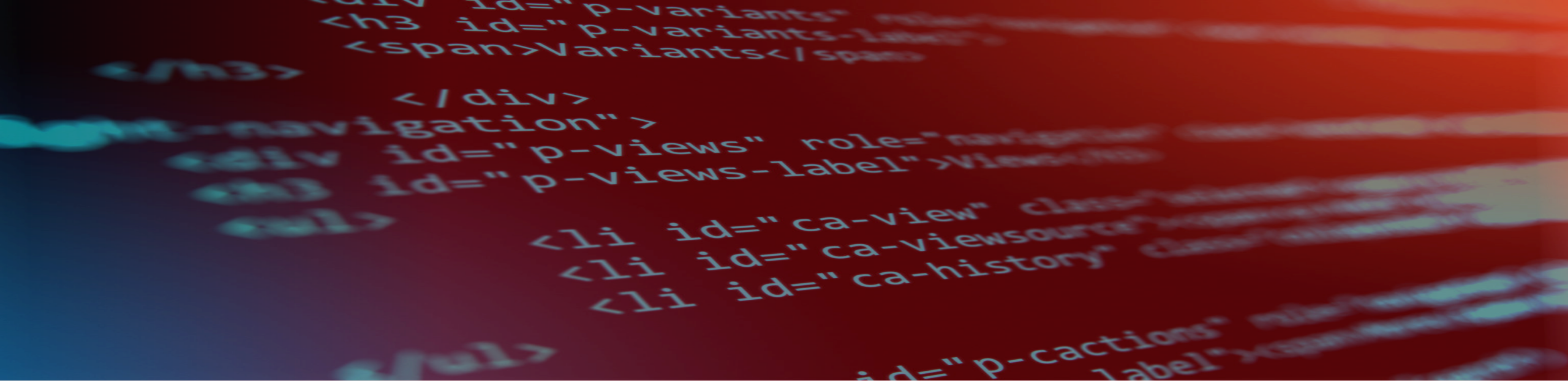


Introduction to Angular Concepts

Presenter:

K.C.Ashish Kumar

BROADCOM

Agenda

- Development Approach - *Framework* vs *Vanilla JavaScript*
- Benefits of Angular
- Getting started with the Angular Framework
  - Essential Tools
  - Prerequisites
  - Two-way Data Binding
  - Understanding Components / Directives, Services, Modules
  - Routing
- An example application (with source code)
- Q & A

Development Approach - *Framework* vs *Vanilla JavaScript*

- Managing smaller projects with Vanilla JS might seem easier, but as the size & complexity of the project grows, it gets difficult to do so
- Frameworks enforce code conventions and design patterns
- State Management
- “Cross Browser Compatible” code generation
- Frameworks have better documentation & community support
- Performance & Reliability
- Vanilla JS ➔ Reinventing the Wheel

Benefits of Angular

- Easily create Web Applications like SPA (Single Page App), Native App (using NativeScript for IOS & Android), SSR (Server-Side Rendering with Angular Universal).
- Component based development encourages modularity
- Out-of-box support for Unit-Testing the code
- Out-of-box support for Automation
- Data Binding capabilities
- Easy form validations
- Component libraries (Material, PrimeNG, . . .)

& much more . . . . . .

AngularJS or Angular ?

AngularJS = Angular 1.x (Obsolete & Deprecated)

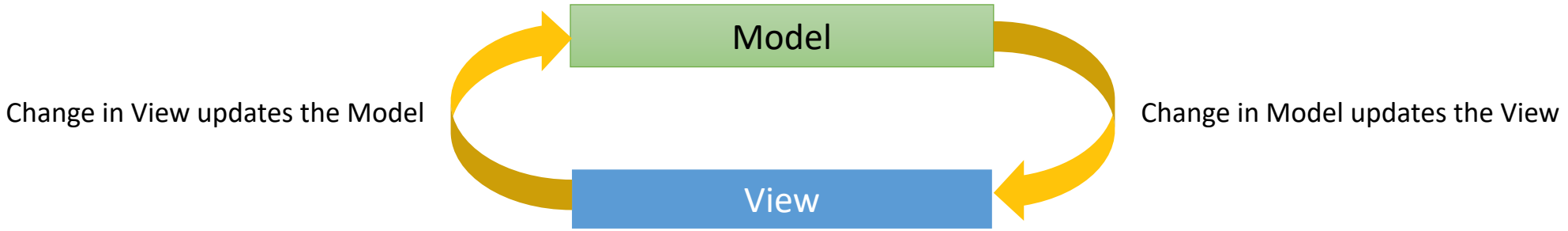
Angular = Angular 2.x, 3.x, ... 10.x

Getting started with the Angular Framework

- **Essential Tools:**
  - NodeJS (Download from https://nodejs.org/)
  - Angular CLI (Run the below commands for installing the Angular CLI and creating a new project)
    npm i -g @angular/cli
    ng new my-first-project
  - IDE**: VSCode (** https://code.visualstudio.com/ **) / Webstorm (** https://www.jetbrains.com/webstorm/ **)**
  - Browsers**: Google Chrome / Mozilla Firefox / Apple Safari / Opera / Microsoft Edge**
  - Version Control: **Git**
  - Git Client: **Sourcetree / Github Desktop / Git CLI**

Getting started with the Angular Framework (Contd . . .)

