Building Business Objects

Deborah Kurata

@deborahkurata | blogs.msmvps.com/deborahk/

pluralsight

Business Objects

- Think about our application as a set of business objects (entities)
- Create classes to define the prototype for those business objects
- Keeps logic out of the Angular controllers
- Use the classes anywhere in the application
- More natural approach for some

pluralsight

Interfaces and Classes

- Clearly state our intent with an interface
  - Define properties with their data types
  - Define methods with their signatures
- Implement that intent with a class

pluralsight

Module Overview

Examine OOP in TypeScript

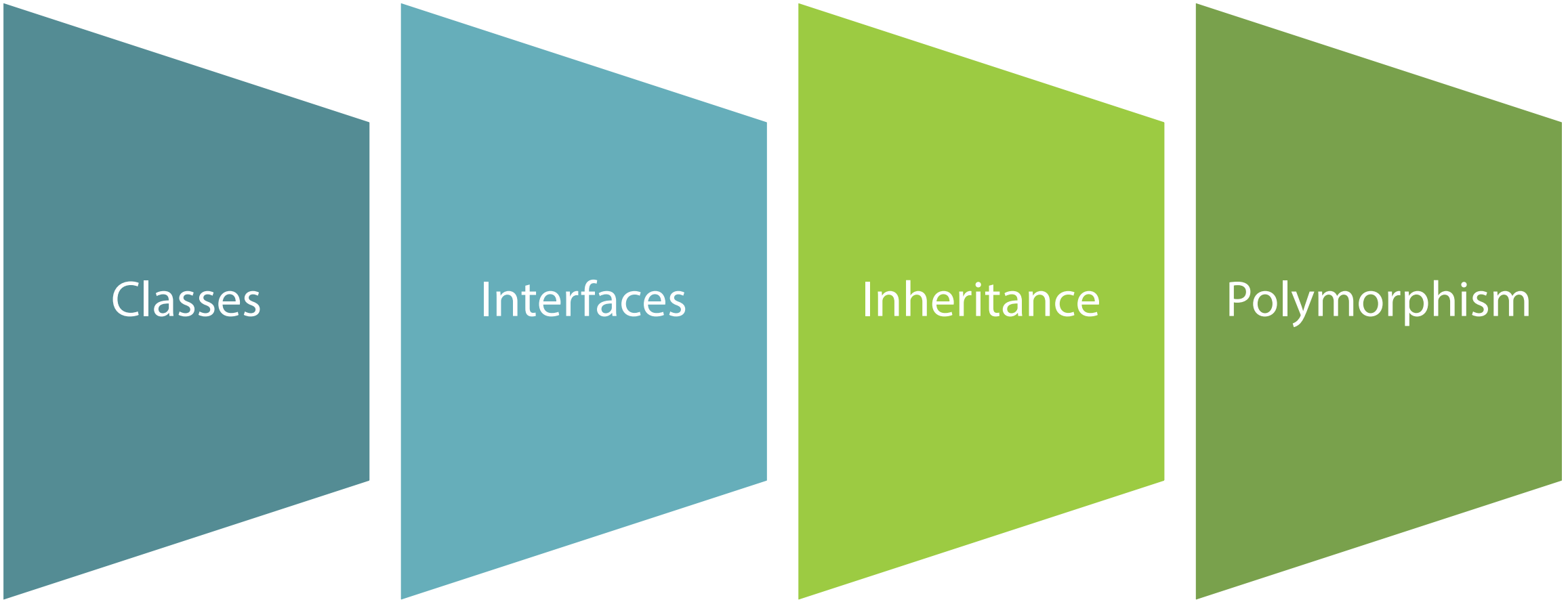
Define Business Objects

Build an Entity Class

Use that Class in an Angular Controller

pluralsight

OOP in TypeScript



pluralsight

Defining Business Object Entities

Analyze the business problem

Product Management System
- Manages products
- Track prices over time
- Maintains search tags
- Manages vendors

pluralsight

Defining Business Object Entities

Analyze the business problem

Extract the nouns

Product Management System
- Manages products
- Track prices over time
- Maintains search tags
- Manages vendors

