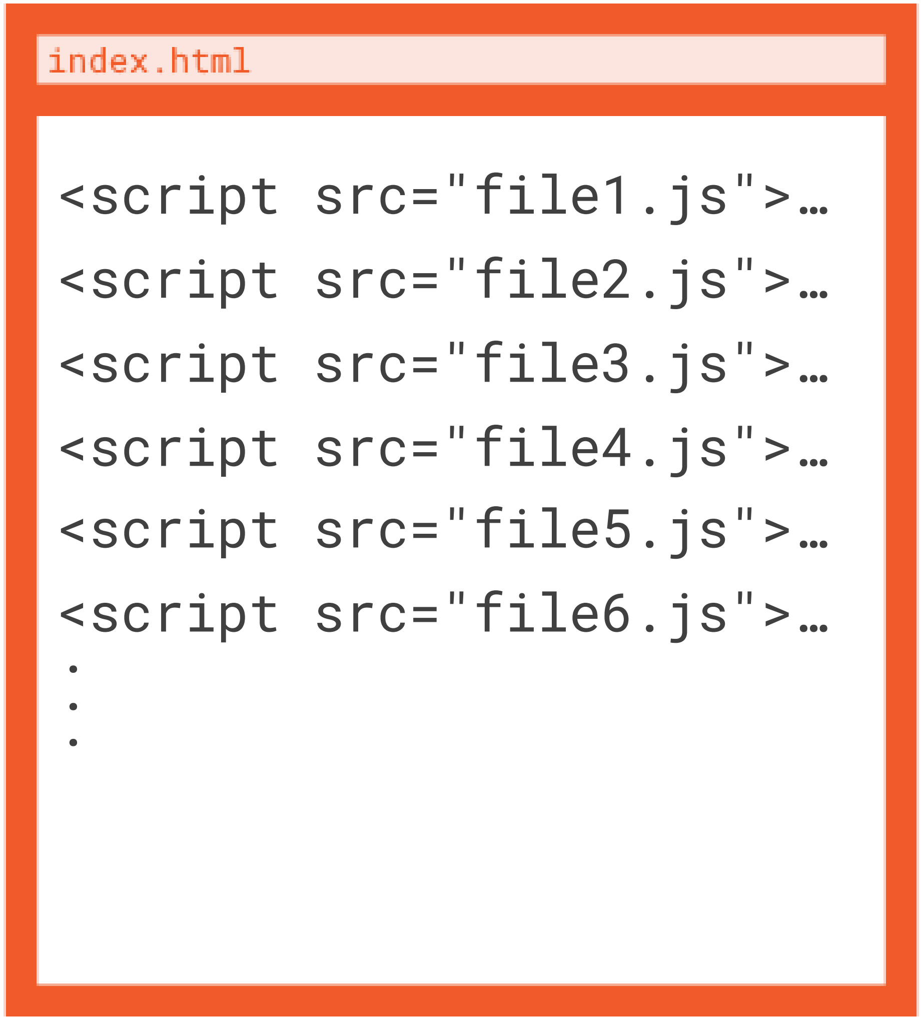


Why Modules are Important



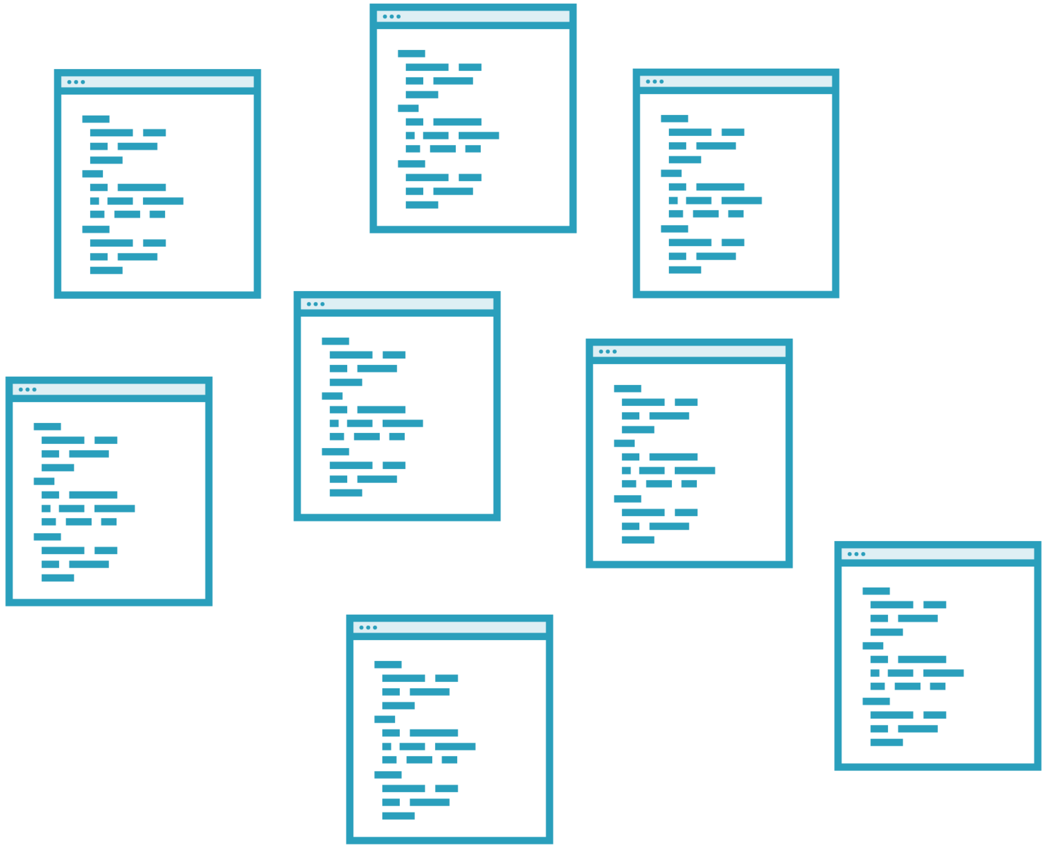
```
import { foo } from
'../folder/some-file.js'

export {
  foo: someFunction()
}
```

What is SystemJs?



index.html

```
<script src="file1.js">…
<script src="file2.js">…
<script src="file3.js">…
<script src="file4.js">…
<script src="file5.js">…
<script src="file6.js">…
.
.
.
```



What is SystemJs?

index.html

```
<script src="system.js">
<script src="config.js">
```

system.config.js

```
var config = {
 map: {
   'app': '/folder/app'
 },
 packages: {
   'app': {main: 'main.js'}
 }
}
```

ES2015 Features

`let` and `const`

Arrow Functions

Array Methods

Classes

```
function doSomething(x) {
  var y = 10
  ...



}
console.log(y) // logs undefined
```

let and const

```
function doSomething(x) {
  if (x) {
    var y = 10
  }
  console.log(y) // logs 10
}
```

let and const

```
function doSomething(x) {
  if (x) {
    let y = 10
  }
  console.log(y) // logs undefined
}
```

let and const

```
function doSomething() {
  const y = 10
  y = 20  // exception
}
```

let and const

```
function(x) {
  if (x)
    return 10
  else
    return 20
}
```

```
(x) => {
  if (x)
    return 10
  else
    return 20
}
```

Arrow Functions

```javascript
var cats = [ {name: 'Fluffy'}, {name: 'Muffin'} ]
var muffin = cats.find((cat) => {return cat.name == 'Muffin'})
//var muffin = cats.find((cat) => {return cat.name == 'Muffin'})
var muffin = cats.find(cat => cat.name == 'Muffin')
console.log(muffin.name) // logs 'Muffin'
```

Find and Filter Array Methods

```
var cats = [ {name: 'Fluffy'}, {name: 'Muffin'} ]
var cats = cats.filter(cat => cat.name.indexOf('u') > -1)
console.log(cats[0].name) // logs 'Fluffy'
console.log(cats[1].name) // logs 'Muffin'
```

Find and Filter Array Methods

```
var cat = {name: 'Fluffy', color:  'White'}
```

