Exploring Angular 2

Lecture 1

Course Prerequisites

<HTML
CSS}

JS
JavaScript

Course Objectives

Learn about Angular 2 and it amazing features

Learn how to treat the web app as set of blocks that integrate with each other

Angular 2

Intro

AngularJS is a structural framework for dynamic web apps

extend HTML's syntax to express your application's components.

All things happens within the browser

eliminate much of the code you would otherwise have to write.

TypeScript

It just a new language

Variable Declaration

```
let
     identifier : type = value
var
```

```
var name: string = "Ahmed"

let age: number = 17

let isStudent: boolean = True

var salary: any = "1900"
```

Data Types

Functions

```
function add(x: number, y: number): number{
    return x + y;
}
```

Classes

```
class Animal {
    private name: string;
    constructor(aname: string) {
        this.name = aname;
    }
    move(distanceInMeters: number = 0)
    {}
    static doStatic()
    {}
}
class Horse extends Animal {
    constructor(name: string) { super(name);}
}
```

Modules

index.ts

```
export function sum (x, y) {
        return x + y
}

export var dept = "OS"
```

home.ts

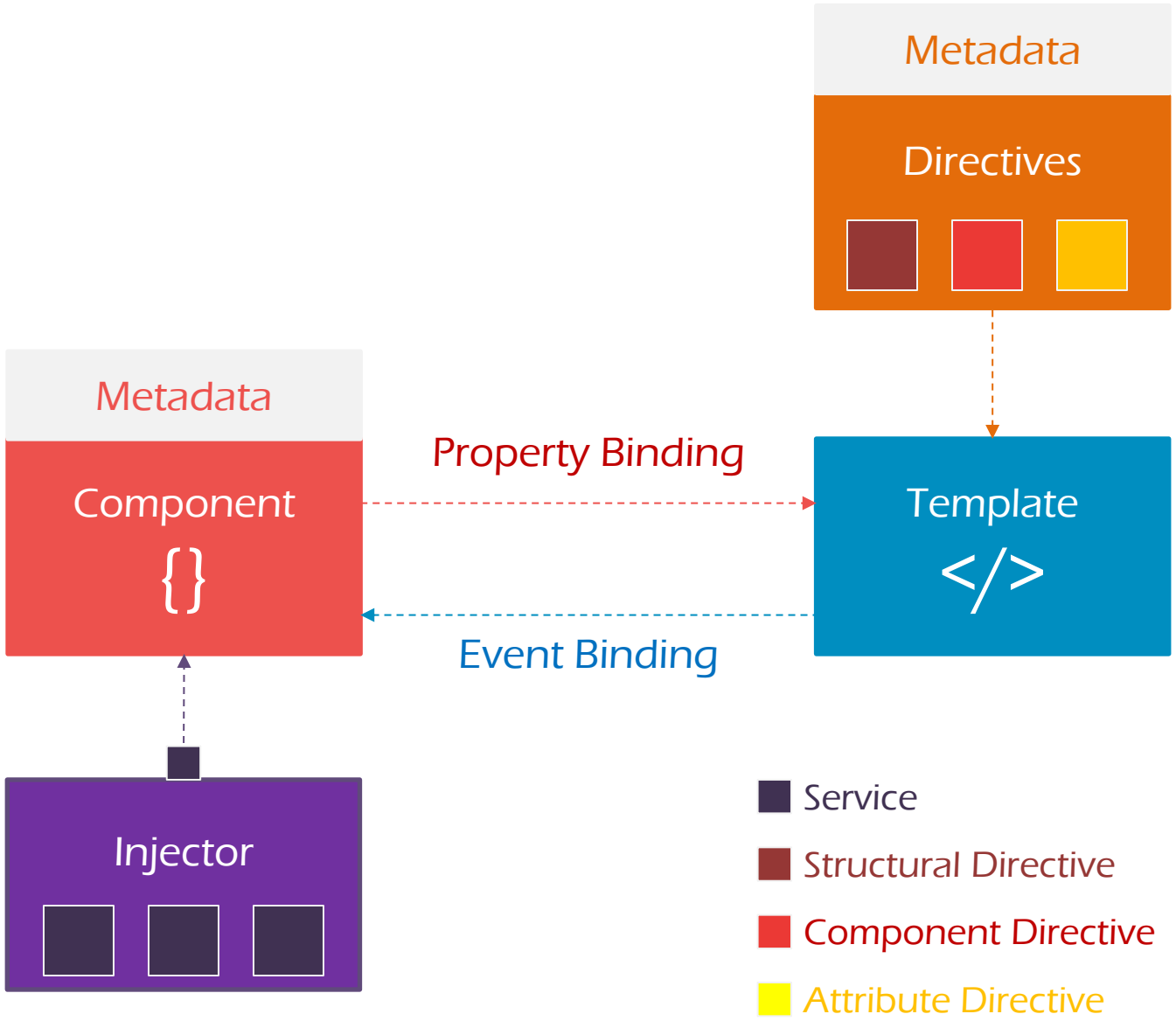
```
import * as i from "index";

console.log(i.dept);


//OR


import {sum} from "index";

sum(1,2);
```

