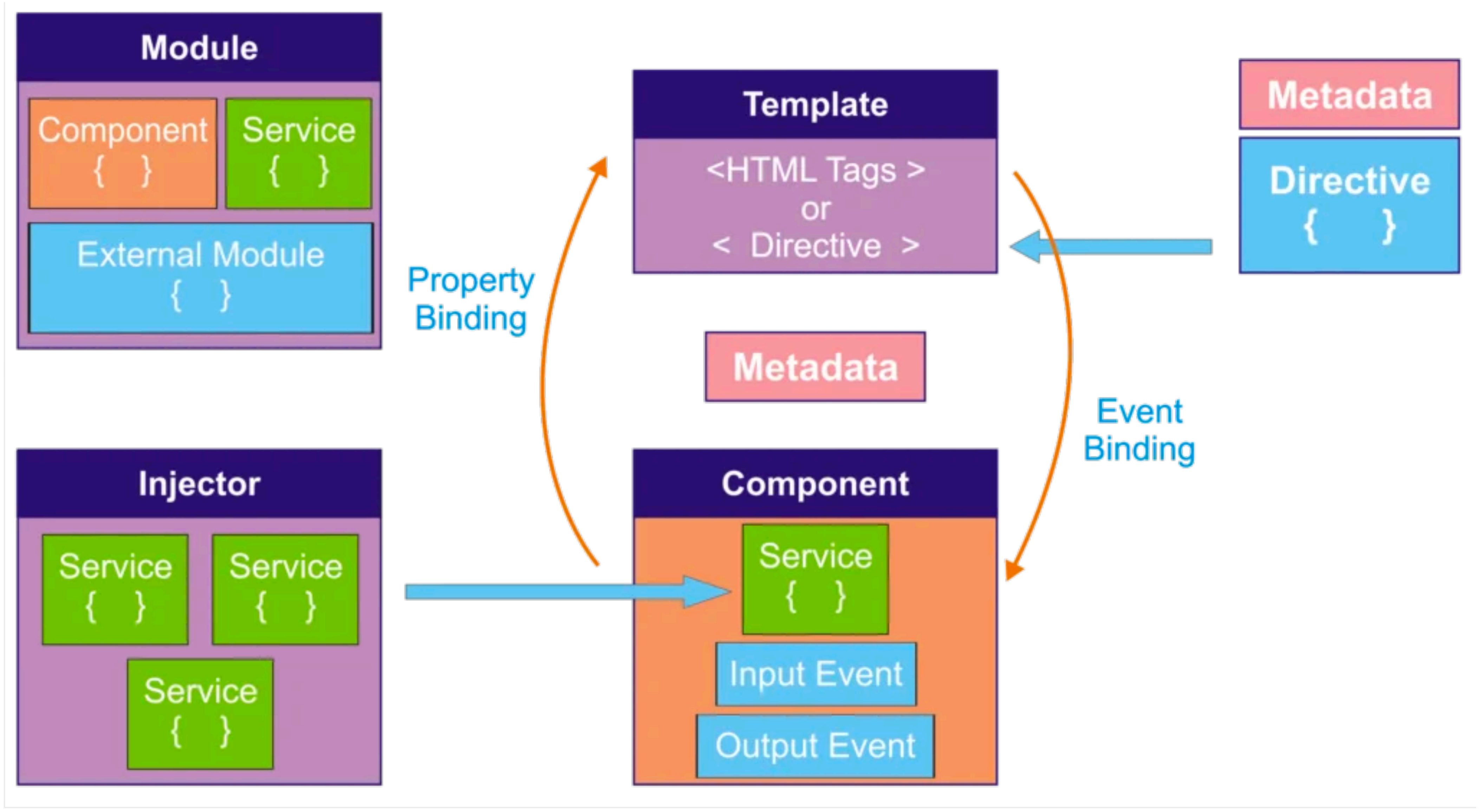
Angular Modules & Components

523419

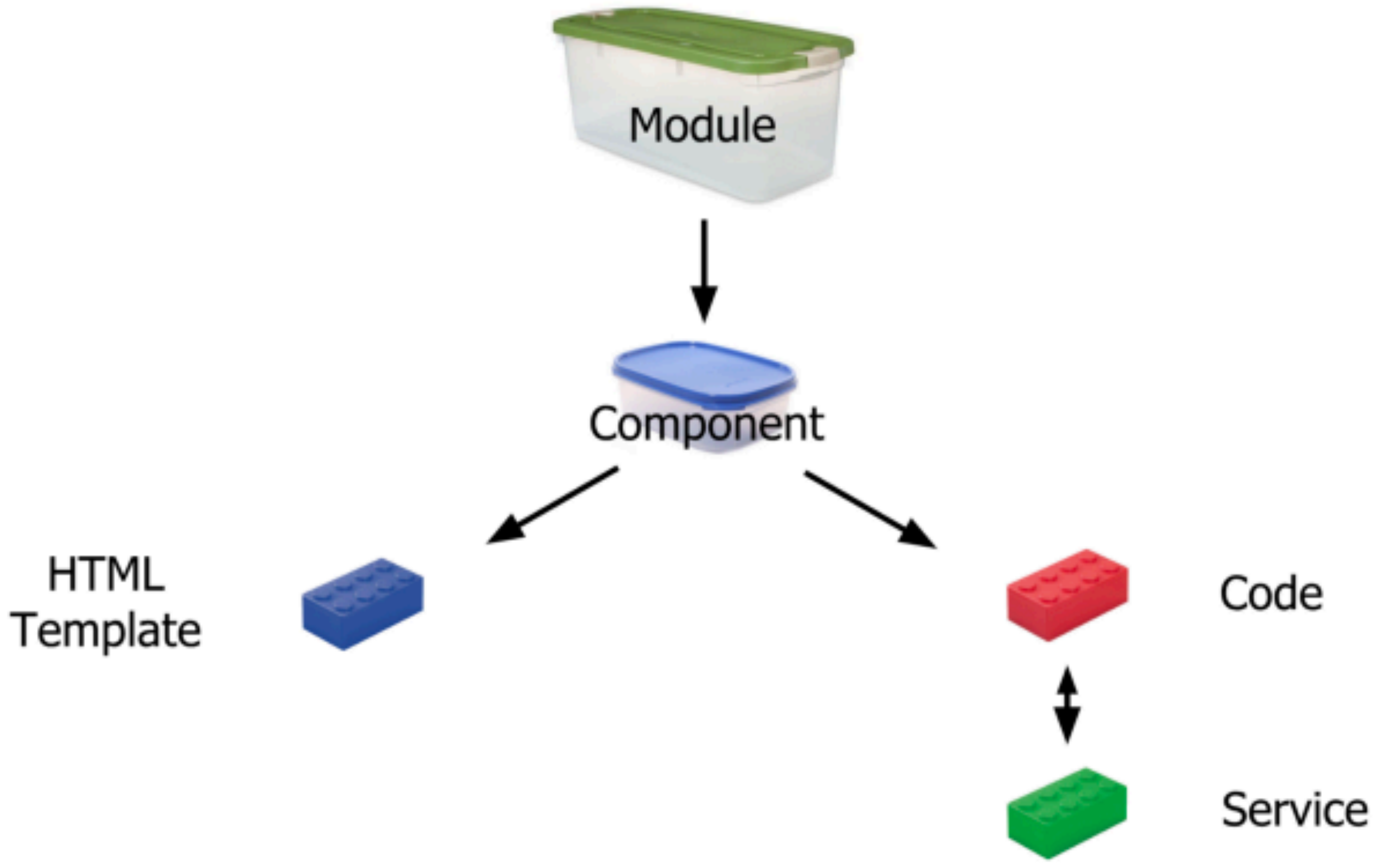
(Advanced Web Application Development)

Dr. Nuntawut Kaoungku
Assistant Professor of Computer Engineering

Angular Architecture



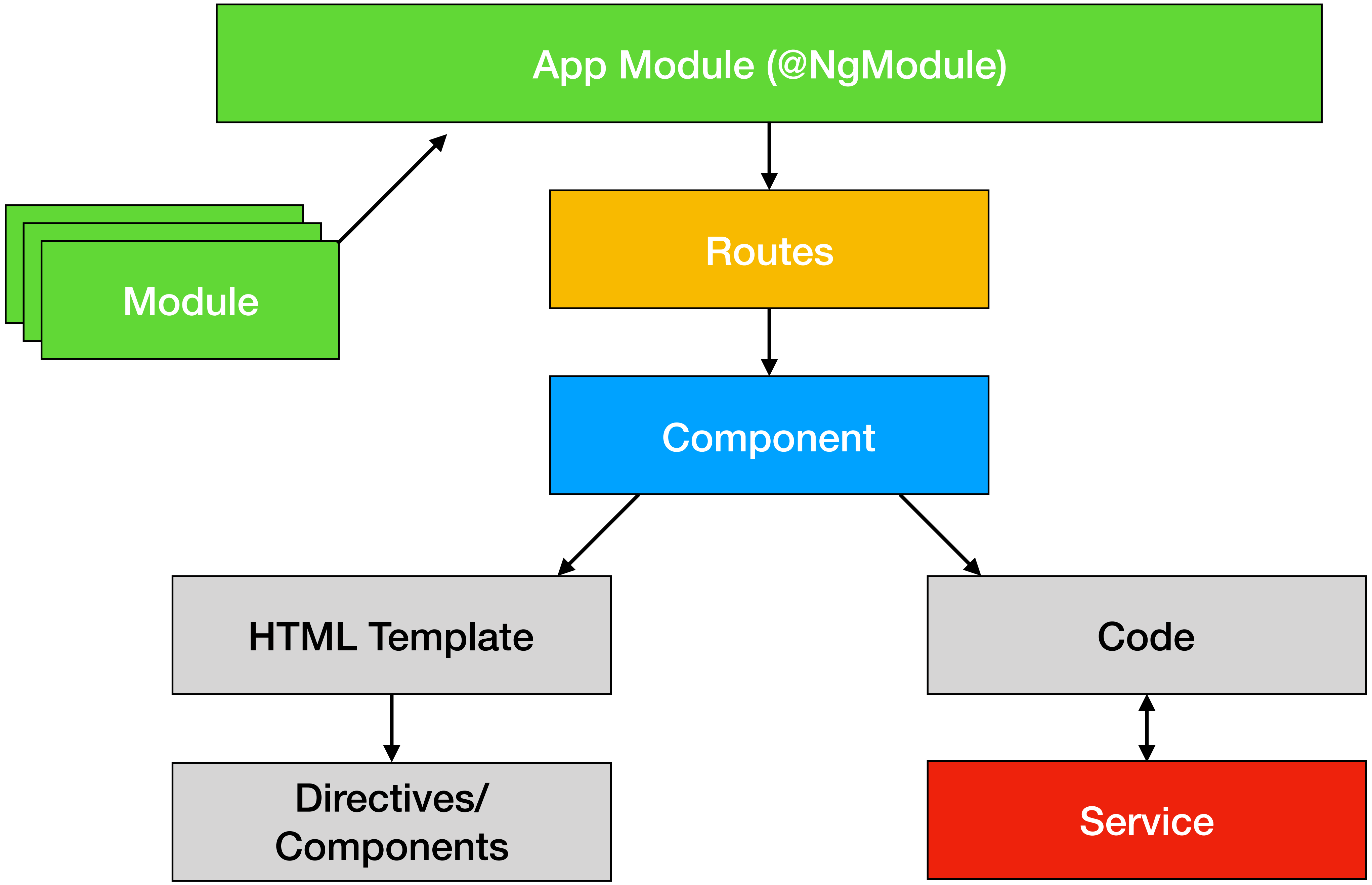
The Big Picture in Angular



Introduction to Modules

- Application is Module
- Angular Applications are modular and Angular has its own modularity system called NgModule.
  - Every Angular application has at least one class with a @NgModule decorator, it is the root module, conventionally named as AppModule.
- To use any component into an application you need to declare it into the related module
  - Angular Module is class with a @NgModule decorator.