ES2015 Classes

```
var Cat = new function(name, color) {
  this.name = name
  this.color = color
}

var fluffy = new Cat('Fluffy', 'White')
```

ES2015 Classes

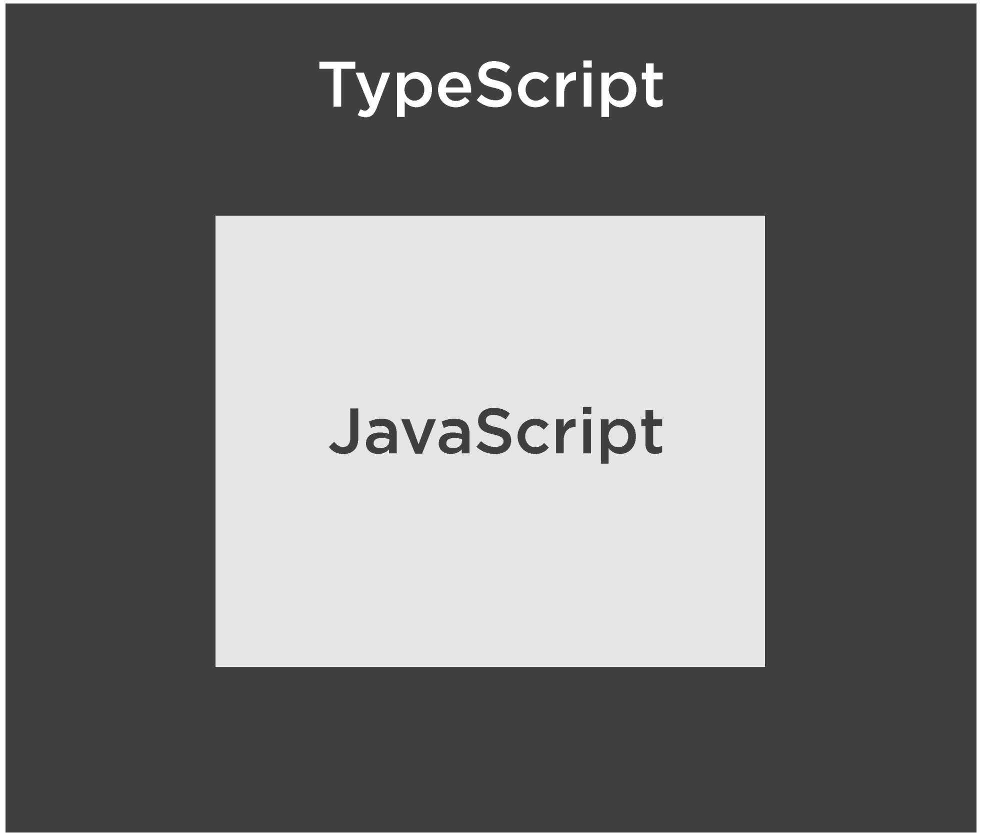
```
var Cat = new function(name, color) {
  this.name = name
  this.color = color
}
Cat.prototype.speak = function() {
  console.log('meow')
}
var fluffy = new Cat('Fluffy', 'White')
```

ES2015 Classes

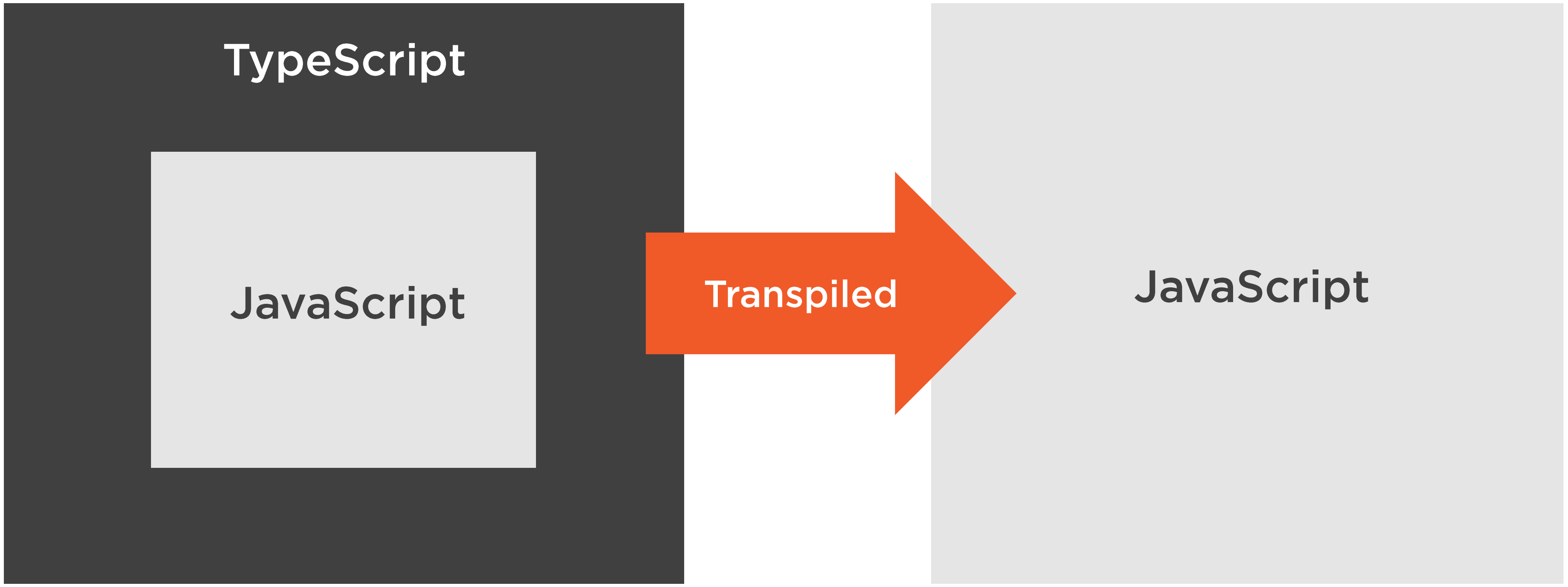
```
class Cat {
  constructor (name, color) {
    this.name = name;
    this.color = color;
  }
  speak() { console.log('meow') }
}
 var fluffy = new Cat('Fluffy', 'White')
```

ES2015 Classes

TypeScript Overview



TypeScript Overview



ES2015 Features

Static Typing

Interfaces

Class Properties

Public/Private Accessibility

```
let name : string
let age : number
let birthDate : date
```

Static Typing

```
interface ICat {
 name:string
 age:number
}
```

TypeScript Interfaces

```
interface ICat {
 name:string
 age:number
}

let fluffy:ICat
```

```
fluffy =
  {
    name: 'Fluffy',
    age: 'seven'
  }
```

TypeScript Interfaces

```
interface ICat {
 name:string
 age:number
}

let fluffy:ICat
```

```
fluffy =
  {
    name: 'Fluffy'
  }
```

TypeScript Interfaces

```
interface ICat {
 name:string
 age?:number
}

let fluffy:ICat
```

```
fluffy =
  {
    name: 'Fluffy'
  }
```

TypeScript Interfaces

```
class Cat {
  constructor (name) {
    this.name = name
  }
}
```

TypeScript Class Properties

```
class Cat {
  name:string
  constructor (name) {
    this.name = name;
  }
}
```

TypeScript Class Properties

```
class Cat {
  name:string
  color:string
  constructor (name) {
    this.name = name;
  }
}
```

TypeScript Class Properties

```
class Cat {
  name
  color
  constructor (name) {
    this.name = name;
  }
}
```

TypeScript Class Properties

```
class Cat {
  name:string
  color:string
  constructor (name) {
    this.name = name;
  }
  speak() { console.log('meow') }
}
```

Public and Private Accessibility

```
class Cat {
  name:string
  color:string
  constructor (name) {
    this.name = name;
  }
  speak() { console.log('My name is: ' + this.name) }
}
```

