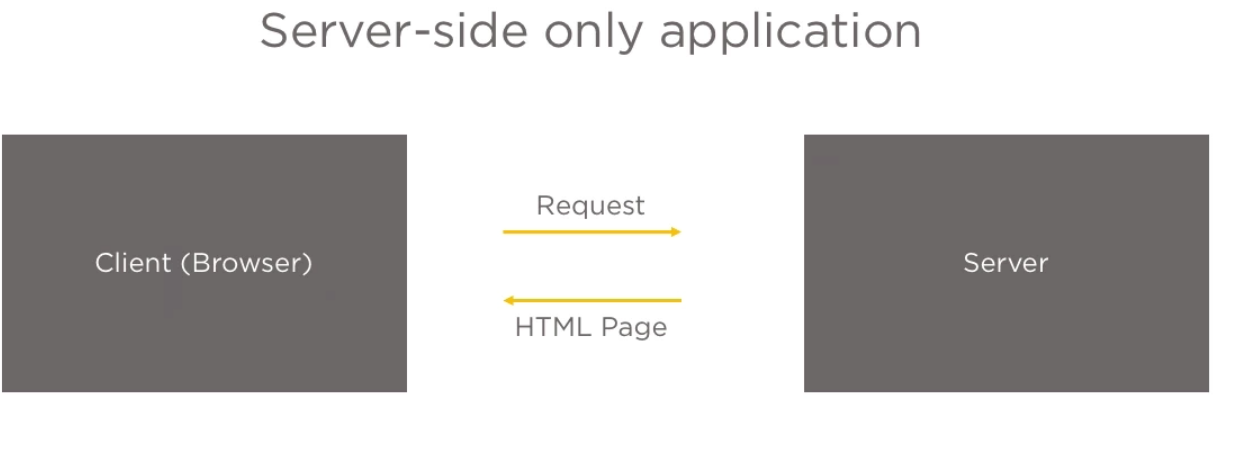
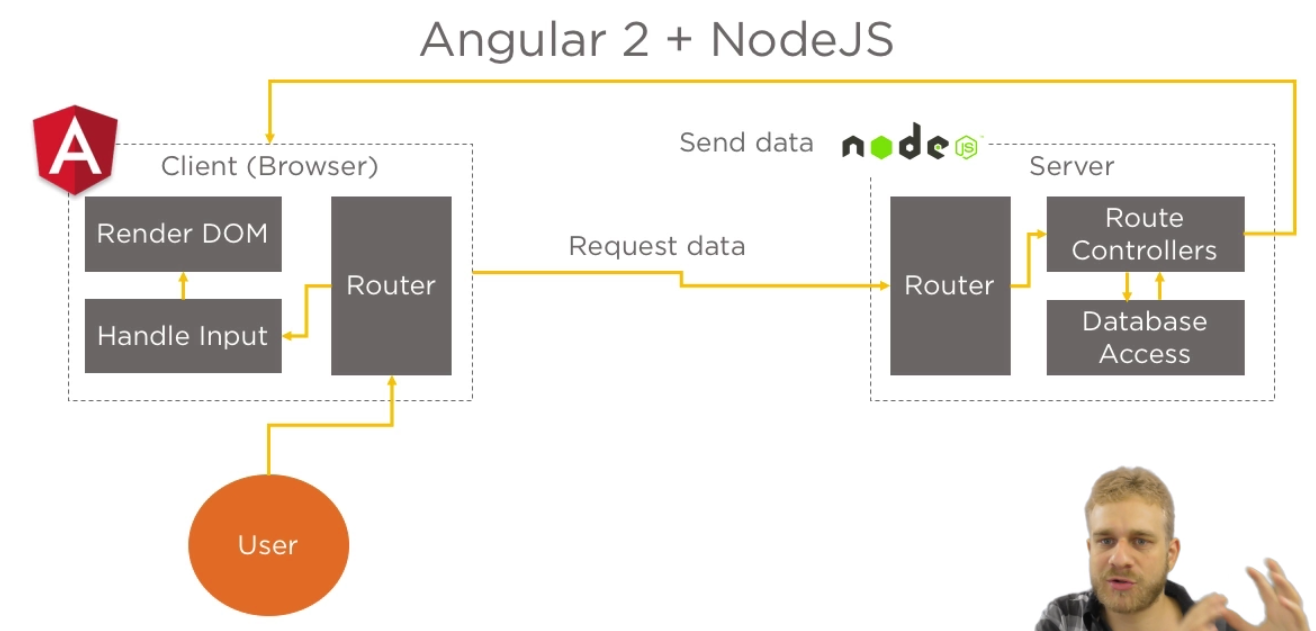
How Angular 2 and Node JS works



