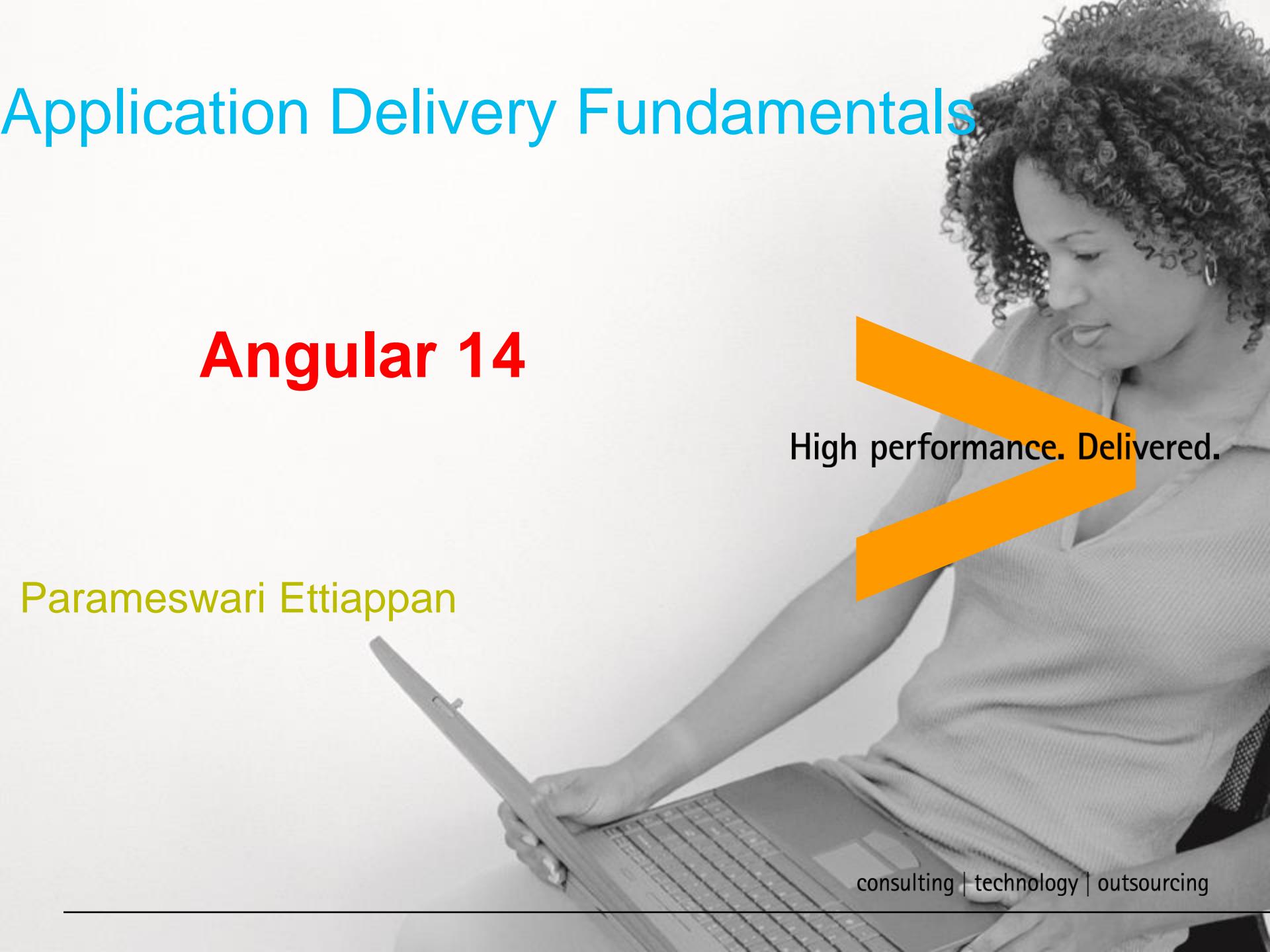


Application Delivery Fundamentals

Angular 14



High performance. Delivered.

Parameswari Ettiappan

consulting | technology | outsourcing



Angular 12

Goals

- CROSS PLATFORM
 - Progressive Web Apps
 - Native
 - Desktop
- SPEED AND PERFORMANCE
 - Code Generation(Template to code)
 - Universal
 - Code Splitting



Angular 12

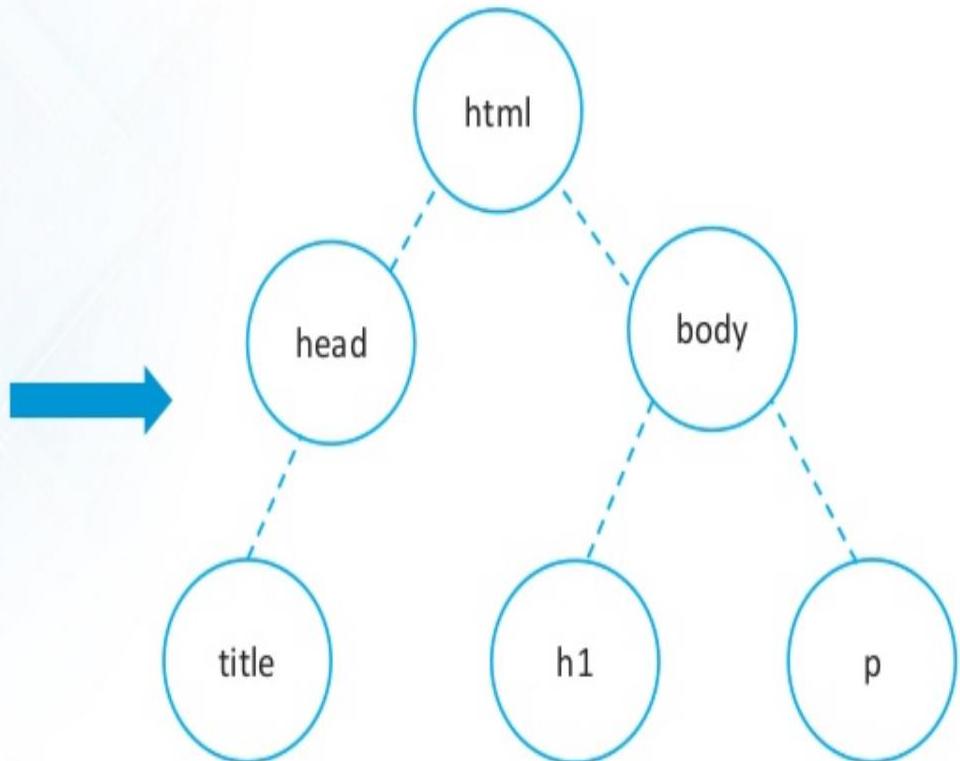
Goals

- PRODUCTIVITY
 - Templates
 - Angular CLI
 - IDEs
- FULL DEVELOPMENT STORY
 - Testing
 - Animation
 - Accessibility(RPA enabled components)

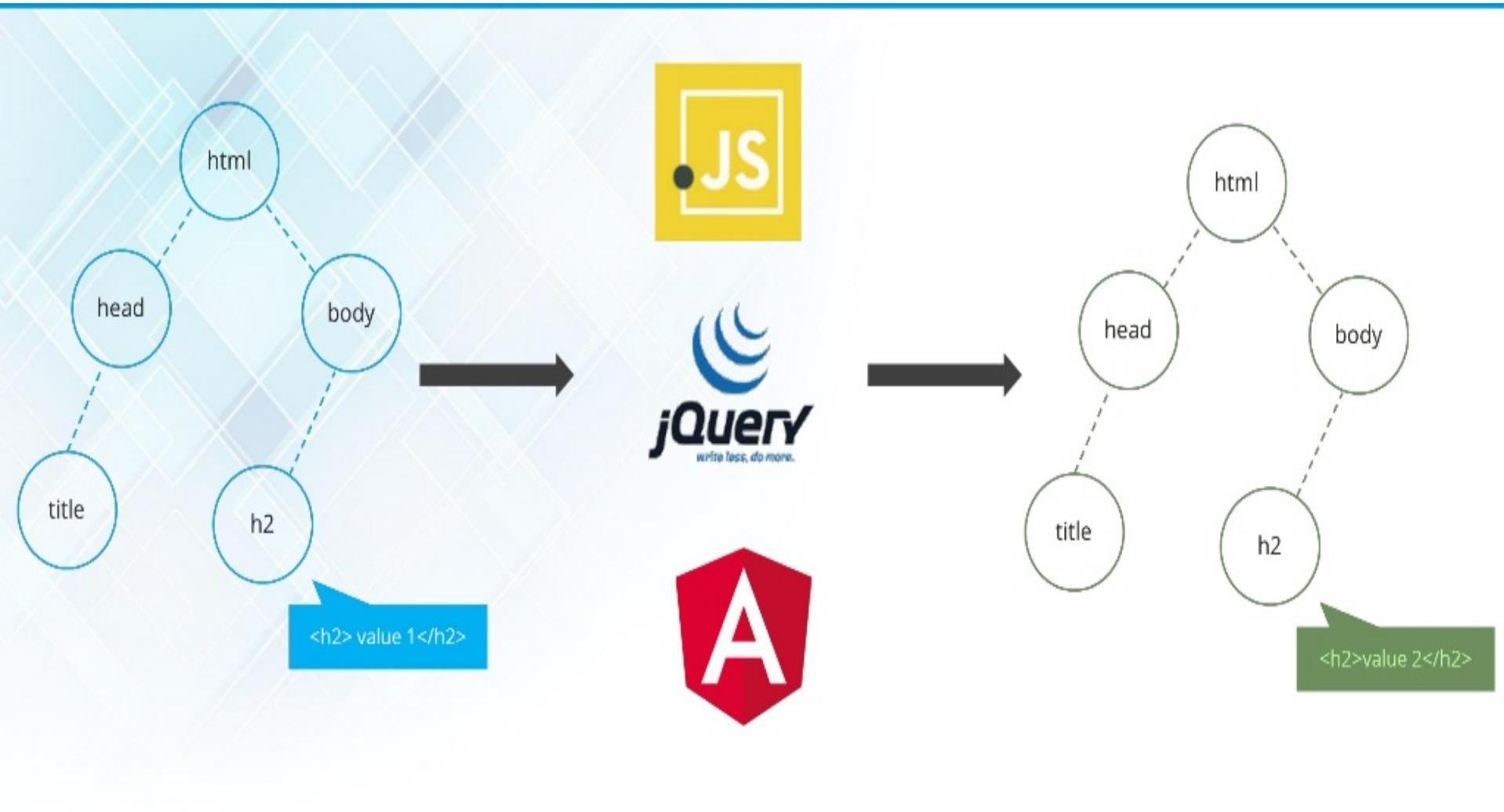


```
<html>
  <head>
    <title> Angular 2 Tutorial </title>
  </head>
  <body>
    <h1> Welcome to Angular 2 Tutorial </h1>
    <p>Angular is a development platform for creating
       applications using modern web standards.</p>
  </body>
</html>
```

HTML Markup



DOM Tree of the HTML document





- JavaScript is a programming language designed for use in a web browser.
- Used for manipulating DOM
- Example:

```
Document.body.style.background = red;
```

- jQuery is a library built in JavaScript to automate and simplify common tasks.
- Used for manipulating DOM
- Example:

```
$(‘body’).css(‘background’, ‘#ccc’);
```

Demands of Modern Front-end Applications



- Step 1. Verify you have a working & stable data API.
- Step 2. Select software management tools(github,slack etc.,)
- Step 3. Create a foundational specification(device,browser, language,cultures etc.,)
- Step 4. Select a software development process
- Step 5. Select a development platform & a host platform (i.e. staging & production platforms)
- Step 6. Select package managers (node and bower)

Demands of Modern Front-end Applications



- Step 7. Select site & user analytic tools(Chart analytics and user analytics)
 - analytics tools example
 - site analytics chartbeat
 - user analytics mixpanel
- Step 8. Select error, quality, and style code enforcement tools
- error, quality, and style enforcement code tools
 - example
- linters/hinters CSSLint, HTMLhint, JShint
- code style checker JavaScript Code Style
- code editor configuration file .editorconfig

Demands of Modern Front-end Applications



- Step 9. Select an automated task running tool
- The most common JavaScript task runners are Grunt.js, Gulp.js, or npm run using your package.json file.

Demands of Modern Front-end Applications



- Step 10. Select application architecture/structure (and corresponding tools/solutions)
- app architecture/structure example
- MV* AngularJS
- templates MVVM from AngularJS
- module communication AngularJS \$rootScope.\$broadcast and \$scope.\$on for a PubSub communication.
- data abstraction Kendo UI dataSource
- app structure NG seed
- routing AngularUI Router
- dependency management AngularJS Dependency Injection
- widgets/components Kendo UI (using built-in directives)
- css UI frameworks Material Design for Angular

Demands of Modern Front-end Applications

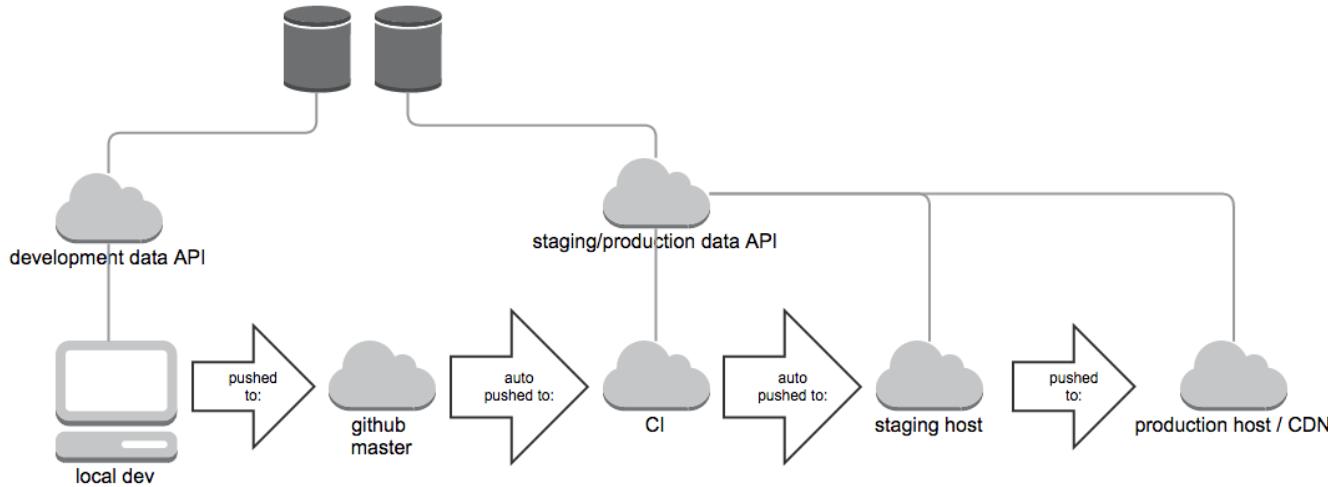


- Step 11. Select testing methodologies & tools
- testing tools example
- test runner (for unit testing) karma
- unit testing framework mocha
- unit testing assertions chai
- unit testing mocking sinon.js
- unit testing coverage blanketjs
- browser testing platform (for unit & functional testing)
browserstack
- browser automation framework (for functional testing)
nightwatch.js

Demands of Modern Front-end Applications



- Step 12. Select code quality/complexity tools
- quality/complexity example
- JavaScript Source Analysis Plato
- Step 13. Define deployment strategy



Demands of Modern Front-end Applications



- Step 14. Select package monitoring solution
- A tool like gemnasium can monitor both your npm package.json file as well as your bower.json file.
- Step 15. Select a JavaScript error monitoring solution
- Step 16. Select a performance monitoring solution
- Use something like Pingdom to capture uptime and performance stats.