Public and Private Accessibility

```
class Cat {
  name:string
  speak() { console.log('My name is: ' + this.name) }
}

let fluffy = new Cat()
console.log(fluffy.name)
fluffy.speak()
```

Public and Private Accessibility

```
class Cat {
  private name:string
  private speak() { console.log('My name is: ' + this.name) }
}

let fluffy = new Cat()
console.log(fluffy.name)
fluffy.speak()
```

Public and Private Accessibility

```
class Cat {
  private name:string
  private speak() { console.log('My name is: ' + this.name) }
}

let fluffy = new Cat()
console.log(fluffy.name) //compile-time error
fluffy.speak() // compile-time error
```

Public and Private Accessibility

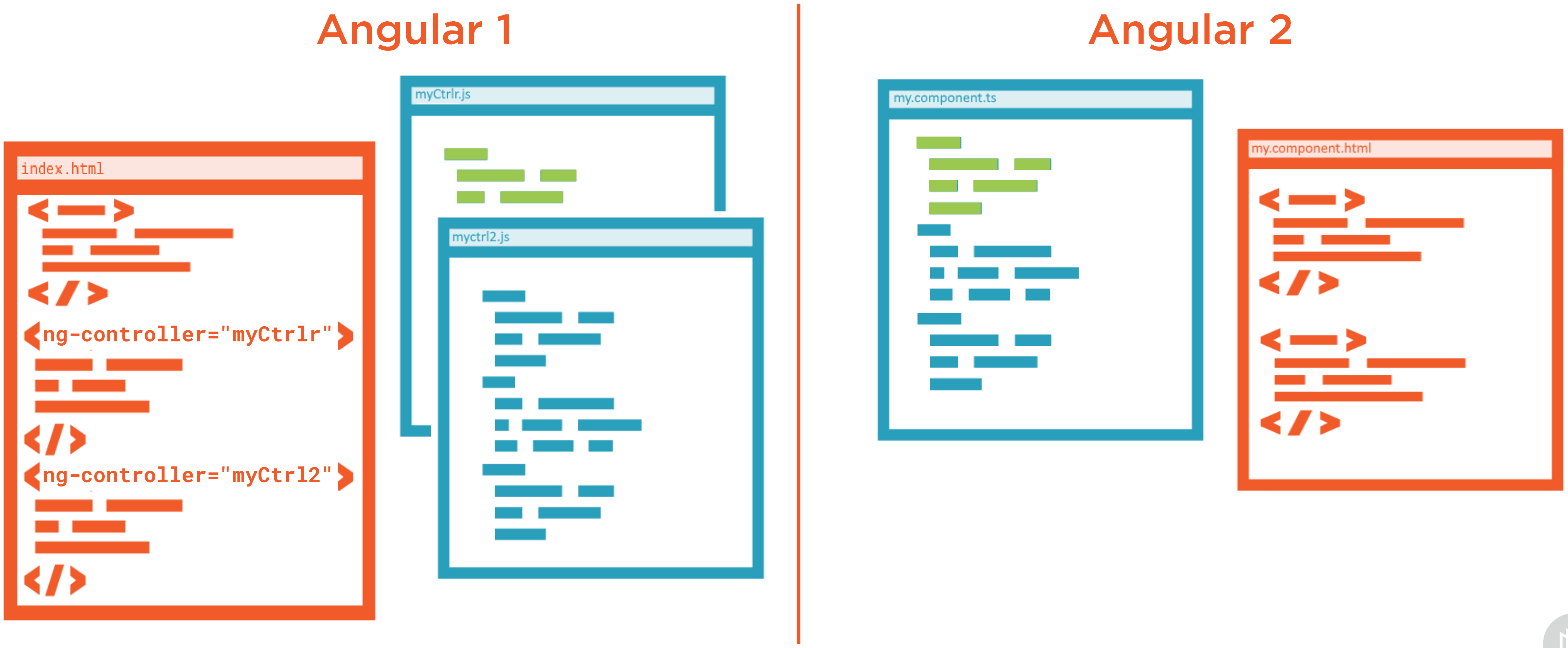
```
class Cat {
  private name:string
  private color:string
  constructor(name, color) {
    this.name = name
    this.color = color
  }
}
```

```
class Cat {
  constructor(private name, private color) {

  }
}


let fluffy = new Cat('Fluffy', 'White')
```

