**Angular Animations**

* Angular provides a library for configuring CSS animation dynamically.
* Angular can implement CSS 2D and 3D animations.
* Angular uses CSS
  + Transition
  + Transform
  + Animate
* Angular animation library is configured with

“**BrowserAnimationsModule”**

* It is provided with “**@angular/animations**”
* Angular animation methods

|  |  |
| --- | --- |
| **Animation Method** | **Description** |
| trigger() | It is used to configuration animations effects in a component. |
| state() | It is used to configure animation state.  Angular animations have 2 states   * Initial State * Final State   In **initial state** we define effects for element to apply before transformation. It is represented by using “[void=>\*]”    In **final state** we define effects for element to apply after transformation.  It is represented by using “[\*=>void]”  Note: CSS uses them as “@keyframes” |
| style() | It is used to define the effects to apply. You can use all CSS attributes. |
| transition() | It configures the state and time for animation. |
| animation() | It configures the animation duration. |

* Animations for any component are configured in the component meta data by using “animations[]”

Syntax:

@Component({

selector: ‘’,

templateUrl: ‘’,

styleUrls: [],

animations:[]

})

**Animations collection uses animation functions.**

Syntax:

animations:[

trigger(‘EffectName’,[

state(‘initial’, styles({attribute:value, attribute:value})),

state(‘final’, styles({attribute:value, attribute:value)),

transition(‘initial=>final/void=>\*’, animate(timeInterval)),

transition(‘final=>initial/\*=>void’, animate(timeInterval))

]) /**/ Trigger End**

]

**You have to apply trigger to any HTML element**

<div [@TriggerName/EffectName]> </div>

Ex:

* Add Angular Material to your application
* Import BrowserAnimations Module in “app.module.ts”
* Add a new component

**Animationdemo.component.ts**

import { animate, state, style, transition, trigger } from '@angular/animations';

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-animationdemo',

templateUrl: './animationdemo.component.html',

styleUrls: ['./animationdemo.component.css'],

**animations: [**

trigger('ZoomEffect', [

state('initial', style({

width: '200px',

height: '200px',

transform: 'rotate(0deg)'

})),

state('final', style({

width: '400px',

height: '400px',

transform: 'rotate(360deg)'

})),

transition('initial=>final', animate('3000ms')),

transition('final=>initial', animate('3000ms'))

])

]

})

export class AnimationdemoComponent implements OnInit {

public animationState = 'initial';

public zoomText = 'Zoom In';

constructor() { }

ngOnInit(): void {

}

public ZoomClick(){

this.animationState = (this.animationState=='initial')?'final':'initial';

this.zoomText = (this.zoomText=='Zoom In')?'Zoom Out':'Zoom In';

}

}

**Animationsdemo.component.html**

<div class="container-fluid">

<h2>Animations</h2>

<div class="form-group text-center">

<button (click)="ZoomClick()" class="btn btn-primary">{{zoomText}}</button>

</div>

<div class="form-group">

<div (mouseover)="ZoomClick()" style="margin:auto; justify-content: center; align-items: center;">

<img [@ZoomEffect]="animationState" src="assets/shirt.jpg">

</div>

</div>

</div>

**Angular Security**

* Reporting vulnerabilities
* Preventing XSS [Cross-Site-Scripting] Attacks
* Preventing Request Forgery
* Angular provides built-in protections against web application attacks.
* Angular also provides options to authenticate and authorize users to give access to resources in application.

**Preventing Cross-Site-Scripting Attacks (XSS)**

* *HTML* is used when interpreting a value as HTML component.

**innerHTML = “<b>Hello !</b>”**

* *Styles* is used when binding to CSS into style property.
* *URL* is used for url properties such as “<a href=’’>”.
* **Resources URL as** URL that will be loaded when code is executed.

