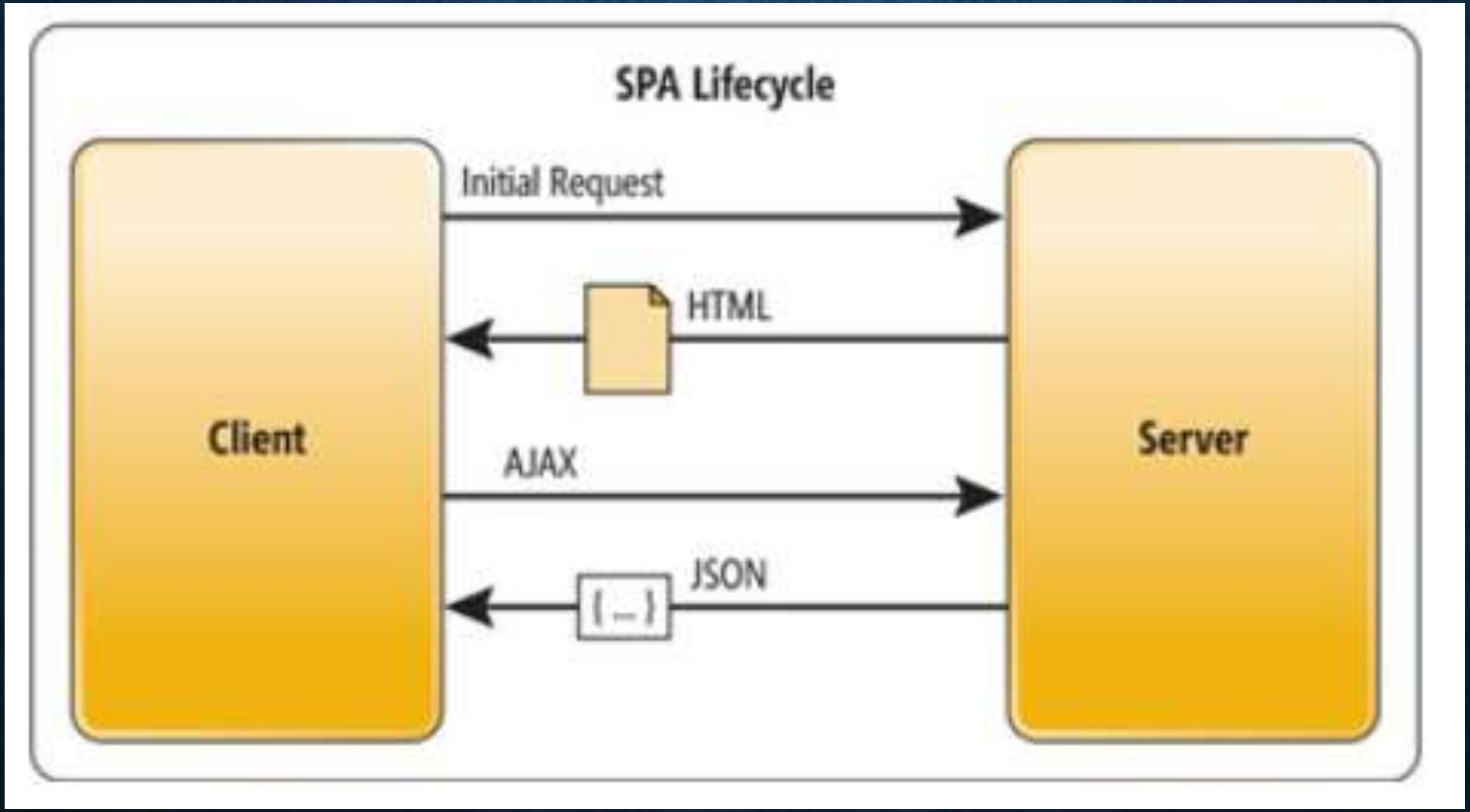
ANGULAR 18

WHAT IS ANGULAR?

