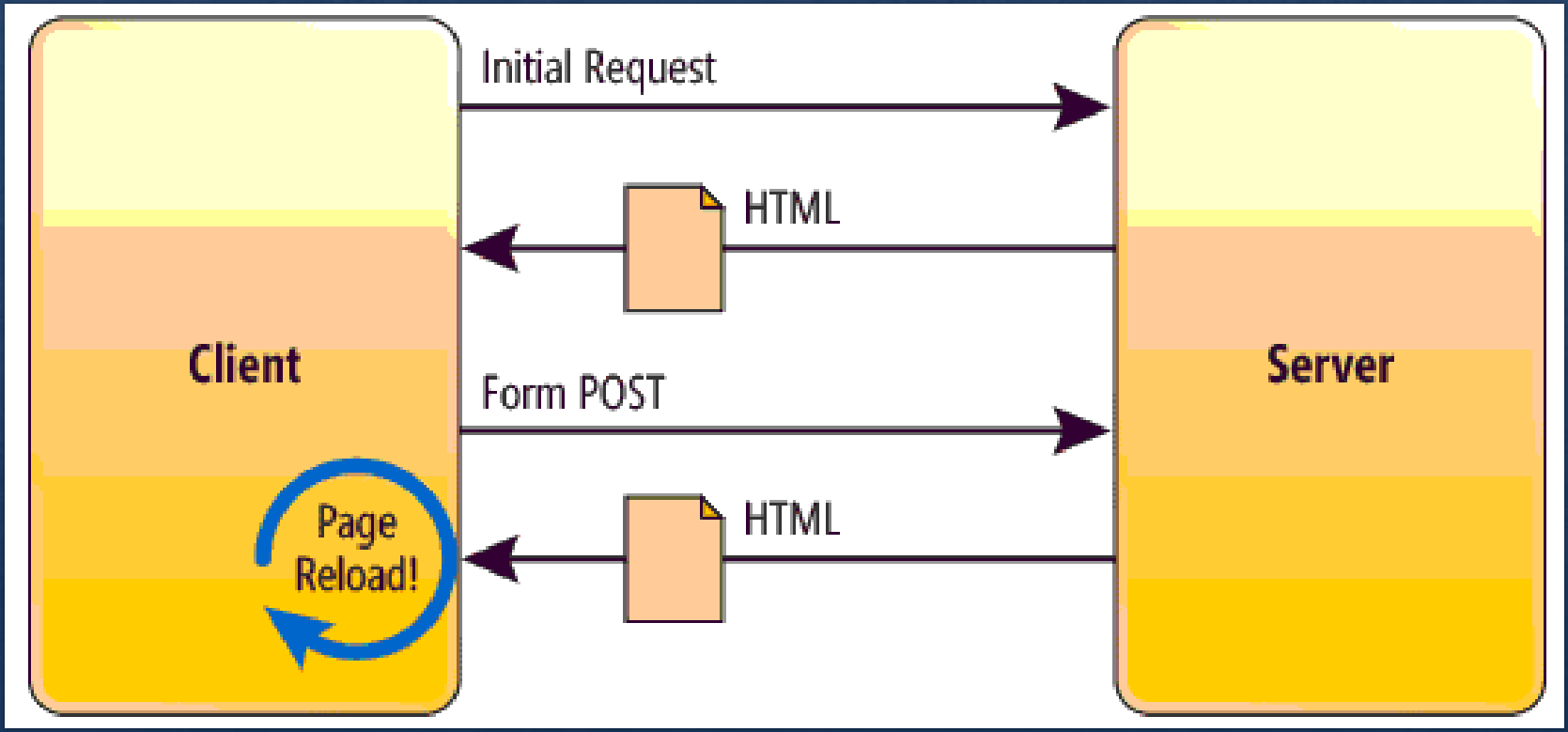