What is a Functional Requirement?



- In software engineering, a functional requirement defines a system or its component.
- It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs.
- It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.
- Functional software requirements help you to capture the intended behavior of the system.
- This behavior may be expressed as functions, services or tasks or which system is required to perform.

What is a Functional Requirement?



- Example of Functional Requirements
- The software automatically validates customers against the ABC Contact Management System
- The Sales system should allow users to record customers sales
- The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF.
- Only Managerial level employees have the right to view revenue data.
- The software system should be integrated with banking API
- The software system should pass Section 508 accessibility requirement.

What is Non-Functional Requirement?



- A non-functional requirement defines the quality attribute of a software system.
- They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?
- A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system.
- Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.
- Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs.
- Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

What is Non-Functional Requirement?



- Examples of Non-functional requirements
- Here, are some examples of non-functional requirement:
- Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
- Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
- Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
- A website should be capable enough to handle 20 million users without affecting its performance
- The software should be portable. So moving from one OS to other OS does not create any problem.
- Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

Functional vs Non Functional Requirements



| Parameters | Functional Requirement | Non-Functional Requirement |
|----------------|-----------------------------|--|
| What it is | Verb | Attributes |
| Requirement | It is mandatory | It is non-mandatory |
| Capturing type | It is captured in use case. | It is captured as a quality attribute. |
| End-result | Product feature | Product properties |
| Capturing | Easy to capture | Hard to capture |

Functional vs Non Functional Requirements



| | | |
|-----------------|--|--|
| Objective | Helps you verify the functionality of the software. | Helps you to verify the performance of the software. |
| Area of focus | Focus on user requirement | Concentrates on the user's expectation. |
| Documentation | Describe what the product does | Describes how the product works |
| Type of Testing | Functional Testing like System, Integration, End to End, API testing, etc. | Non-Functional Testing like Performance, Stress, Usability, Security testing, etc. |
| Test Execution | Test Execution is done before non-functional testing. | After the functional testing |
| Product Info | Product Features | Product Properties |

Advantages of Functional Requirement



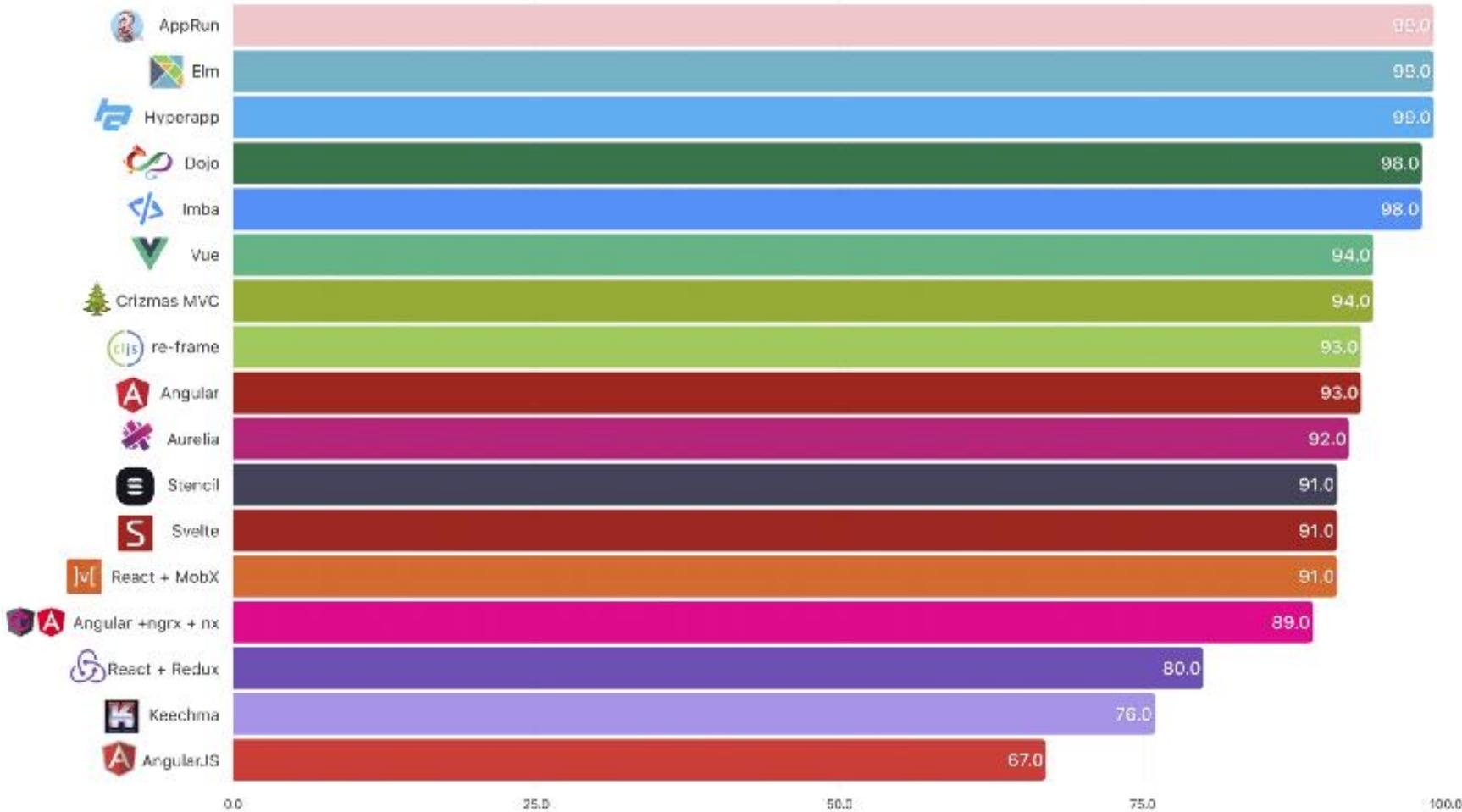
- Here, are the pros/advantages of creating a typical functional requirement document-
- Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application
- A functional requirement document helps you to define the functionality of a system or one of its subsystems.
- Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior.
- Errors caught in the Functional requirement gathering stage are the cheapest to fix.
- Support user goals, tasks, or activities for easy project management
- Functional requirement can be expressed in Use Case form or user story as they exhibit externally visible functional behavior.

Advantages of Non-Functional Requirement



- Benefits/pros of Non-functional testing are:
- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.







| | Community | Size | Ecosystem | Learnability |
|-------------|------------------------------------|--------|------------------|--------------|
| Angular.js | 48,787 stars 1,449 contributors | 80 kb | 1,877 ngModules | Difficult |
| Backbone.js | 24,650 stars 285 contributors | 61 kb | 257 backplugs | Easy |
| React.js | 41,038 stars 666 contributors | 450 kb | 4,442 components | Difficult |
| Ember.js | 16,115 stars 580 contributors | 269 kb | 725 emberaddons | Easy |
| Aurelia.js | 5,958 stars 30 contributors | 295 kb | 12 plugins | Easy |

1. React



- Usage Statistics
- 475k websites on the web are powered by React.
- 64.8% of javascript developers are currently using React.
- Ranked 1st most popular front-end javascript framework in State Of JS survey.
- Github – 120.5k stars, 21k forks and 1200 contributors.

1. React



- Why REACT Is One Of The Top JavaScript Frameworks For 2019?
- High performance – React is renowned for its high efficiency and flexibility. It can be easily integrated with other frameworks without any hassle. Can be used for both client side as well as server side.
- Abundant Resources – As react is maintained by Facebook, there is a huge trove of documentation and resources available on the web which makes learning curve of react much smoother than its chief rival AngularJS.
- Backward Compatibility – Transitioning or migrating from older to new versions is fairly easy in react using CodeMods.
- Easy to maintain component structure – Component based architecture of ReactJS helps in increasing the reusability of code and makes maintenance of large scale projects quite a bit easier.
- Rich community – React has close to 1300 contributors on GitHub which is more than any other library/framework.
- Unidirectional Data Flow – One way/Unidirectional downward only data binding in react via flux controls helps to make sure that any changes made to child element structure don't affect the parent element structure

1. React



- Why REACT Is One Of The Top JavaScript Frameworks For 2019?
- High performance – React is renowned for its high efficiency and flexibility. It can be easily integrated with other frameworks without any hassle. Can be used for both client side as well as server side.
- Abundant Resources – As react is maintained by Facebook, there is a huge trove of documentation and resources available on the web which makes learning curve of react much smoother than its chief rival AngularJS.
- Backward Compatibility – Transitioning or migrating from older to new versions is fairly easy in react using CodeMods.
- Easy to maintain component structure – Component based architecture of ReactJS helps in increasing the reusability of code and makes maintenance of large scale projects quite a bit easier.
- Rich community – React has close to 1300 contributors on GitHub which is more than any other library/framework.
- Unidirectional Data Flow – One way/Unidirectional downward only data binding in react via flux controls helps to make sure that any changes made to child element structure don't affect the parent element structure

1. React



- Cons Of Using REACT
- Bloated and complex – Many developers might find react too complex and bloated for their purpose as compared to Vue JS. Added complexity that comes along with react might be unnecessary for small scale projects
- JSX – The use of JSX adds another layer of complexity. JSX is a pre-processor, it adds XML syntax extension to JS. Although JSX helps to code react in a safer and faster manner, it is difficult to grasp for new developers.
- Need for Assembly-Tools – React requires a wide array of assembly tools to function properly and be compatible with other libraries and frameworks.
- SEO problems – webpages built with react are known to face issues with indexing by search engine crawlers/bots.



1.React

- PROMINENT USERS
- LambdaTest, Netflix, Yahoo, Airbnb, Instagram, Facebook, WhatsApp , Sberbank, PayPal, Microsoft, BBC,



2.Vue

- Usage Statistics
- 64k websites are powered by Vue.
- 28.8% of javascript developers are currently using vue.
- Ranked 2nd most popular front-end javascript framework in State Of JS survey.
- Github – 125k stars, 18k forks and 253 contributors.



2. Vue

- Why VUE.JS Is One Of The Top JavaScript Frameworks For 2019?
- Fast Configuration: Vue has an inbuilt MVC model which makes configuration quick and easy compared to angular and react.
- Flexibility: Flexibility and modular nature of Vue helps developers with past experiences on other frameworks and libraries to quickly adopt Vue with minimal effort.
- Easy Learning curve: Learning curve of Vue is much easier than Angular and react. Only knowledge of HTML and JavaScript is required.
- Lightweight Size: Vue is highly lightweight and compact in size compare to all other major JavaScript frameworks like angular and react. Compressed version of Vue is only 18 kb in size.
- Integration Capability: Vue can be used to build both SPAs and large-scale complex applications as well with easy integration with server-side frameworks like Laravel, Symfony Django etc.
- Transitions: Vue makes it easy to add transitions to elements when they are added or removed from the DOM.



2. Vue

- Cons Of Using VUE
- Poor Support: Vue doesn't offer a big resource collection to new users on a scale similar to that of angular and react. Considering a comparatively lower market share, developers who are new to Vue js might not find adequate support and resources.
- Small Community:: A large chunk of Vue developer community is from non-English speaking eastern European countries. Community engagement used to be fairly low, However that is expected to change in 2019 as Vue saw a mammoth rise in popularity by the end of 2018.



2. Vue

- **PROMINENT CLIENTS**
- StackOverflow, 9gag, Adobe, Xiaomi, Grammarly, Alibaba.



3. Angular

- Usage Statistics
- 350k websites on the web are powered by AngularJS/Angular2+ versions.
- 23.9% of javascript developers are currently using Angular.
- Ranked 3rd most popular front-end javascript framework in State Of JS survey.
- Github – 44.5k stars, 11.5k forks and 840 contributors.



3. Angular

- Why ANGULAR Is One Of The Top JavaScript Frameworks For 2019?
- Data binding Apart from 2-way binding, angular 2 and beyond support 3 other types of data-binding namely – one-way property binding, event binding and interpolation.
- Enhanced RXJS has led to lightning fast compilation time of approx. 2.9 sec and modified start HttpClient.
- Abundant resources and support: Google offers a rich treasure trove of resources and rolls out new updates and improvements every 6 months.
- Support for Progressive web applications: Angular is the first framework to integrate features for development of progressive web applications.
- MVVM (Model-View-ViewModel) allows developers to work on the same application and data pool in isolation.
- Native mobile app: NativeScript powered by angular enables development of native mobile apps for both iOS and android.



3. Angular

- Difficult Syntax: Angular has made a big leap in terms of syntax complexity after adopting TypeScript in angular2 and beyond. Yet it poses a steep learning curve for anyone making a transition from vanilla JS or jQuery.
- Migration: It is complex to migrate from older angular version especially from AngularJS to angular 7.
- Steep learning curve: out of the three top javascript front-end frameworks, Angular has the steepest learning curve and suffers from a very high rate of abandonment. A large base of angular users has moved to Vue and react.
- Knowledge of MVC: A crucial prerequisite for using angular is the need to have a thorough understanding of model view controller architecture



3. Angular

- **PROMINENT CLIENTS**
- **Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.**



4. Ember

- Usage Statistics
- 7k websites on the web are powered by Ember.
- 6.3% of javascript developers are currently using Ember. Its popularity has stagnated over the last few years.
- Ranked 4rd most popular front-end JavaScript framework in State Of JS survey.
- Github – 20k stars, 4k forks and 750 contributors.



4. Ember

- Why EMBER Is One Of The Top JavaScript Frameworks For 2019?
- Ember templates: One of the most striking features of ember is templates which help to create stunning UI. Templates are coded with Handlebars language which drastically reduces the necessary code. Templates are updated automatically as soon as moderation is made to their underlying data.
- Ember CLI: Ember-CLI is a command line addon that is packed alongside Ember stack. Ember-CLI provides support for techs like Sass/Less, CoffeeScript, Handlebars etc.
- Ember is shipped with Inspector tool to helps with debugging
- Convention over Configuration: Ember is based on Convention over Configuration (CoC). This helps devs to focus more on functionality and building apps faster and worrying less about planning High performance.



4. Ember

- Problems With Ember
- Losing Market share: Ember has seen massive stagnation over past 5 years and faces obscurity in future. Big companies like Netflix and Airbnb have dumped ember and moved to other frameworks like react.
- Lack of Up to date resources: Ember has a very poor support offering due to a shrinking community and is plagued with outdated resources.
- Too complex to be implemented in small projects. Takes too much time to configure and use which eats away into productivity.
- In comparison to react and Vue, ember is extremely complicated and difficult to learn especially for beginners.



4. Ember

- **PROMINENT USERS**
- Apple music, Groupon, Vine, Twitch, Chipotle, Atlas.io, sitepoint



5. Polymer

- **Usage Statistics**
- **4k websites on the internet are powered by Polymer.**
- **3.8% of javascript developers are currently using Polymer.**
- **Ranked 6th most popular front-end javascript framework in State Of JS survey.**
- **Github – 20k stars, 1.9k forks and 140 contributors.**



5. Polymer

- **Why POLYMER Is One Of The Top JavaScript Frameworks For 2019?**
- **Polymer.js enables developers to build their own custom HTML elements.**
- **Lightning Speed – compared to other front end JavaScript frameworks, polymer clocks lightning fast speed especially on chrome and safari browser.**
- **Supports both One-way and two-way data binding.**
- **Cross browser compatibility:** polymer was designed especially with keeping cross browser compatibility testing in mind.



5. Polymer

- **Problems With Polymer**
- **Unlike its peers, it lacks server-side rendering.**
- **Not abundant resources:** As compared to react Vue and angular, polymer has scarce resources available on the net and has a very small community which has shunted its growth in popularity .



5. Polymer

- **PROMINENT CLIENTS**
- **Google Apps like youtube, play music, Electronic Arts, ING, Coca Cola, McDonald's, IBM, General Electric**



Why Angular?



edureka!

