Experiment: Registration Page with Event Handling using Angular 9

Aim

To design and implement a **Registration Page** using **Angular 9**, incorporating event handling, form validation, and user interaction features.

Software / Hardware Requirements

- Node.js (LTS version)
- Angular CLI 9
- Visual Studio Code
- Web Browser (Chrome recommended)

Theory

Angular 9 is a powerful front-end framework that supports creating interactive and dynamic **Single Page Applications (SPAs)**. It uses components, event binding, and reactive features to handle form inputs and user events.

Key Angular Concepts Used

- **Angular Components** – UI building blocks.
- **Template-driven Forms** – for user input validation.
- **Event Handling** – `(click)`, `(keyup)`, `(submit)` etc.
- **Two-way Data Binding** – using `[(ngModel)]`.
- **Form Validation** – required fields, email validation.

Procedure

1. Install Angular CLI

```
npm install -g @angular/cli
```

2. Create a New Angular Project

```
ng new registration-demo
cd registration-demo
```

Choose **yes** for routing and **CSS** for styling.

---

3. Create Registration Component

```
ng generate component registration
```

---

4. Enable FormsModule

In `app.module.ts`, import:

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

@NgModule({
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
})
export class AppModule {}
```

---

5. Registration Form Implementation

registration.component.html

```
<h2>User Registration</h2>
<form (ngSubmit)="register()" #regForm="ngForm">

  <label>Name:</label>
  <input type="text" name="name" [(ngModel)]="user.name" required>
  <br><br>

  <label>Email:</label>
  <input type="email" name="email" [(ngModel)]="user.email" required>
  <br><br>

  <label>Password:</label>
  <input type="password" name="password" [(ngModel)]="user.password" required
minlength="6">
  <br><br>

  <button type="submit" [disabled]="!regForm.form.valid">Register</button>
</form>

<p *ngIf="message">{{ message }}</p>
```

---

registration.component.ts

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

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

  user = {
    name: '',
    email: '',
    password: ''
  };

  message = '';

  // Event Handler
  register() {
    this.message = `Registration Successful! Welcome, ${this.user.name}`;
  }
}
```

---

6. Add Route to Registration Page

In `app-routing.module.ts`:

```
import { RegistrationComponent } from
'./registration/registration.component';

const routes: Routes = [
  { path: 'register', component: RegistrationComponent }
];
```

---

7. Add Navigation (Optional)

`app.component.html`:

```
<nav>
  <a routerLink="register">Register</a>
</nav>
<hr>
<router-outlet></router-outlet>
```

Output

- A functional **Registration Page**.
- Inputs validated using Angular form controls.
- Event handling performed on **form submit**.
- Success message displayed dynamically.

Result

The registration page was successfully designed using Angular 9 with form validation and event handling.

If you want, I can add: - PDF export - Flowchart / Diagram - Angular Material UI - Additional events like keyup, blur, mouseover - Login & Dashboard module