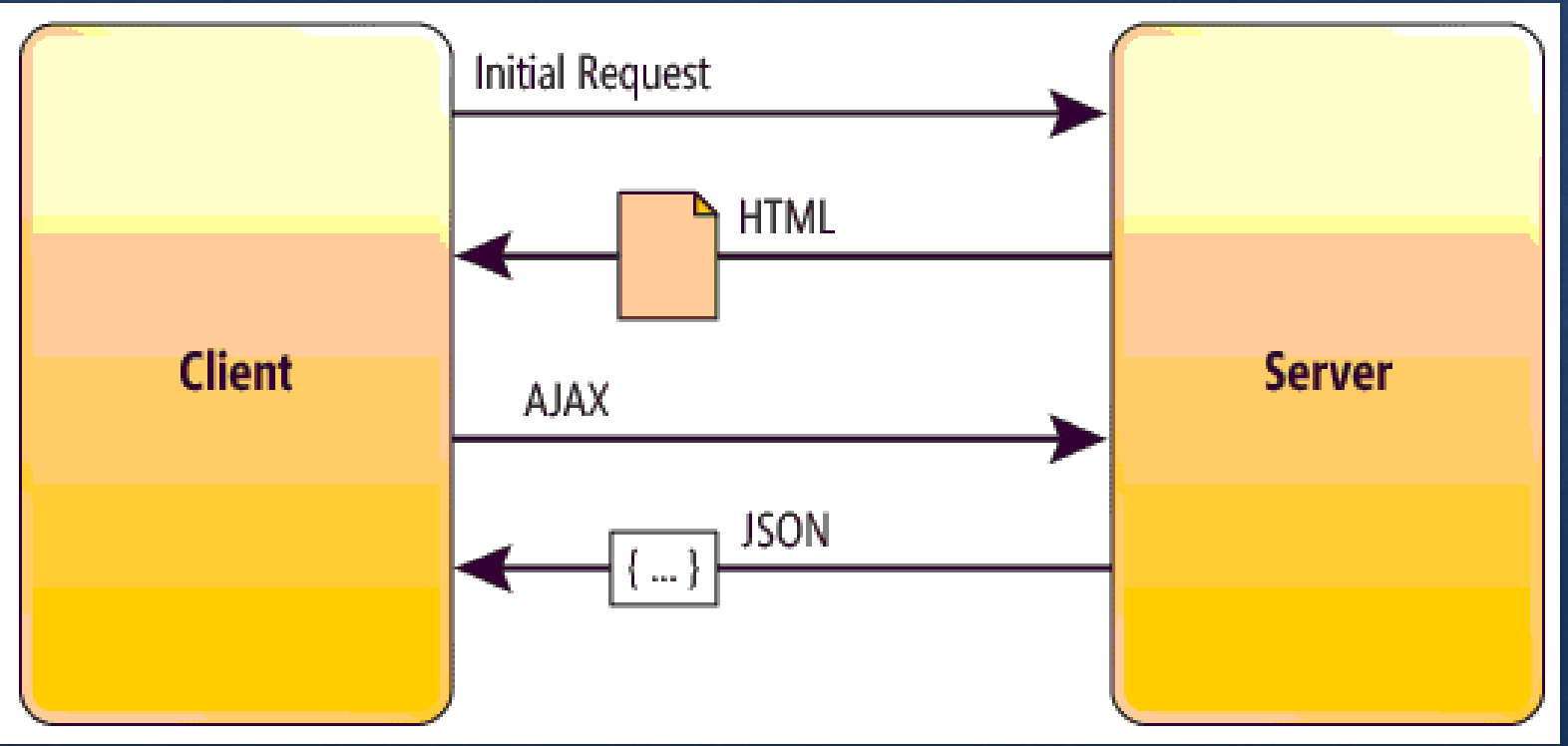
Angular 13