Public and Private Accessibility

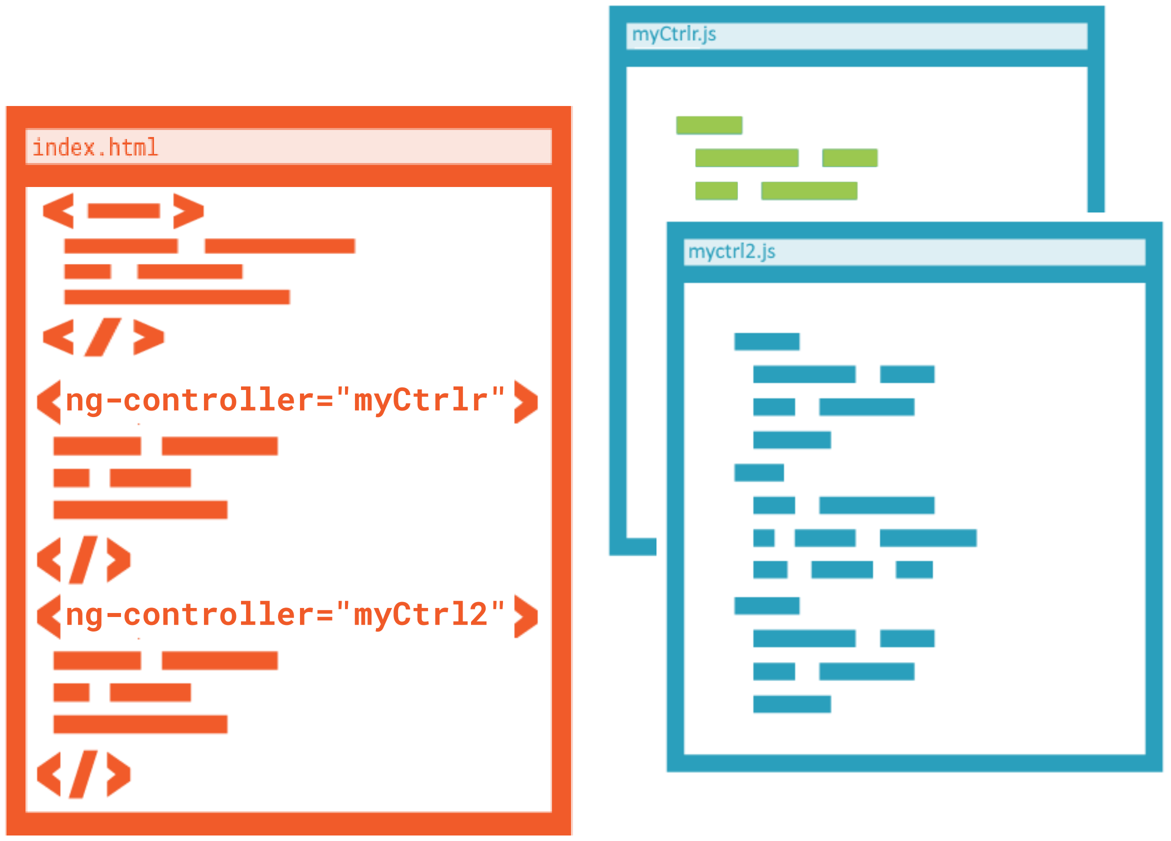
Angular 2 Conceptual Overview

MVC vs Components

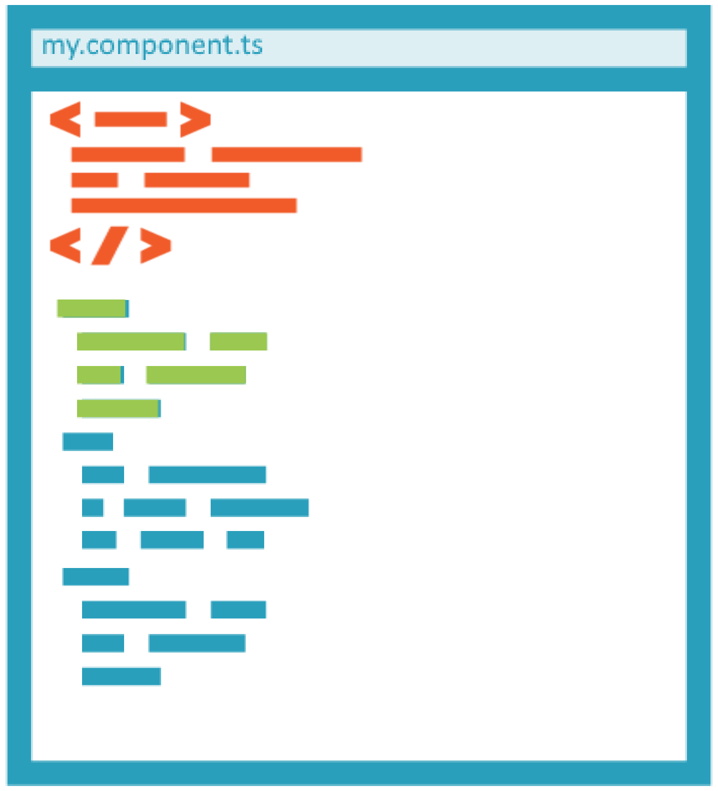


MVC vs Components

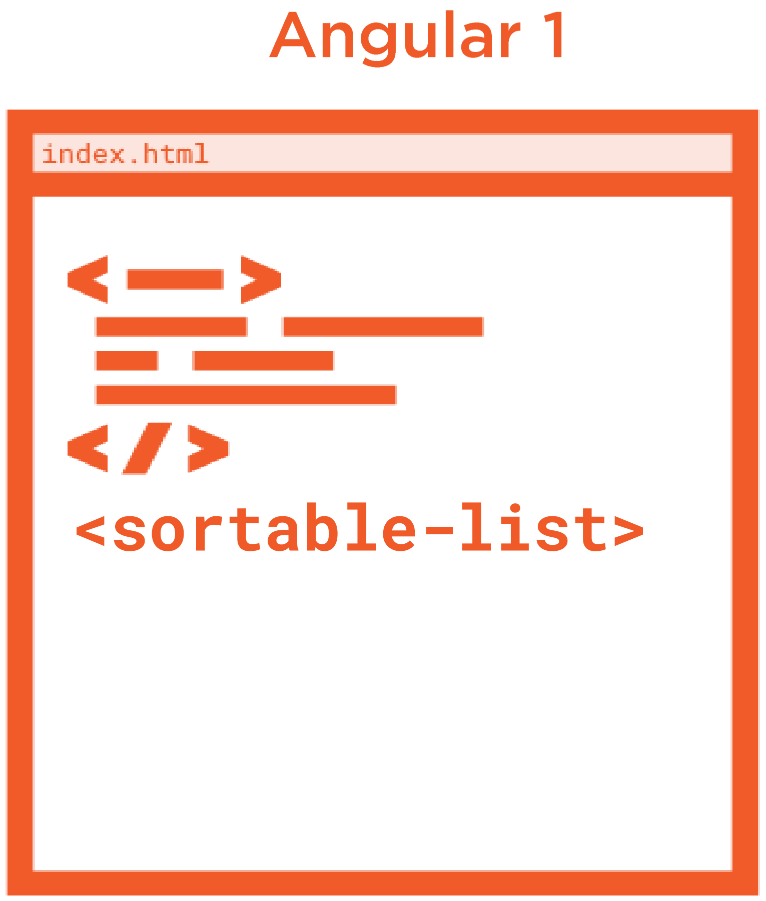