YouTube
video inside.

Why Angular?



| | jQuery | Angular |
|----------------------|--------|---------|
| DOM Manipulation | ✓ | ✓ |
| RESTful API | ✗ | ✓ |
| Animation Support | ✓ | ✓ |
| Deep Linking Routing | ✗ | ✓ |
| Form Validation | ✗ | ✓ |
| 2 Way Data Binding | ✗ | ✓ |
| AJAX/JSONP | ✓ | ✓ |



What is a Single Page Application?



SPA

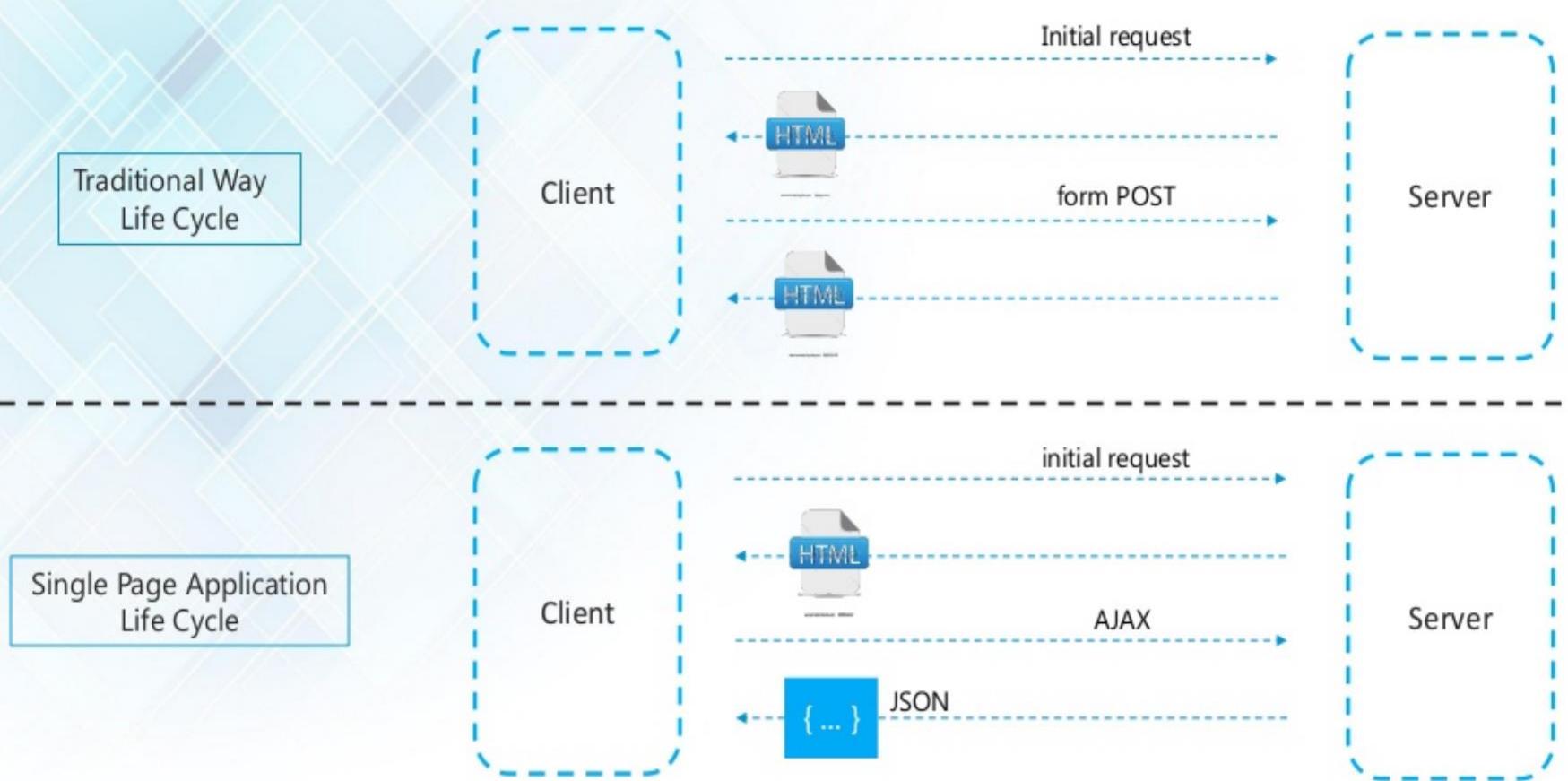
A Single Page Application is a web application that requires only a single page load in a web browser.

- Whole page is not reloaded every time
- Your browser fully renders the DOM once
- Later any server interactions is performed by JavaScript which modifies the view

The screenshot shows the Gmail inbox interface. A blue dashed box highlights the top navigation bar with the Google logo, search bar, and search icon. Below it, a blue box highlights the left sidebar menu with 'COMPOSE' at the top, followed by 'Inbox (2)', 'Starred', 'Sent Mail', 'Drafts', 'Less ▾', 'Important', 'Chats', 'All Mail' (which is bolded), 'Spam', 'Trash', and 'Categories'. A blue arrow points from the 'Categories' link to the text 'mail categories' below. Another blue arrow points from the 'Inbox (2)' link to the text 'mail inbox' below. The main content area shows two emails in the inbox: one from 'Ashish Bakshi' with the subject 'Angular 4 is fun -- Thanks and regards, Ashish Bakshi' and another from 'Andy from Google' with the subject 'John, get more out of your new Google Account - Hi John.' Below the inbox, there is a message about storage usage: '0 GB (0%) of 15 GB used' with a 'Manage' link, and links for 'Terms - Privacy'.



Traditional Vs SPA





YouTube
video inside.

Angular Introduction



Angular Introduction

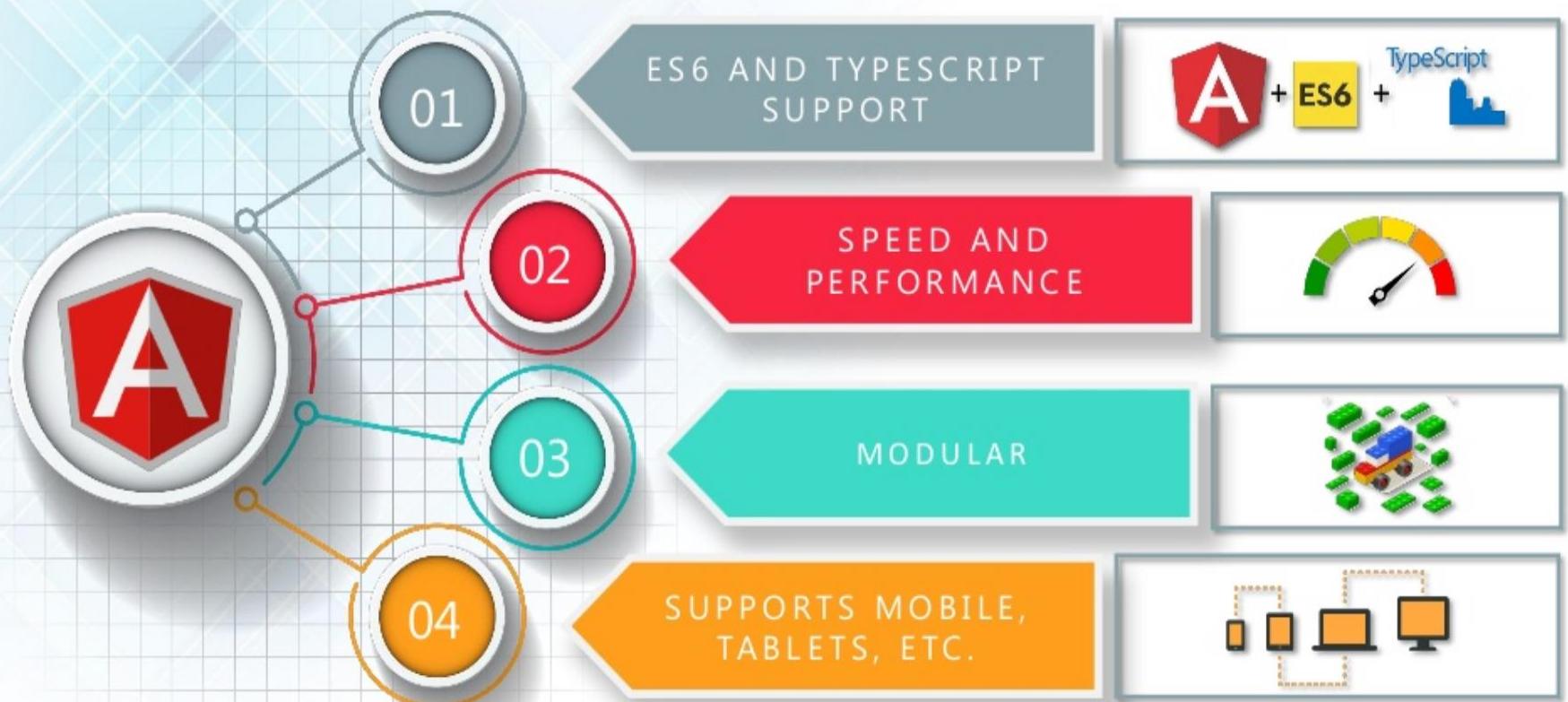




Angular Features



Angular Features





Angular Installation



IDEs and Tools

IDEs

- Amexio Canvas Web Based Drag and Drop IDE by MetaMagic
 - Amexio Canvas is Drag and Drop Environment to create Fully Responsive Web and Smart Device HTML5/Angular Apps. Code will be auto generated and hot deployed by the Canvas for live testing. Out of the box 50+ Material Design Theme support. Commit your code to GitHub public or private repository.
- Angular IDE by Webclipse
 - Built first and foremost for Angular. Turnkey setup for beginners; powerful for experts.
- IntelliJ IDEA
 - Capable and Ergonomic Java * IDE
- Visual Studio Code
 - VS Code is a Free, Lightweight Tool for Editing and Debugging Web Apps.
- WebStorm
 - Lightweight yet powerful IDE, perfectly equipped for complex client-side development and server-side development with Node.js



IDEs and Tools

- Tooling
- Angular CLI
 - The official Angular CLI makes it easy to create and develop applications from initial commit to production deployment. It already follows our best practices right out of the box!
- Angular Playground
 - UI development environment for building, testing, and documenting Angular applications.
- Angular Universal
 - Server-side Rendering for Angular apps.
- Augury
 - A Google Chrome Dev Tools extension for debugging Angular applications.



IDEs and Tools

- Celerio Angular Quickstart
 - Generate an Angular CRUD application from an existing database schema
- Codelyzer
 - Static analysis for Angular projects.
- Compodoc
 - This tool generates dedicated documentation for Angular applications.
- Lite-server
 - Lightweight development only Node.js® server
- NinjaCodeGen - Angular CRUD Generator
 - Generate several types of CRUD apps complete with e2e testing using template-sets for Angular, Material Design, Bootstrap, Kendo UI, Ionic, ...



IDEs and Tools

- Nx
 - Nx (Nrwl Extensions for Angular) is an open source toolkit built on top of Angular CLI to help enterprise teams develop Angular at scale.
- Protractor
 - The official end to end testing framework for Angular apps
- UI-jar - Test Driven Style Guide Development
 - A drop in module to automatically create a living style guide based on the test you write for your components.
- Universal for ASP.NET
 - This package provides facilities for developers building Angular applications on ASP.NET.



IDEs and Tools

- Angular Fire
 - The official library for Firebase and Angular
- AngularCommerce
 - Angular Commerce is a solution for building modern e-commerce applications with power of Google Firebase. Set of components is design agnostic and allows to easily extend functionality.
- Apollo
 - Apollo is a data stack for modern apps, built with GraphQL.
- Meteor
 - Use Angular and Meteor to build full-stack JavaScript apps for Mobile and Desktop.
- ngrx
 - Reactive Extensions for Angular
- ngx-api-utils
 - ngx-api-utils is a lean library of utilities and helpers to quickly integrate any HTTP API (REST, Ajax, and any other) with Angular.



IDEs and Tools

- UI Components
- ag-Grid
 - A datagrid for Angular with enterprise style features such as sorting, filtering, custom rendering, editing, grouping, aggregation and pivoting.
- Amexio - Angular Extensions
 - Amexio (Angular MetaMagic EXtensions for Inputs and Outputs) is a rich set of Angular components powered by Bootstrap for Responsive Design.
 - UI Components include Standard Form Components, Data Grids, Tree Grids, Tabs etc. Open Source (Apache 2 License) & Free and backed by MetaMagic Global Inc



IDEs and Tools

- Angular Material
 - Material Design components for Angular
- Ant Design Mobile of Angular (`ng-zorro-antd-mobile`)
 - A set of enterprise-class mobile UI components based on Ant Design Mobile and Angular
- Material Community Components
 - Material components made by the community



IDEs and Tools

- Ant Design of Angular (ng-zorro-antd)
 - A set of enterprise-class UI components based on Ant Design and Angular
- Blox Material
 - A lightweight Material Design library for Angular, based upon Google's Material Components for the Web
- Clarity Design System
 - UX guidelines, HTML/CSS framework, and Angular components working together to craft exceptional experiences
- DevExtreme
 - 50+ UI components including data grid, pivot grid, scheduler, charts, editors, maps and other multi-purpose controls for creating highly responsive web applications for touch devices and traditional desktops.



IDEs and Tools

- Ant Design of Angular (ng-zorro-antd)
 - A set of enterprise-class UI components based on Ant Design and Angular
- Blox Material
 - A lightweight Material Design library for Angular, based upon Google's Material Components for the Web
- Clarity Design System
 - UX guidelines, HTML/CSS framework, and Angular components working together to craft exceptional experiences
- DevExtreme
 - 50+ UI components including data grid, pivot grid, scheduler, charts, editors, maps and other multi-purpose controls for creating highly responsive web applications for touch devices and traditional desktops.



IDEs and Tools

- Essential JS 2
 - Essential JS 2 for Angular is a collection modern TypeScript based true Angular Components. It has support for Ahead Of Time (AOT) compilation and Tree-Shaking. All the components are developed from the ground up to be lightweight, responsive, modular and touch friendly.
- Ignite UI for Angular
 - Ignite UI for Angular is a dependency-free Angular toolkit for building modern web apps.
- jQWidgets
 - Angular UI Components including data grid, tree grid, pivot grid, scheduler, charts, editors and other multi-purpose components
- Kendo UI
 - A professional grade library of Angular UI components written in TypeScript that includes our Data Grid, TreeView, Charts, Editors, DropDowns, DatePickers, and many more. Features include support for AOT compilation, Tree Shaking for high-performance, localization, and accessibility.