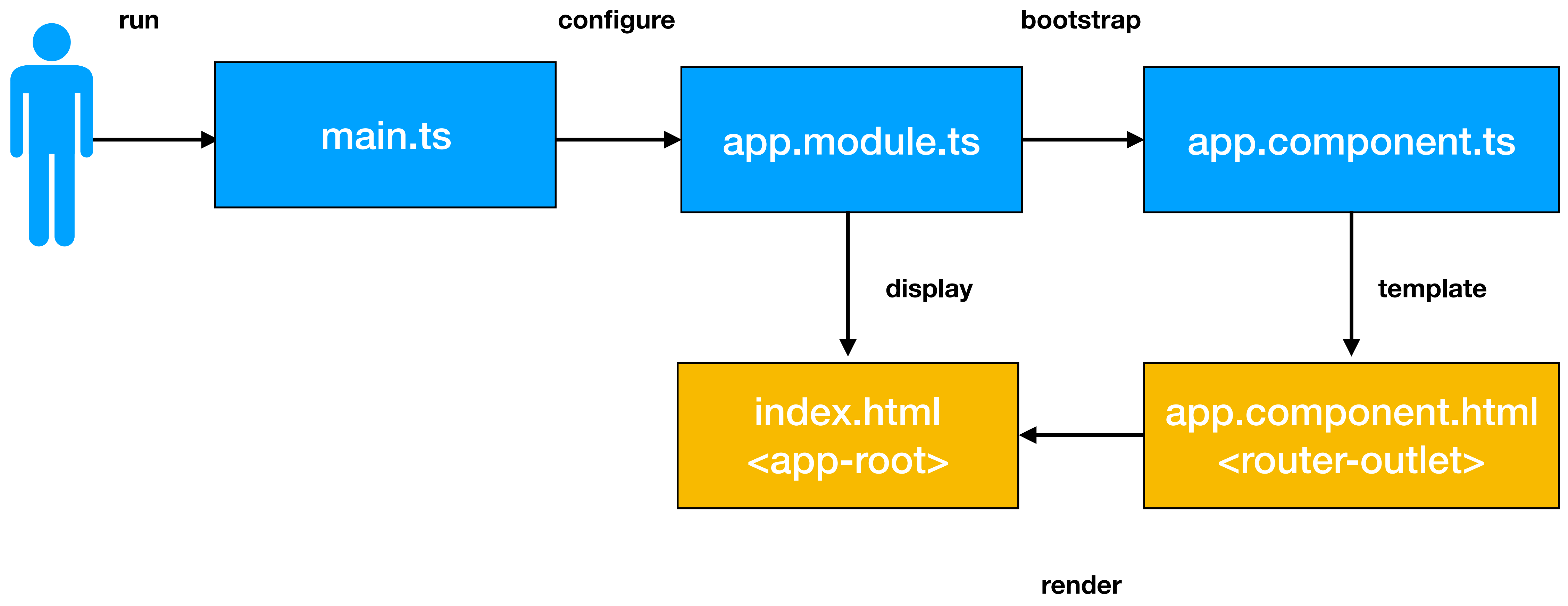
Introduction to Modules (contd.)



Module execution flow

1. Application executes *main.ts* file.
2. File *main.ts* configure app using *app.module.ts* file
3. File *app.module.ts* defines application module
4. Application displays *index.html* file.
5. File *index.html* bootstraps root component from *app.component.ts*

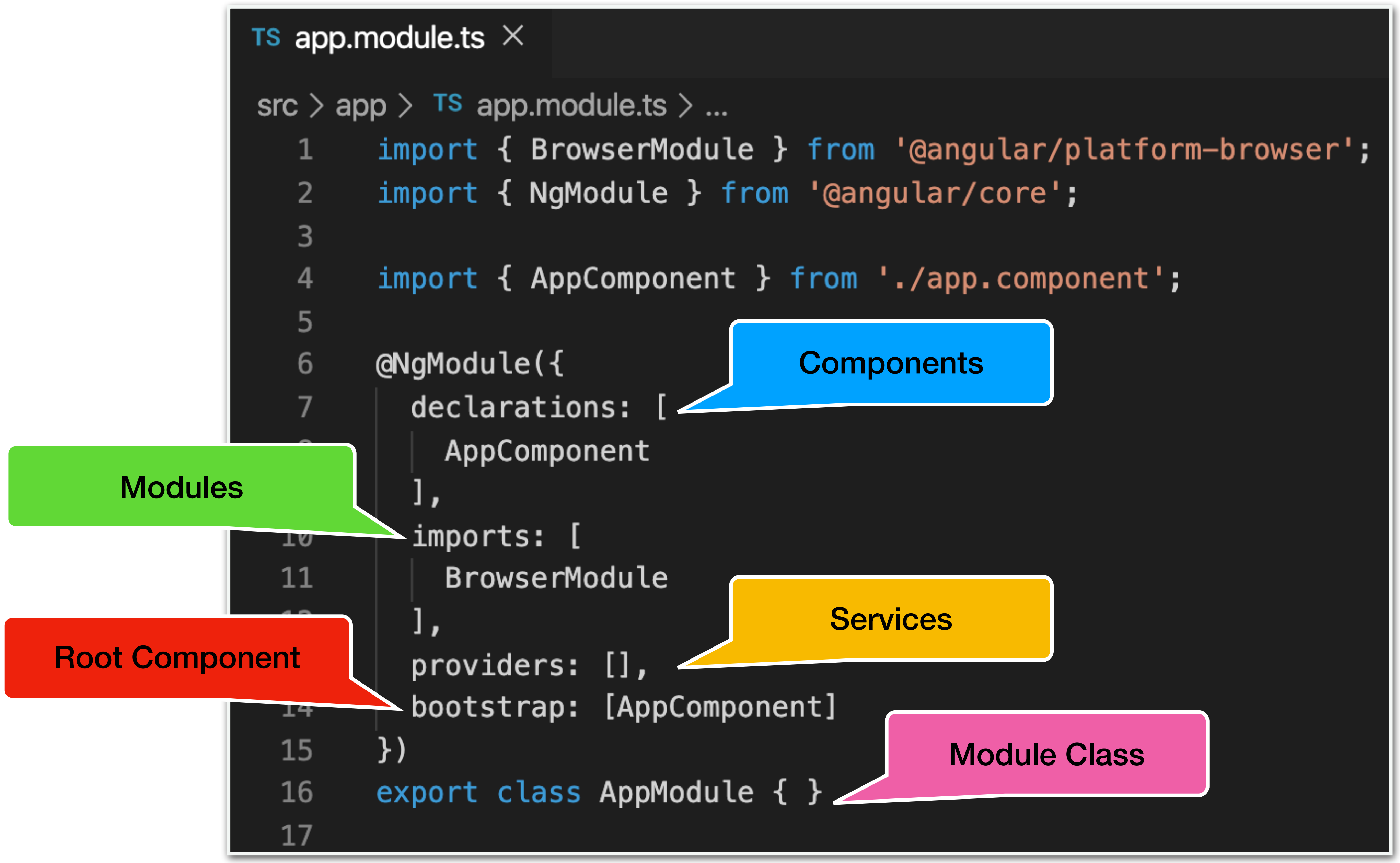
Module execution flow (contd.)



The most important properties of @NgModule

- **declarations**: declaration property contains a list of the component which you define for this module.
- **imports**: if you want to use external modules(libraries) like FormsModule, RouterModule etc then you need to add that module name here.
- **providers**: whatever service you create in that module you need to provide it here.
- **bootstrap**: you need to provide the name of the component which you want to load when the application loaded on the browser. Generally, it is the name of the root component.
- **exports**: if you want to use component or directive of this module into another module then you need to add that component or directive name here.

app.module.ts



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

import { AppComponent } from './app.component';

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

Introduction to components

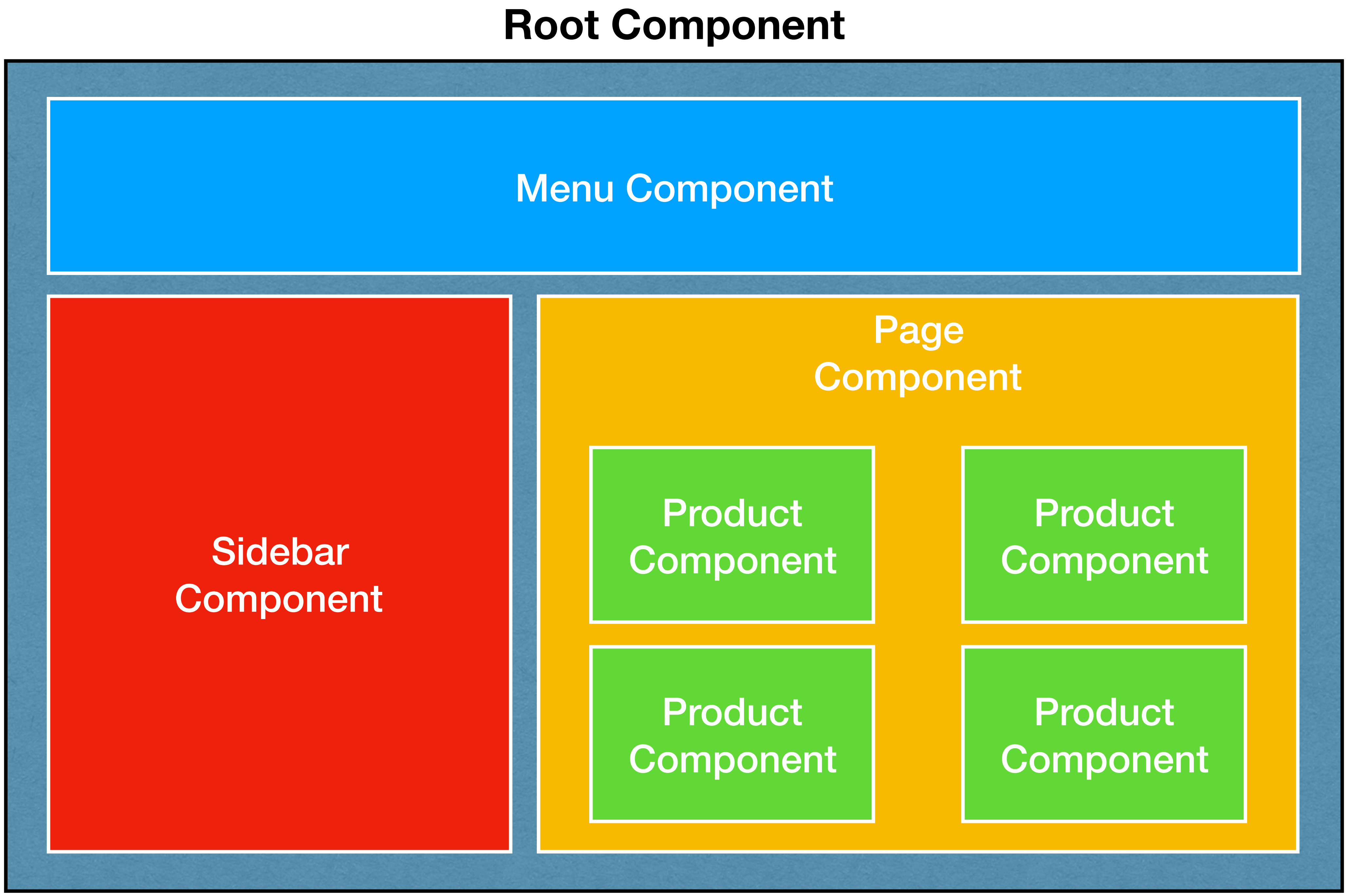
- The component is the basic building block of User Interface(UI).