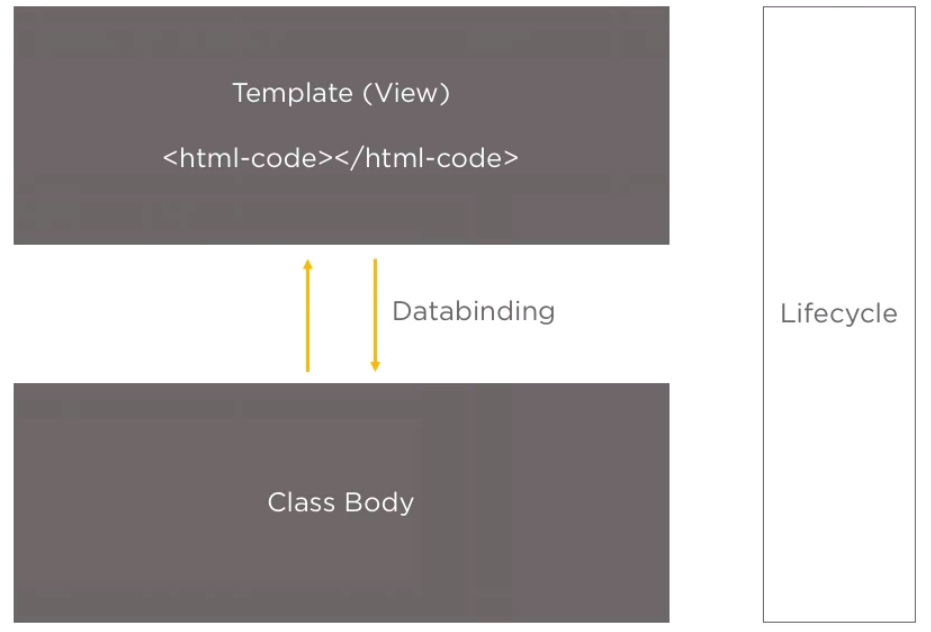


Rendering an Angular 2 App with Node JS /Express

We will return only one view from node JS. Angular allows us to create single page applications

bundle.js: It is the Bundled Angular 2 App.

Angular2 breaks the code into multiple component. Each component contains piece of code which contains the template (HTML code) which will be displayed in the browser in the end.



Important Files:

**polyfill.js**: Application runs in multiple browser.

**main.js**: Used to bootstrap the Angular application

**app.module.ts**: Describe what the application consists of. It bootstraps the root component which holds all the required informations.

Angular application can be grouped together by multiple components. Each component will have the template so that HTML code can be rendered.

**Decorator**: It is the typescript feature that allows us to attach additional information to the class.

Two-way data binding can be achieved using ngModel. ngModel is present in FormsModule

**Syntax**: [(ngModel)] =”fieldname”.

**Create a model:**

Syntax1:

export class Message{

content :string;

constructor(content :string){this.content=content;}

}

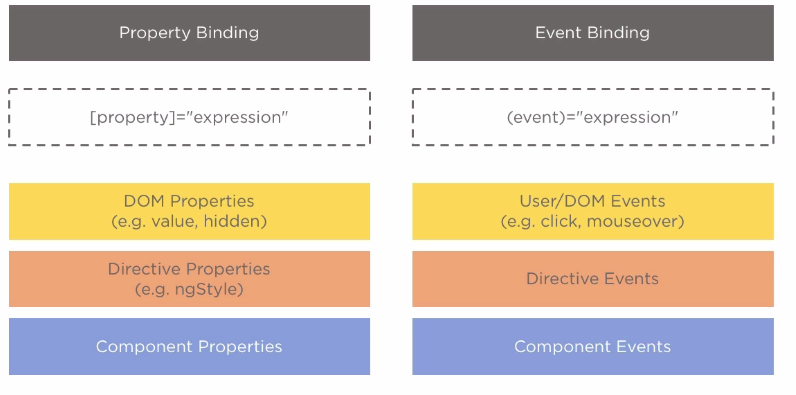
**Syntax2**:

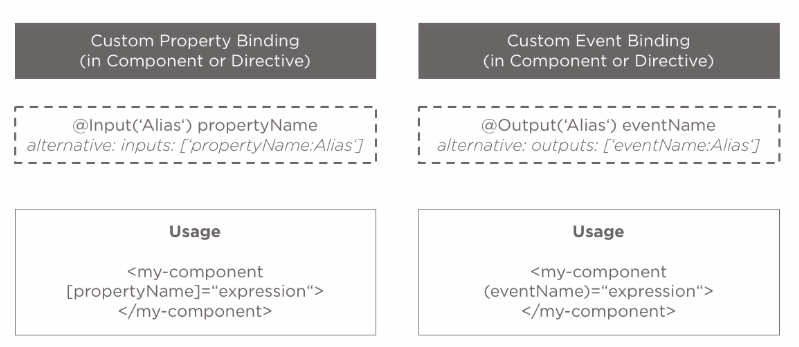
export class User{

constructor(public email:string, public firstname?:string){}

}

Making a field optional by adding “?”.





**Passing property from parent component to child component.**

We can either use:

inputs: [‘message’]

or

export class MessageComponet{

//Alias Name

//@Input('inputMessage') message:Message;

Or

//@Input message:Message;

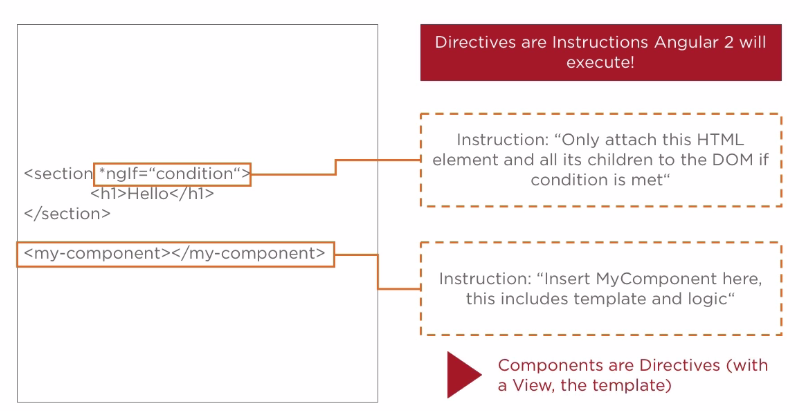
}

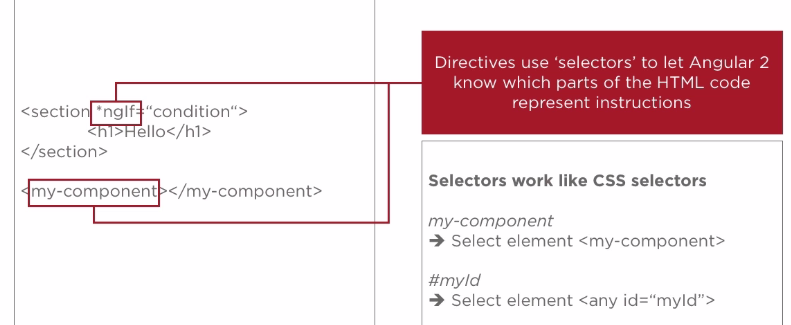
**Passing value from child component to parent component:**

Using Event Emitter, we can emit an event from child to parent.

**Directives:**

Directive are instructions Angular 2 will execute at run time. When it finds, them it knows what to do with them.





Attribute Directive: [ngStyle]="{backgroundColor:color}"

Structural Directive: \*ngFor = "let message of messages "

#var: This expression allows to refer to the user input value.