pluralsight

Defining Business Object Entities

Analyze the business problem

Extract the nouns

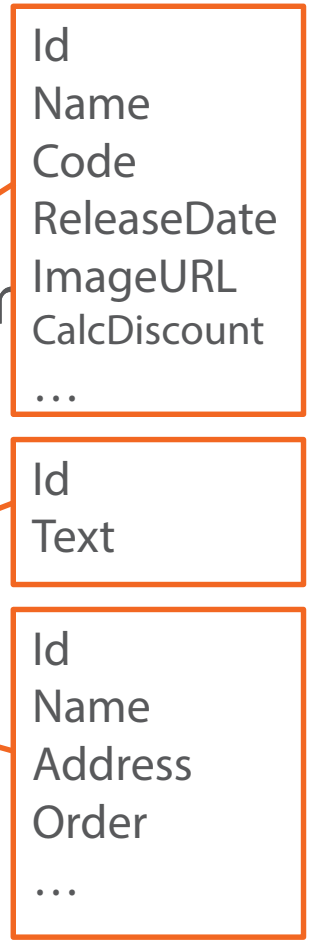
Define the members

Properties

Methods

Product Management System

- Manages products
- Track prices over time
- Maintains search tags
- Manages vendors



pluralsight

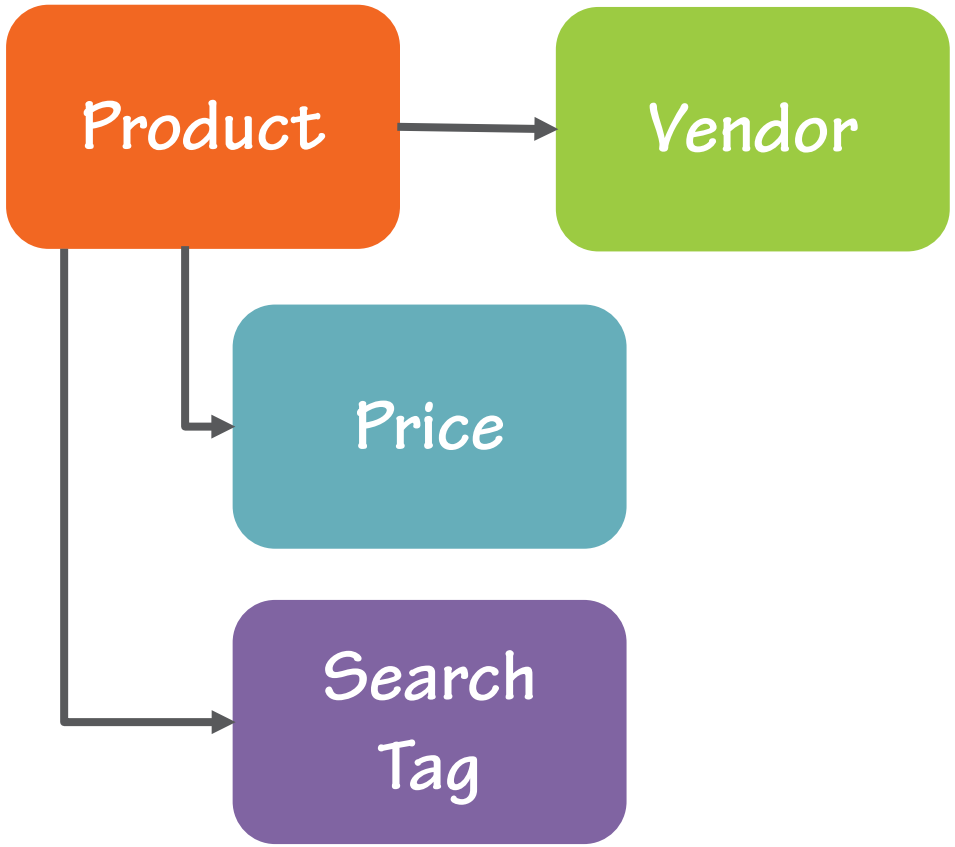
Defining Business Object Entities

Analyze the business problem

Extract the nouns

Define the members
Properties
Methods

Establish relationships



pluralsight

Building an Entity Class

- Start with a TypeScript module
  - No Angular module needed
- Optionally, define an interface
  - Defines our intent
- Create the class
  - Implement the interface with the **implements** keyword
- Declare the properties
- Add the constructor function
- Implement the methods

pluralsight

Export the Interface/Entity Class

```
module app.domain {
     export interface IProduct {
          productId: number;
          productName: string;
          calculateDiscount(percent: number): number;
     }
     export class Product implements IProduct {

          constructor(public productId: number,
                      public productName: string) {
          }

          calculateDiscount(percent: number): number {
               return this.price - (this.price * percent/100);
          }
     }
}
```

pluralsight

Using an Interface/Entity Class

- Use the interface as a type

```
products: app.domain.IProduct[];
currentProduct: app.domain.IProduct;
```