- Every Angular application always has at least one component known as root component

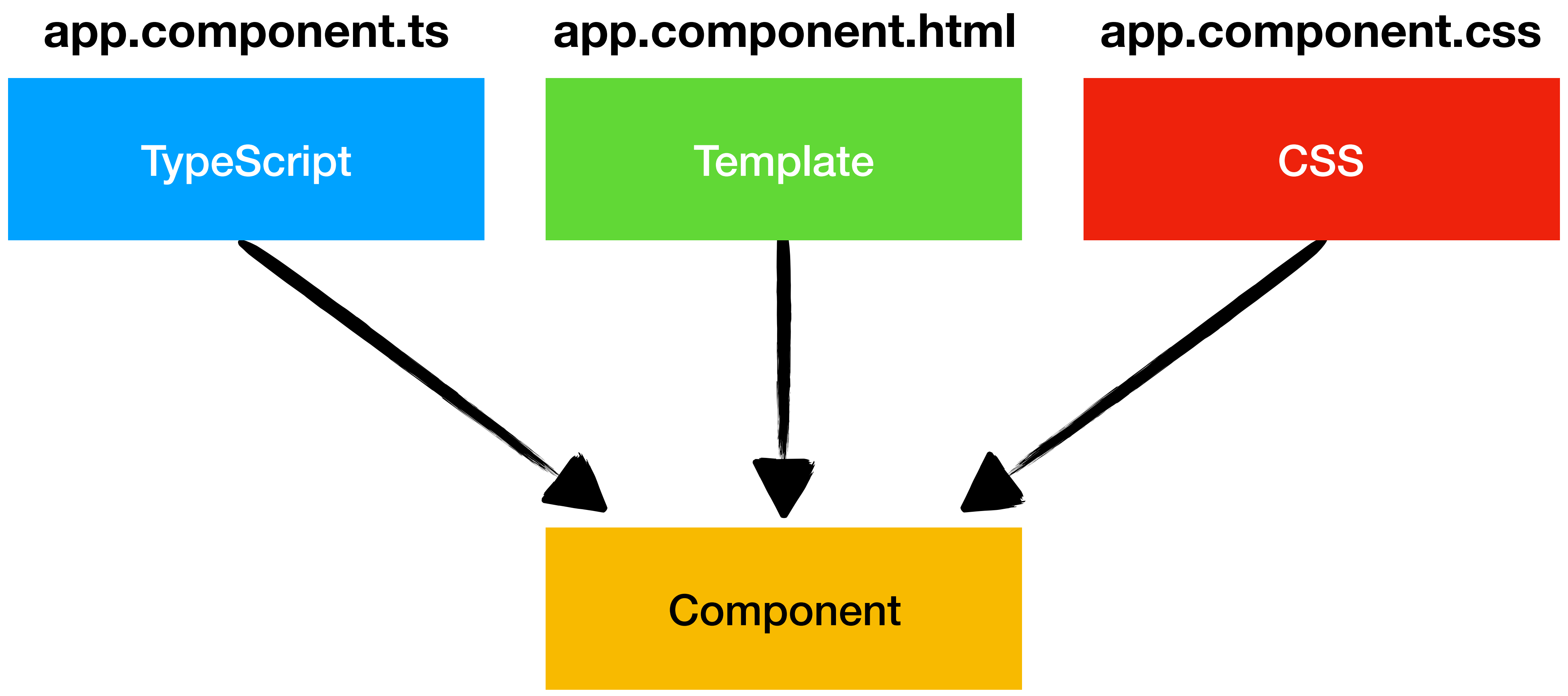
- Each component defines a class that contains application data and logic,

    - and is associated with an HTML template that defines a view to be displayed in a target environment.

Basic Angular Application view with Multiple Components

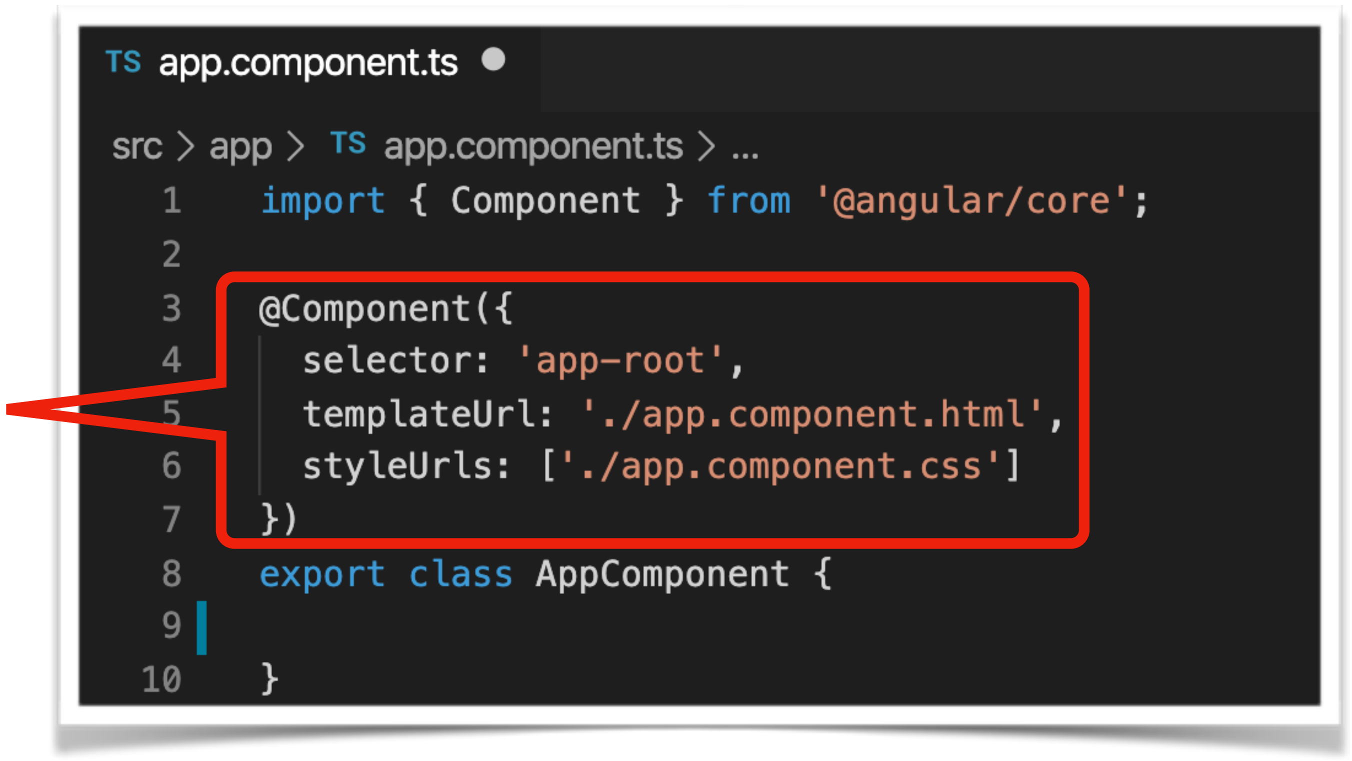


Angular Component files



Important properties of the @Component decorator

- **selector:** the name given in this property is used on HTML page as a tag to load that component the screen.
- **templateUrl:** templateUrl is used to map an external HTML page to that component
- **styeUrls:** styleUrls is used to insert the list of CSS files which you want to use for that component.