IDEs and Tools

- ng-bootstrap
 - The Angular version of the Angular UI Bootstrap library. This library is being built from scratch in Typescript using the Bootstrap 4 CSS framework.
- ng-lightning
 - Native Angular components & directives for Lightning Design System
- ngx-bootstrap
 - Native Angular directives for Bootstrap
- Onsen UI
 - UI components for hybrid mobile apps with bindings for both Angular & AngularJS.
- Prime Faces
 - PrimeNG is a collection of rich UI components for Angular

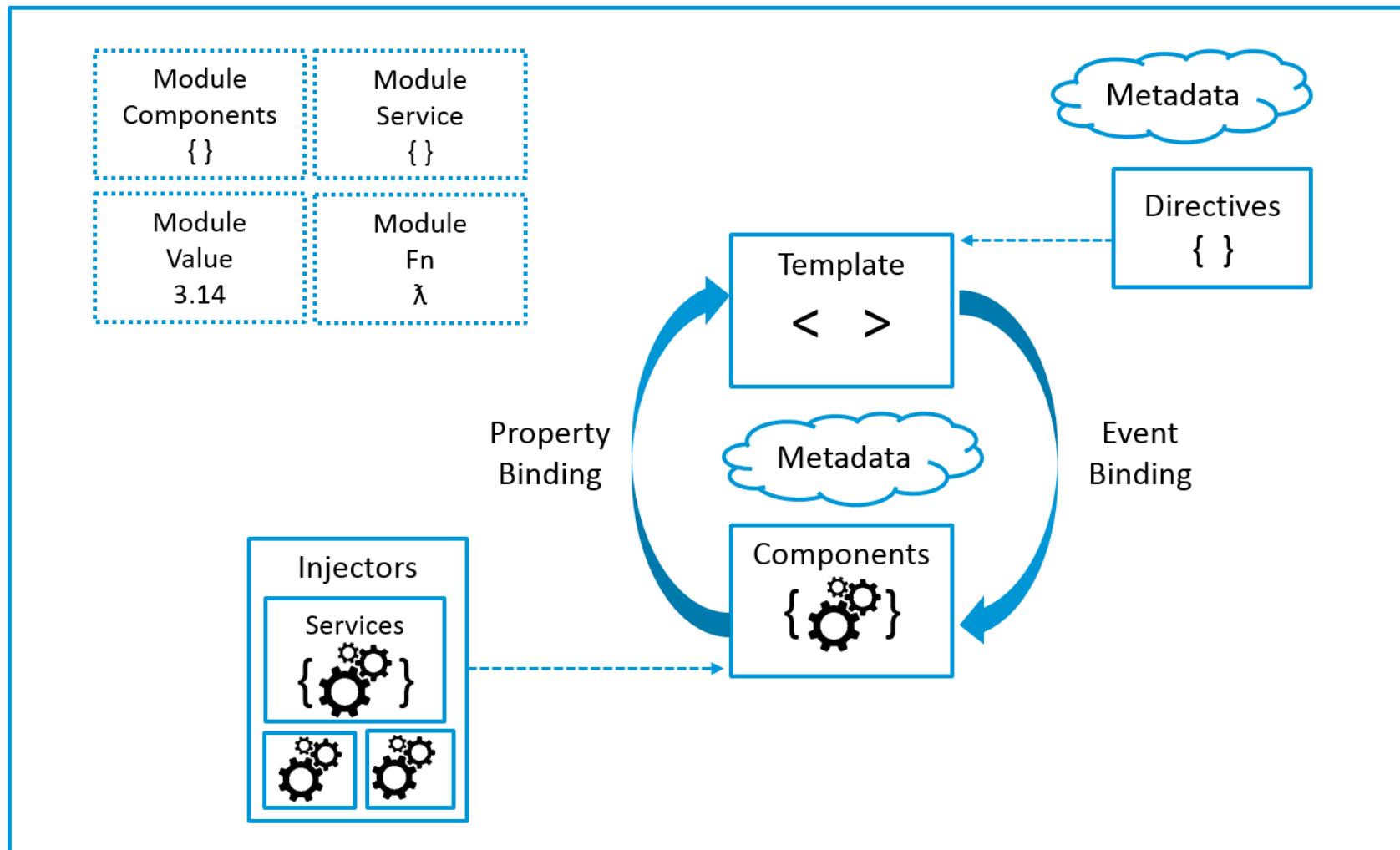


The Angular Journey





Architecture overview





Architecture overview

- Angular is a platform and framework for building client applications in HTML and TypeScript.
- Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.
- The basic building blocks of an Angular application are NgModules, which provide a compilation context for components.
- NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules.



Architecture overview

- An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.
- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.



Architecture overview

- Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.
- The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).
- An app's components typically define many views, arranged hierarchically. Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.



Architecture overview

- Modules
- Angular NgModules differ from and complement JavaScript (ES2015) modules.
- An NgModule declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities.
- An NgModule can associate its components with related code, such as services, to form functional units.
- Every Angular app has a root module, conventionally named AppModule, which provides the bootstrap mechanism that launches the application.
- An app typically contains many functional modules.



Architecture overview

- Components
- Every Angular application has at least one component, the root component that connects a component hierarchy with the page document object model (DOM).
- Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.
- The `@Component()` decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.
- Decorators are functions that modify JavaScript classes. Angular defines a number of decorators that attach specific kinds of metadata to classes, so that the system knows what those classes mean and how they should work.



Architecture overview

- Templates, directives, and data binding
- A template combines HTML with Angular markup that can modify HTML elements before they are displayed.
- Template directives provide program logic, and binding markup connects your application data and the DOM.
There are two types of data binding:
- Event binding lets your app respond to user input in the target environment by updating your application data.
- Property binding lets you interpolate values that are computed from your application data into the HTML.



Architecture overview

- Services and dependency injection
- For data or logic that isn't associated with a specific view, and that you want to share across components, you create a service class.
- A service class definition is immediately preceded by the `@Injectable()` decorator. The decorator provides the metadata that allows your service to be injected into client components as a dependency.
- Dependency injection (DI) lets you keep your component classes lean and efficient. They don't fetch data from the server, validate user input, or log directly to the console; they delegate such tasks to services.



Architecture overview

- Routing
- The Angular Router NgModule provides a service that lets you define a navigation path among the different application states and view hierarchies in your app. It is modeled on the familiar browser navigation conventions:
- Enter a URL in the address bar and the browser navigates to a corresponding page.
- Click links on the page and the browser navigates to a new page.
- Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen.



Building Blocks of Angular

Module

Component

Metadata

Template

Data Binding

Services

Directives

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

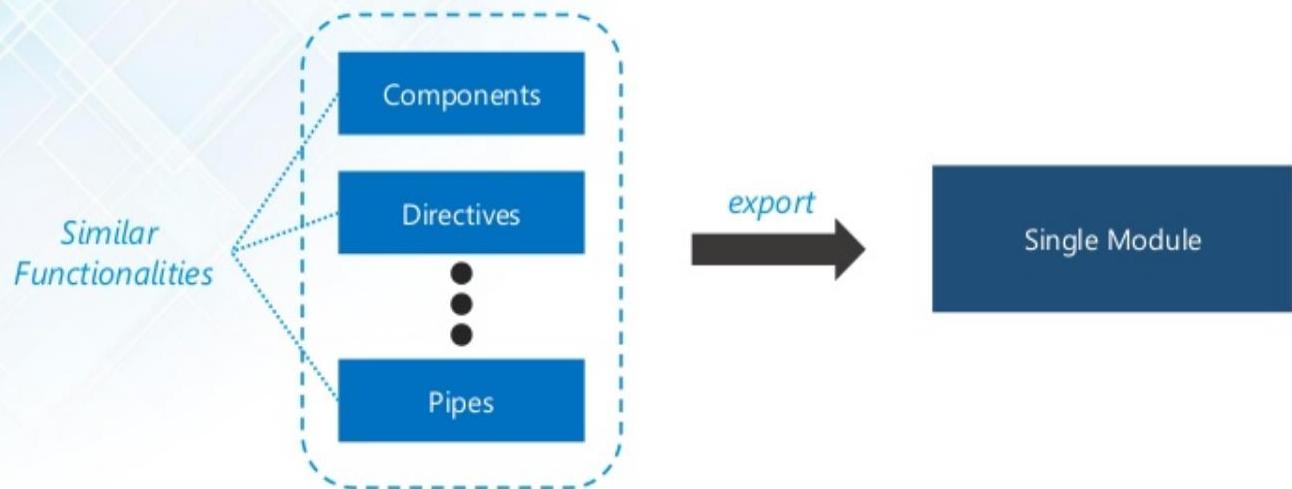
Services

Directives

Module is a class with
@NgModule metadata

Every Angular app has at
least one root module

Encapsulation of different
similar functionalities





Building Blocks of Angular

Module

Component

Metadata

Template

Data Binding

Services

Directives

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/http';

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

@NgModule({
  declarations: [
    AppComponent,
    TaskComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Decorator

Declaring all the components

Importing Modules

Provide Services to all module's component

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

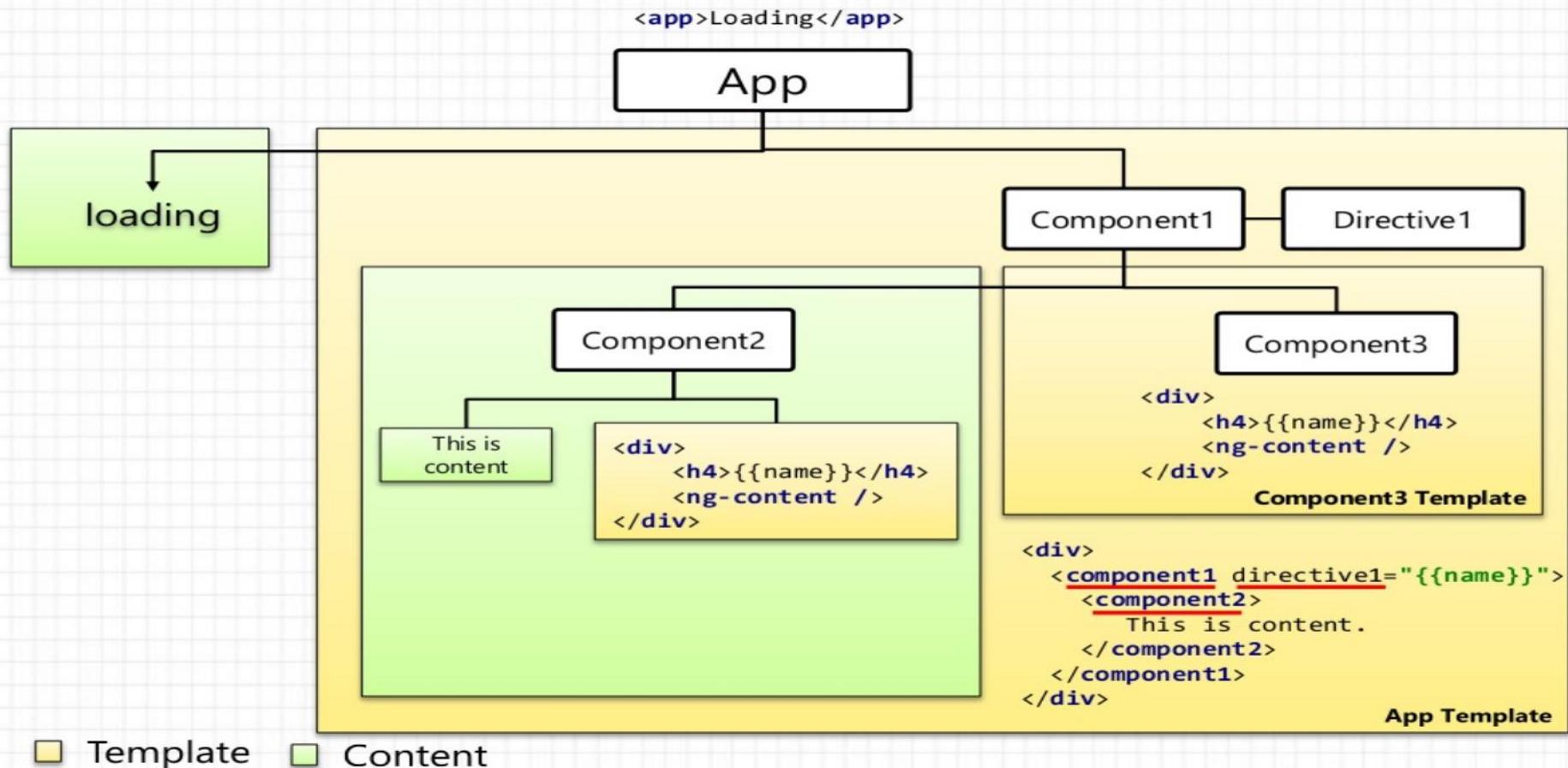
Services

Directives

The screenshot shows the homepage of theguardian.com. At the top, there is a dark blue header with a sign in button, a search icon, and links for jobs, more, and International edition. The main title "the guardian" is prominently displayed. Below the header is a navigation bar with categories: home, tech (which is highlighted in red), UK, world, sport, football, opinion, culture, business, lifestyle, and all. A red arrow points from this bar to the text "Nav Bar". The main content area features a green sidebar on the left with the heading "technology" and a news article about a drone near a plane at Heathrow. To the right of the sidebar are two news cards: one showing an airplane on a runway and another showing a person holding a Samsung Galaxy S8 phone. A green arrow points from this section to the text "News Feed".



Tree Components



Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  styleUrls: ['./example.component.css']
})
export class ExampleComponent implements OnInit {
  constructor() { }
  ngOnInit() {
  }
}
```

Importing Component Decorator → `import { Component, OnInit } from '@angular/core';`

Decorator → `@Component({`

Meta Data → `selector: 'app-example',
 templateUrl: './example.component.html',
 styleUrls: ['./example.component.css']`

Exporting Component Class → `export class ExampleComponent implements OnInit {`

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

Metadata describes how
to process the class

Decorator is used to
attach metadata

Example:

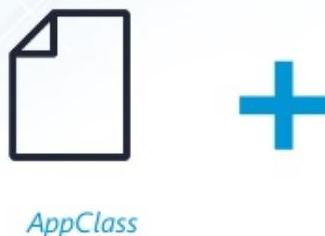


@Component({
.....
})

Decorator



Component
{ }



@NgModule({
.....
})

Decorator



Module
{ }

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