- Create an instance of the class

```
currentProduct = new app.domain.Product;
```

- Access properties

```
currentProduct.productName;
```

- Call methods

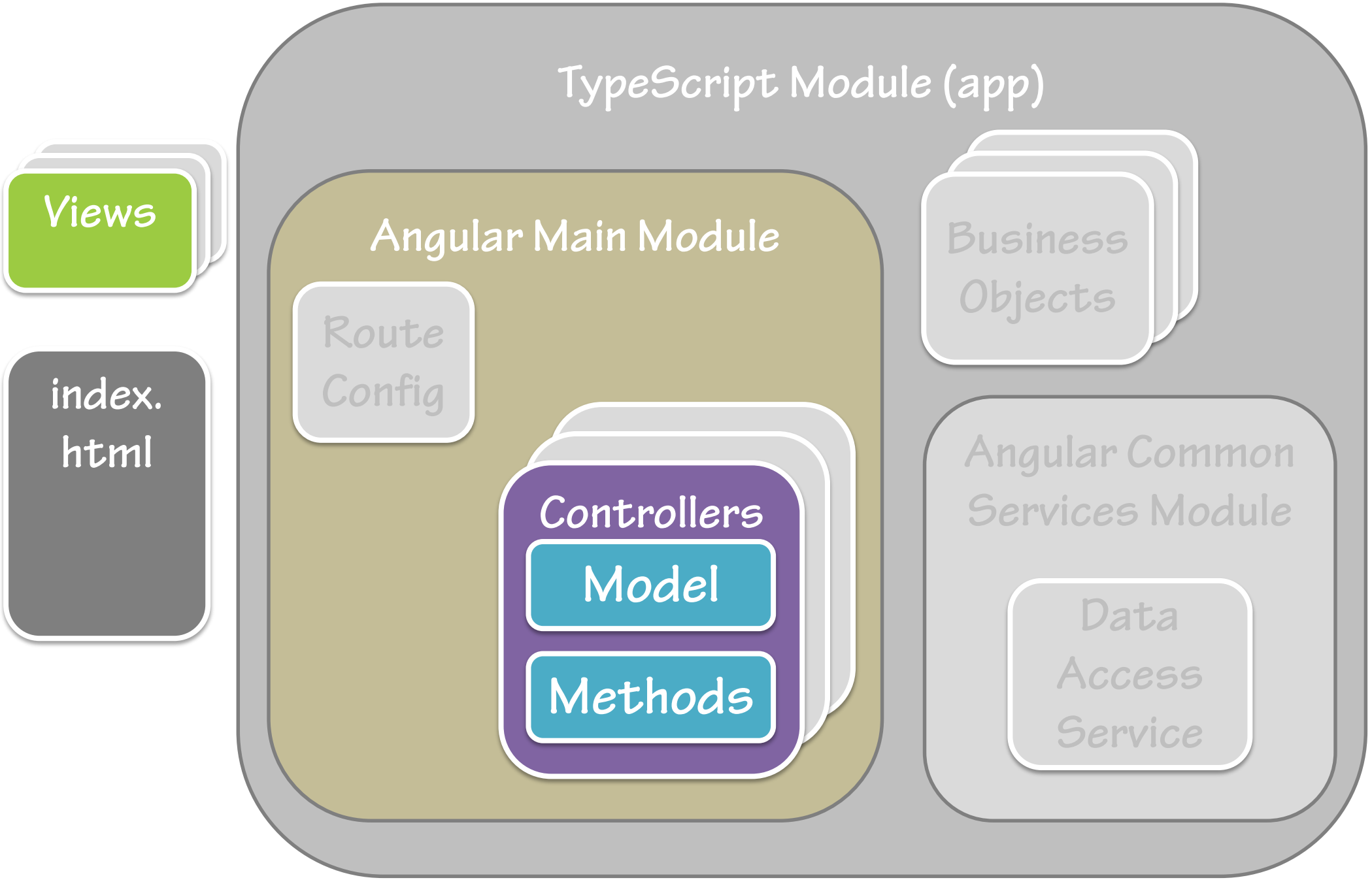
```
currentProduct.calculateDiscount(10);
```