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

}
```

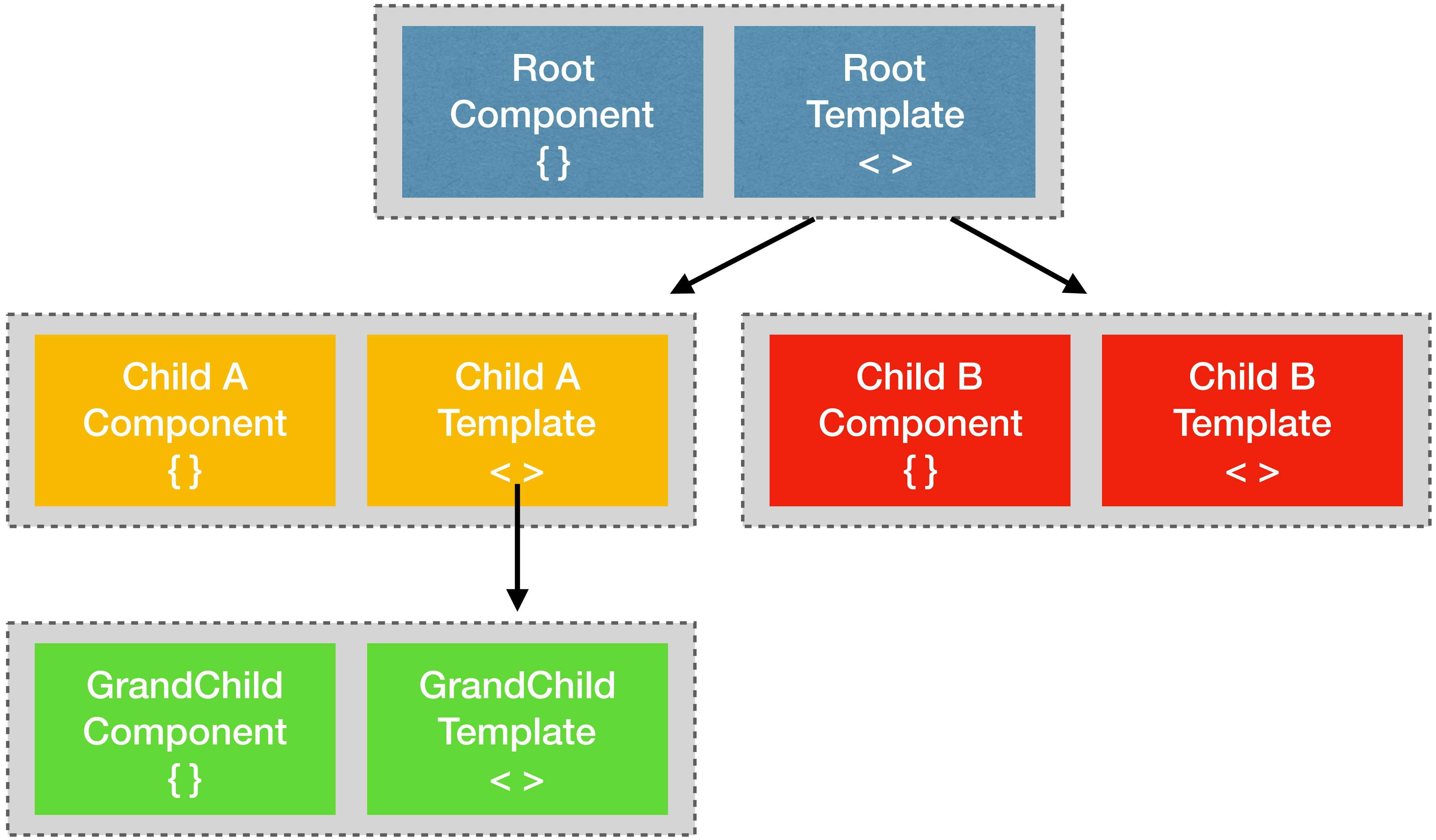
Template

- As we have seen in Component, each component is mapped to one template.

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

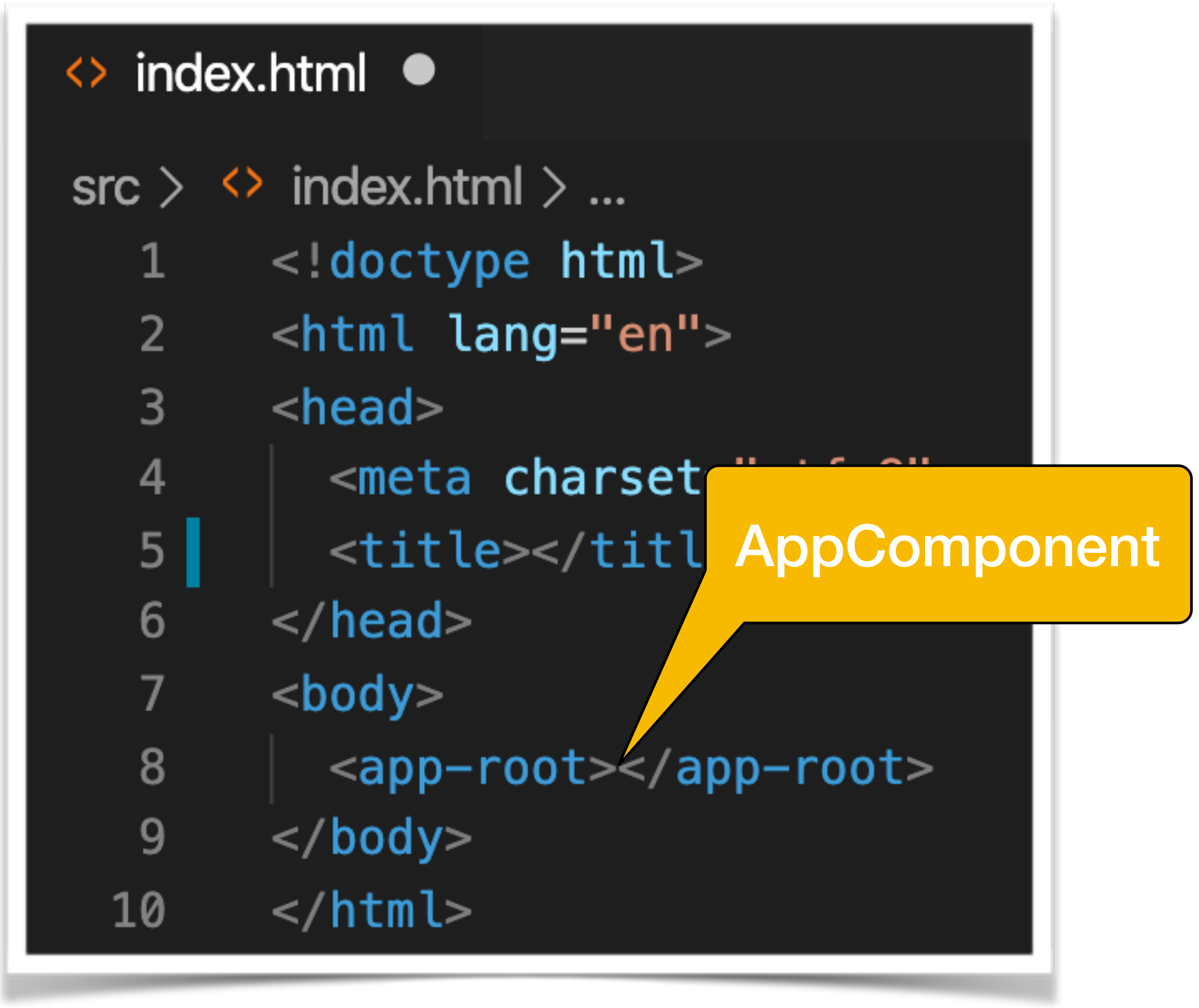
- A template looks like a regular HTML, except for a few differences, like directives, events, interpolation, data binding, other component tags.

Template is a tree like structure



Root Component

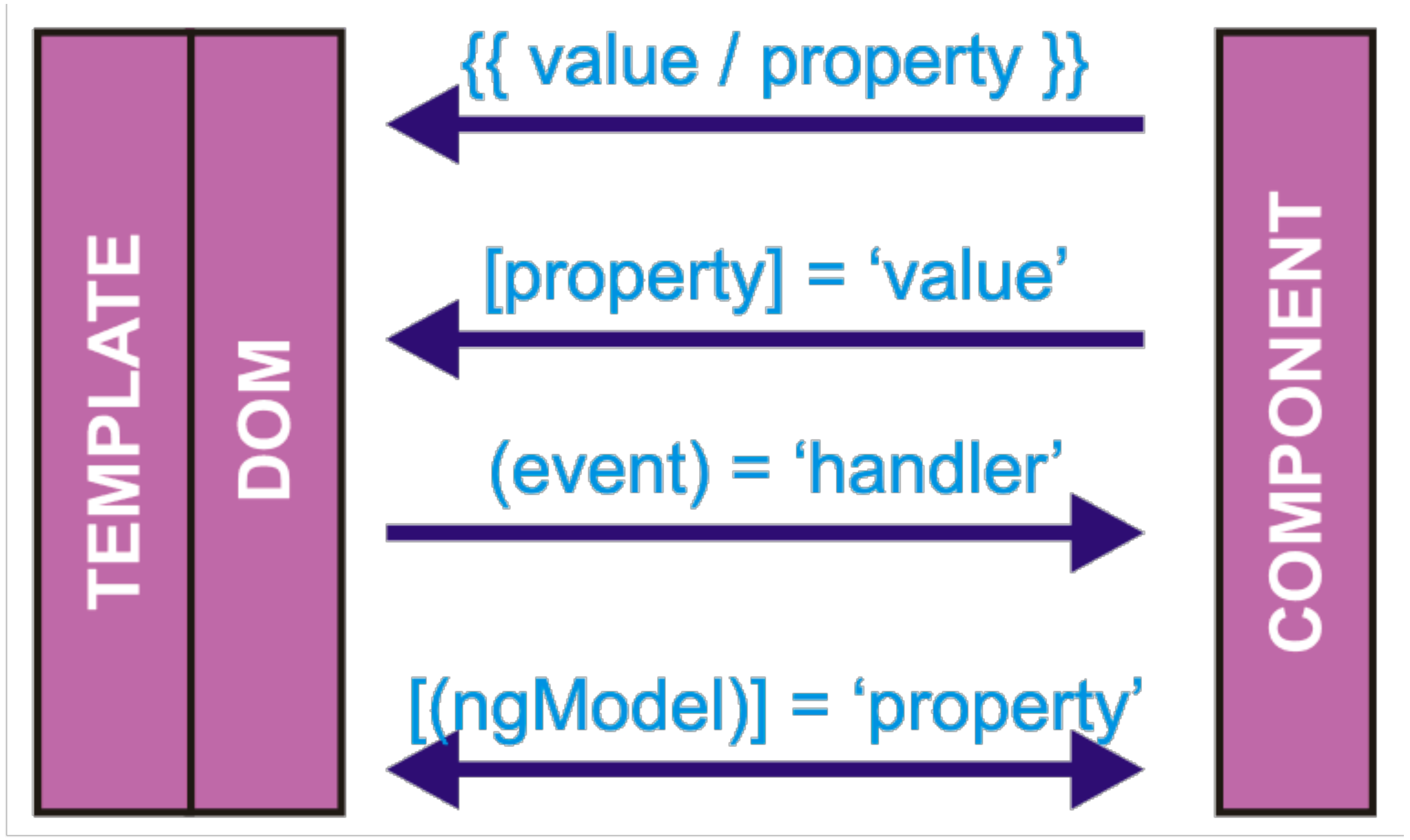
- Application has one root-component app.component.ts
- Root component is bootstrapped with index.html
- Html template of root-component app.component.html has *<app-root></app-root>* tag.
- Tag *<app-root>* is replaced by sub-components at runtime.



```
<!doctype html>
<html lang="en">
<head>
  <meta charset
  <title></titl
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Data Binding

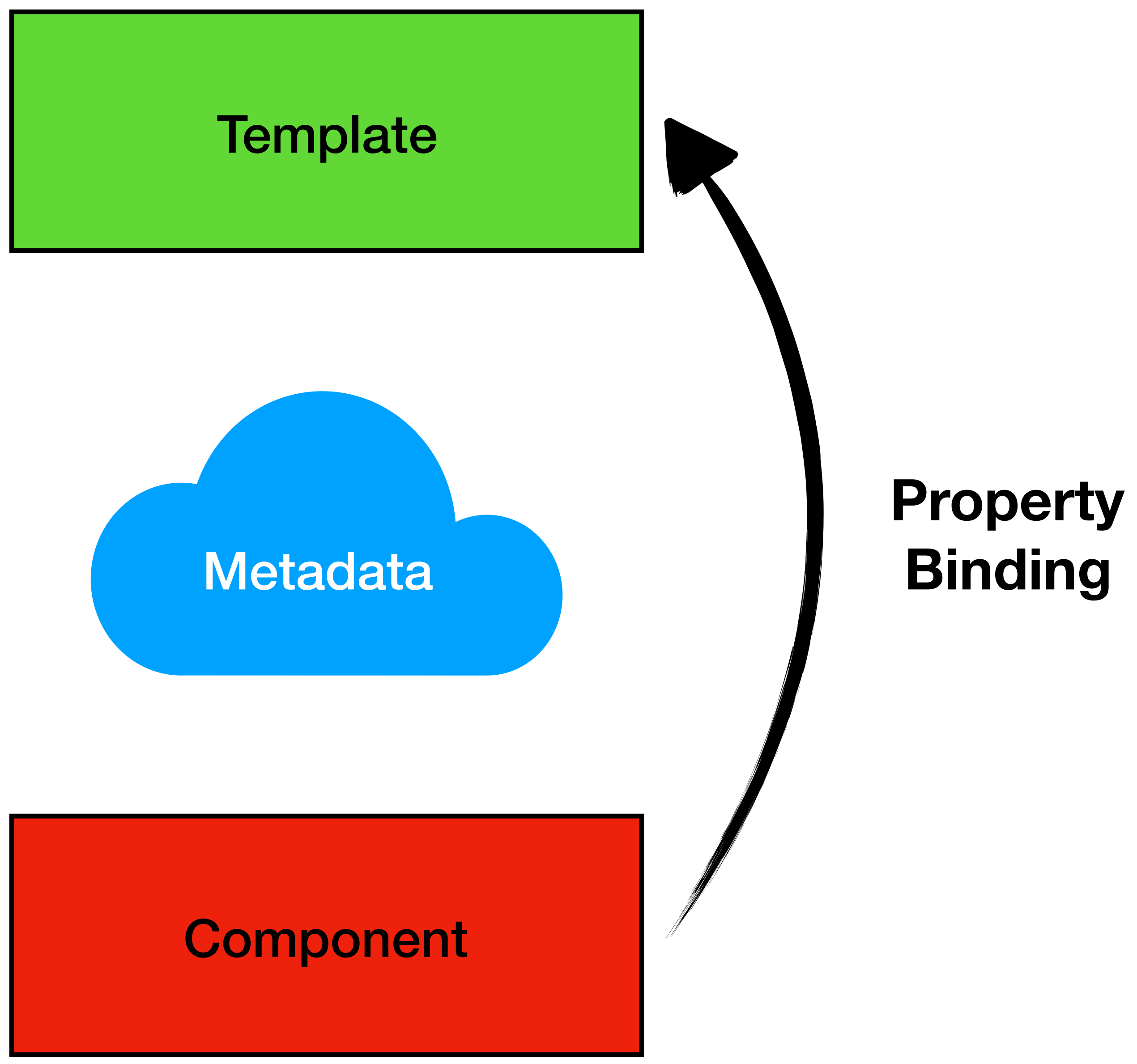
- Angular supports the data binding for coordinating parts of a template with the parts of a component.



