Angular
"formerly Angular 4"
Benefits worth the cost

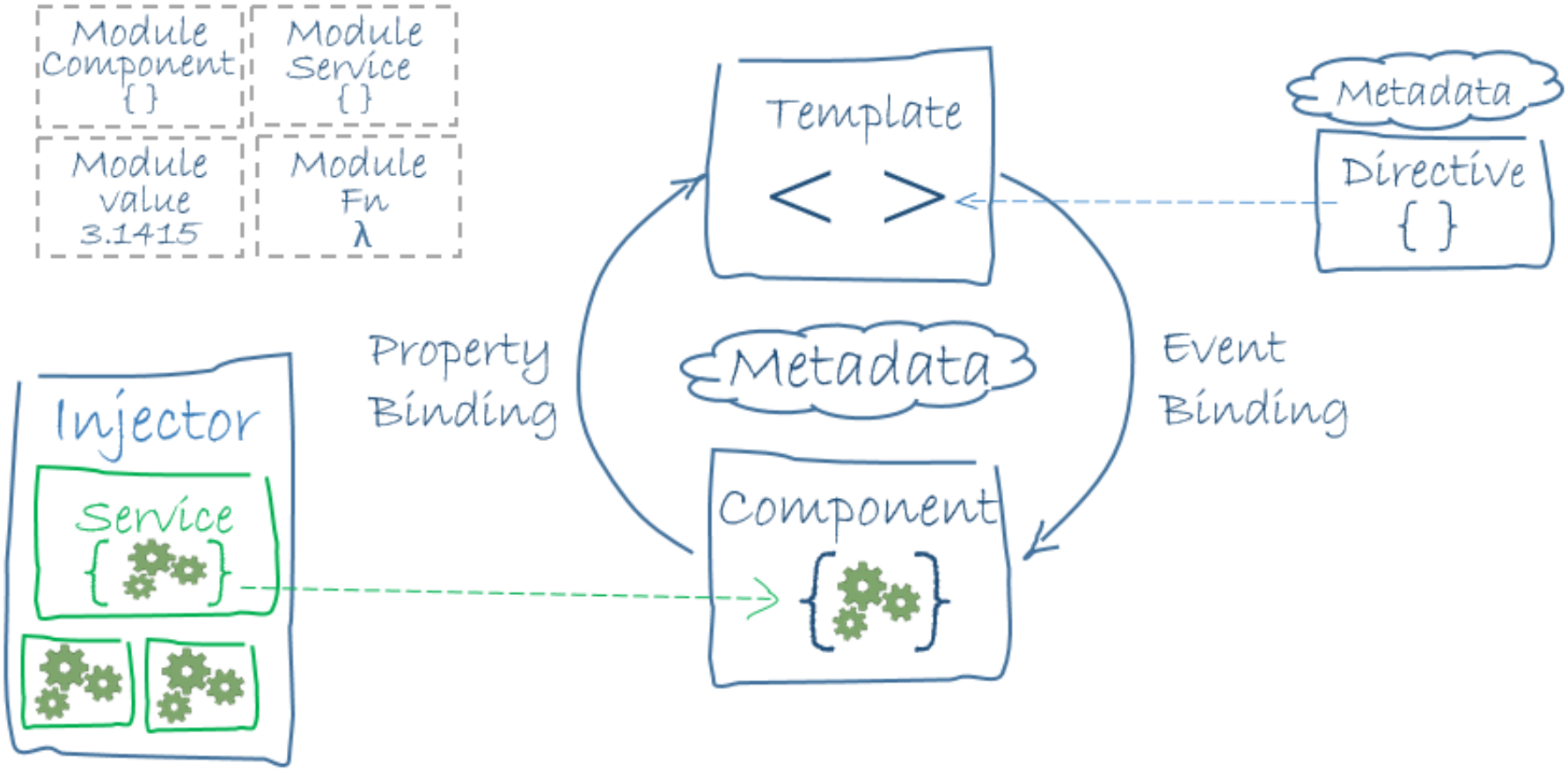
Eng. Niveen Nasr El-Den
SD & Gaming CoE
iTi

Day 2

Component Based Architecture

- Angular application should be composed of well encapsulated, loosely coupled components.
  - Components can be easily replaced with alternative implementations
- Advantages:
  - Reusability
  - Readability
  - Testability
  - Maintainability