Moumita Asad
Lecturer
IIT, DU.

Traditional Page



Single Page Application



Angular 13

- An open-source JavaScript framework written in TypeScript
- Maintained by Google
- Primary purpose is to develop single-page applications
- As a platform, Angular includes:
  - A component-based framework for building scalable web applications
  - A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication
  - A suite of tools to develop, build, test, and update code

World Tech Giants using Angular

Angular Prerequisites

1. NodeJS
   - **Download URL:** https://nodejs.org/en/download/
   - **Command to check installation:** node -v
2. Angular CLI
   - npm install -g @angular/cli
   - **Command to check installation:** ng --version
3. Text Editor (Visual Studio Code)
   - **Download URL:** https://code.visualstudio.com/download

Creating The First Project!

- **Create new project:** ng new project-name
- **Go to the workspace directory:** cd project-name
- **Build and run the project:** ng serve

Troubleshoot: Running Scripts Is Disabled On This System

- Execute the following 3 commands:
  1. powershell set-ExecutionPolicy RemoteSigned -Scope CurrentUser
  2. powershell Get-ExecutionPolicy
  3. powershell Get-ExecutionPolicy -list

Component

- Fundamental building block of Angular applications
- Responsibility:
  1. display data on the screen
  2. listen for user input
  3. take action based on that input
- Consists of 3 things:
  1. A component class that handles data and functionality
  2. An HTML template that determines the UI
  3. Component-specific styles that define the look and feel

Example of Component

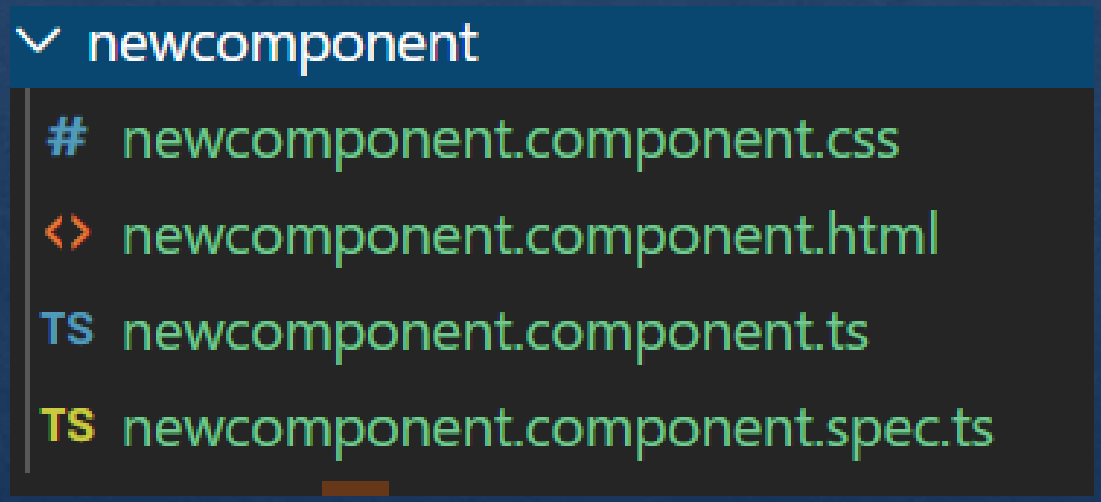
```
import { Component } from '@angular/core';

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