Data Binding (contd.)

- Data Binding can be **One-way**, where data change in controller is reflected at view, or **Two-way**, where data changes are reflected in both directions; controller and view.
- The following types of bindings are supported by Angular:
- **One-way binding**
  - Case 1:
    - Interpolation - {{attribute-name}}
    - Property Binding - [attribute-name]
  - Case 2:
    - Event Binding - (event)
- **Two-way binding** - [(attribute-name)]

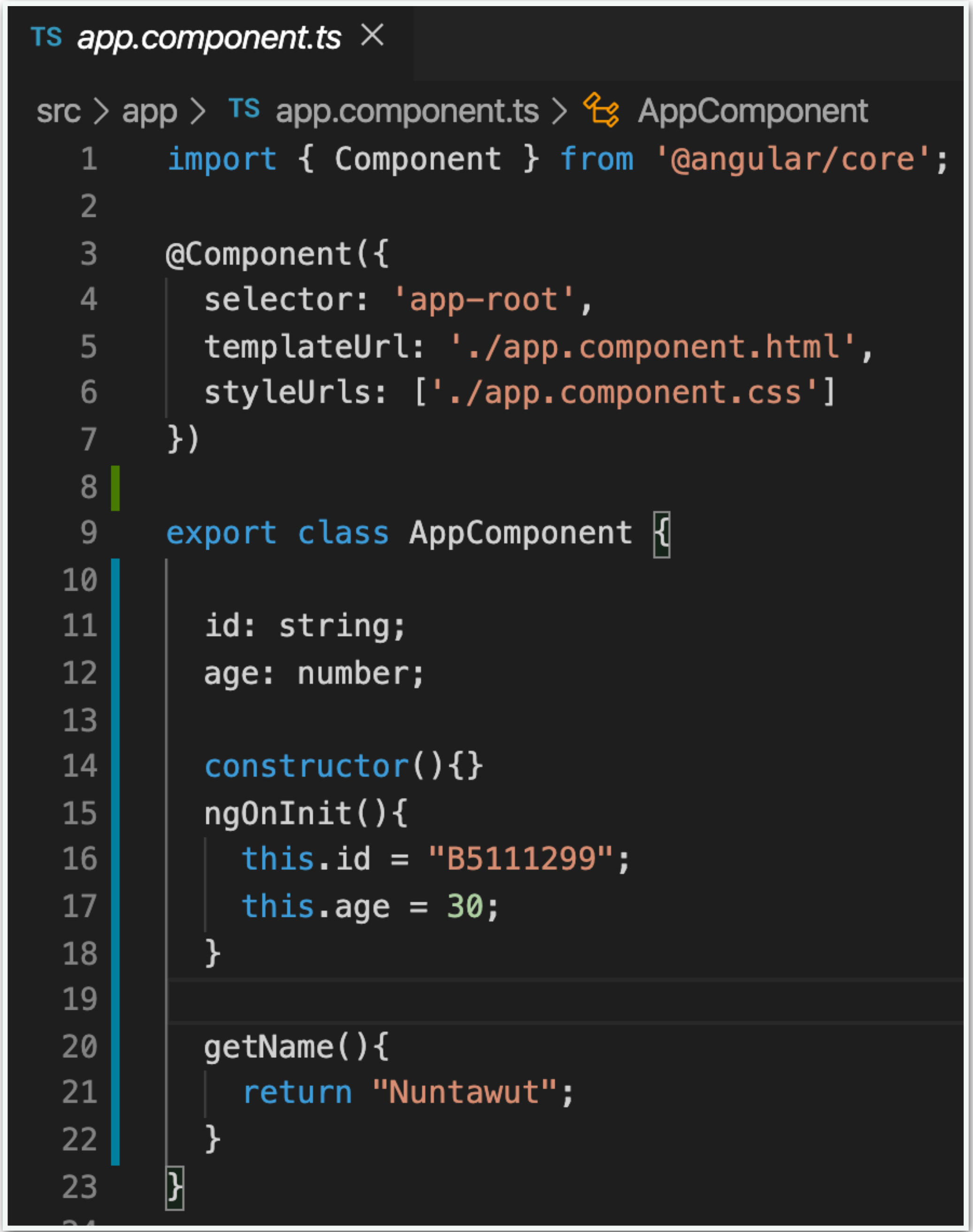
One-way binding



Interpolation

- One-way data binding is done by directive {{}}, called interpolation.
- Attributes defined in controller can be displayed in html using {{}}
- Interpolation pattern:
  - {{ propertyName }} or
  - {{ Expression }} or
  - {{ methodName( ) }}

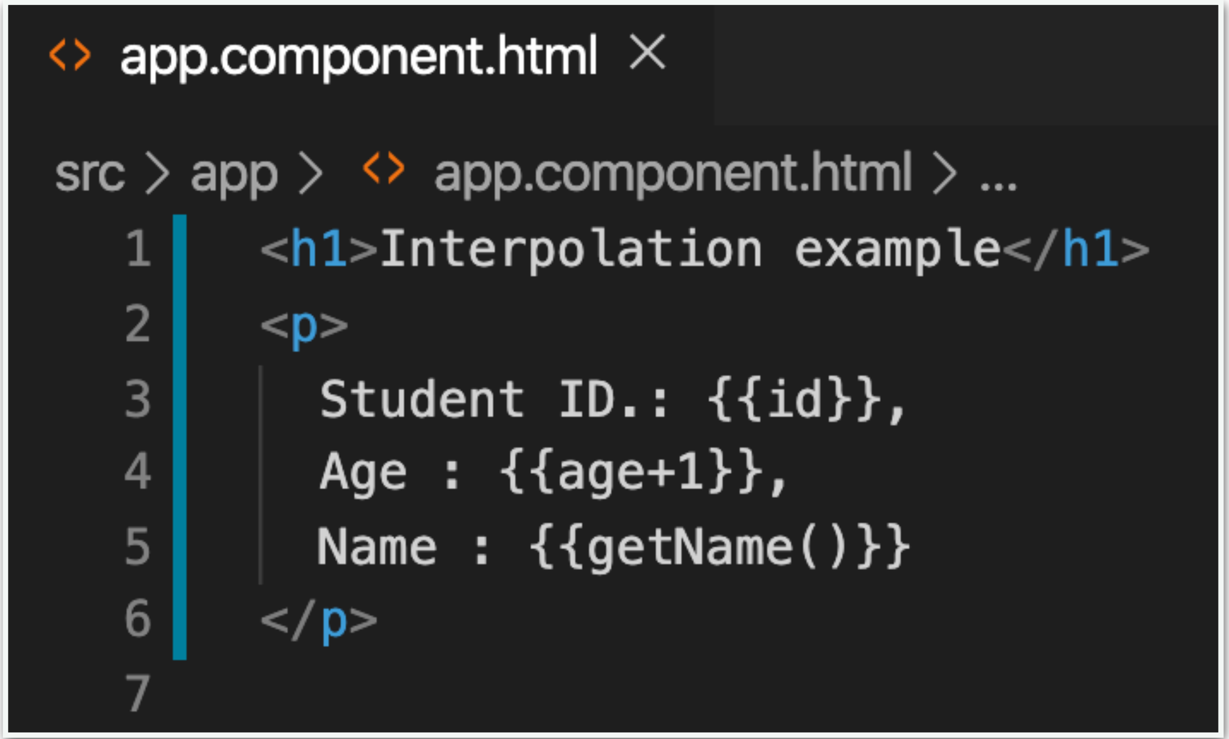
Interpolation example



app.component.ts

src > app > app.component.ts > AppComponent

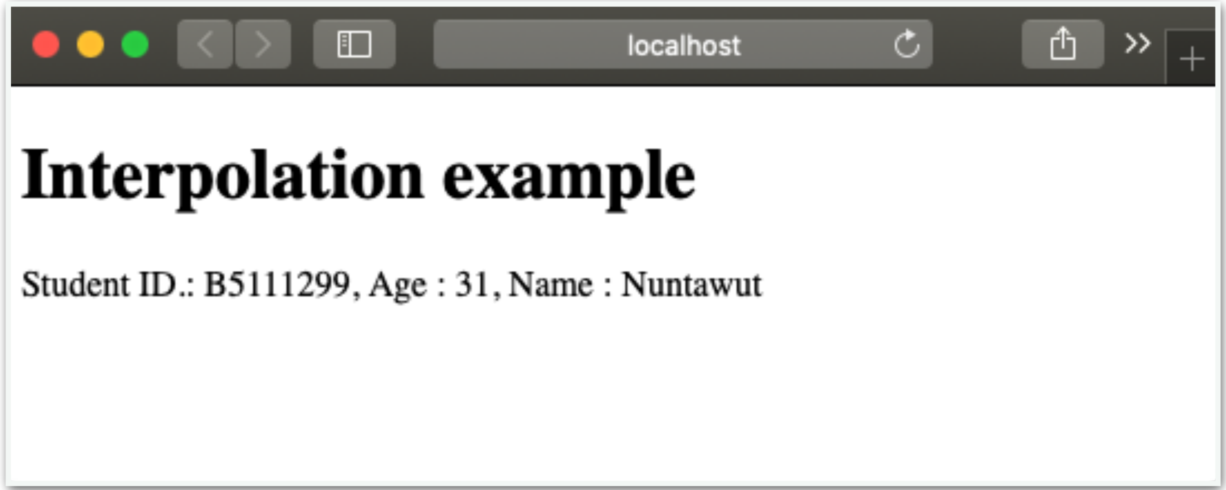
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  id: string;
  age: number;

  constructor(){}
  ngOnInit(){
    this.id = "B5111299";
    this.age = 30;
  }

  getName(){
    return "Nuntawut";
  }
}
```



app.component.html

src > app > app.component.html > ...