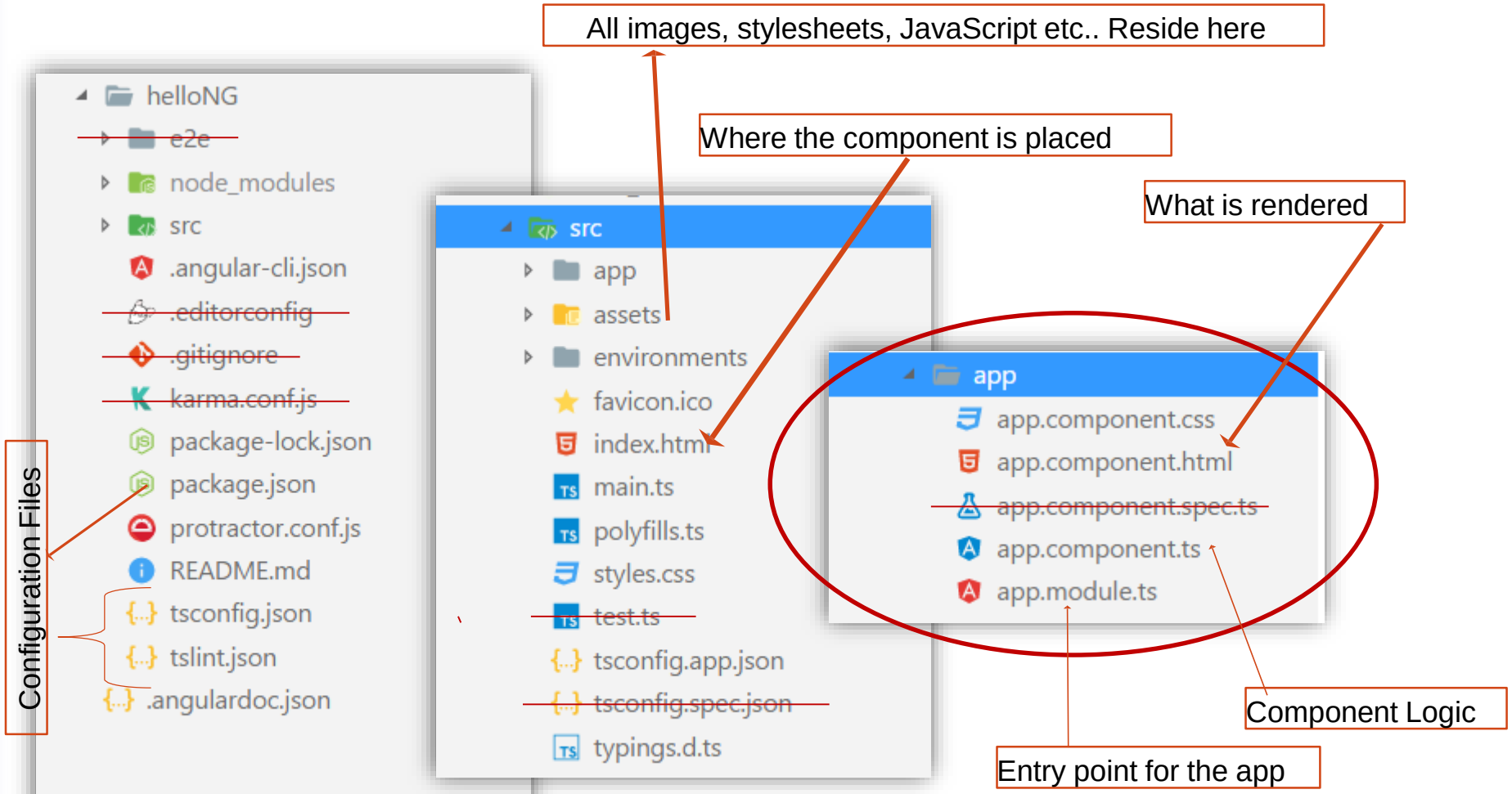
The Big Picture



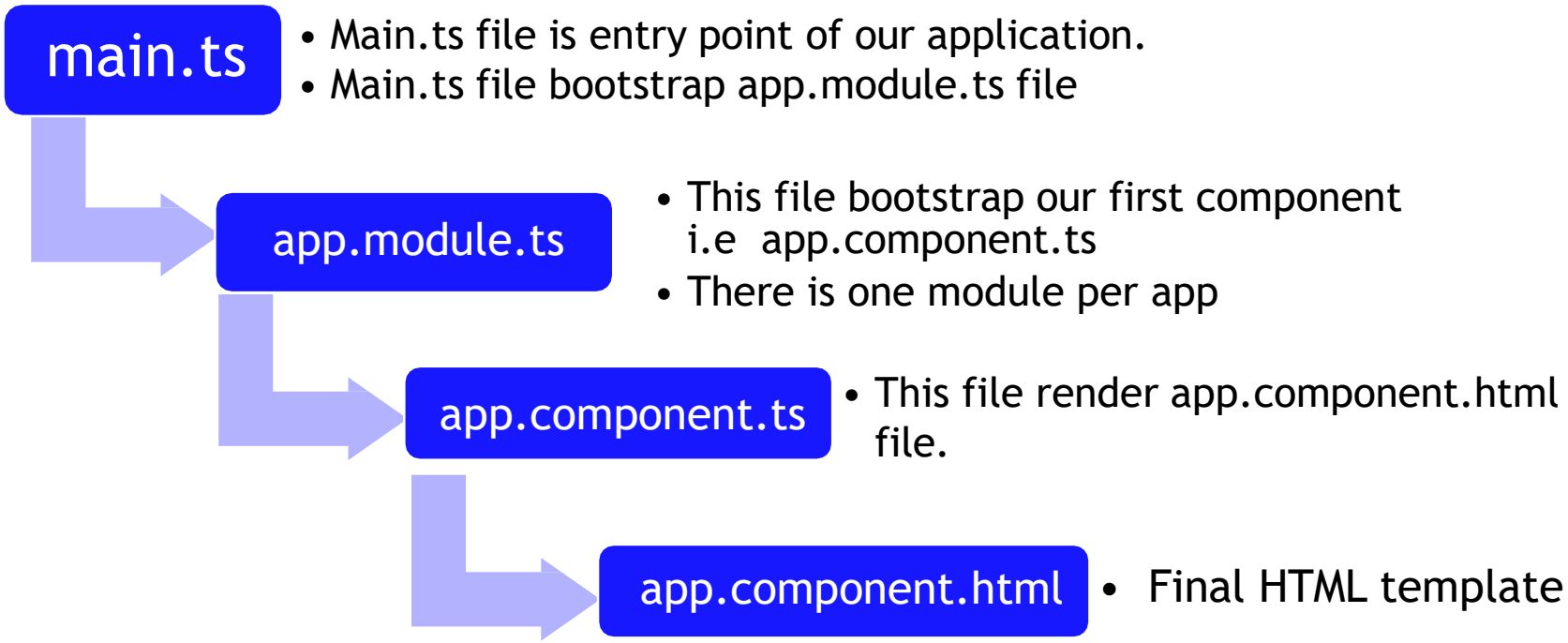
ng Commands

- ng new projectName
- cd projectname
- ng serve -p port -o
- ng g component componentName
- etc.

Angular Project Structure



App Starting Point



Bootstrapping the App

- Import the **bootstrap** module
- Import your top-level component
- Import application dependencies
- Call **bootstrap** and pass in your top-level component as the first parameter and an array of dependencies as the second

Angular Expression

- Expressions usually placed in interpolation bindings such as

  {{expression}}

- It inserts dynamic values into your HTML.
- It allows executing some computation in order to return a desired value

- Example:

  {{ 1 + 1 }}
  {{ 946757880 | date : ‘medium’ }}
  {{ user.name }}
  {{[1,2,3][0]+1}}

The Main Building Elements

- Module
- Component
- Metadata
- Template
- Data Binding
- Directive
- Pipes
- Service

Angular Module

- An Angular Module is a class decorated by @NgModule
- Every app begins with one Angular Module
- Modules declaratively specify how an application should be bootstrapped.
  - It Organize Functionality
- Module is a container for the different parts of our app
  - components, services, pipes, directives, etc

Module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { MyCompComponent } from './my-comp/my-comp.component';


@NgModule({
  declarations: [
    AppComponent,
    MyCompComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Declare components, directives, pipes

Import modules we depend on

Provide services to app root injector

Bootstrap a component

Class to define the NgModule

Component

- Components are classes decorated with @Component decorator
- It contains application logic that controls a region of the user interface that we call a view.
- Properties and methods of the component class are available to the template
- Components have templates, which may use other components

Component.ts

Imports (use other modules)