Angular 1

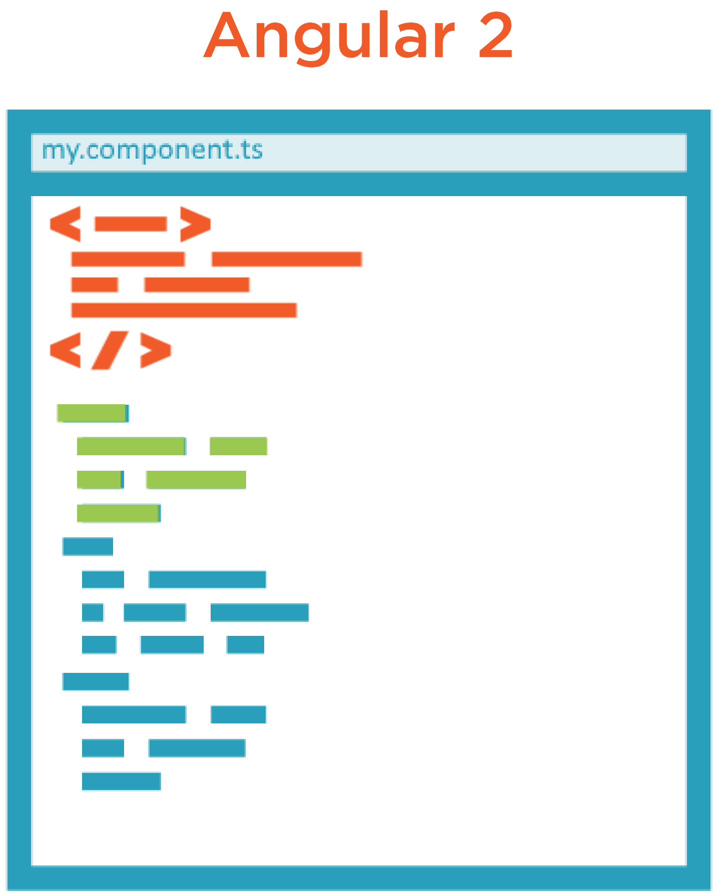


Angular 2

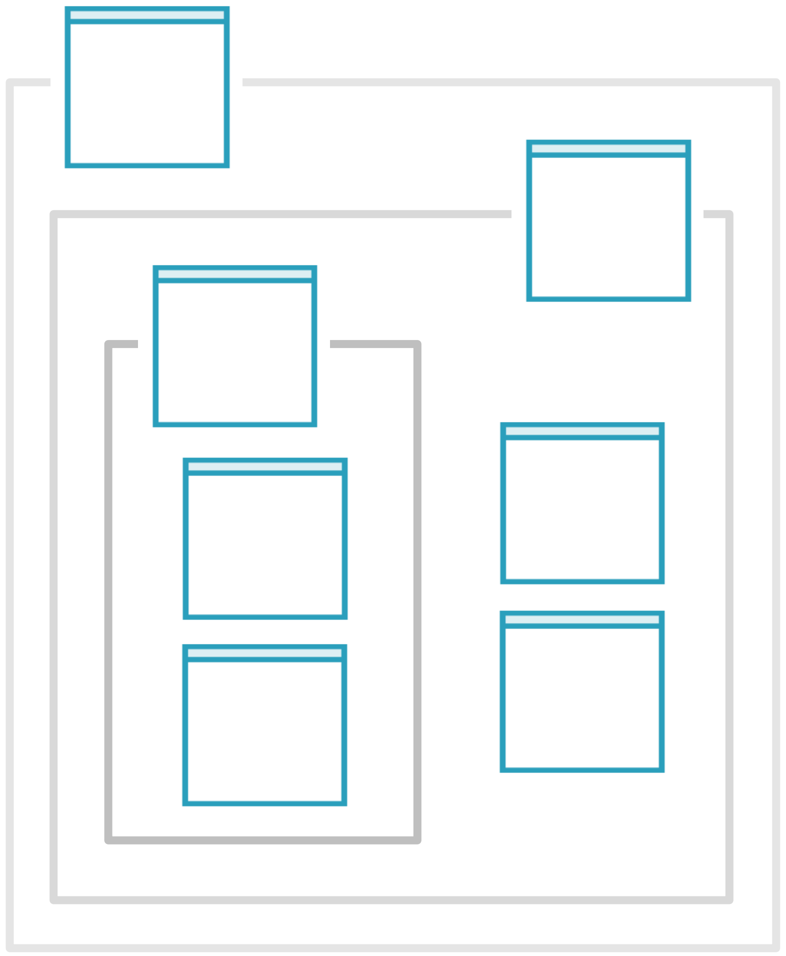


MVC vs Components

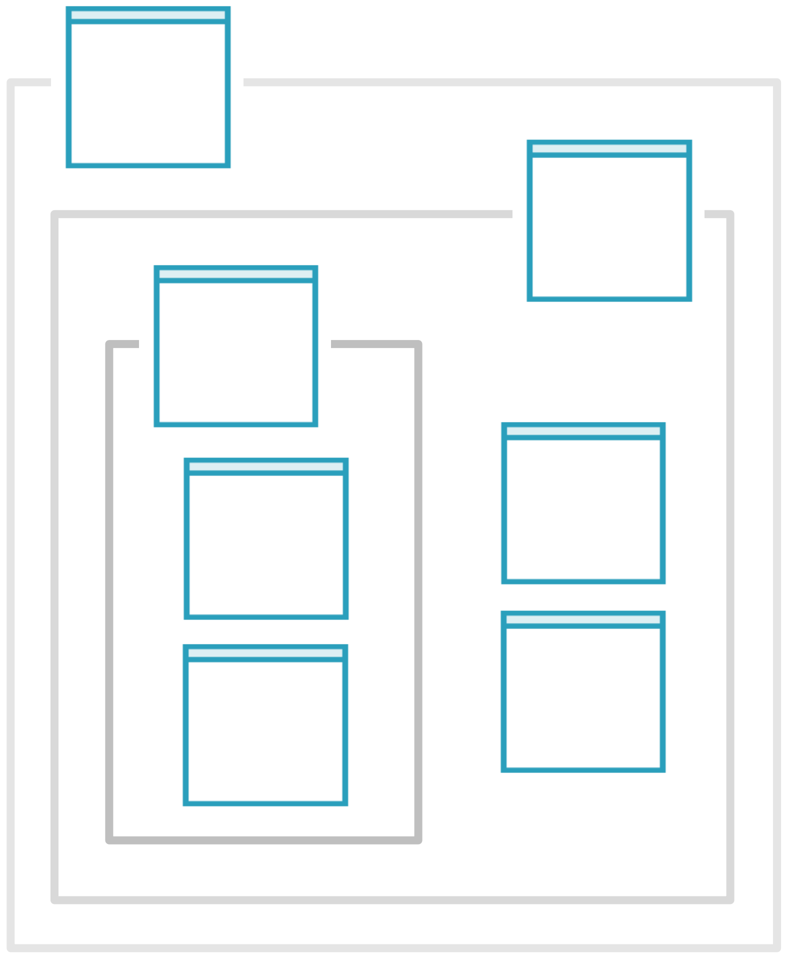




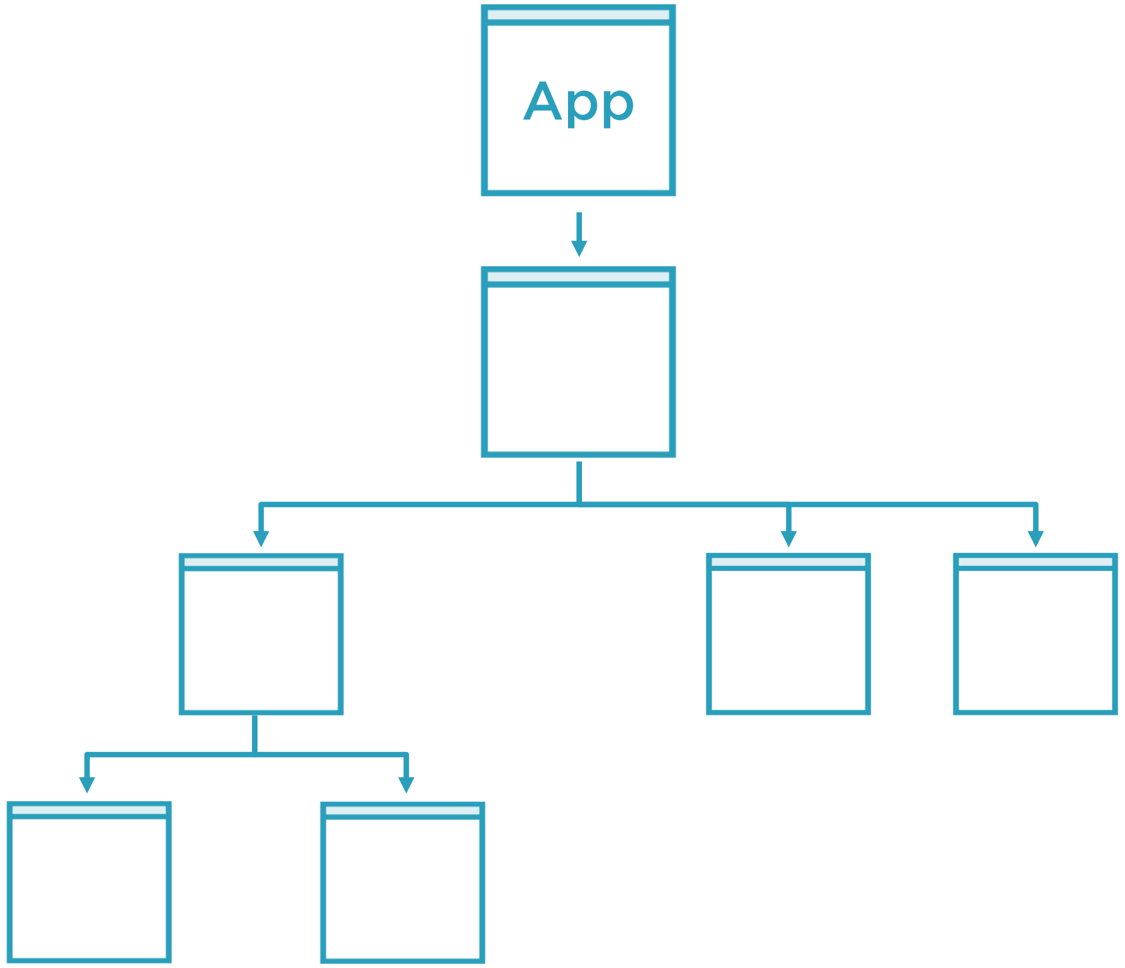