```
<h1>Interpolation example</h1>
<p>
   Student ID.: {{id}},
   Age : {{age+1}},
   Name : {{getName()}}
</p>
```



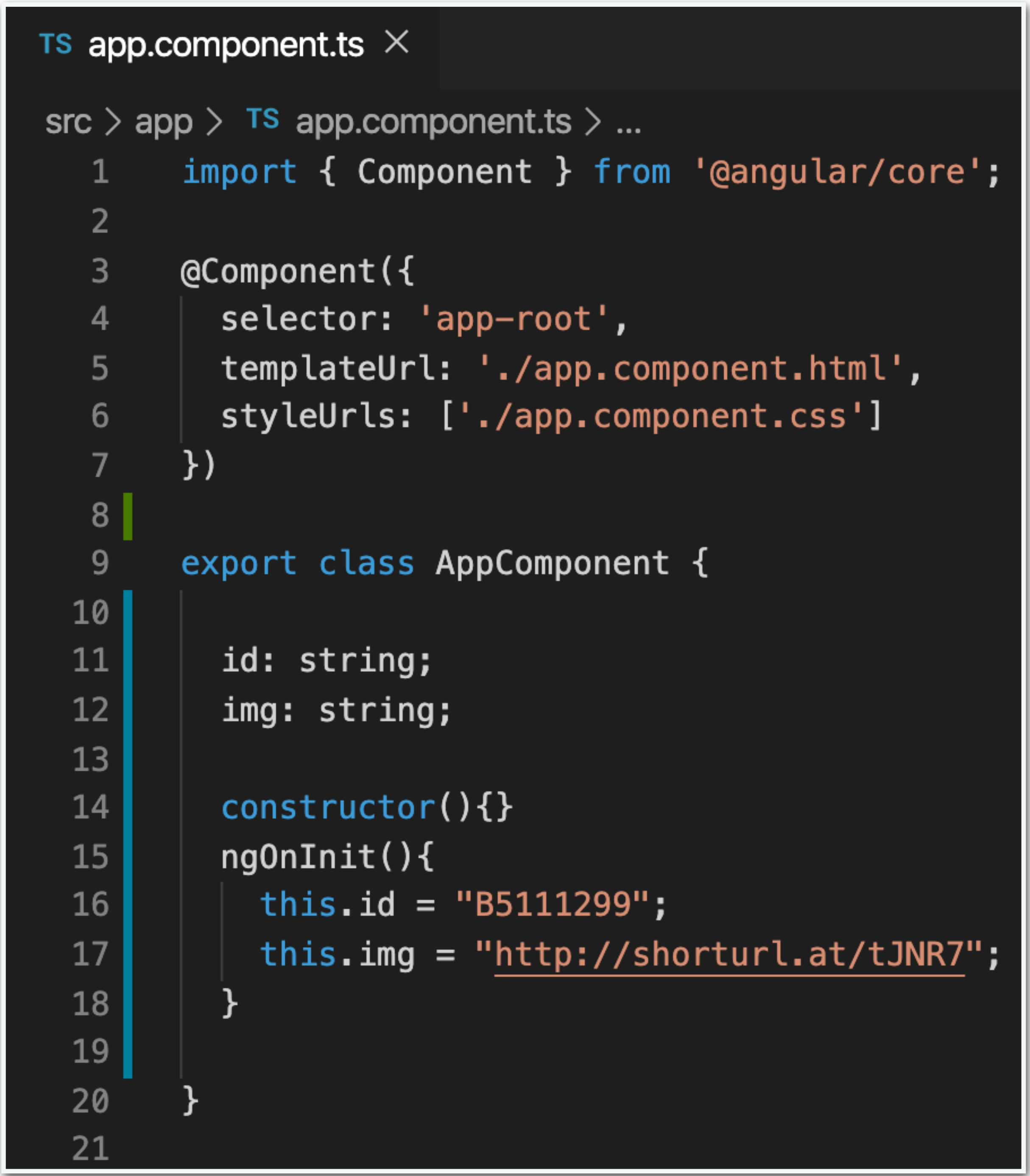
Interpolation example

Student ID.: B5111299, Age : 31, Name : Nuntawut

Property Binding

- Property binding is used for one-way data binding
- It binds controller attribute with DOM property of HTML elements
- Interpolation pattern:
    - [attributeName] = {{ propertyName }} or
    - [attributeName] = {{ Expression }} or
    - [attributeName] = {{ methodName( ) }}

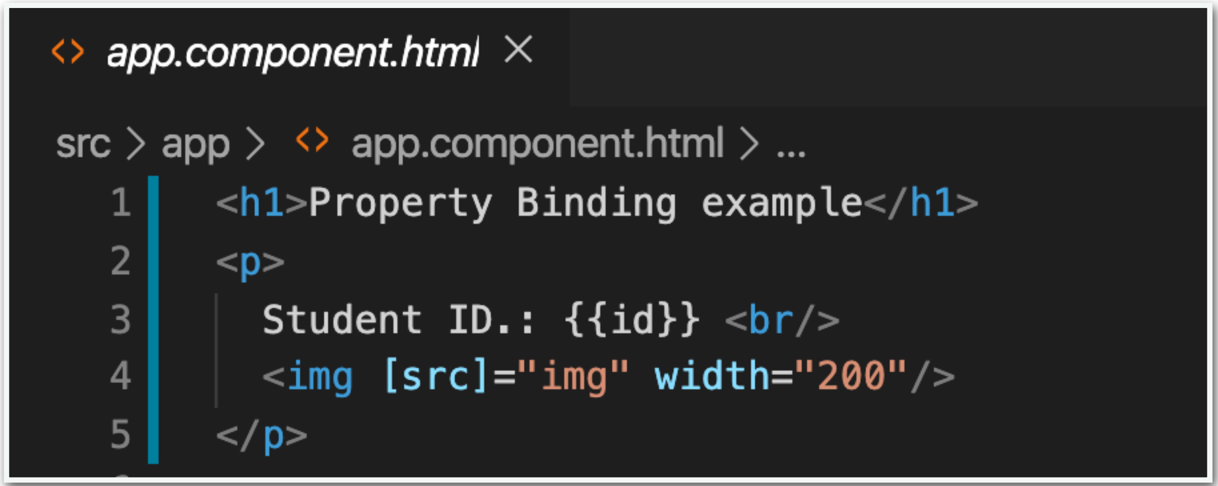
Property Binding Example



app.component.ts

src > app > app.component.ts > ...

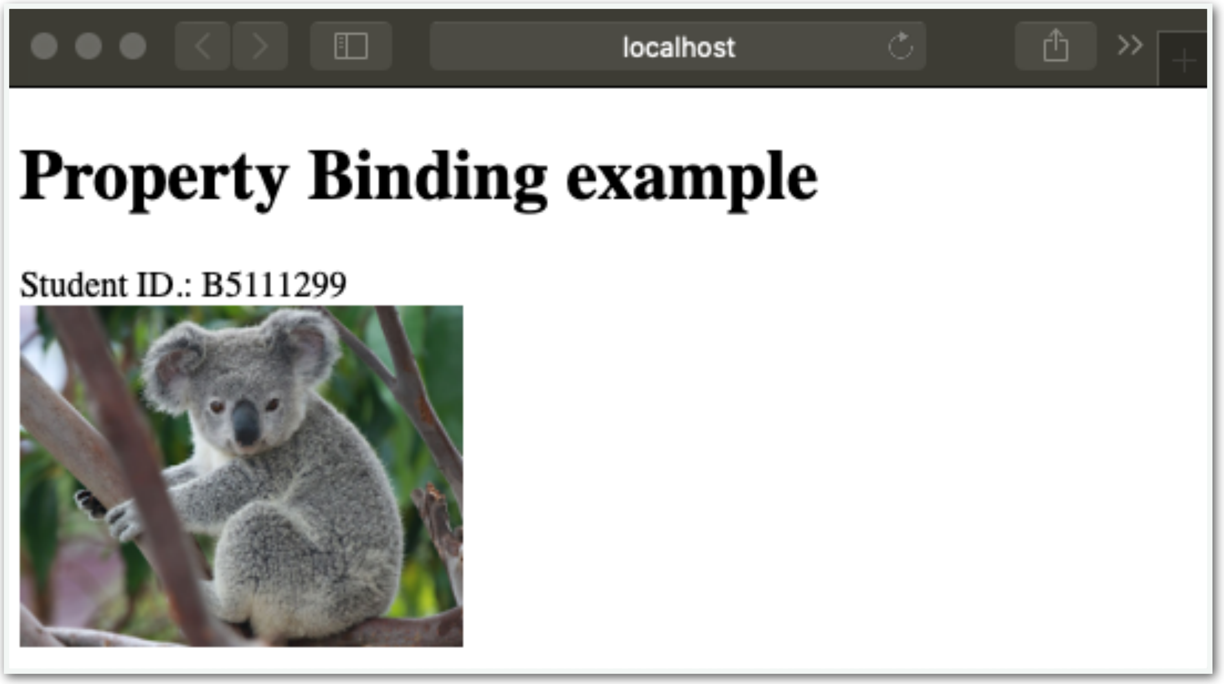
```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  id: string;
  img: string;

  constructor(){}
  ngOnInit(){
    this.id = "B5111299";
    this.img = "http://shorturl.at/tJNR7";
  }