- **Prerequisites:**
  - Knowledge of JavaScript & OOJS, ES6, TypeScript, HTML5, CSS3
  - Knowledge / experience in working with NodeJS i.e. managing NPM Modules
- **Two-way Data Binding:**
  - Two-way binding gives your app a way to share data between a component class and its template
  - It basically does two things:
    - Sets a specific element property on the component class.
    - Listens for an element change event (i.e. update the visual aspects of data on the screen as well as data stored in the element property)



Getting started with the Angular Framework (Contd . . .)

- **Understanding Components / Directives, Services, Modules:**

  - *Component*: A component controls a patch of screen called a view.
    e.g.: Header, Body, Footer, Navigation, Menu, Detail, . . .

  - *Directive*: There are 3 kinds of directives:
    - Components
      e.g.: <product-detail id="1234"></product-detail>, <employee-list></employee-list>

    - Structural Directives - change the DOM layout by adding and removing DOM elements
      e.g.: *ngFor, *ngIf, ngSwitch with *ngSwitchCase & *ngSwitchDefault

    - Attribute Directives - change the appearance / behavior of an element, component, or another directive.
      e.g.: <p **appHighlight**>Hello World</p>

  - *Services*: Services facilitate data sharing among different classes.

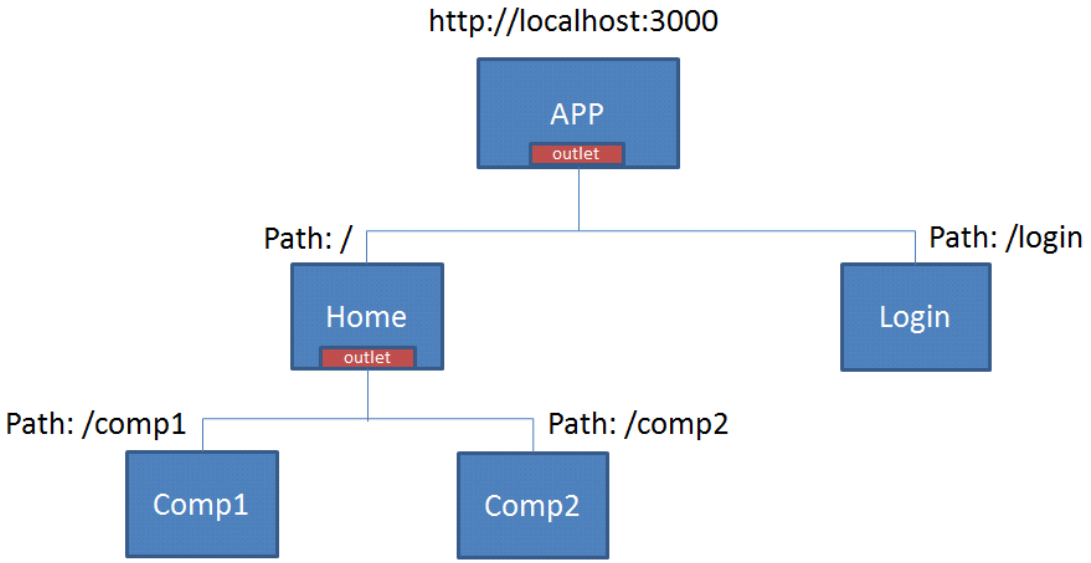
  - *Modules*: Angular apps are modular and Angular has its own modularity system called NgModules. NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities. They can contain components, service providers, and other code files whose scope is defined by the containing NgModule. They can import functionality that is exported from other NgModules, and export selected functionality for use by other NgModules.

    Angular launches an app by bootstrapping its root NgModule. While a small application might have only one NgModule, most apps have many more feature modules.

Getting started with the Angular Framework (Contd . . .)

- **Routing:** In a single-page app, you change what the user sees by showing or hiding portions of the display that correspond to particular components, rather than going out to the server to get a new page. As users perform application tasks, they need to move between the different views that you have defined.

  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.



An example application (with source code)

Recipe Book:

An example SPA built using Angular 10.1.5. The app allows a user to view, manage & create recipes. Also a user can add the recipe ingredients to a shopping list.

- **Application URL:**
  https://recipe-book.ashishkumarkc.com

- **Source code:**
  https://github.com/ashishkumarkc/recipe-book

  Steps to run the app in local machine:

  $> git clone https://github.com/ashishkumarkc/recipe-book

  $> cd recipe-book

  $> npm i

  $> ng serve

  Launch the url **http://localhost:4200/** in your browser.

- **Online Playground:**
  https://stackblitz.com/github/ashishkumarkc/recipe-book

That’s all for now. Thank You.