Let's go back to

Angular 2

Architecture

Big Picture

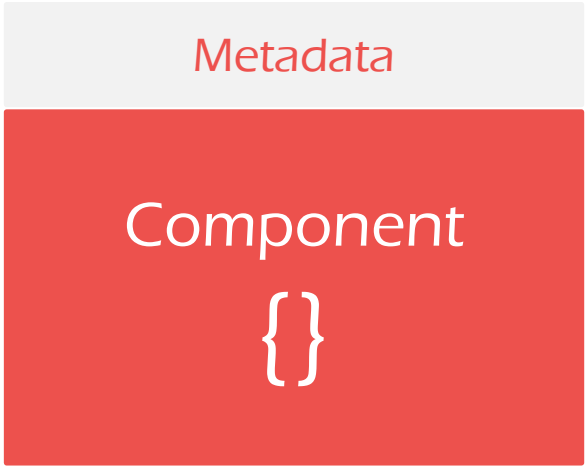

Metadata
Directives
Metadata
Component
{}
Property Binding
Template
</>
Event Binding
Injector
Service
Structural Directive
Component Directive
Attribute Directive

Components

The basic unit of Angular 2

Intro

A component controls a patch of screen called a view.
Angular 2 App is constructed of components that integrate with each other.



Decorator

Component decorator allows you to mark a class as an Angular component and provide additional metadata that determines how the component should be processed, instantiated and used at runtime.

```
@Component({
        //Metadata Properties
})
export class AppComponent{
        //The Component Class
}
```

Metadata Properties

Metadata Properties are the data that Angular 2 use to prepare the Component

selector	Declare the name that we select the component by in html.
template	Declare the component inline-defined html template.
templateUrl	Declare the url of html file that contain the template.
styles	Declare the component inline-defined style.
styleUrls	Declare the list of urls of style files that applied on the view.

```
@Component({
        selector: "app-comp"
})
export class AppComponent{}
```

Class

Class is the blue-print of the Component that Angular 2 will insatiate the Component from.

```
@Component({
        selector: "app-comp"
        template: `<p>Hello World</p>`
})
export class AppComponent{
        name: string = "Ahmed";
}
```

Naming Conventions

File: <Component Name>.component.ts
Class: <Component Name>Component

app.component.ts

```
@Component({
        selector: "app-comp"
        template: `<p>Hello World</p>`
})
export class AppComponent{
        name: string = "Ahmed";
}
```