}

```



app.component.html

src > app > app.component.html > ...

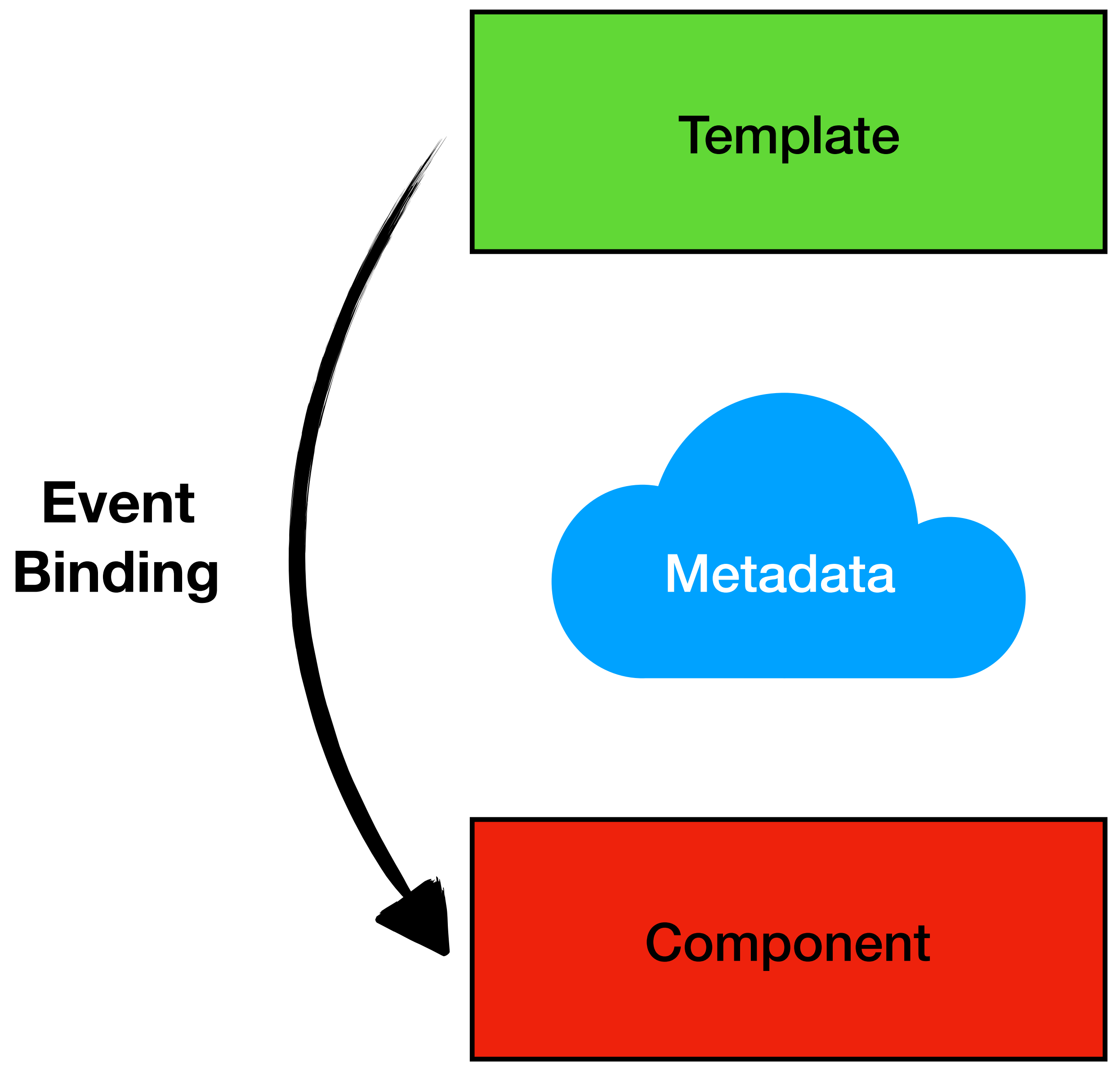
```html
<h1>Property Binding example</h1>
<p>
  Student ID.: {{id}} <br/>
  <img [src]="img" width="200"/>
</p>
```



Property Binding example

Student ID.: B5111299

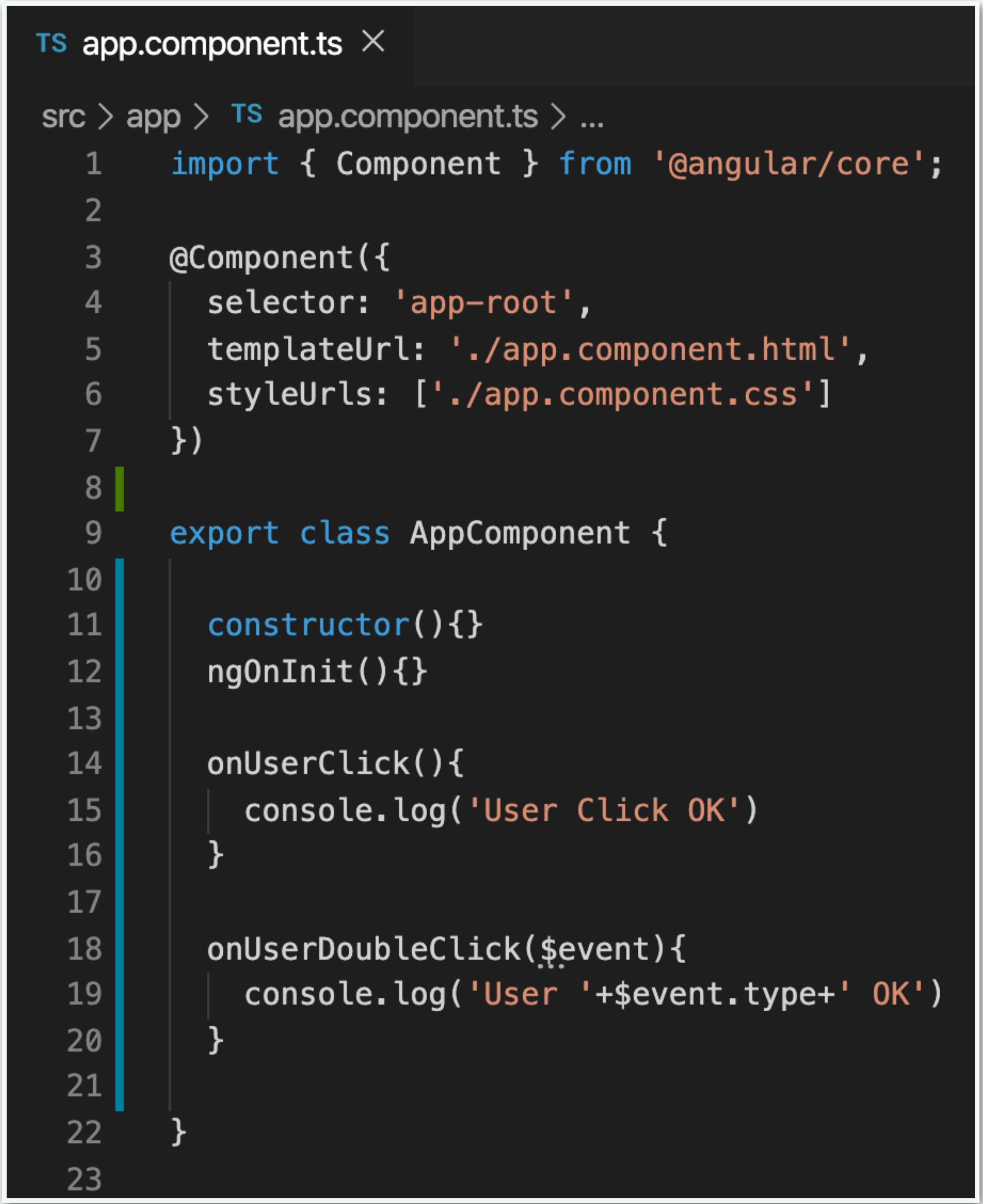
Event Binding



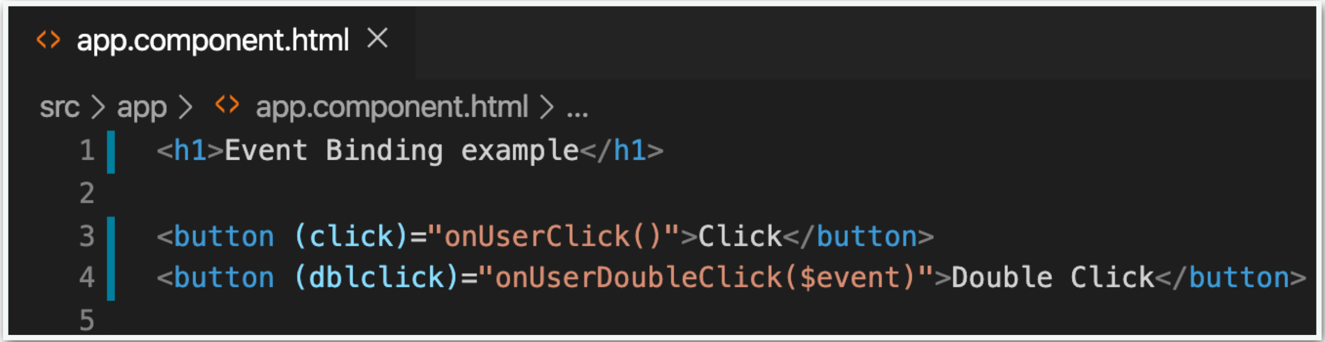
Event Binding (contd.)

- Html form events can be bound with component class methods using (event) directive.
- Followings are form events to be bind: https://www.w3schools.com/jsref/dom_obj_event.asp
- Pattern:
    - (eventName) = “actionName( )” or
    - (eventName) = “actionName($event)”

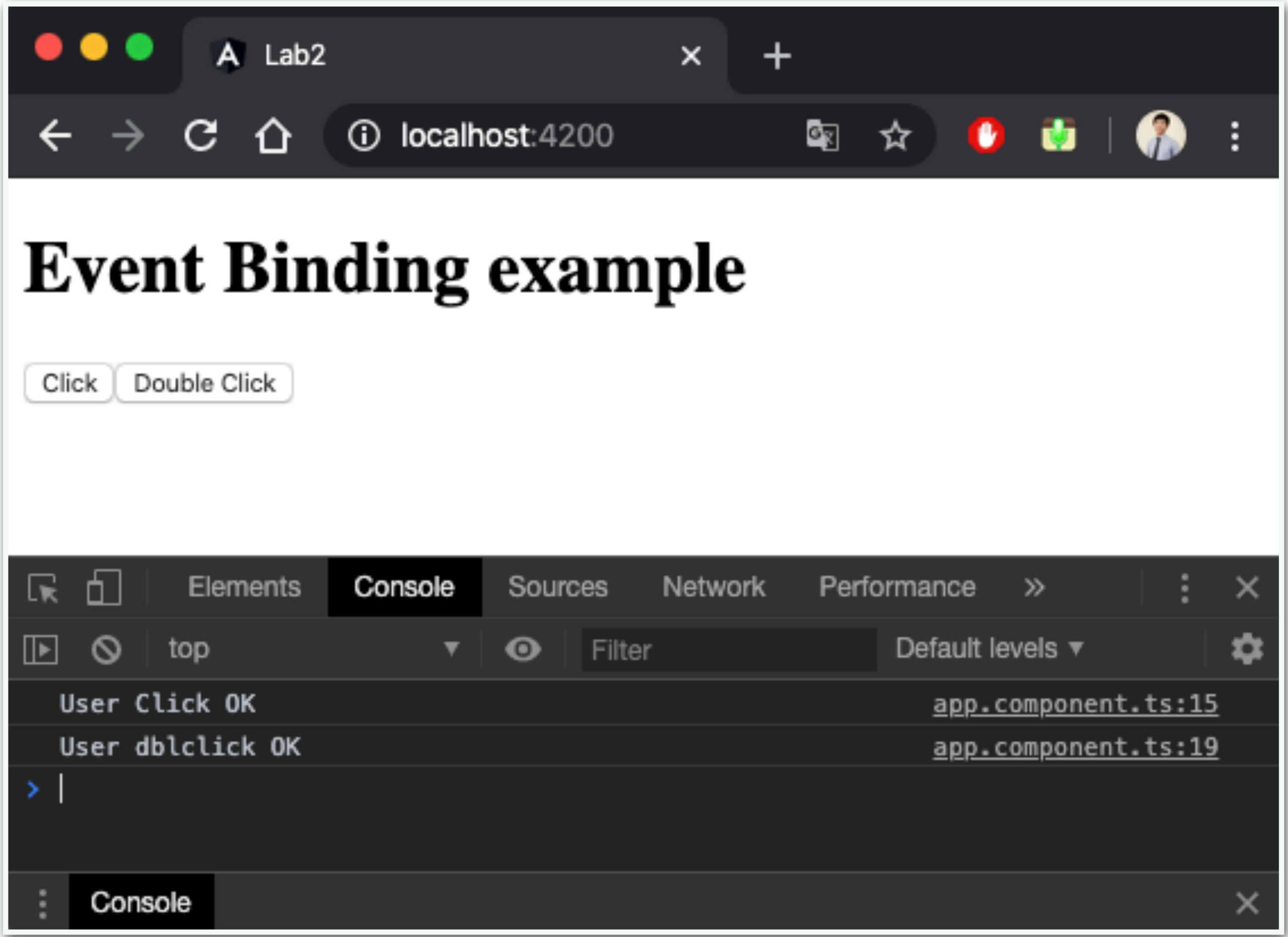
Event Binding Example


app.component.ts
src > app > app.component.ts > ...

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  constructor(){}
  ngOnInit(){}

  onUserClick(){
    console.log('User Click OK')
  }

  onUserDoubleClick($event){
    console.log('User '+$event.type+' OK')
  }

}
```