- Angular is an open-source, JavaScript framework written in TypeScript. Google maintains it, and its primary purpose is to develop single-page applications.
- Website that use angular
- 1) IRCTC
- 2) Gemini
- 3) Battle.net

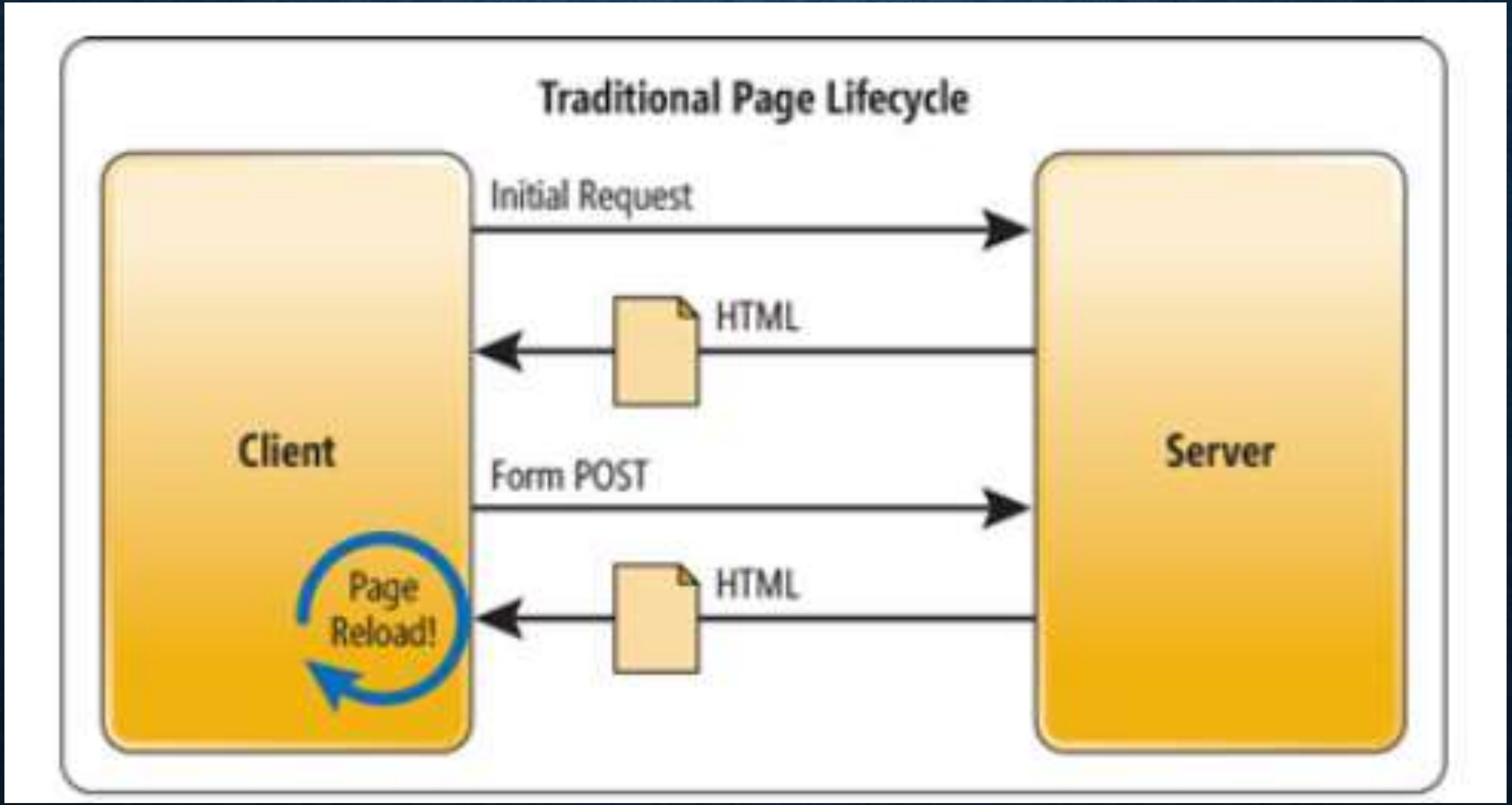
SINGLE PAGE APPLICATION(SPA)

- An SPA (Single-page application) is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs such as Fetch when different content is to be shown.