Hello World Component

app.component.ts

```
import { Component } from '@angular/core';

@Component({
        selector: "app-comp"
        template: `<p>Hello {{name}}!</p>`
})
export class AppComponent{ name: string = "Ahmed"; }
```

index.html

```
<html>
  <body>
     <app-comp></app-comp>
  </body>
</html>
```

Output

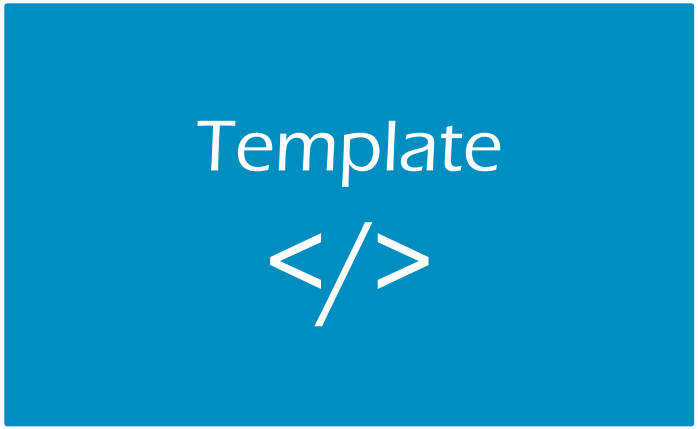
```
Hello Ahmed
```

Templates

The View of the Component

Intro

A template is a form of HTML that tells Angular how to render the component.



Naming Conventions

File: <Component Name>.component.html

app.component.html

```
<p>Age: {{age+15}}</p>
```

Interpolation

```
{{ expression }}
```

Example

app.component.ts

```
@Component({
        selector: "app-comp"
        templateUrl: "./app.html"
})
export class AppComponent{
        name: string = "Ahmed";
}
```

app.component.html

```
<p>Hello {{name}}</p>
```

index.html

```
<html>
  <body>
      <app-comp></app-comp>
  </body>
</html>
```

Output

```
Hello Ahmed
```

Expressions

A template **expression** produces a value.

Angular executes the expression and assigns it to a property of a binding target

```
{{ expression }}
```

Template Expressions look like JavaScript Expressions except that we cant use the following:

1. assignment operators (=, += , -=).
2. **new** operator.
3. ; or , .
4. increment (++) or decrement operators (--) .

Example

app.component.ts

```
@Component({
        selector: "app-comp"
        templateUrl: "./app.html"
})
export class AppComponent{
        age: number = 13;
}
```

app.component.html

```
<p>Age: {{age+15}}</p>
```

index.html

```
<html>
  <body>
      <app-comp></app-comp>
  </body>
</html>
```