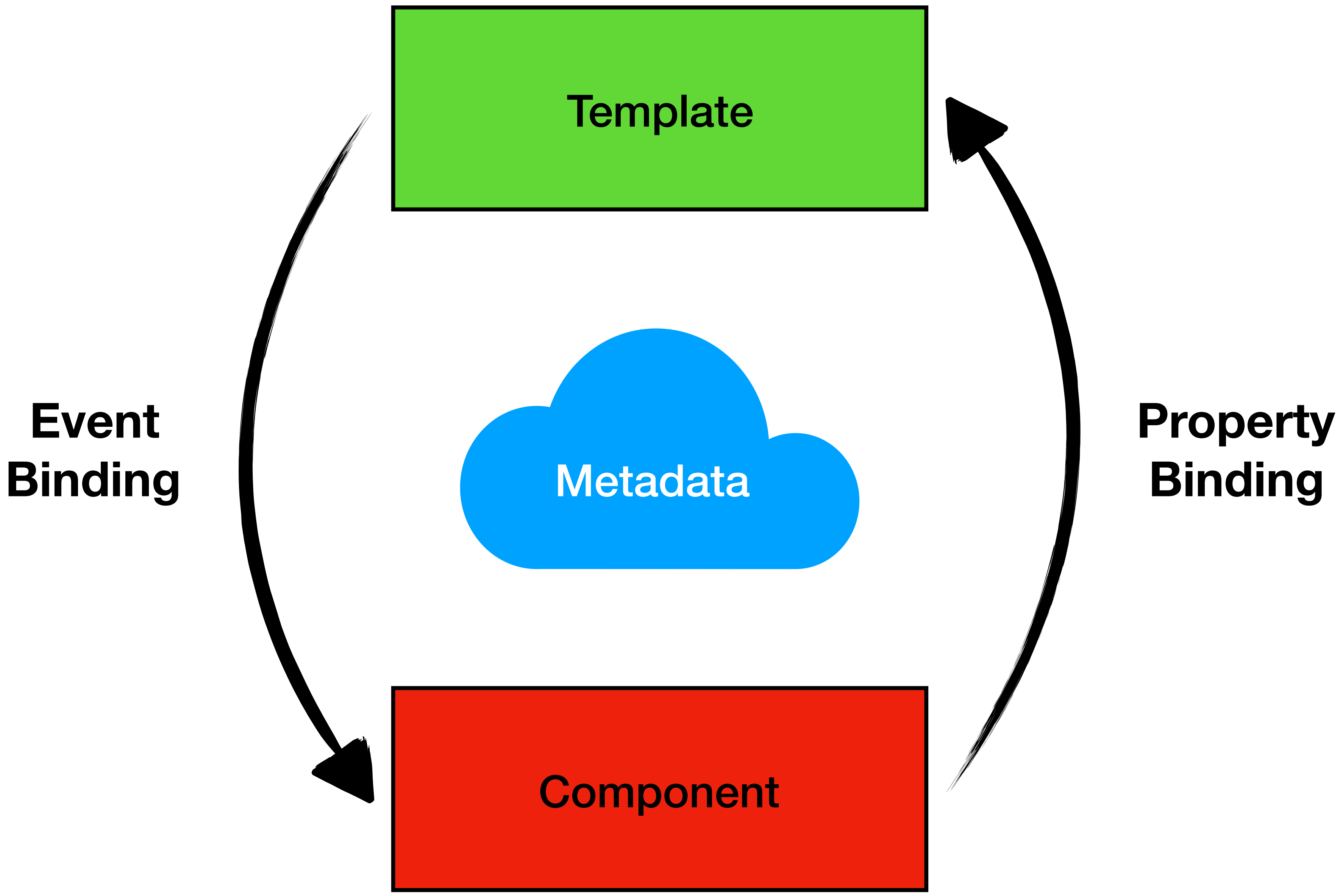

app.component.html
src > app > app.component.html > ...

```html
<h1>Event Binding example</h1>

<button (click)="onUserClick()">Click</button>
<button (dblclick)="onUserDoubleClick($event)">Double Click</button>
```



```
User Click OK                                  app.component.ts:15
User dblclick OK                               app.component.ts:19
```

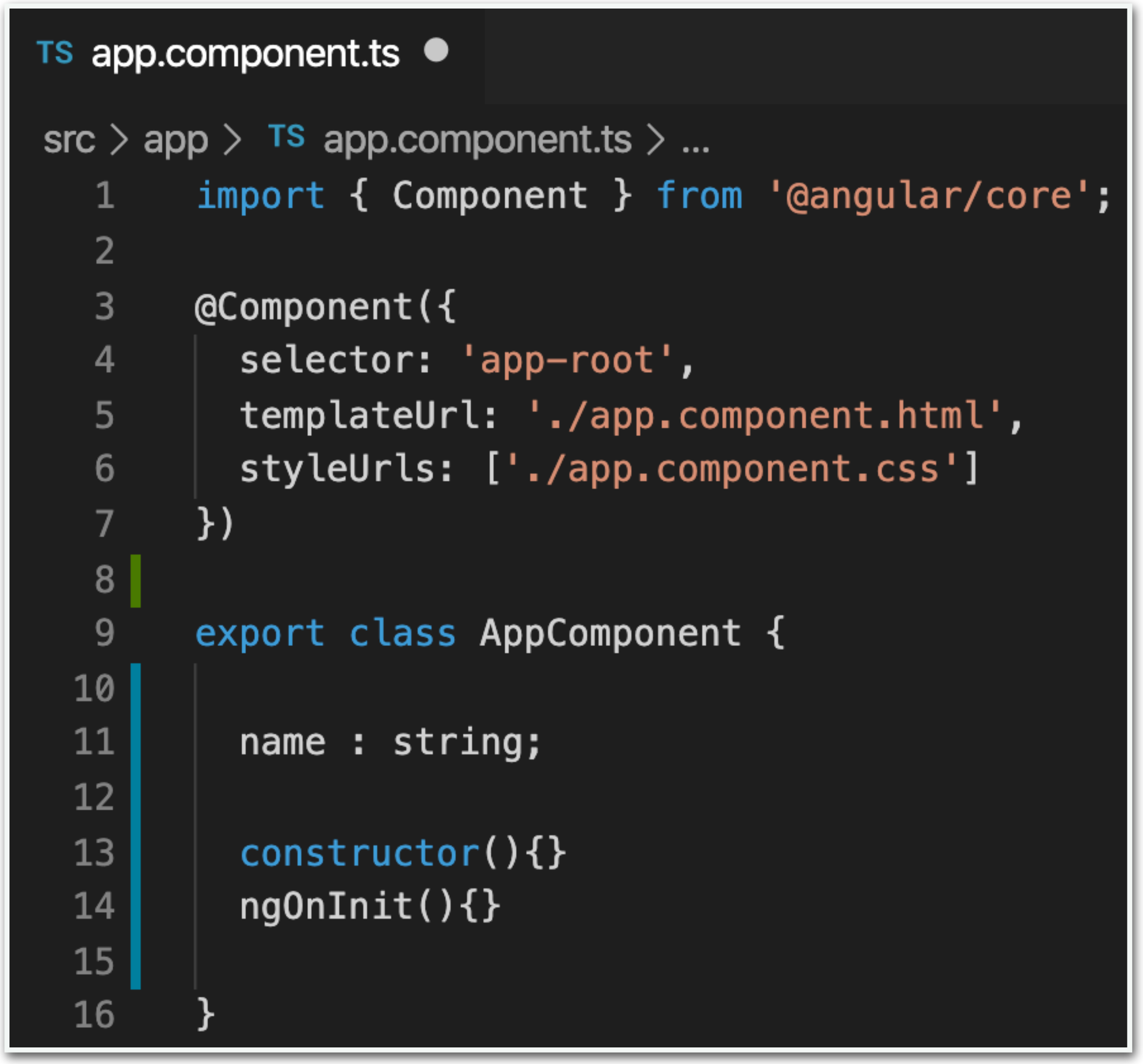
Two-way binding



Two-way binding (contd.)

- In two-way data binding, data will be changed in both directions; controller and view.
- If you change data at view then controller will be changed. If you change data at controller then view will be changed.
- Two-way data binding is done by directive [(ngModel)].
- It is used to bind html form input elements with controller class attributes.

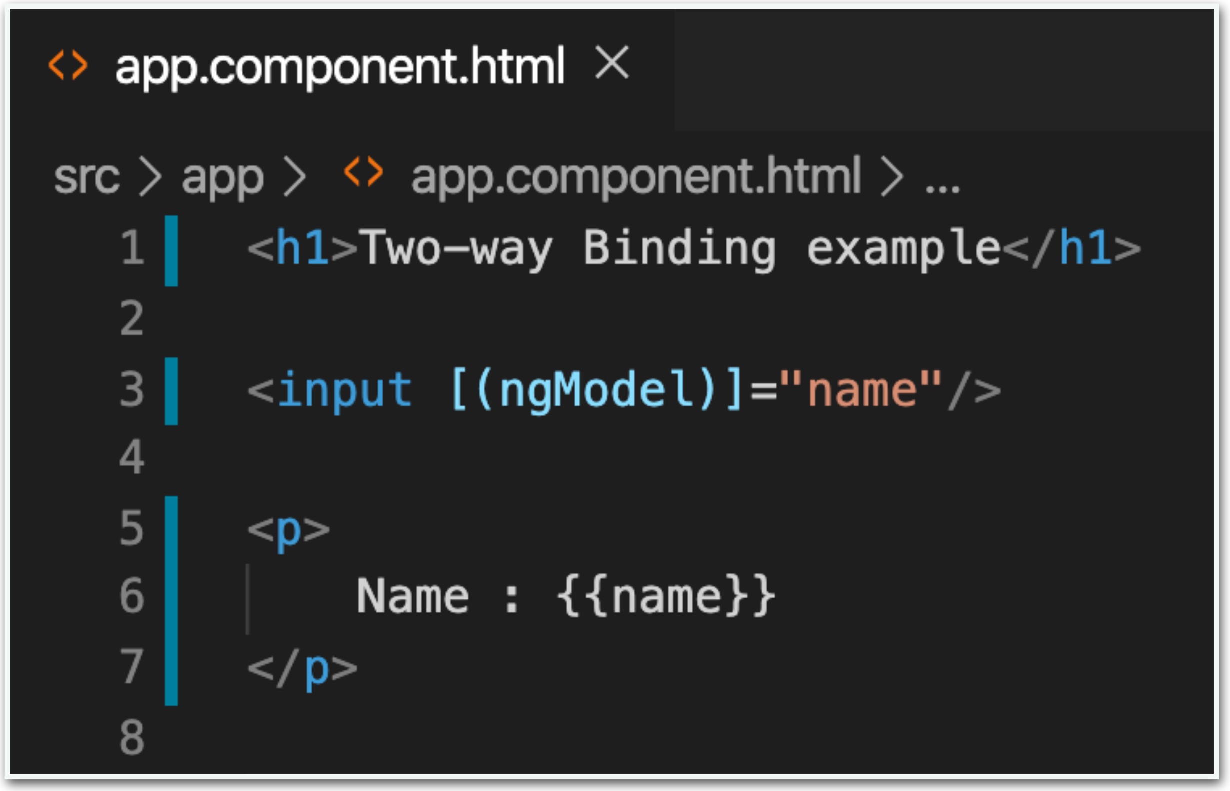
Two-way binding Example



TS app.component.ts ●

src > app > TS app.component.ts > ...

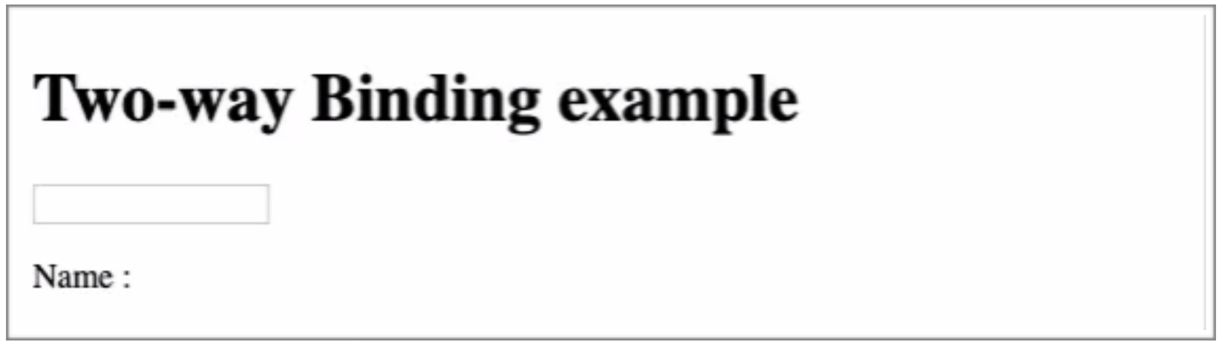
```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  name : string;

  constructor(){}
  ngOnInit(){}

}
```



app.component.html ×

src > app > app.component.html > ...

```html
<h1>Two-way Binding example</h1>

<input [(ngModel)]="name"/>

<p>
    Name : {{name}}
</p>

```



Two-way Binding example

Name :