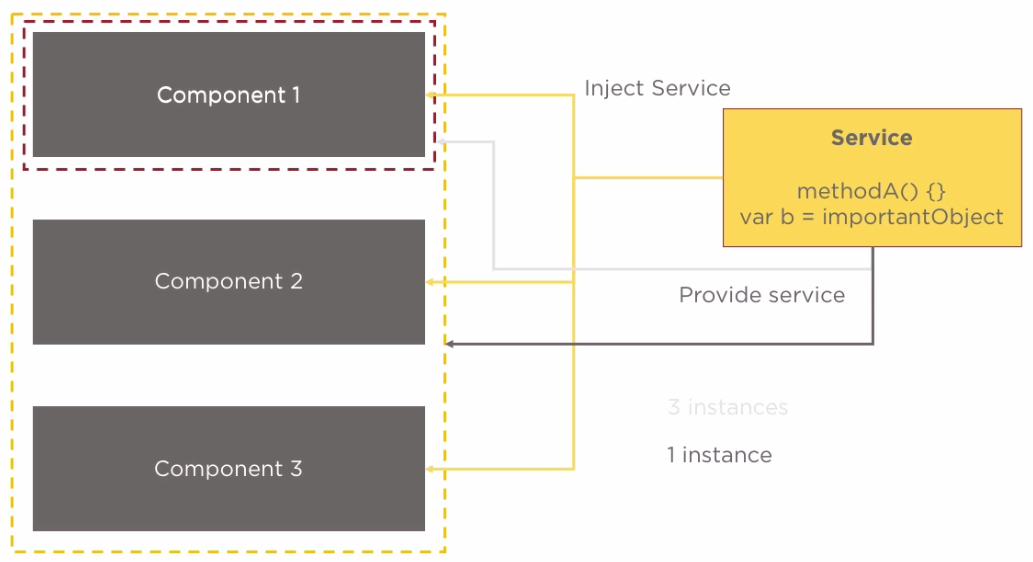
Ex: In the following example #input is used to refer the input text value. On click the value will be passed to the “onSave” method.

<input type="text" id="content" class="form-control" #input>

<button class="btn btn-primary" type="submit" (click)="onSave(input.value)">Save</button>

**Services:**





OnInit life cycle hook method.

Component will implement the OnInit interface.

ngOnInit () method is used as an implemented method. The method will be called when the component and property are initialized.

Array and Object in JavaScript are referenced type.

export class MessageListComponent implements OnInit

ngOnInit () {}

Dependency Injector is a hierarchical injector which means it can be setup in a way where one instance can be create for all the component or individual instance for individual component.

The input with which ngModel is placed should be registered as a control of the form which was automatically derived by Angular 2.

How to pass the form Angular 2 created:

<form (ngSubmit)="onSubmit()" #f="ngForm">

#f: Local reference to the form. This will give access to the form HTML element.

To get access to the form created in the back ground we write #f = “ngForm”.

With this command, we tell angular that don’t give me access to the form HTML element. Instead give the access to the JavaScript object created by Angular.

Path: messages it means messages will be appended to the current URL.

Path: /messages = it means the messages url will be replaced by base url+messages

{path:'',redirectTo:'/messages',pathMatch:'full'},

{path:'messages',component:MessagesComponent},

{path:'auth',component:AuthenticationComponent}

pathMatch:’full’== Without this Angular will always redirect to messages even if the path is auth.

To register the routes in Angular2 we use the RouterModule.

Ex: const APP\_ROUTES:Routes = [

{path:'',redirectTo:'/messages',pathMatch:'full'},

{path:'messages',component:MessagesComponent},

{path:'auth',component:AuthenticationComponent}

]

RouterModule.forRoot(APP\_ROUTES);

To display the router loaded component we make use of <router-outlet>

routerLinkActive="active"

The above code will apply “active” class to the active link.

**Adding Child Routes:**

export const AUTH\_ROUTES:Routes = [

{path:'',redirectTo:'signup',pathMatch:'full'},

{path:'signup',component:SignUpComponent},

{path:'signin',component:SignInComponent},

{path:'logout',component:LogoutComponent}

]

**Use it in the main routing component**

const APP\_ROUTES:Routes = [

{path:'',redirectTo:'/messages',pathMatch:'full'},

{path:'messages',component:MessagesComponent},

{path:'auth',component:AuthenticationComponent,children:AUTH\_ROUTES}

]

**Template Form approach: Angular will create the Form**

**Reactive Form approach: We can create Form group**

Angular2 can send request behind the scenes. It does not re-render the page but it can still send ajax request. The request can be handled in the back ground.

Such a request is sent with a concept/Object called Observables. These are objects coming from third party library which allow us to handle asynchronous task.

Such a HTTP request is asynchronous task. Angular HTTP services are the built-in service which allow us to connect to web and handle request in the background.

Once we get the response another method provided by the observables, which allow us to map the response.