```
@Component({  
  selector: 'app-example',  
  templateUrl: './example.component.html',  
  styleUrls: ['./example.component.css'],  
  providers: [ExampleService]  
})
```

Decorator that specifies how to process an Angular Class

Creates an instance of the component

HTML template for the component

CSS Styling

Provides Service for the Component



Component Metadata

- **Names:**

- selector? : string
- exportAs? : string

- **Binding:**

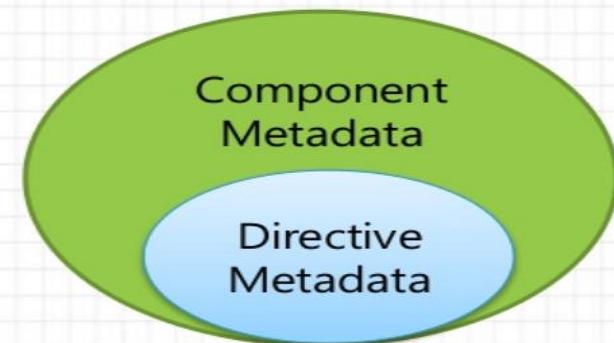
- inputs? : string[]
- outputs? : string[]
- host? : {[key: string]: string}
- changeDetection?: ChangeDetectionStrategy

- **Injector:**

- providers? : any[]
- viewProviders? : any[]
- directives? : Array<Type | any[]>
- pipes? : Array<Type | any[]>
- queries? : {[key: string]: any}

- **View:**

- templateUrl? : string
- template? : string
- styleUrls? : string[]
- styles? : string[]
- encapsulation?: ViewEncapsulation



Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

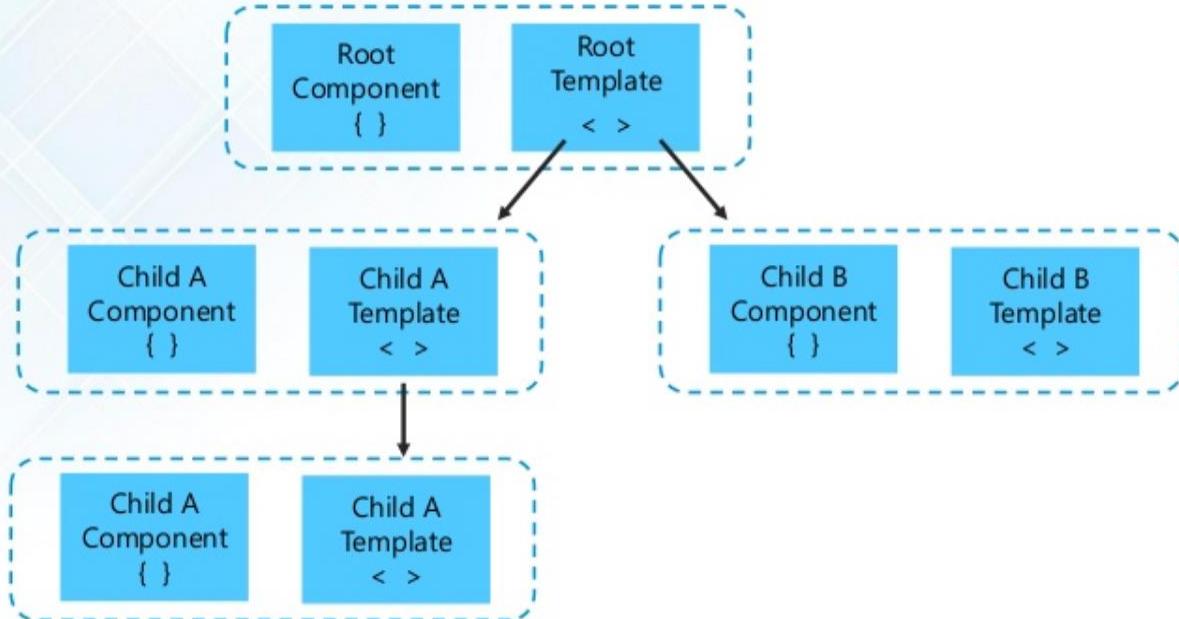
Services

Directives

Used to define view of a component

Looks like HTML, except for a few differences.

Describes how the component is rendered on the page.



Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

TYPES OF DATA BINDING

Data binding plays an important role in communication between a template and its component

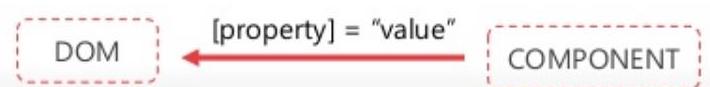
INTERPOLATION

01



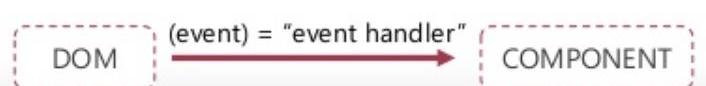
PROPERTY BINDING

02



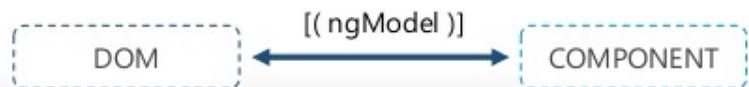
EVENT BINDING

03



2 WAY DATA BINDING

04



Building Blocks of Angular



Module

Component

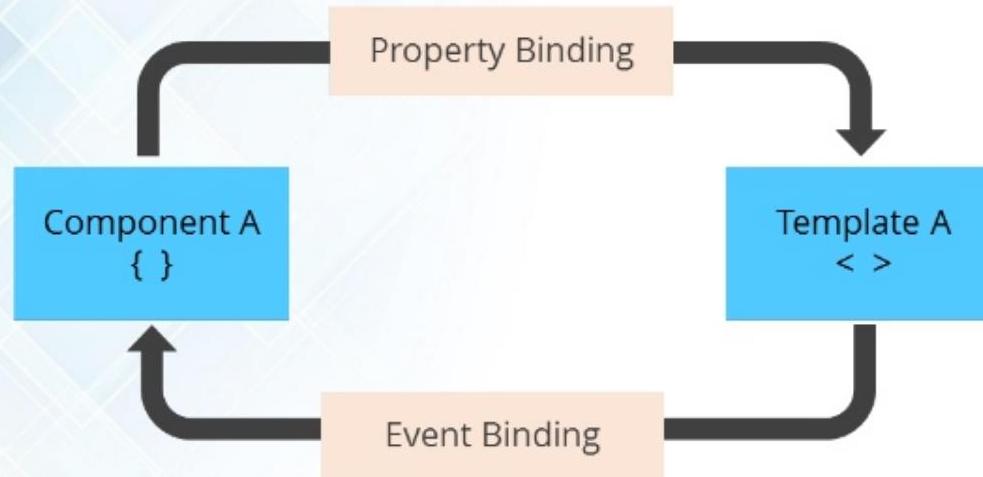
Metadata

Template

Data Binding

Services

Directives



Data binding plays an important role in communication
between a template and its component.

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

Parent Component
{ }

Property Binding

Parent Template
< >

Child Component
{ }

Event Binding

Data binding is also important for communication between
parent and child components.

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

Service is a broad category encompassing any value, function, or feature that your application needs.

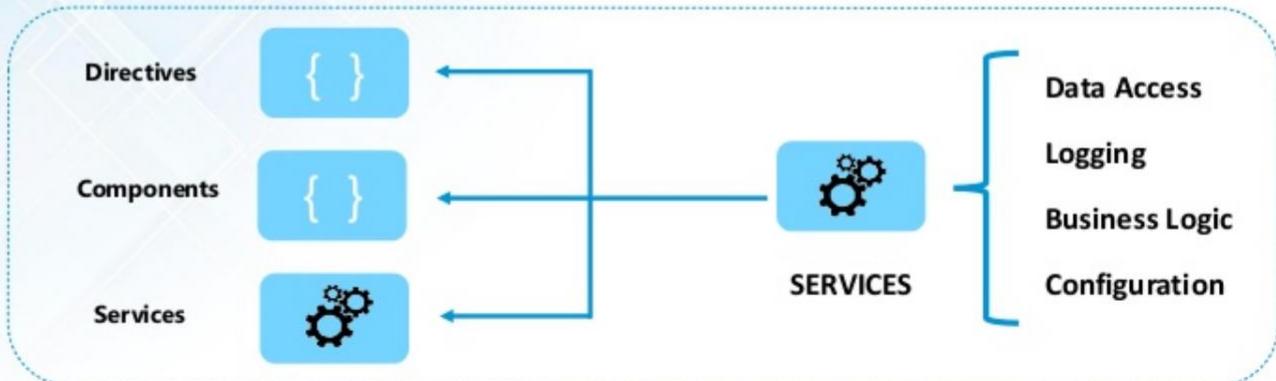
Example:

logging service

data service

message bus

tax calculator



Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

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

@Injectable()
export class ExampleService {

  movies: string[] = ["Inception", "Dark Knight", "Shutter Island"];

  constructor() { }

  getMovies(): string[]
  {
    return this.movies;
  }
}
```

Service Class

Service Method for retrieving data

Building Blocks of Angular



Module

Component

Metadata

Template

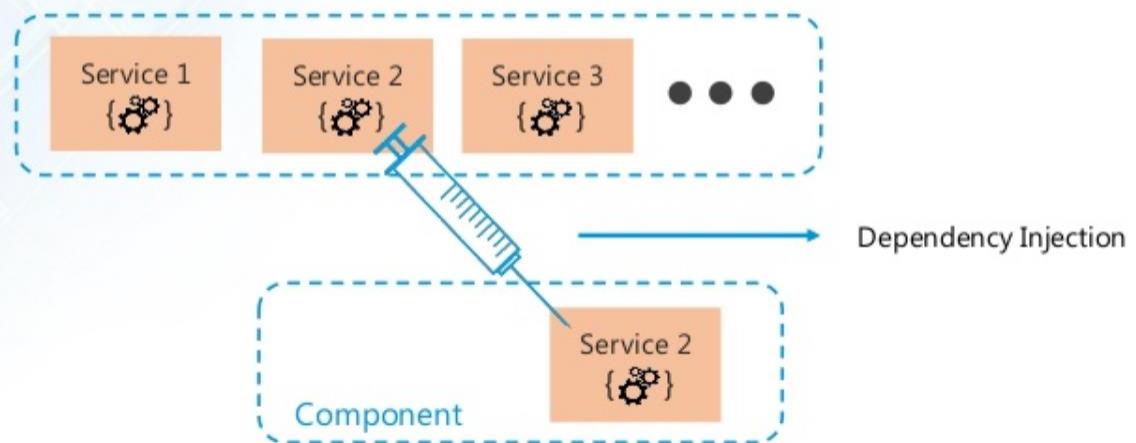
Data Binding

Services

Directives

Creates a new instance of class along with its required dependencies

Used to provide services to a component



Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

```
import { Component, OnInit } from '@angular/core';
import {ExampleService} from '../example.service';

@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  styleUrls: ['./example.component.css'],
  providers: [ExampleService]
})
export class ExampleComponent implements OnInit {

  movies: string[];
  constructor(private exampleService: ExampleService) {}

  ngOnInit() {
    this.movies = this.exampleService.getMovies();
  }
}
```

Importing Service Class

Injecting Service into the Component

Retrieving data

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

Changes the appearance or behavior of a DOM element

1

COMPONENTS

Directives with a template

2

STRUCTURAL DIRECTIVE

Adds & removes DOM elements
to change DOM layout

3

ATTRIBUTE DIRECTIVE

Changes the appearance or
behavior of an element

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

2

STRUCTURAL DIRECTIVE

Adds & removes DOM elements
to change DOM layout

```
<ul>
  <li *ngFor = "let movie of movies">{{movie}}</li>
</ul>
```

Iterating over
the movies list

Building Blocks of Angular



Module

Component

Metadata

Template

Data Binding

Services

Directives

3

ATTRIBUTE DIRECTIVE

Changes the appearance or behavior of an element

```
import { Directive, ElementRef, HostListener } from '@angular/core';  
  
@Directive({  
  selector: '[appBoldText]'  
})  
export class BoldTextDirective {  
  
  constructor(private elementRef: ElementRef) {}  
  
  @HostListener('mouseenter') onMouseEnter() {  
    this.elementRef.nativeElement.style.fontWeight = 'bold';  
  }  
  
  @HostListener('mouseleave') onMouseLeave() {  
    this.elementRef.nativeElement.style.fontWeight = null;  
  }  
}
```

Importing Directive, ElementRef & HostListener

Directive Metadata

Injecting ElementRef to access the DOM elements

Bold the text on cursor hover

Un-bold the text



Installation

- `npm install -g @angular/cli`
- `npm install -g @angular/cli@8.3.21`
- `npm install -g @angular/cli@12.2.12`
- `ng new ecommerceapp`
- `ng serve`



Lifecycle Hooks

- Angular calls lifecycle hook methods on directives and components as it **creates**, **changes**, and **destroys** them.

Creates:

- OnInit
- AfterContentInit
- AfterViewInit

Changes:

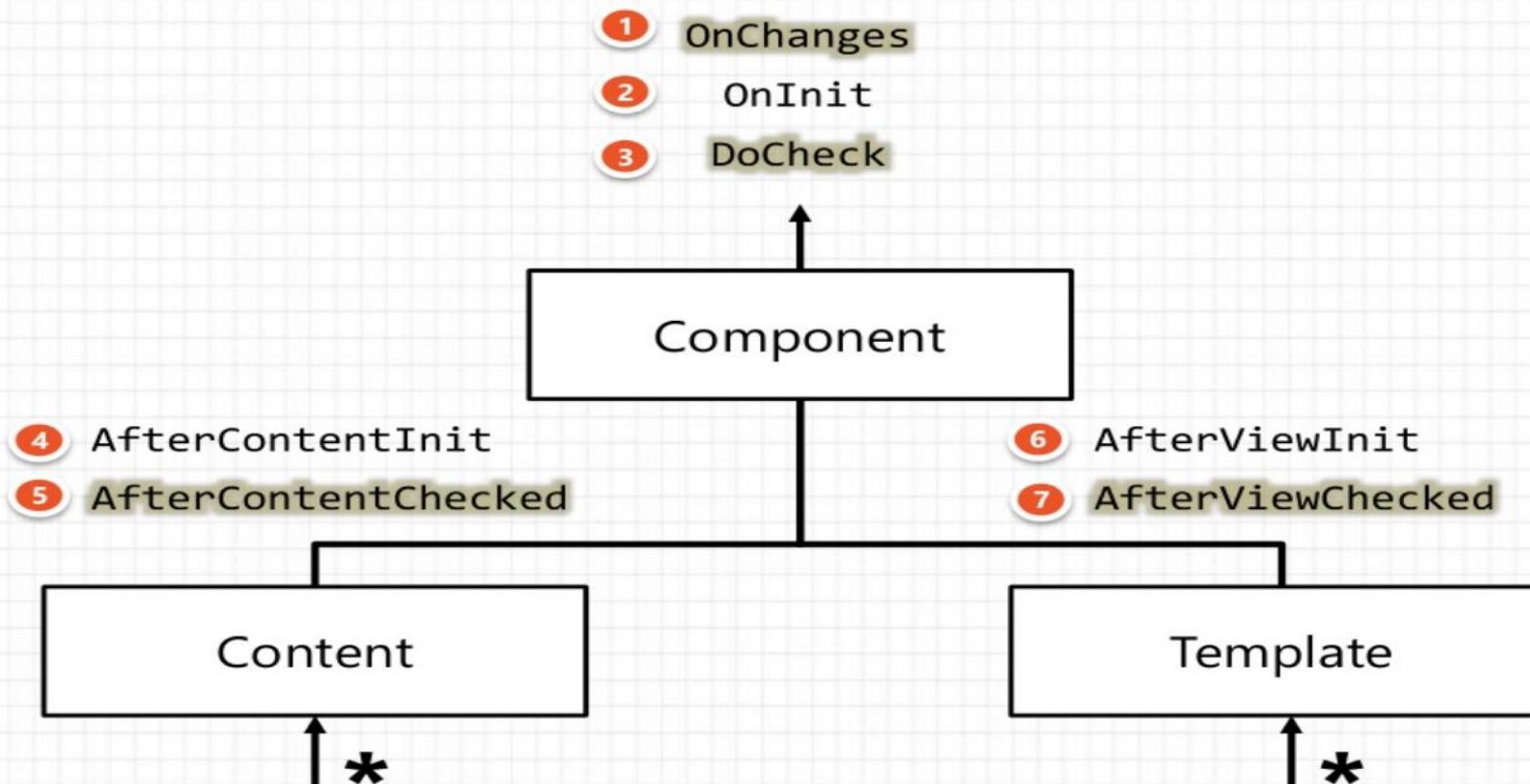
- DoCheck
- OnChanges
- AfterContentChecked
- AfterViewChecked

Destroys:

- OnDestroy



Hooks Order



What Is View Encapsulation in Angular



Emulated

Component level style

- no Shadow DOM
- style encapsulation
- value : 0

Native

Shadow DOM level Style

- Shadow DOM
- style encapsulation
- value : 1

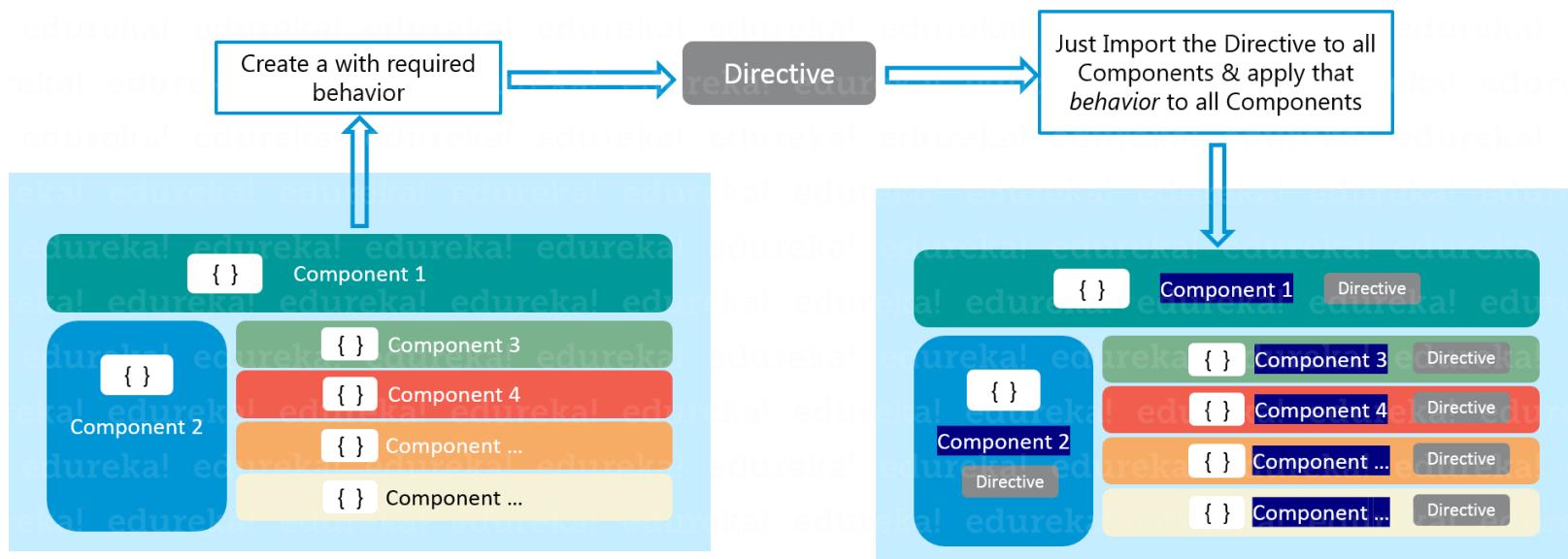
None

Main Html level head section style

- no Shadow DOM
- no Style encapsulation
- value : 2



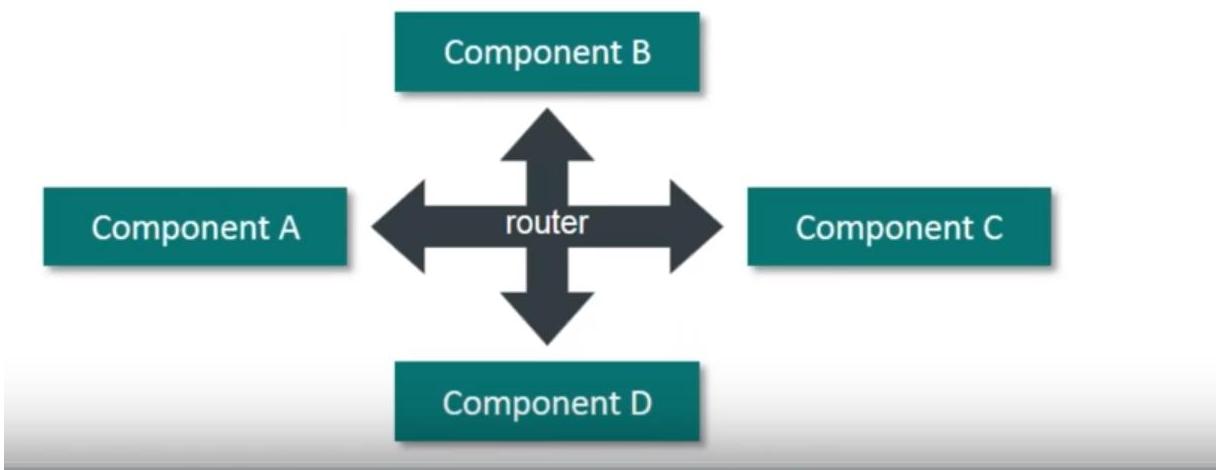
Angular Directives



Directives can be imported anywhere providing similar functionality

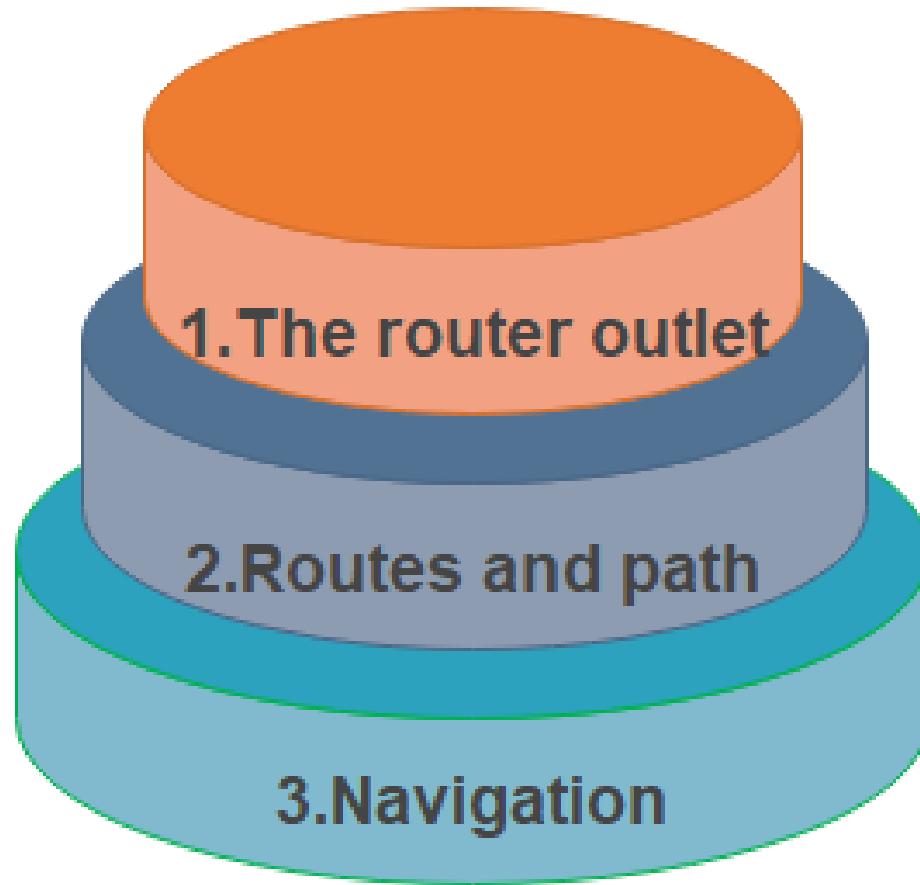


Angular Routing



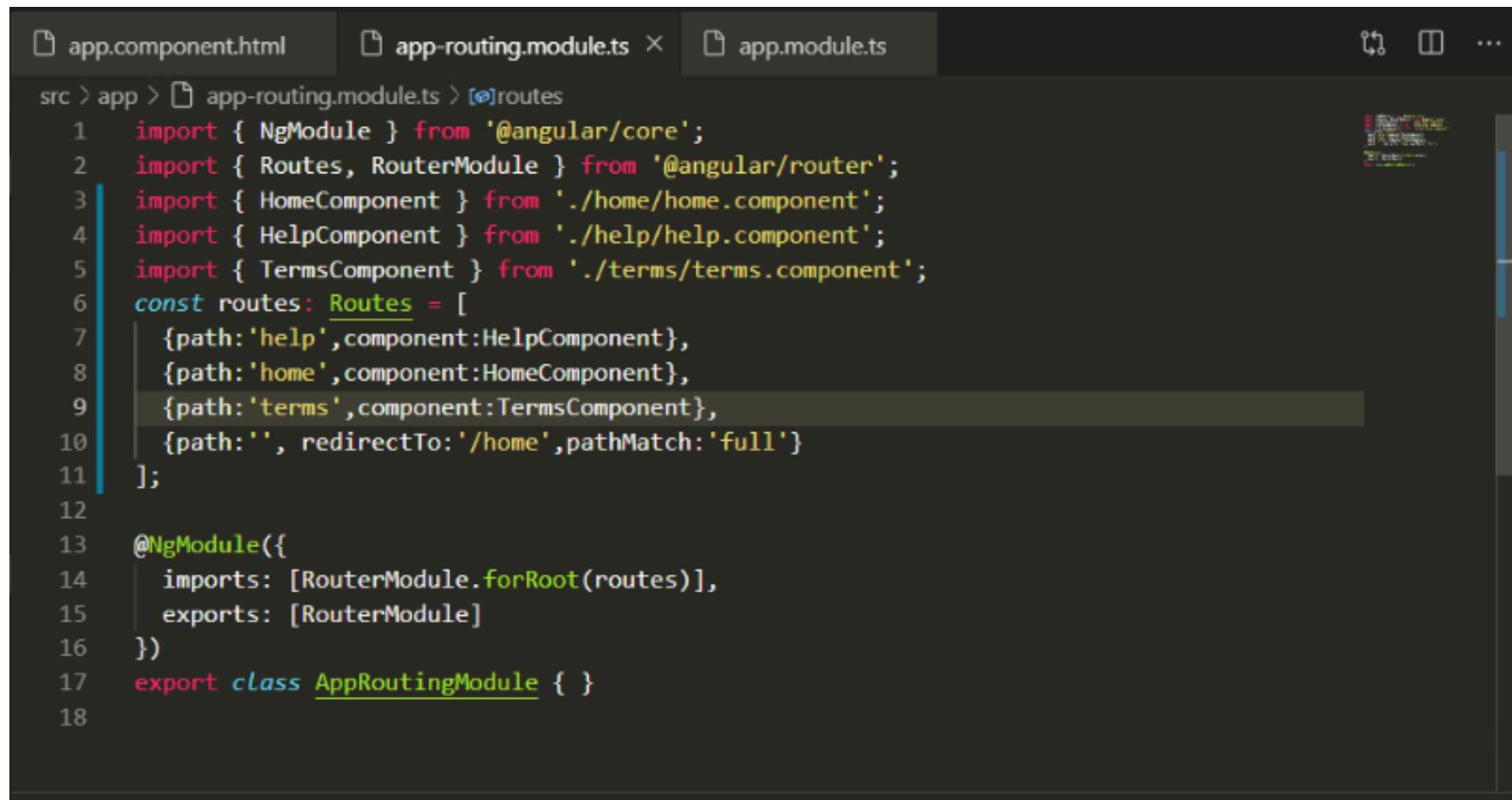


Angular Routing





Angular Routing



```
src > app > app-routing.module.ts > routes
1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3 import { HomeComponent } from './home/home.component';
4 import { HelpComponent } from './help/help.component';
5 import { TermsComponent } from './terms/terms.component';
6 const routes: Routes = [
7   {path: 'help', component: HelpComponent},
8   {path: 'home', component: HomeComponent},
9   {path: 'terms', component: TermsComponent},
10  {path: '', redirectTo: '/home', pathMatch: 'full'}
11 ];
12
13 @NgModule({
14   imports: [RouterModule.forRoot(routes)],
15   exports: [RouterModule]
16 })
17 export class AppRoutingModule { }
```



Lazy Loading

Eager | Lazy

Shell

Feed

Notifications

Messages

Moments



Lazy Loading

- * Inside of app-routing.module.ts > route > children */
- {
- path: 'notifications',
- loadChildren: () =>
 import('./innerComponents/notifications/notifications.m
 odule').then(mod => mod.NotificationsModule),
- }



PreLoading

Eager | Preload | Lazy

Shell

Feed

Notifications

Messages

Moments



PreLoading

```
imports: [
  RouterModule.forRoot(
    routes,
    { preloadingStrategy: CustomPreloaderStrategy }
  )
]
```



Parameterized Routing

```
{ path: "product/:id", component:  
ProductDetailComponent }
```

```
<h1>Products List</h1>  
  
<ul>  
  <li *ngFor="let product of products">  
    <a [routerLink]=["'/product',product.id]"></a>  
  </li>  
</ul>
```



Parameterized Routing

- import { ActivatedRoute } from '@angular/router';
- constructor(private route: ActivatedRoute) {}
- ngOnInit() {
 - this.route.paramMap.subscribe(params => {
 - this.products.forEach((p: Product) => {
 - if (p.id == params.id) {
 - this.product = p;
 - }
 - });
 - });
 - }



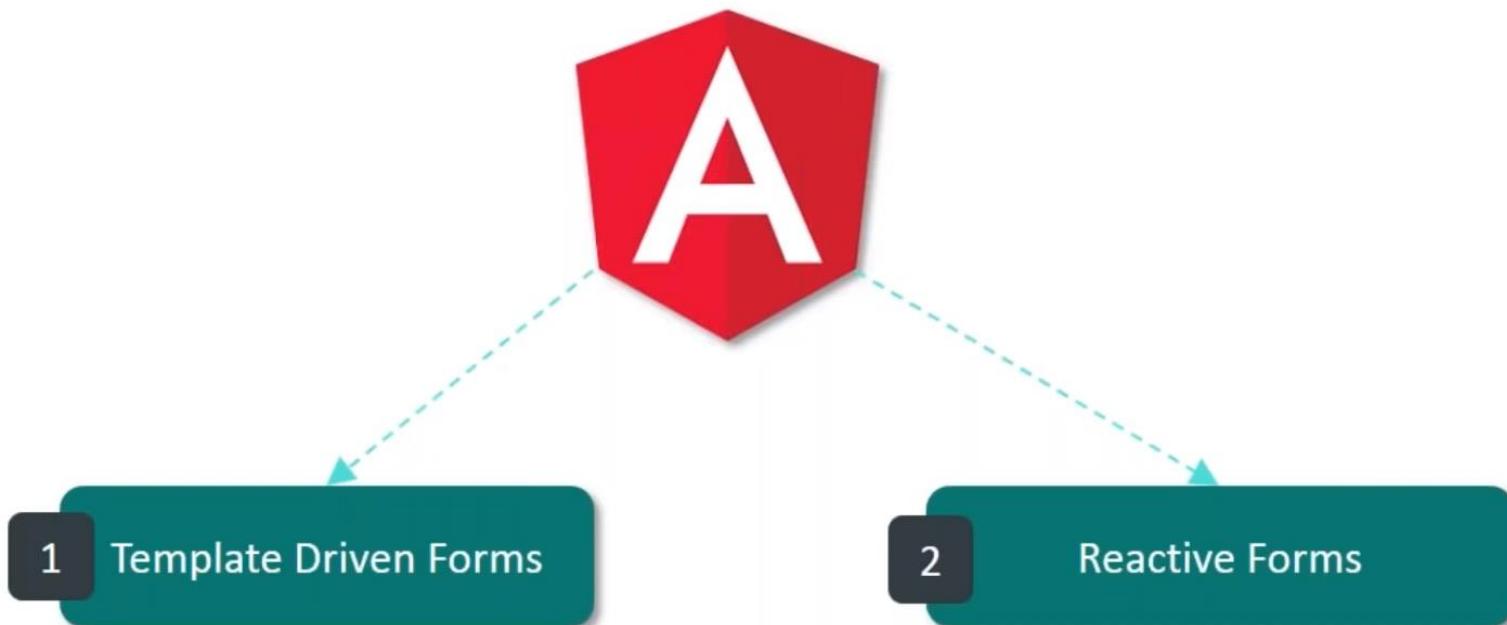
Routing Guards

- Guard Types
- There are four different guard types we can use to protect our routes:
- CanActivate - Decides if a route can be activated
- CanActivateChild - Decides if children routes of a route can be activated
- CanDeactivate - Decides if a route can be deactivated
- CanLoad - Decides if a module can be loaded lazily



Angular Forms

Types of Angular Forms





Template Driven Forms

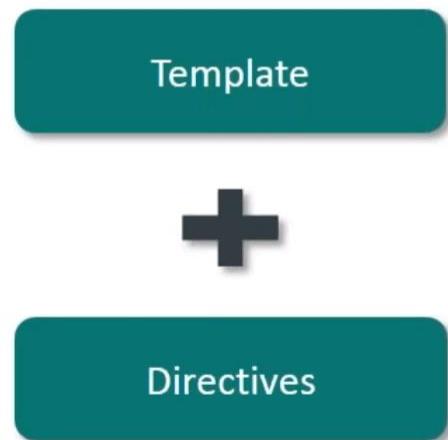
Name:

Contact:

Email:

Address:

Submit





Reactive Forms

Name:

Contact:

Email:

Address:

Submit



Component



Underlying APIs



Template vs Reactive Forms

- Template-driven forms make use of the "FormsModule", while reactive forms are based on "ReactiveFormsModule".
- Template-driven forms are asynchronous in nature, whereas Reactive forms are mostly synchronous.
- In a template-driven approach, most of the logic is driven from the template, whereas in reactive-driven approach, the logic resides mainly in the component or typescript code. Let us get started by generating a component and then we'll update our form code.