SPA Lifecycle
Client
Server
Initial Request
HTML
AJAX
JSON
{ ... }

MULTI PAGE APPLICATION(MPA)

- A Multi-Page Application, or MPA, is a type of website or web application where each page is separately created and sent to our web browser by a server. Unlike Single Page Applications (SPAs), which build pages in our browser using JavaScript, MPAs depend on the server to generate and refresh pages.


Traditional Page Lifecycle
Client
Server
Initial Request
HTML
Form POST
HTML
Page Reload!

FEATURES OF ANGULAR

- 1) Component based architecture
- 2) Two way data-binding
- 3) Dependency Injection
- 4) Typescript
- 5) Routing
- 6) It is based on Change detection
- 7) Ivy Engine

ADVANTAGES OF ANGULAR

- It is a product of google
- Large community
- It creates a single Page Application
- Component Based Architecture
- Improved speed and performance

DISADVANTAGES OF ANGULAR

- Steep learning curve
- large build size
- Limited SEO (Search Engine Optimization)

INSTALLATION

REQUIREMENTS

- --> Nodejs is needed
- cmd command to check node version: node -v
- --> NPM must be available
- cmd command to check npm Version: npm –v

NPM: Node Package Manage

It contains different libraries available for javascript

ANGULAR CLI

- The Angular CLI is a command-line interface tool which allows you to scaffold, develop, test, deploy, and maintain Angular applications directly from a command shell.
- Command to install angular cli : npm install -g @angular/cli
- Commands invoking ng are using angular cli
- Command to check angular version: ng v

IMPORTANT COMMANDS

- ng new projectName : starts the installation setup
- ng serve : command to run our project

  the default port on which our project runs in 4200

Command to stop the project

ctrl + c

PROJECT FOLDER STRUCTURE

- Node Modules: contains all the different packages for running angular application
- Public Folder: Contains the files such as images, video or static assets which may be needed in the project
- .gitignore: file to tell git which folders and file it should track or leave untracked
- Angular.json: Provides the workspace configuration for angular
- Package.json: It contains human-readable metadata about the project (like the project name and description) as well as functional metadata like the package version number and a list of dependencies required by the application.

- Package-lock.json : package-lock.json is an auto-generated file that provides a detailed, deterministic record of the dependency tree.
- Tsconfig.app.json: configuration files for typescript with respect to angular
- Tsconfig.json: configuration for typescript
- Tsconfig.spec.json: configuration for testing

SRC FOLDER IN ANGULAR PROJECT

- This folder is the core directory of your Angular application, containing all the essential files and folders required to build, develop, and maintain your app
- App folder: This is where all the components, services, pipes, routes etc. are kept
- Index.html: the main html file which is the entry point for the application
- Main.ts: the main entry point for angular project
- Styles.css: the global css file

COMPONENTS IN ANGULAR

- components are the fundamental building blocks of an application.
- They encapsulate the HTML, CSS, and logic that define a particular piece of the user interface (UI).
- Each Angular component is a TypeScript class that is associated with a template (HTML), styles (CSS/SCSS), and a selector that defines how the component can be used in other templates.

COMMAND TO CREATE A COMPONENT

- ng generate component componentName
- ng g c componentName

STANDALONE COMPONENT

- In Angular, a **standalone component** is a type of component that does not require being declared in an Angular module (NgModule).
- Traditionally, Angular components had to be declared in an NgModule to be part of an Angular application.
- However, with the introduction of standalone components, Angular provides a more modular and flexible approach to component architecture.