Angular 2 Component Hierarchy



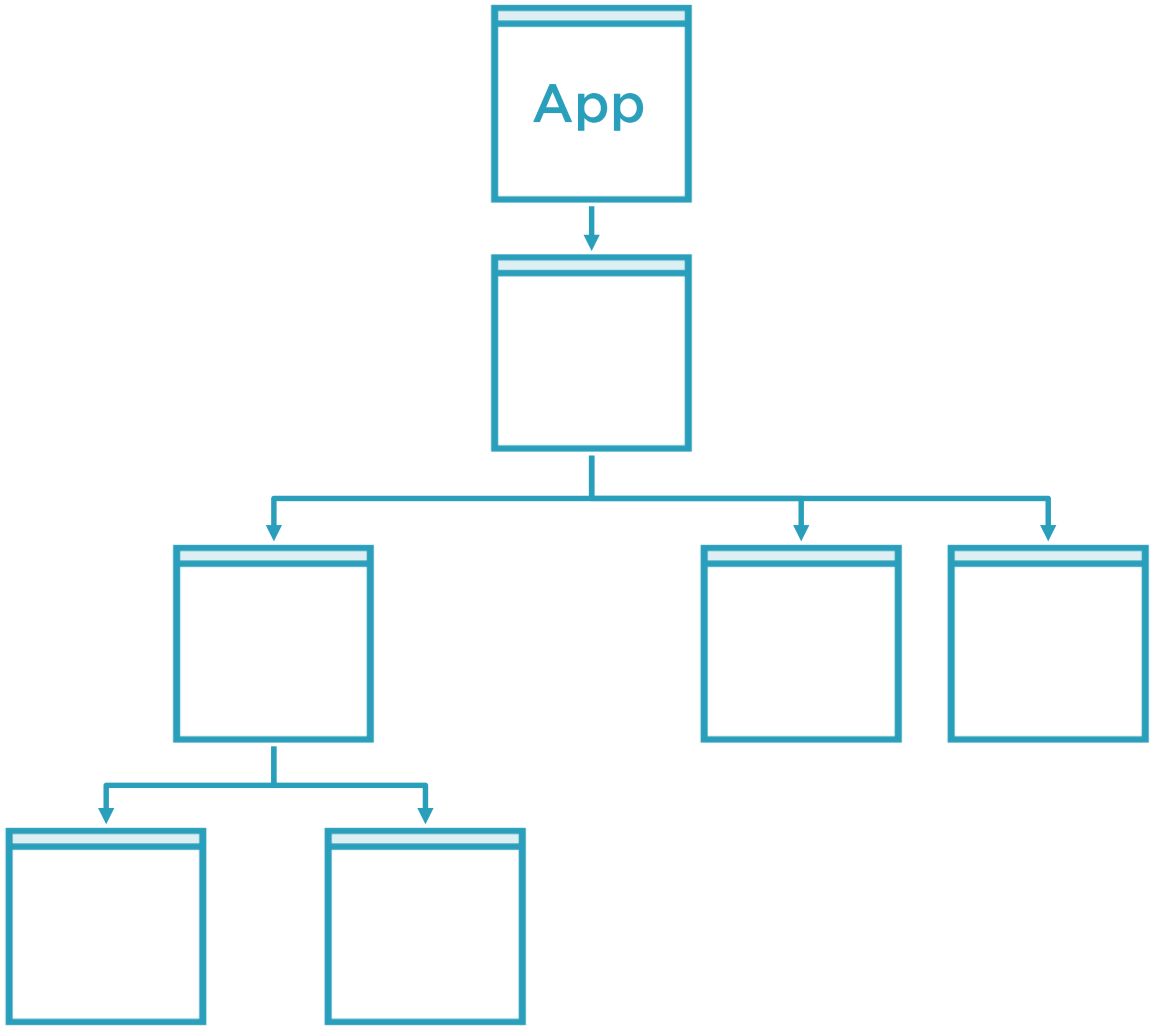
Angular 2 Component Hierarchy



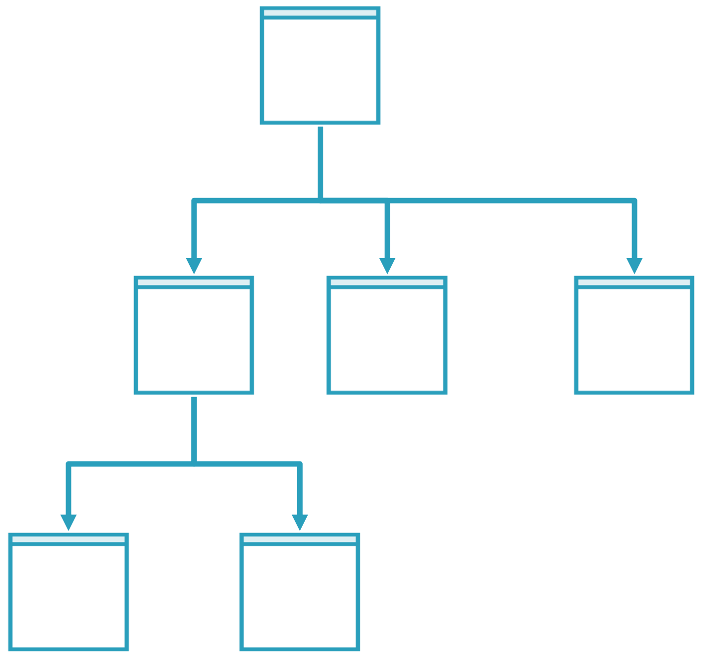
Angular 2 Component Hierarchy

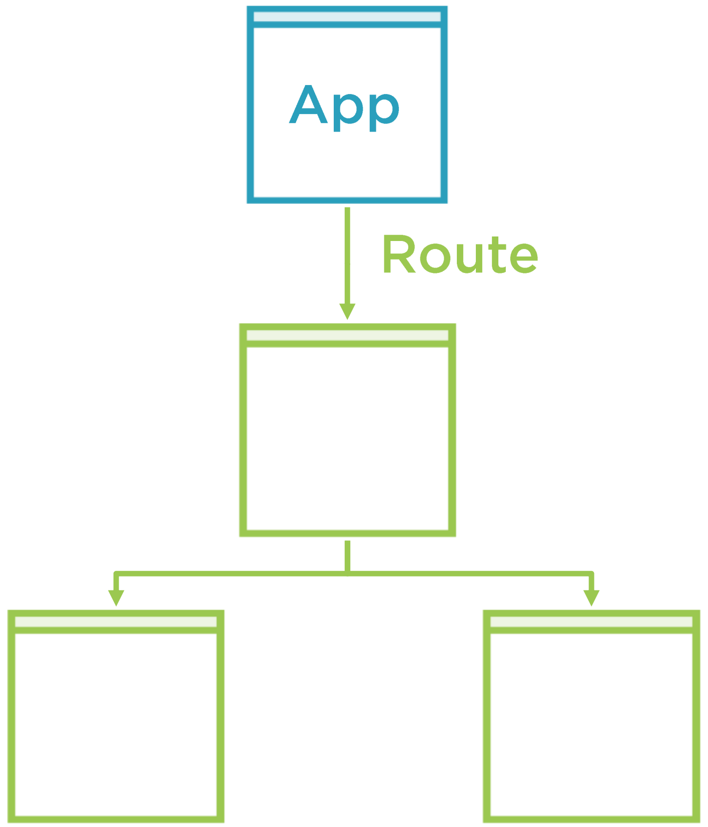


Angular 2 Component Hierarchy