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myApp';
  isavailable = true;
  sty = { color: 'red', 'background-color': 'black' };
  ctnClickHandler(e) {
    console.log(e);
    console.log(this.title);
  }

  onDoneDone() {
    console.log('fired');
  }
}
```

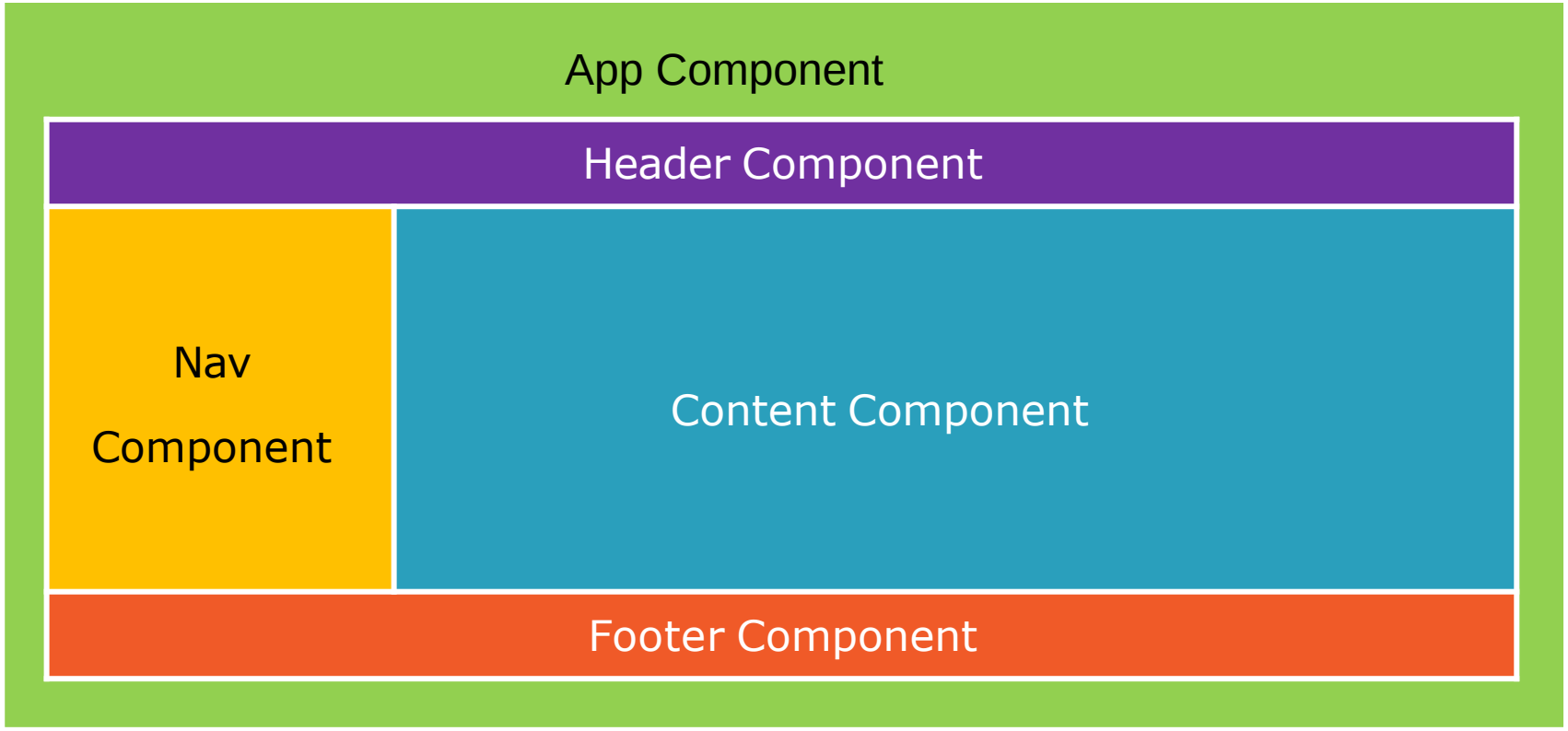
Metadata/Decorator (describe the component)

Class (define the component)

Metadata

- Metadata allows Angular to process a class
- We can attach metadata using decorators
  - Note: decorators are just functions
- Most common is the **@Component()** decorator
  - It takes a config option with the selector, template(Url), providers, directives, pipes and styles…

Simple Example



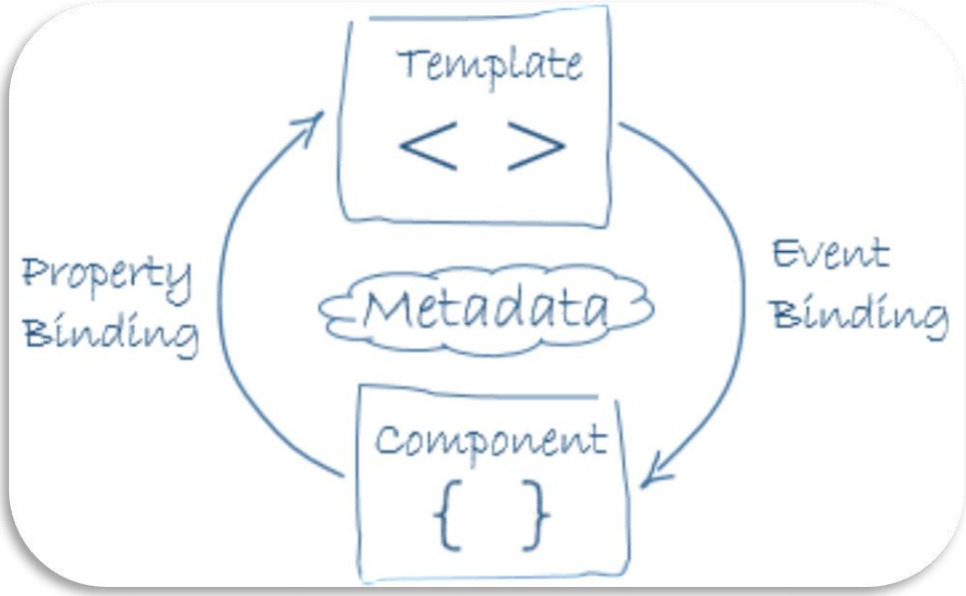
Template

- A template is HTML that tells Angular how to render a component
- Templates include data bindings as well as other components and directives
- Angular leverages native DOM events and properties which dramatically reduces the need for a ton of built-in directives
- Angular leverages shadow DOM to do some really interesting things with view encapsulation

Template

- Inline Templates
  - template defines an embedded template string
  - Use back-ticks for multi-line strings
- Linked Templates
  - templateUrl links the Component to its Template

Data-Binding

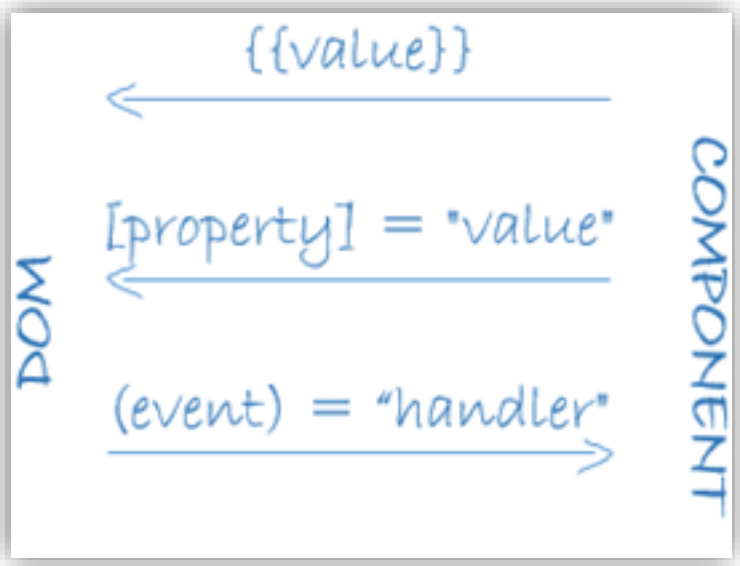


- Its the automatic synchronization of data between the Component and its Template