pluralsight

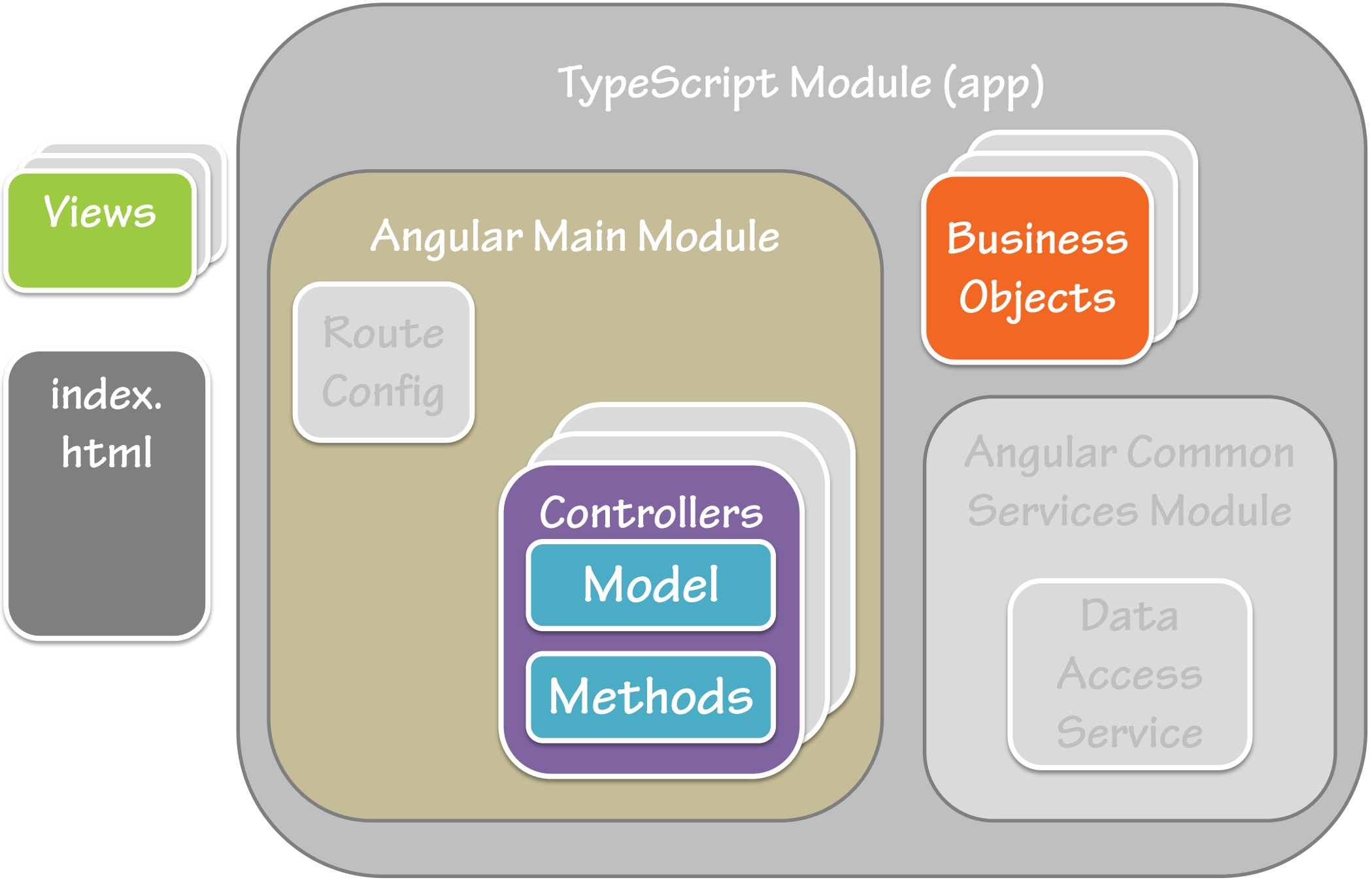
This Module Covered

- Basic OOP features in TypeScript
- Defining business objects
- Building an entity class
- Using the interface/class in an Angular controller

pluralsight



pluralsight



pluralsight