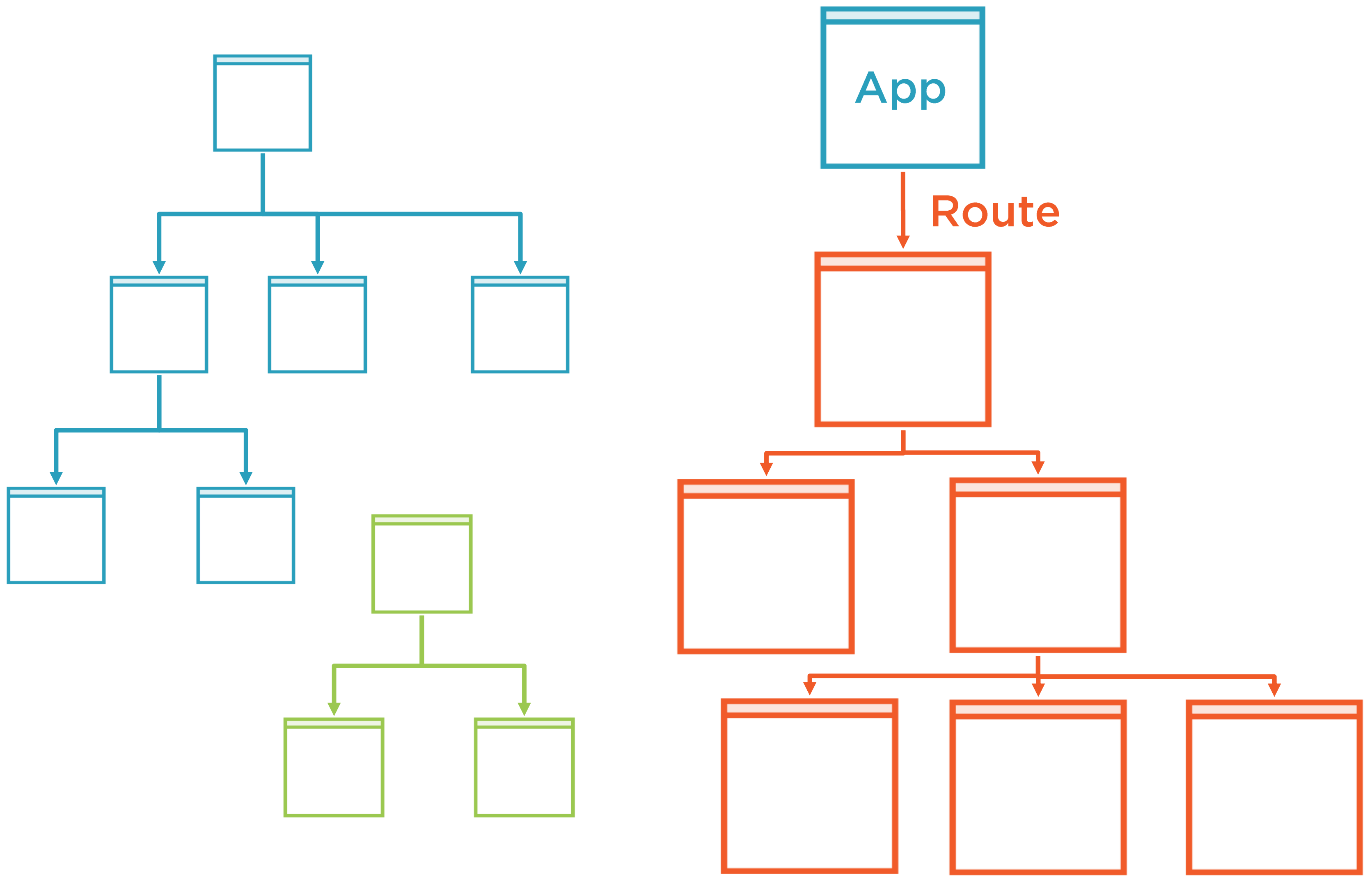


Angular 2 Component Hierarchy

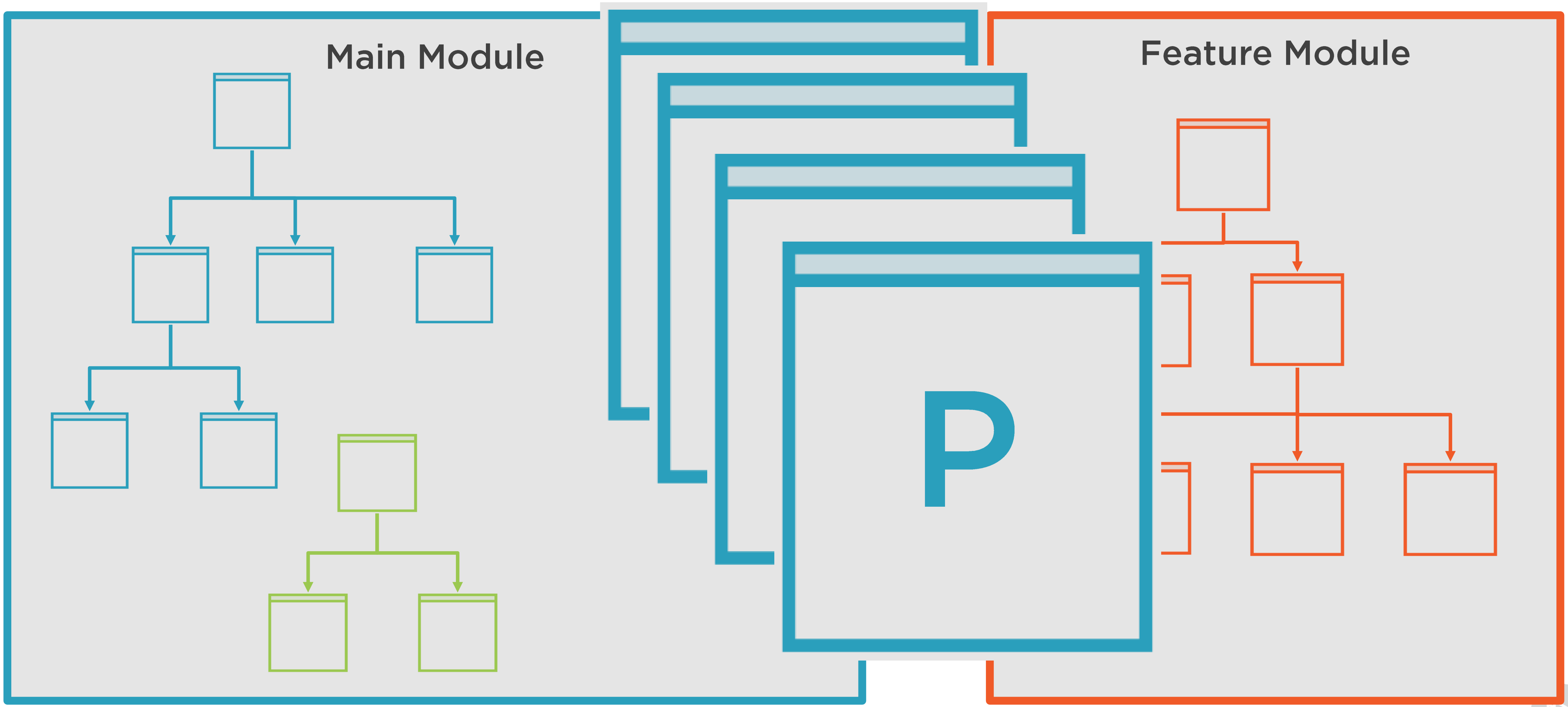




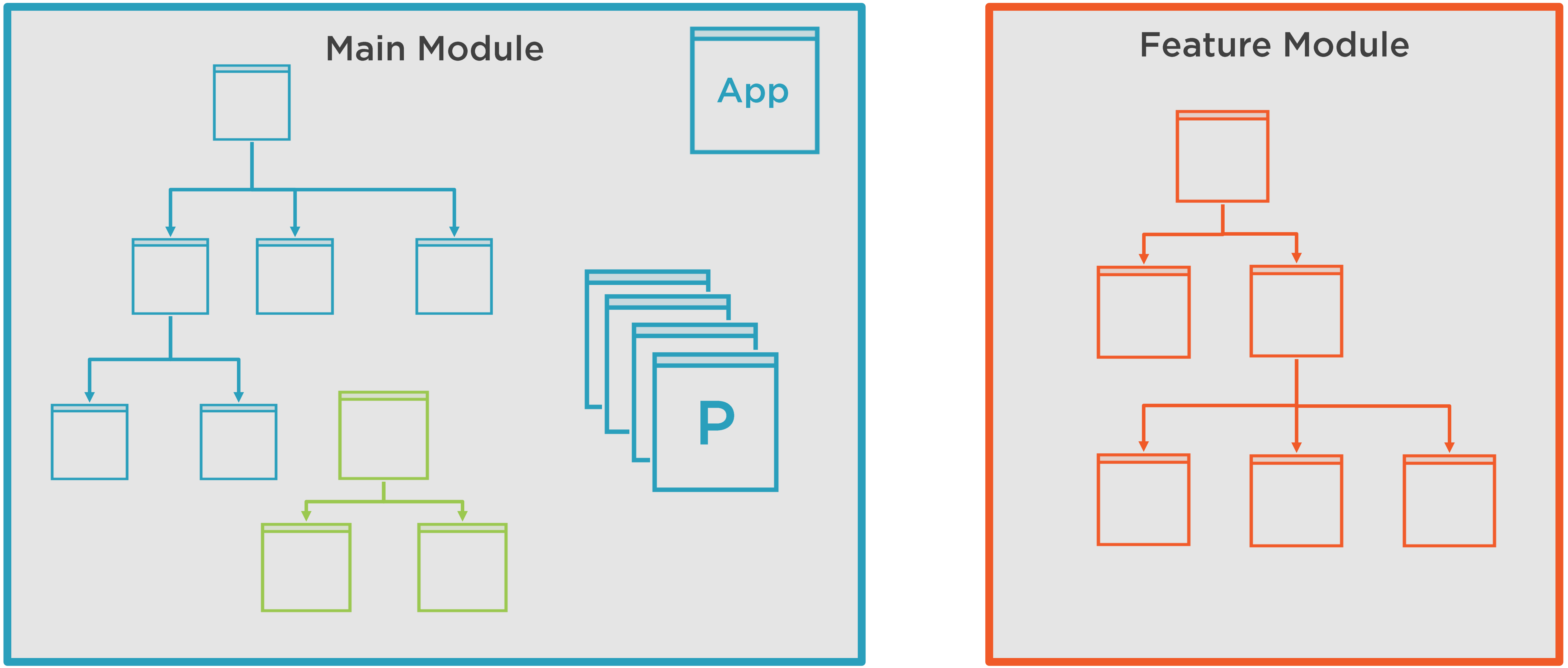
Angular 2 Component Hierarchy



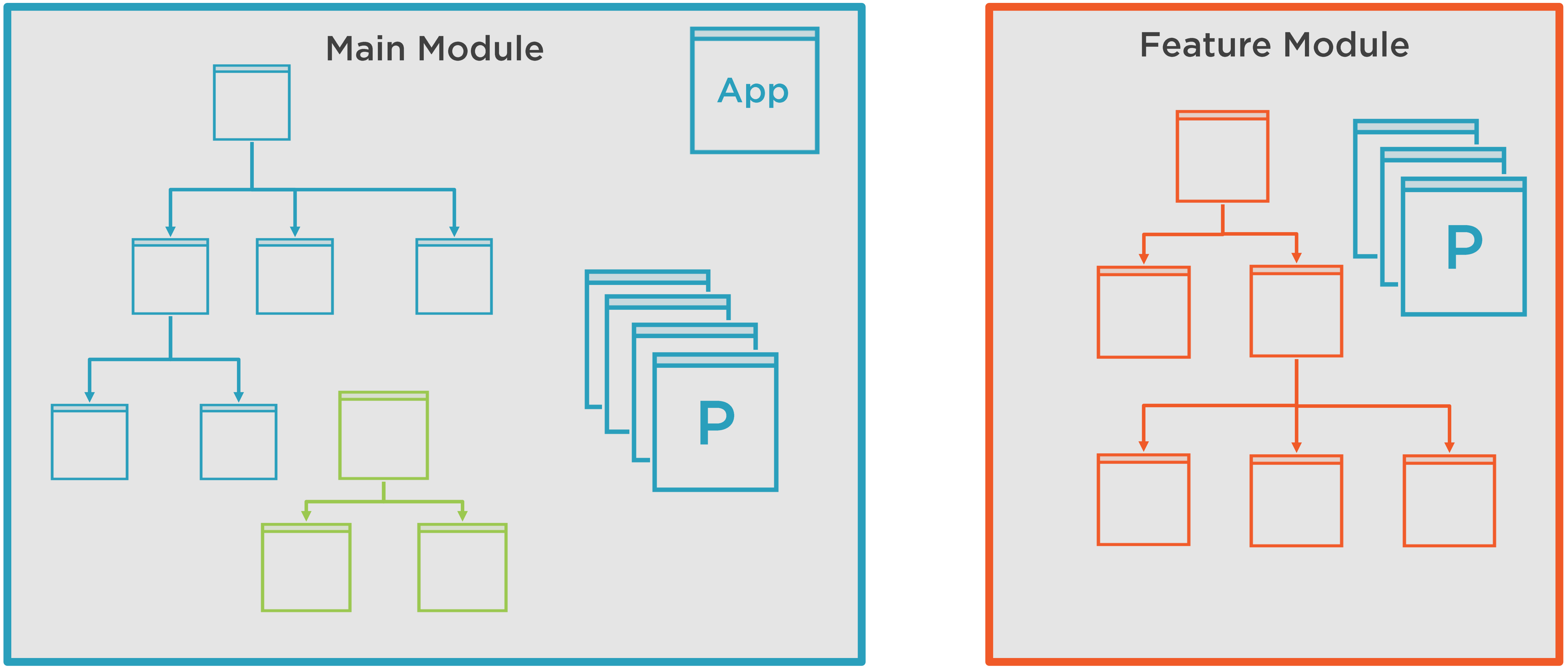