**<script src=””>**

* Angular uses “**DomSanitizer.Sanitize()” method** for handling XSS.
* Angular also provides methods to by-pass the security XSS
  + bypassSecurityTrustHTML
  + bypassSecurityTrustScript
  + bypassSecurityTrustUrl
  + bypassSecurityTrustResourceUrl
  + bypassSecurityTrustStyleUrl

**Ex:**

**Securitydemo.component.ts**

import { Component, OnInit } from '@angular/core';

import { DomSanitizer } from '@angular/platform-browser';

import { timeStamp } from 'console';

@Component({

selector: 'app-securitydemo',

templateUrl: './securitydemo.component.html',

styleUrls: ['./securitydemo.component.css']

})

export class SecuritydemoComponent implements OnInit {

public xssUrl = 'javascript:alert("Hello !")';

public trustedUrl;

public xssVideoUrl = 'https://www.youtube.com/embed/xDwNeVEIOeU';

public trustedVideoUrl;

constructor(private sanitizer: DomSanitizer) { }

ngOnInit(): void {

this.trustedUrl = this.sanitizer.bypassSecurityTrustUrl(this.xssUrl);

this.trustedVideoUrl = this.sanitizer.bypassSecurityTrustResourceUrl(this.xssVideoUrl);

}

}

**Securitydemo.component.html**

<div class="container-fluid">

<h2>XSS URL - Don't Trust</h2>

<a [href]="xssUrl">Click Here - Don't Trust</a>

<h2>XSS URL - Trust Me</h2>

<a [href]="trustedUrl">Click Here - Trust Me</a>

<h2>YouTube Video</h2>

<iframe [src]="trustedVideoUrl" width="400" height="300"></iframe>

</div>

**Authorization**

* Authorization is the process restricting access to the resources in application.
* You can configure components so that they are accessible only to the user when authentication is successful.
* Authentication is the process of verifying user credentials, security token etc.
* You can restrict access to any component by using “Route Guards”.
* It prevents users from navigating to any specific location without proper authentication.
* You can secure the route path.
* A route guard allows to configure custom logic and functionality, where we can verify the user credentials and allow or restrict access.
* For any component if you want a restricted access then you can generate a route guard.
* **Angular provides various route guards:**

|  |  |
| --- | --- |
| **CanActivate** | It restricts access to specific route. |
| **CanActivateChild** | It restricts access to child route. |
| **CanDeactivate** | It is used to restrict the user to exit the route. |
| **Resolve** | It is used to access data from any API before route activation. |
| **CanLoad** | It is authorizing lazy routes. |

* All route guards and return different types of values. Usually a boolean value is used to confirm user authentication.
* Route Guards are generated for every component by using

> ng generate guard <guard-name>

**Ex:**

* Go to your project enabled with routing
* Add a new component

> ng g c manager-home

* Generate a guard for component

> ng generate guard manager-guard

**Manager-guard.guard.ts**

import { Injectable } from '@angular/core';

import { Router, CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree } from '@angular/router';

import { Observable } from 'rxjs';

import { DataService } from './data.service';

@Injectable({

providedIn: 'root'

})

export class ManagerGuardGuard implements CanActivate {

constructor(private data: DataService, private router: Router){}

public users = [];

public username = 'David';

public password = 'mng12';

canActivate(

next: ActivatedRouteSnapshot,

state: RouterStateSnapshot): boolean {

// your authentication logic here..

this.users = this.data.GetUsers();

for(var user of this.users) {

if(user.Name==this.username && user.pwd==this.password) {

return true;

} else {

this.router.navigate(['login']);

}

}

return false;

}

}

(or)

this.users = this.data.GetUsers();

for(var user of this.users) {

if(user.Name!==this.username && user.pwd!==this.password) {

this.router.navigate(['login']);

return false;

}

}

return true;

* Maintain data in “data.service.ts”

public GetUsers(){

return [

{Name: 'John', Role: 'Admin', pwd: 'admin12'},

{Name: 'David', Role: 'Manager', pwd: 'mng12'}

];

}

* **Go to app-routing.module.ts**

{path: 'manager', component: ManagerHomeComponent, canActivate: [ManagerGuardGuard]},

* Try changing the user name and password if it is mismatching it will restrict.

**Unit Testing**

**Building and Deploying**