DATA BINDING

- Data binding in Angular is a mechanism that allows communication between the component (TypeScript code) and the view (HTML template).
- It enables you to bind data from your component to your view and vice versa, allowing for dynamic updates and interactions in your application.

TYPES OF DATA BINDING

- One way data binding
- Property binding
- Event Binding
- Two way data binding

ONE WAY BINDING(INTERPOLATION)

- This binds the data from the component to the view. It’s typically used to display data in the HTML template.
- Syntax: {{ expression }}.
- Example: <h1>{{ title }}</h1> where title is a property in the component.

PROPERTY BINDING

- This binds data from the component to the property of an HTML element or directive.
- Syntax: [property]="expression".
- Example: <img [src]="imagePath"> where imagePath is a property in the component.

EVENT BINDING

- This binds an event in the view to a method in the component. It's used to handle user interactions.
- Syntax: (event)="method()".
- Example: <button (click)="onClick()">Click me</button> where onClick() is a method in the component.

TWO WAY BINDING

- This allows for both property binding and event binding simultaneously, meaning changes in the view update the component and vice versa.
- Syntax: [(ngModel)]="property".
- Example: <input [(ngModel)]="name"> where name is a property in the component. This binding keeps the input field and the name property in sync.

DIRECTIVES IN ANGULAR

- In Angular, directives are special classes that can modify the structure or behavior of elements in the DOM. There are three main types of directives in Angular:
- There are three types of directives
    - Component directives
    - Attribute directives
    - Structural directives

COMPONENT DIRECTIVES

- These are the most common type of directive in Angular. Components are directives with a template.
- Angular components are essentially directives with a @Component decorator that defines their view (HTML template) and behavior.

ATTRIBUTE DIRECTIVES

- These directives change the appearance or behavior of an element, component, or another directive.
- Attribute directives are used to modify the behavior or styling of the DOM element to which they are applied.

STRUCTURAL DIRECTIVE

- Structural directives change the DOM layout by adding or removing elements.
- These directives are typically used to manipulate the structure of the view.
- Structural directives are of the following type
    - ngIf
    - ngSwitch
    - ngFor

CONTROL FLOW STATEMENTS IN ANGULAR

- The control flow statements are recent introduction in Angular 17
- They are as follows
  - @if
  - @if …@else
  - @if …@else if… @else
  - @for
  - @switch

PIPES IN ANGULAR

- Pipes in Angular are a powerful feature used to transform data in templates.
- They are simple functions that accept an input value, process it, and return a transformed value.
- Pipes can be used to format strings, dates, numbers, or even perform custom data transformations.

TYPES OF PIPES

- Date Pipe
- Uppercase Pipe
- Lowercase pipe
- Currency pipe
- Json pipe
- Async pipe
- Custom pipes

HANDLING FORMS IN ANGULAR

- In Angular, there are two main approaches to handling forms
  - **template-driven forms**
  - **reactive forms**.

TEMPLATE DRIVEN FORMS

- Template-driven forms rely on Angular's directives and are more declarative, using the Angular template syntax to create and manage forms.
- The structure and logic of the form are primarily defined in the template (HTML) file.
- Angular automatically creates and manages the form model based on the directives you use.
- This approach is ideal for simple forms where you want to leverage Angular's two-way data binding and automatic form state management.

FEATURES

- **Custom Validation**: You can create custom validation logic using Angular's built-in validation mechanisms.
- **Two-Way Data Binding**: Angular’s ngModel directive supports two-way data binding out of the box, so changes to the model automatically update the view and vice versa.

EXPLANATION

- **ngModel Directive**: Binds the form input fields to the model in your component. It tracks the state of the form control and updates the model in your component automatically.
- **Form Validations**: You can add HTML5 validations or use Angular's built-in validators like required, email, etc. The form is considered valid when all controls are valid.
- **#myForm="ngForm"**: This local template variable references the form instance in the template, allowing you to access the form's validity and other properties.