Angular 2 Component Hierarchy



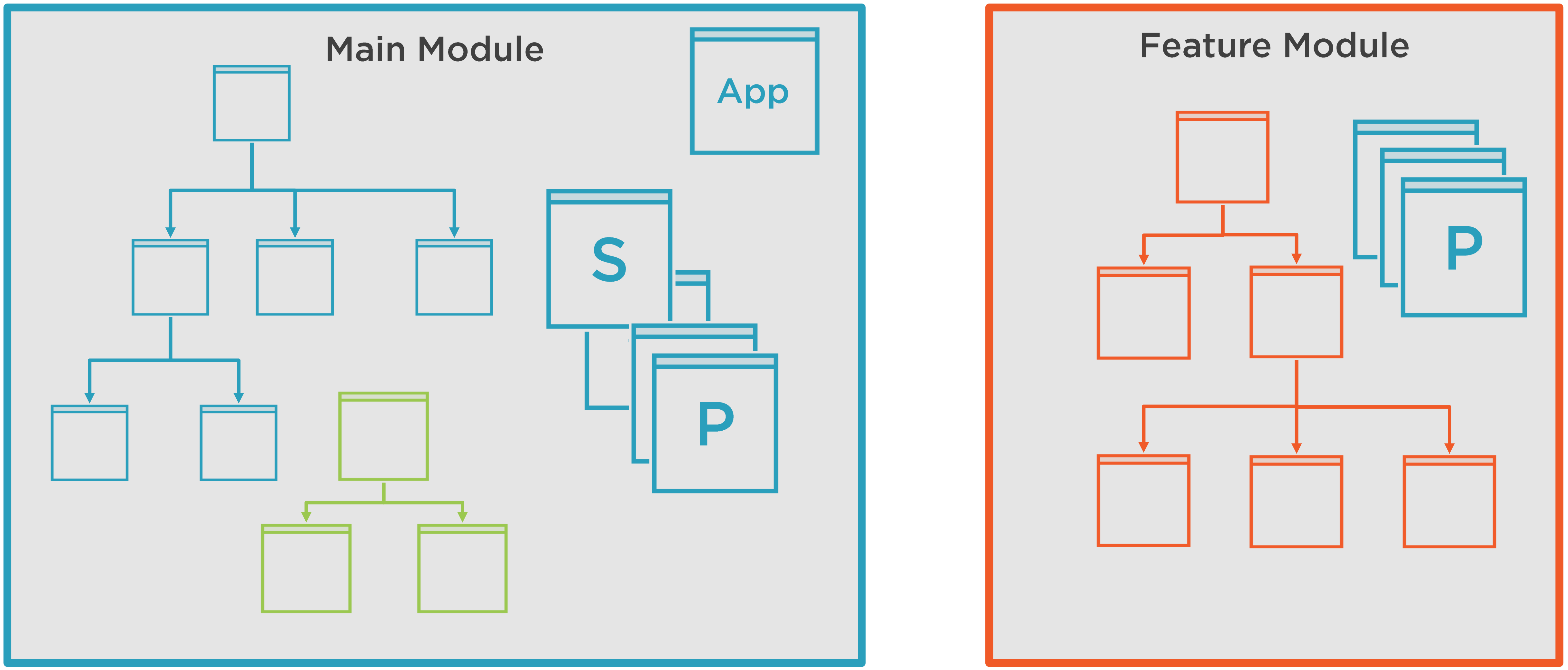
Angular 2 Component Hierarchy



Angular 2 Component Hierarchy



Angular 2 Component Hierarchy


Main Module
App
S
P
Feature Module
P