1. **Create a textbox**

app.component.html

<input type="text" [(ngModel)]="name">

<br>

{{name}}

app.module.ts

import { FormsModule } from '@angular/forms';

 imports: [

    BrowserModule,

    FormsModule

  ],

1. **Install bootstrap**

npm install --save bootstrap

1. **Creating a new component**

**Server.component.ts**

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

//Decorator

@Component({

    //Metadata

    selector: 'app-server',

    //Template:

    templateUrl: './server.component.html'

}

)

export class ServerComponent {

}

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

app.module.ts

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

import { FormsModule } from '@angular/forms';

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

import {ServerComponent} from './server/server.component';

@NgModule({

  declarations: [

    AppComponent,

    ServerComponent

  ],

  imports: [

    BrowserModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

1. **Creating components with CLI and nesting components**

**ng generate component servers**

1. **Inline style**

template: '<app-server></app-server><app-server></app-server>',

 template: `

  <app-server></app-server>

  <app-server></app-server>`

1. **Component styles**

<div class="container">

    <div class="row">

        <div class="col-xs-12">

            <h2>Hello from app component</h2>

            <hr>

            <app-servers></app-servers>

        </div>

    </div>

</div>

1. **app.component.css**

h2{

color: darkblue;

}

1. **Inline styles**

@Component({

  selector: 'app-root',

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

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

  styles: [`

  h2{

    color: red;

  }`

  ]

1. **Understanding Component Selectors**

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

@Component({

  //selector: 'app-root',

  selector: '[app-root]',

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

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

  styles: [`

  h2{

    color: red;

  }`

  ]

})

export class AppComponent {

  name = 'Sindhu';

}

1. **Using the selectors in div**

**Servers.component.ts**

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

@Component({

  selector: '[app-servers]',

  template: `

  <app-server></app-server>

  <app-server></app-server>`,

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

})

export class ServersComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

1. **App.component.html**

<div class="container">

    <div class="row">

        <div class="col-xs-12">

            <h2>Hello from app component</h2>

            <hr>

            <div app-servers></div>

        </div>

    </div>

</div>

1. **Assignment**

**Create two components**

1. **Warning Component and SuccessComponent**
2. **Output each one below the other in the appcomponent**
3. **Use styles**
4. **Have red and green color**
5. **String interpolation**

**Server.component.ts**

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

//Decorator

@Component({

    //Metadata

    selector: 'app-server',

    //Template:

    templateUrl: './server.component.html'

}

)

export class ServerComponent {

    serverId: number = 10;

    serverStatus: string = "offline";

    getServerStatus() : string{

        return this.serverStatus;

    }

}

**Server.component.html**

<h4>

    {{'Server'}} with id {{serverId}} is {{getServerStatus()}}

</h4>

1. **Property binding**

**Server.component.ts**

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

//Decorator

@Component({

    //Metadata

    selector: 'app-server',

    //Template:

    templateUrl: './server.component.html'

}

)

export class ServerComponent {

    serverId: number = 10;

    serverStatus: string = "offline";

    allowNewServers: boolean = false;

    constructor() {

        setTimeout(() => {

            this.allowNewServers = true;

        }, 2000)

    }

    getServerStatus(): string {

        return this.serverStatus;

    }

}

**Server.component.html**

<h4>

    {{'Server'}} with id {{serverId}} is {{getServerStatus()}}

</h4>

<button class="btn btn-primary" [disabled]="!allowNewServers">

    Add New Server

</button>

1. **Property Binding Vs String Interpolation**

**Server.component.html**

<h4>

    {{'Server'}} with id {{serverId}} is {{getServerStatus()}}

</h4>

<button class="btn btn-primary" [disabled]="!allowNewServers">

    Add New Server

</button>

<!-- <p>{{allowNewServers}}</p> -->

<p [innerText]="allowNewServers"></p>

1. **Event Binding**

**Server.component.html**

<h4>

    {{'Server'}} with id {{serverId}} is {{getServerStatus()}}

</h4>

<button class="btn btn-primary" [disabled]="!allowNewServers" (click)="onServerCreated()">

    Add New Server

</button>

<!-- <p>{{allowNewServers}}</p> -->

<p [innerText]="allowNewServers"></p>

{{serverCreateStatus}}

**Server.component.ts**

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

//Decorator

@Component({

    //Metadata

    selector: 'app-server',

    //Template:

    templateUrl: './server.component.html'

}

)

export class ServerComponent {

    serverId: number = 10;

    serverStatus: string = "offline";

    allowNewServers: boolean = false;

    serverCreateStatus: string = "No server was created";

    constructor() {

        setTimeout(() => {

            this.allowNewServers = true;

        }, 2000)

    }

    getServerStatus(): string {

        return this.serverStatus;

    }

    onServerCreated(): void {

        this.serverCreateStatus = "Server was created";

    }

}

1. **Paasing and using Data with Event binding**

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

//Decorator

@Component({

    //Metadata

    selector: 'app-server',

    //Template:

    templateUrl: './server.component.html'

}

)

export class ServerComponent {

    serverId: number = 10;

    serverStatus: string = "offline";

    allowNewServers: boolean = false;

    serverCreateStatus: string = "No server was created";

    serverName: string = '';

    constructor() {

        setTimeout(() => {

            this.allowNewServers = true;

        }, 2000)

    }

    getServerStatus(): string {

        return this.serverStatus;

    }

    onServerCreated(): void {

        this.serverCreateStatus = "Server was created";

    }

    onUpdateServerName(event: Event): void {

        this.serverName = (<HTMLInputElement>event.target).value;

    }

}

<!-- <h4>

    {{'Server'}} with id {{serverId}} is {{getServerStatus()}}

</h4> -->

<label for="ServerName">Server Name</label>

<input type="text" id="ServerName" class="form-control" (input)="onUpdateServerName($event)">

<br>

<button class="btn btn-primary" [disabled]="!allowNewServers" (click)="onServerCreated()">

    Add New Server

</button>

<!-- <p>{{allowNewServers}}</p> -->

<p [innerText]="allowNewServers"></p>

{{serverCreateStatus}}

{{serverName}}