Output

```
Age: 28
```

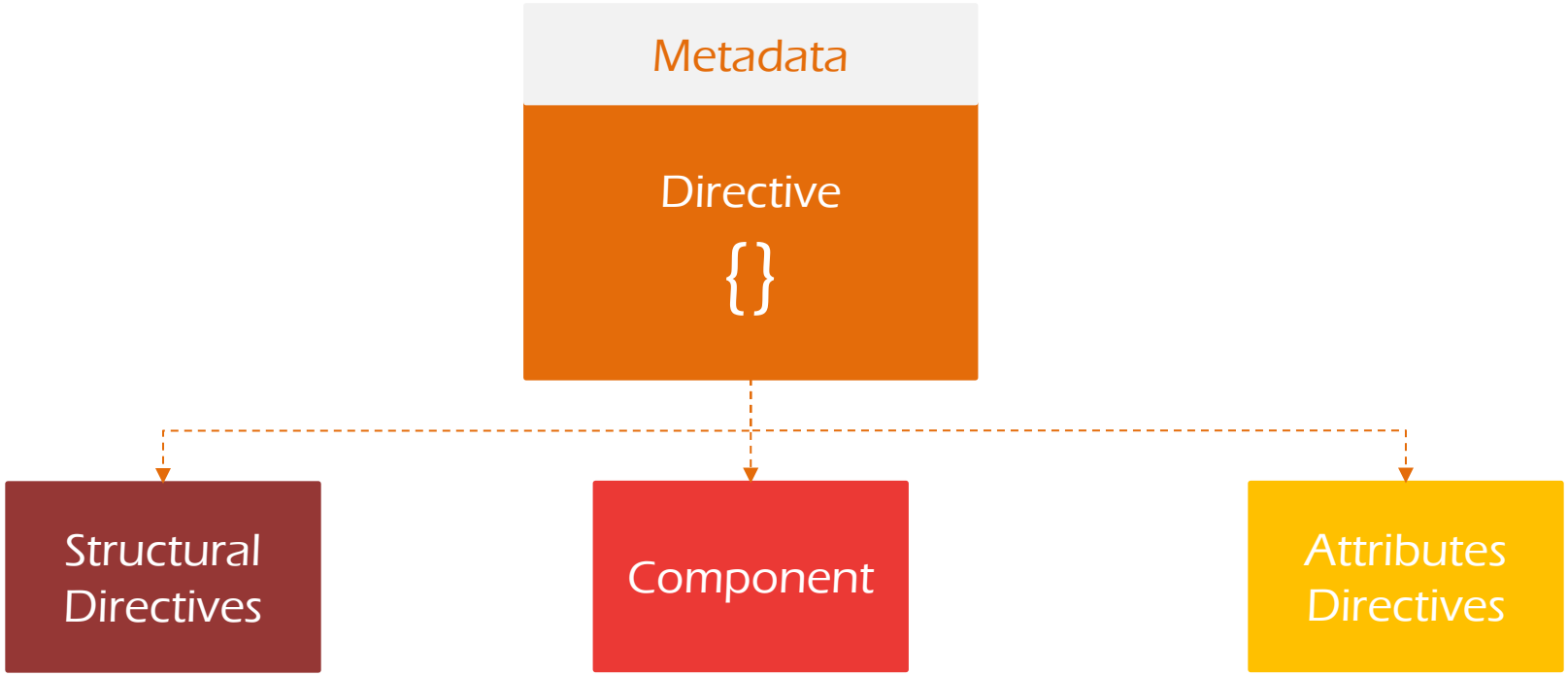
Directives

The parent of all

Intro

Templates of the Angular are dynamic, when these templates are rendered by Angular, it changes the DOM according to the Directive fed instructions.

Types

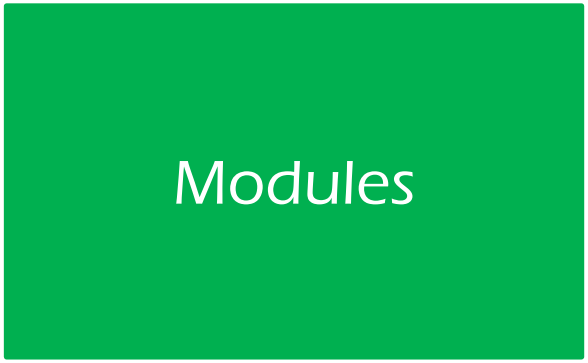

Metadata
Directive
{}
Structural Directives
Component
Attributes Directives

Modules

It helps you when you need it

Intro

Modules help organize an application into cohesive blocks of functionality.



ngModule Decorator

The **@NgModule** decorator identifies Angular module with metadata object.

imports	Import the modules that you need in your App. Example: **BrowserModule**
declarations	Declare the components and directives that belong to your angular App.
bootstrap	Declare the bootstrap component that your application.

Naming Conventions

File: <Bootstrap Component Name>.module.ts
Class: <Bootstrap Component Name>Module

app.module.ts

```
import { NgModule }  from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { AppComponent }  from './app.component';
@NgModule({
        imports: [ BrowserModule ],
        declarations: [ AppComponent ],
        bootstrap: [ AppComponent ]
})
export class AppModule{}
```

Bootstrap Your App

Let's Collect 'em all

Main

main.ts

```
import { platformBrowserDynamic } from

'@angular/platform-browser-dynamic';

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

platformBrowserDynamic().bootstrapModule(AppModule);
```

Thank You