REACTIVE FORMS

- Reactive forms in Angular are a powerful way to manage complex forms with more control over form validation, structure, and reactive changes.
- Unlike template-driven forms, reactive forms are created programmatically in the component class and provide greater flexibility and scalability.

EXPLANATION

- **FormGroup**: Represents the entire form. It's a collection of FormControl instances that can be managed together.
- **FormControl**: Represents a single form control, like an input field. Each form control can be initialized with a value and a set of validators.
- **Validators**: Angular provides a set of built-in validators (e.g., required, minLength, email, min).

OBSERVABLES

- An Observable is a data stream that can emit multiple values over time.
- Observables are used for handling asynchronous operations like HTTP requests, user inputs, or any event-based data.
- Observables comes from a library called Rxjs(Reactive Javascript), which comes preinstalled in the angular

OBSERVER

An Observer is an object with callback functions that handle the data emitted by an Observable.

It can respond to three types of notifications:

next(value): Emits the next value in the stream.

error(error): Emits an error if one occurs.

complete(): Signals that the Observable has completed emitting values.

- Subscription

  A subscription is needed to use any observable.

- Operators

  Operators are methods that allow you to transform, filter, or combine Observables in a declarative way.

  Common operators include map, filter etc.

HTTP CLIENT IN ANGULAR

- In Angular, the HttpClient service is part of the @angular/common/http package and is used to make HTTP requests to backend services.
- It simplifies interaction with external APIs or servers, handling operations like sending requests, receiving responses, and dealing with various data formats (e.g., JSON).

KEY FEATURES

- **Simplified API**: HttpClient provides a straightforward and powerful API to perform HTTP operations such as GET, POST, PUT, DELETE, and PATCH.
- Automatic JSON Parsing: By default, HttpClient automatically parses JSON responses into JavaScript objects, making it easy to work with API data.
- **Observables**: HttpClient methods return Observables, allowing you to work with asynchronous data streams. This integrates well with Angular's reactive programming model.

SERVICES IN ANGULAR

- In Angular, services are a fundamental concept used to organize and share code across your application.
- They allow you to encapsulate business logic, data access, or any other reusable functionality that you may need across multiple components.
- Command to create a service
  - ng g s servicename

LIFE CYCLE METHODS

- In Angular, components and directives have a set of lifecycle methods that provide hooks into different phases of their existence, from creation to destruction.
- Understanding these lifecycle methods helps you manage the initialization, updates, and cleanup of your components effectively.

LIFE CYCLE METHODS

•**ngOnChanges**: Called when an input property changes.
•**ngOnInit**: Called after the first ngOnChanges.
•**ngDoCheck**: Called after ngOnInit and on every change detection run.
•**ngAfterContentInit**: Called after content (ng-content) has been projected.
•**ngAfterContentChecked**: Called after the projected content has been checked.
•**ngAfterViewInit**: Called after the component's view has been initialized.
•**ngAfterViewChecked**: Called after the component's view has been checked.
•**ngOnDestroy**: Called just before the component is destroyed

ROUTING IN ANGULAR

- Routing in Angular is a powerful feature that allows you to navigate between different views or components within your application.
- The Angular Router is responsible for mapping URLs to components, handling navigation, and managing the application's state.

KEY CONCEPTS IN ANGULAR

•**Router Module:**
Angular provides the RouterModule for configuring routes. This module must be imported into your application to set up routing.
•**Routes:**
Routes define the mapping between a URL path and a component. Each route is an object with a path and a component.
•**Router Outlet:**
The <router-outlet> directive acts as a placeholder in your template where the matched component will be displayed.
•**RouterLink:**
•The routerLink directive is used to define navigation links in your templates. Clicking these links will trigger navigation.