- Data binding includes
  - 1 way data binding
  - 2 way data binding

1 Way Data-Binding

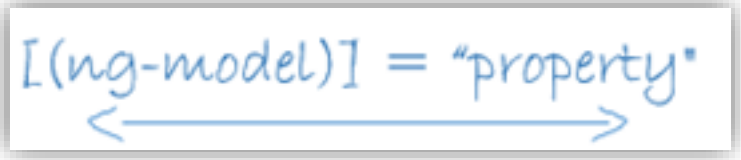
- Data is bounded only from Component to Template via
  - {{interpolation}}.
    - Bind properties & methods
  - [property binding].
    - When there is no element property, prepend with **attr**
    - Canonical form **bind-attribute**



- Data is bounded only from Template to Component via (event binding).
  - Canonical form **on-event**

2 Way Data-Binding

- Data flow from component to template and vice-versa.



- It is done via [(ngModel)] “banana|football in box”
- It is property binding and event binding combination
- FormsModule : must be imported from @angular/forms and added to imports section. This should be implemented in app.module.ts

Directive

- A directive is a class decorated with **@Directive**
- Directives Types
  - structural directives
    - change (the structure of view) the DOM layout by adding and removing DOM elements. e.g. *ngIf, *ngFor
    - Asterisks indicate a directive that modifies the HTML
    - It is syntactic sugar to avoid having to use template elements directly
  - attribute directives
    - change the appearance or behavior of an element, component, or another directive; used as attributes of elements e.g. ngStyle, ngClass
  - Component directive
    - A component is just a directive with added template features

Assignment