Reactive Forms vs. Template-driven Forms

Angular Forms

Template-driven

Easy to use

Similar to Angular 1

Two-way data binding ->
Minimal component code

Automatically tracks form and
input element state

Reactive

More flexible ->
more complex scenarios

Immutable data model

Easier to perform an action
on a value change

Reactive transformations ->
DebounceTime or DistinctUntilChanged

Easily add input elements dynamically

Easier unit testing



Observables

- Observables
- They are cold: Code gets executed when they have at least a single observer.
- Creates copy of data: Observable creates copy of data for each observer.
- Uni-directional: Observer can not assign value to observable(origin/master).



Subject

- They are hot: code gets executed and value gets broadcast even if there is no observer.
- Shares data: Same data get shared between all observers.
- bi-directional: Observer can assign value to observable(origin/master).
- If are using using subject then you miss all the values that are broadcast before creation of observer. So here comes Replay Subject



Replay Subject

- They are hot: code gets executed and value get broadcast even if there is no observer.
- Shares data: Same data get shared between all observers.
- bi-directional: Observer can assign value to observable(origin/master). plus
- Replay the message stream: No matter when you subscribe the replay subject you will receive all the broadcasted messages.
- In subject and replay subject you can not set the initial value to observable. So here comes Behavioral Subject



Behaviour Subject

- They are hot: code gets executed and value get broadcast even if there is no observer.
- Shares data: Same data get shared between all observers.
- bi-directional: Observer can assign value to observable(origin/master). plus
- Replay the message stream: No matter when you subscribe the replay subject you will receive all the broadcasted messages.
- You can set initial value: You can initialize the observable with default value.



Observable vs Subject Behaviour

| Observable | BehaviorSubject/Subject |
|--|--|
| Is just a function, no state | Has state. Stores data in memory |
| Code run for each observer | Same code run only once for all observers |
| Creates only Observable (data producer alone) | Can create and also listen Observable (data producer and consumer) |
| Usage: Simple Observable with only one Obeserver. | Usage: <ul style="list-style-type: none">* Store data and modify frequently* Multiple observers listen to data* Proxy between Observable and Observer |



Observable vs Subject Behaviour

| Observables | Subject | Replay Subject | Behavioral Subject |
|----------------------|----------------|---------------------------|---------------------------|
| Cold | Hot | Hot | Hot |
| Creates copy of data | Shares data | Shares data | Shares data |
| Uni-directional | bi-directional | bi-directional | bi-directional |
| - | - | Replay the message stream | Replay the message stream |
| - | - | - | You can set initial value |

| | |
|-----------------|--|
| | Each next subscribers receive... |
| Subject | ...only upcoming values |
| BehaviorSubject | ...one previous value and upcoming values |
| ReplaySubject | ...all previous values and upcoming values |
| AsyncSubject | ...latest value when stream will close |



| Angular JS | Angular 2 |
|--|--|
| Released by Google in the year 2010. | Released in Sept 2016. |
| JavaScript-based framework for creating SPA. | Complete re-write of AngularJS version. |
| Still supported but no longer will be developed. | It's updated version regularly released because of Semantic Versioning. |
| The architecture of AngularJS is based on MVC. | The architecture of Angular 2 is based on service/controller. |
| AngularJS was not developed with a mobile base in mind. | Angular 2 is a mobile-oriented framework. |
| AngularJS code can write by using only ES5, ES6, and Dart. | We can use ES5, ES6, Typescript to write an Angular 2 code. |
| Based on controllers whose scope is now over. | Nowadays, the controllers are replaced by components, and Angular two is completely component based. |
| Factory, service, provider, value and constant are used for services | The class is the only method to define services in Angular2 |
| Run on only client-side | Runs on client-side & server-side |
| ng-app and angular bootstrap function are used to initialize | bootstrapmodule() function is used to initialize |



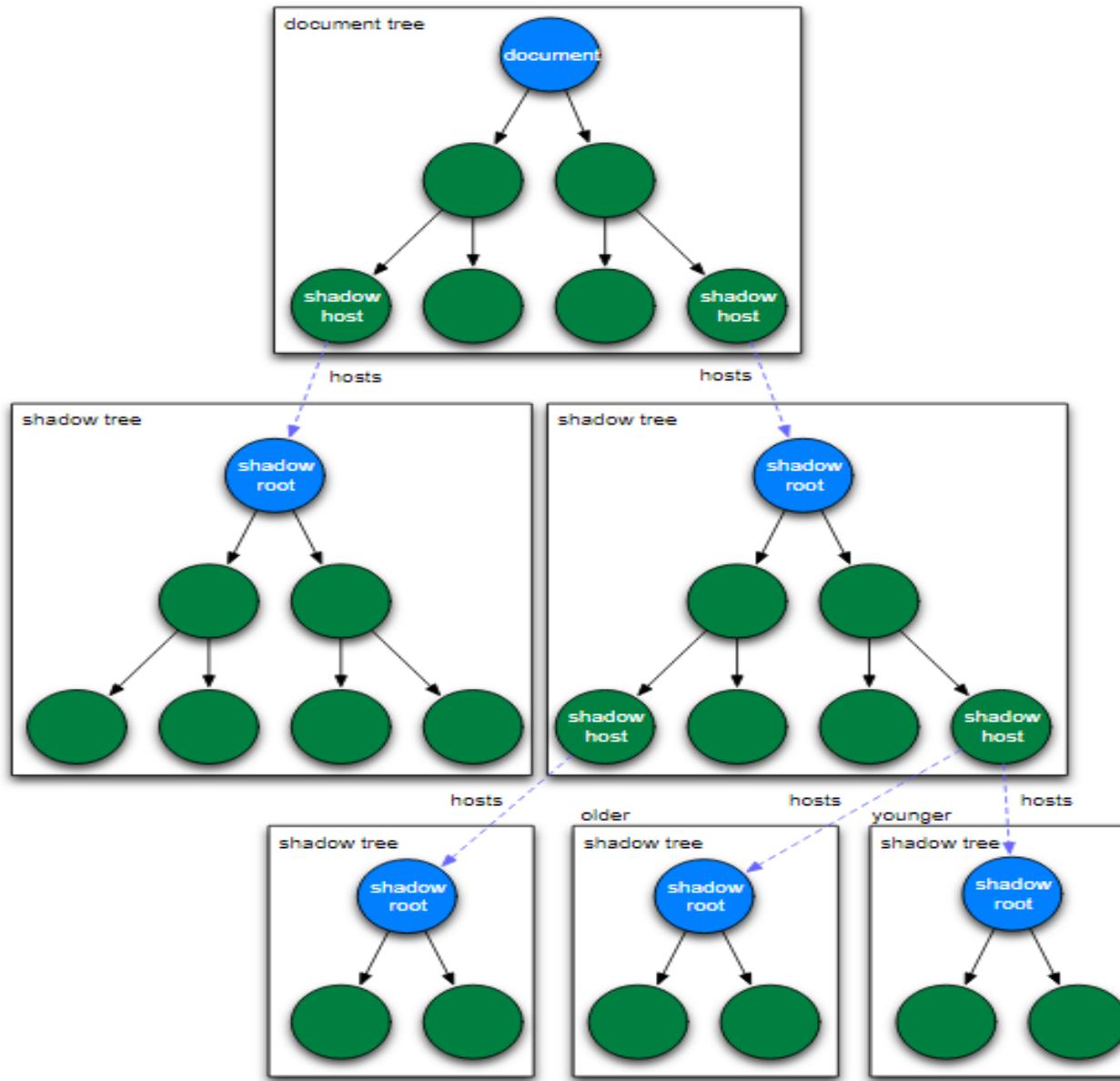
| Angular 2 | Angular 4 |
|---|---|
| The code generated using Angular 2 is bigger, and the file size is also larger. | Angular 4.0 has reduced the bundled file size by 60%. Thus code generated is reduced which helps to accelerate the application performance. |
| Angular two is not backward compatible with Angular JS. | Angular four is backward compatible with Angular 2 for most applications. |
| There is no specific no proper disapproval phases to adjust codes. | There will be proper disapproval phases to allow developers to adjust their code |
| There is no animation feature offers in Angular 2. | Animation features are pulled out of @angular/core and included into their package |

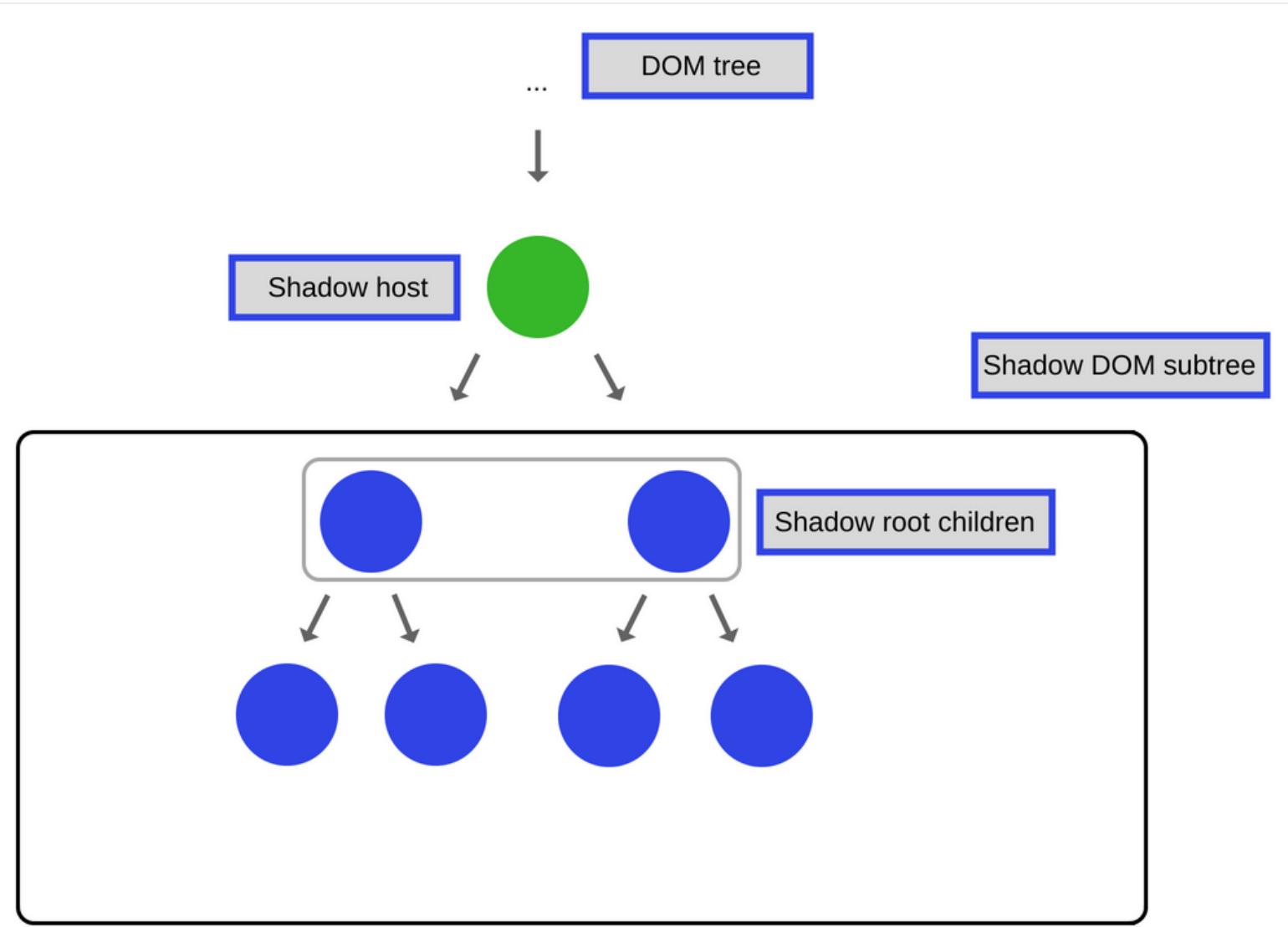


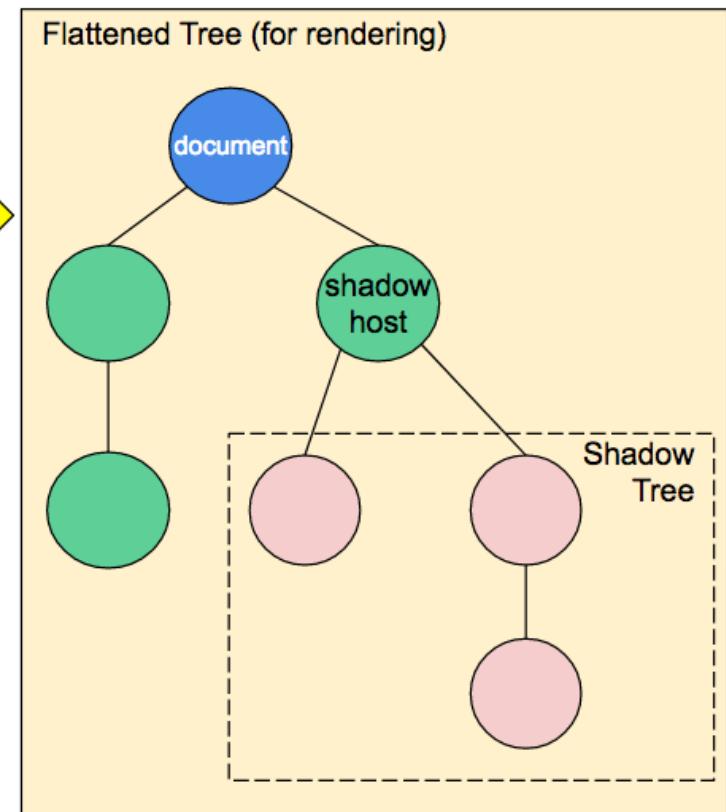
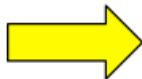
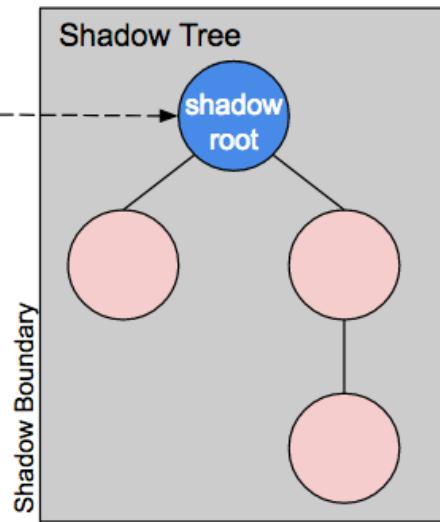
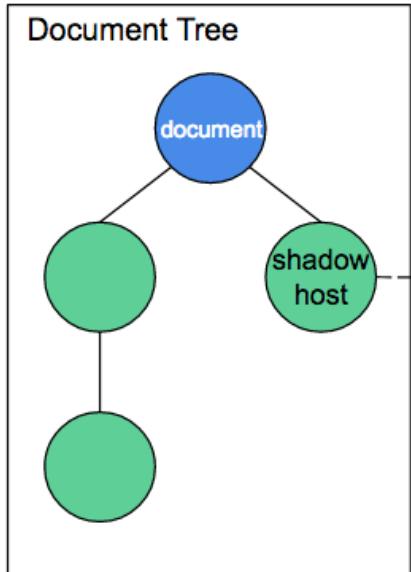
| Angular 4 | Angular 5 |
|---|---|
| Support for Router ParamMap | New Router Lifecycle Event |
| Dynamic Components with NgComponentOutlet | Compiler Improvements |
| TypeScript 2.4 with this version that functions as a JavaScript superset that can be used for optional static typing, interfaces, and classes | Angular 5 comes with build optimizer which is a part of the platform's command like a tool. |
| HTTP Request Simplified | Optimization with HttpClient Feature |
| Includes Animation Package | Internationalized Date & Currency |