```
<body>
  <app-root></app-root>
</body>
</html>
```

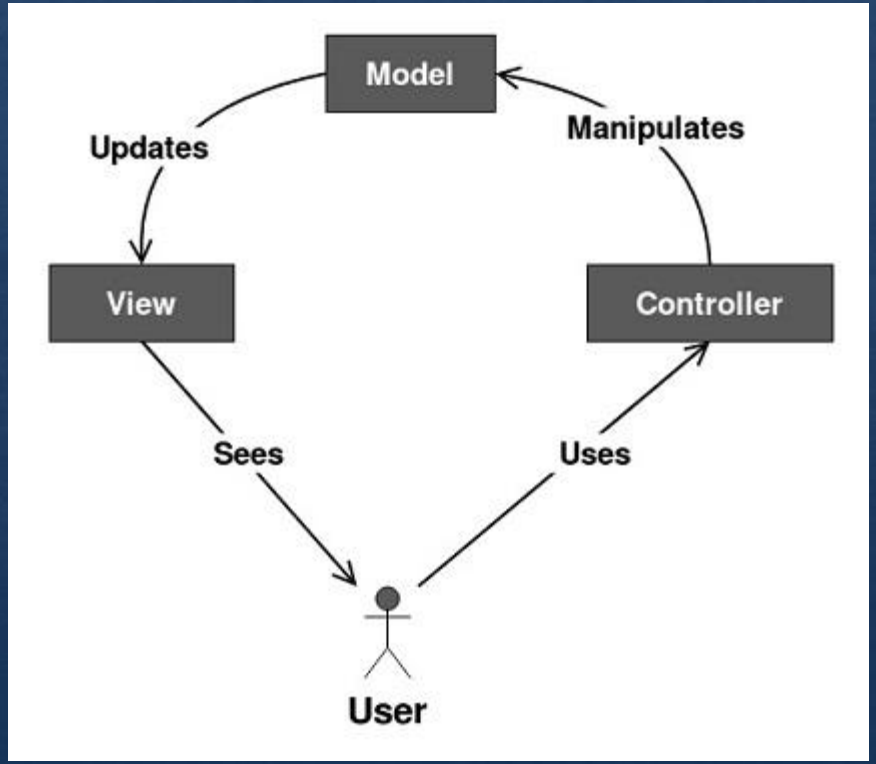
Creating New Component

- **Command:** ng generate component component-name



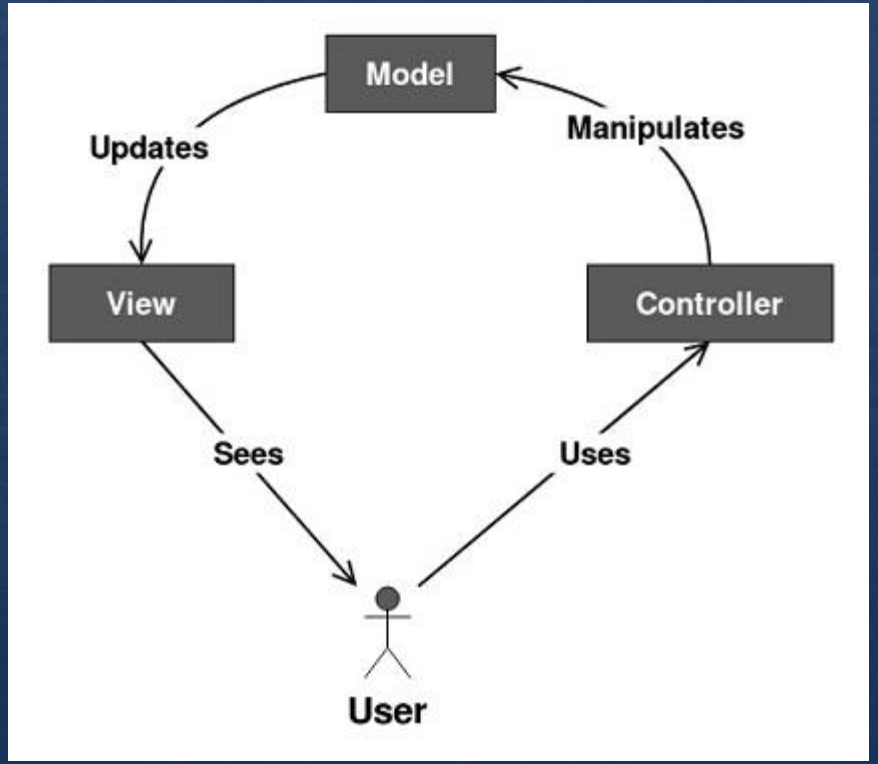
Model-View-Controller (MVC)

- An architectural pattern that separates an application into three main logical components:
  1. model
  2. view
  3. controller
- Each of these components are built to handle specific development aspects of an application.



Model-View-Controller (MVC)

- **Model:** corresponds to all the data-related logic that the user works with
- **View:** presents data to the user or handles user interaction
- **Controller:** An interface between Model and View components



Model (Class)

- TypeScript is object oriented JavaScript.
- A class is a blueprint for creating objects.
- A class encapsulates data for the object.
- **Strict Class Initialization:** checks to ensure that each instance property of a class gets initialized in the constructor body, or by a property initializer.

```
export class book {
    id: number = 0;
    name: string = "";
    year: number = 0;
    availability: boolean = true;
}
```

Service

- Objective: organize and share business logic, models, or data and functions with different components of an Angular application.
- Get instantiated just once during the lifetime of an application.
- Contain methods that maintain data throughout the life of an application, i.e., data is available all the time.
- Usually implemented through **dependency injection**.
- **Command:** ng generate service service-name

```
export class BookService {
  books: Book[] =
          [{id:1, name:"Megh boleche jabo jabo",year:2004,availability:true},
          {id:2, name:"Debi",year:2002,availability:false}]
  constructor() { }
  getBooks(): Book[] {
    return this.books;
  }
}
```

Controller

- focuses on managing the attributes that are connected to the view (template) and invoking the service.

```
export class BookListComponent implements OnInit {

  constructor(private bookService: BookService) { }

  books = this.bookService.getBooks();
  ngOnInit(): void {
    console.log("init!!");
  }
}
```

View

```
<tbody>
  <tr *ngFor ="let book of books">
    <td>{{book.name}}</td>
    <td>{{book.year}}</td>
    <td *ngIf = "book.availability">available</td>
    <td *ngIf = "!book.availability">borrowed</td>
  </tr>
</tbody>
```

Angular Routing

- To handle the navigation from one view to the next, you use the Angular Router.
- The Router enables navigation by interpreting a browser URL as an instruction to change the view.

```
const routes: Routes = [
  {path: '', component: HomepageComponent},
  {path: 'books', component: BookListComponent},
  {path: 'home', component: HomepageComponent},
  {path: 'newbook', component: NewbookComponent},
  {path: 'updatebook', component: UpdateBookComponent},
];
```

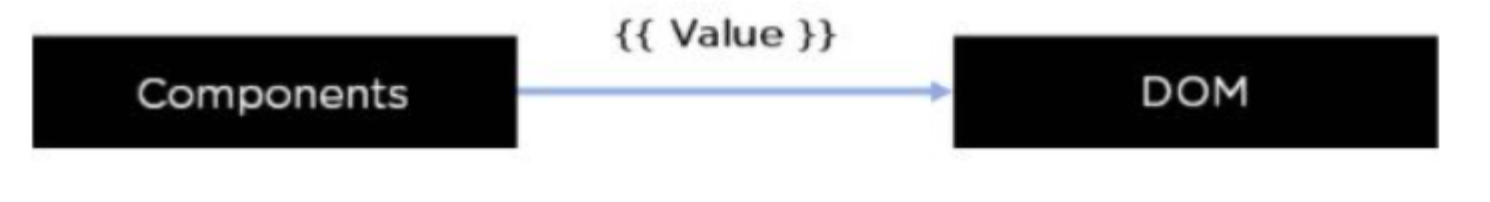
Angular Routing

```
<div class="collapse navbar-collapse" id="navbarResponsive">
    <ul class="navbar-nav ml-auto">
        <li class="nav-item">
            <a class="nav-link" routerLink="/home">Home</a>
        </li>

        <li class="nav-item">
            <nav>
                <a class="nav-link" routerLink="/books" >View Books</a>
            </nav>
        </li>
    </ul>
</div>
```

Data Binding

- **Interpolation Binding**
  - allows the user to bind a value to the user interface element
  - data moves in one direction from the components to HTML elements



Data Binding

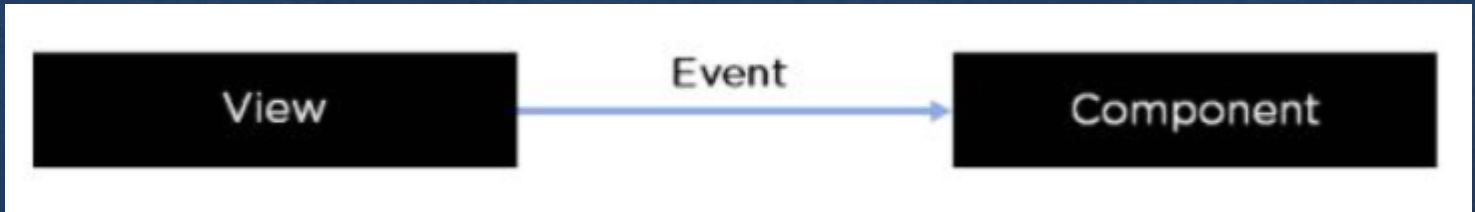
- **Property Binding**
  - set the properties for HTML elements.
  - involves updating a property value in the component and binding the value to an HTML element in the same view

```
In the app.component.ts file:
public image = "/assets/Logo.png"

In the app.component.html file:
<img [src] = "image" alt="" style="height: 100px; width: 250px" class="center">
```

Data Binding

- **Event Binding**
  - when information flows from the view to the component after an event is triggered.



Data Binding

- **Two-way Data Binding**
  - where data flows from the component to the view and back
  - Any changes made on either end are immediately reflected on both

```
<div class="form-group">
  <label for="name">Name</label>
  <input type="text" class="form-control" id="name" [(ngModel)]="bookToBeUpdated.name" [ngModelOptions]="{standalone: true}">
</div>
```

Useful Resources

- https://angular.io/docs
- https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular?source=sl_frs_nav_playlist_video_clicked