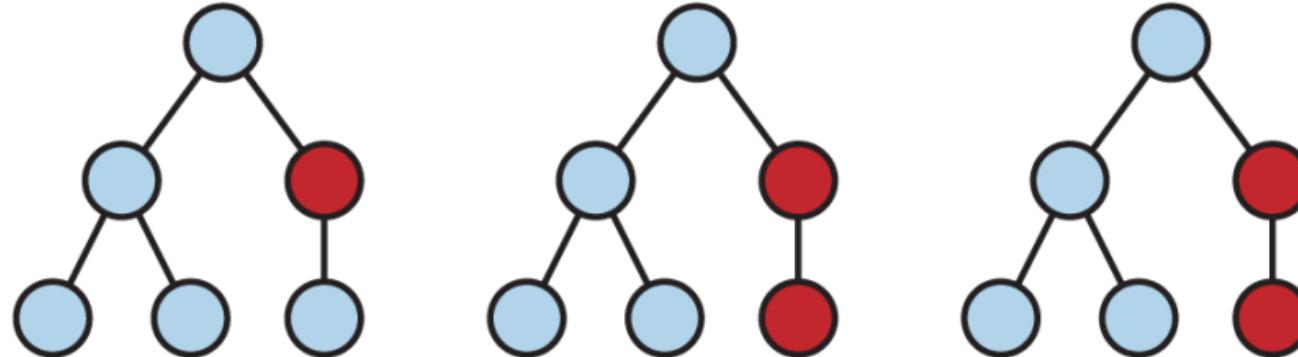


| Technology | React | AngularJS | Angular 2 |
|------------------|--------------------------|------------------------------|------------------------------|
| Author | Facebook community | Google | Google |
| Type | Open source JS library | Fully-featured MVC framework | Fully-featured MVC framework |
| Toolchain | High | Low | High |
| Language | JSX | JavaScript, HTML | TypeScript |
| Learning Curve | Low | High | Medium |
| Packaging | Strong | Weak | Medium |
| Rendering | Server Side | Client Side | Server Side |
| App Architecture | None, combined with Flux | MVC | Component-based |
| Data Binding | Uni-directional | Bi-directional | Bi-directional |
| DOM | Virtual DOM | Regular DOM | Regular DOM |
| Latest Version | 15.4.0 (November 2016) | 1.6.0 | 2.2.0 (November 2016) |

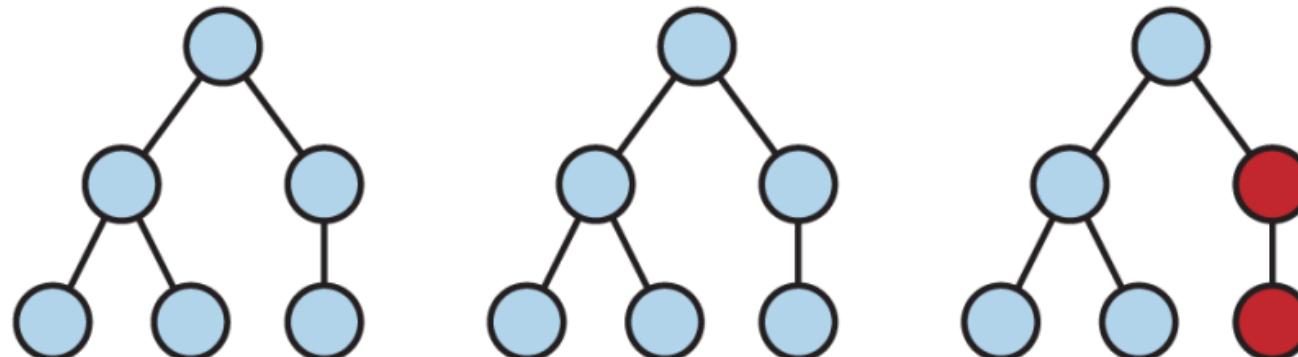






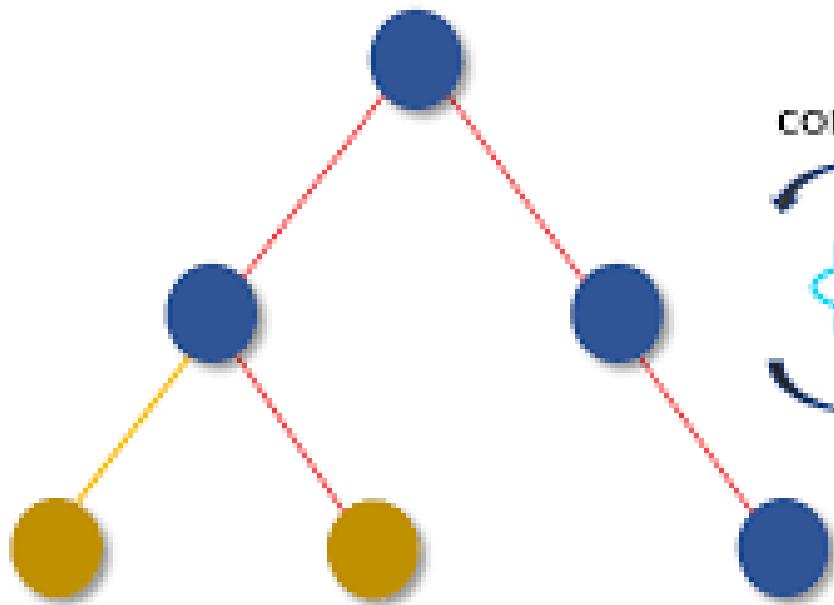


State Change → Compute Diff → Re-render

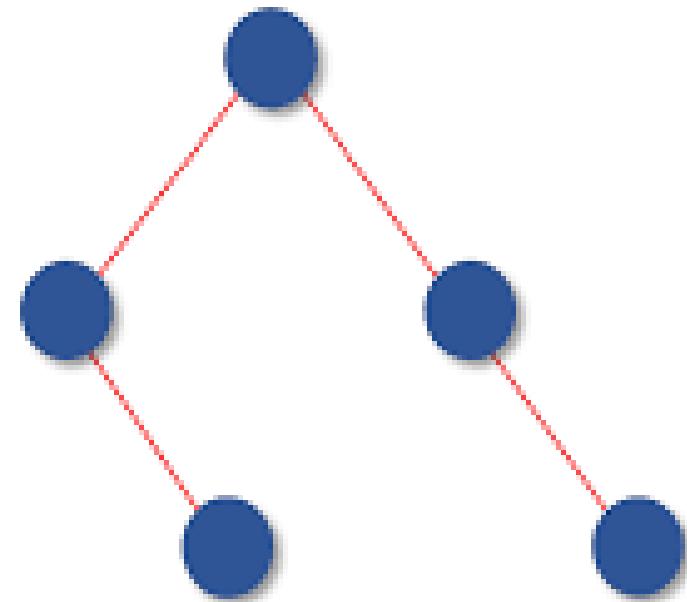




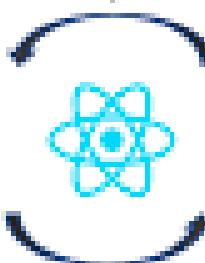
Virtual DOM



Real DOM

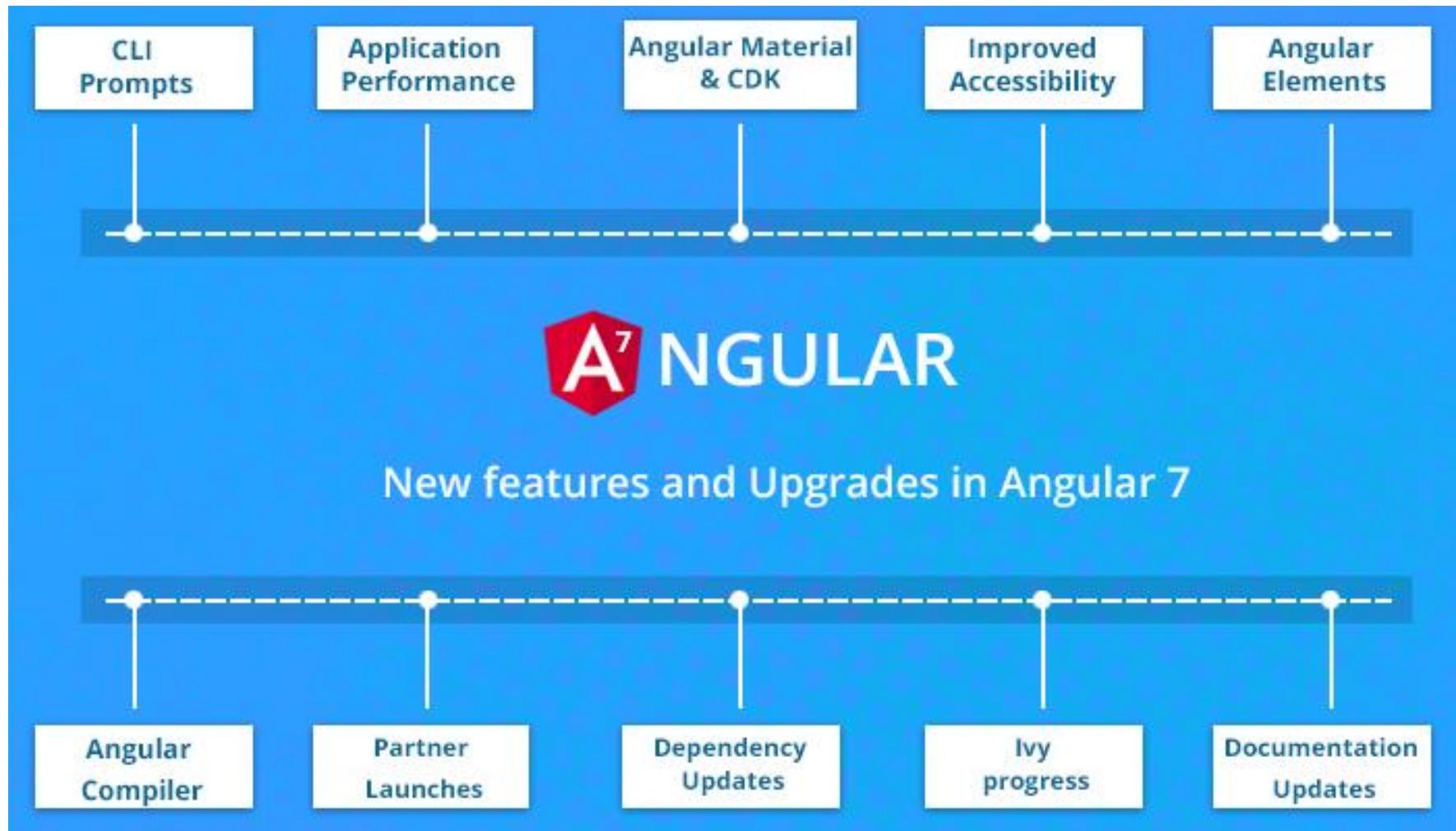


compare





Angular 7





Angular 8

The Top Features of Angular 8





Angular 8



Differential Loading



Ivy Rendering Engine



Bazel Support



Builders API

Top Features of Angular 8



Support for \$location



Web Workers



Router Backwards
Compatibility



Opt-In Usage Sharing



Angular 8

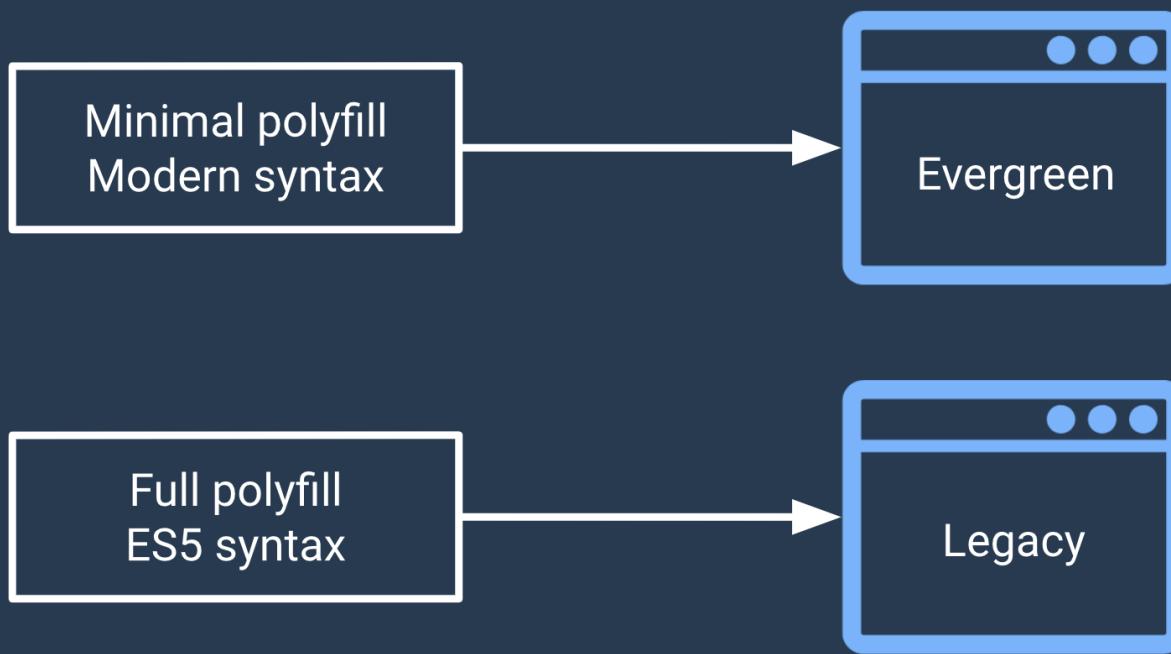
- Angular has a wide ecosystem including:
- High-quality and large community
- Deep Integration
- Fast-growing tools like Bazel
- State management libraries of different flavors
- Code Quality toolings, such as Codelyzer
- UI libraries, Components, Directives, Pipes, etc.
- Large IDE plugins repositories
- Testing Frameworks and utility libraries



- **Differential Loading**
- In spite of angular being a complete framework with its own package for a network request, form validation, and many more, Angular has one disadvantage-the app/bundle size.
- The challenge that we were facing with Angular 7 was the large bundle size due to the conversion of modern JS/TS code in JS. After the conversion, the final application bundle is created for all the browsers (new and old) concerning the app performance on all the browsers.
- This challenge of large bundle size was overcome in Angular 8 by reducing the bundle size by the concept of differential loading.
- When we build apps with ng build, two separate bundles are created for the production dedicated to the older and newer browsers respectively. The correct bundle gets loaded automatically by the browser, thus improving the performance for Angular 8 by loading less code by the newer browsers.



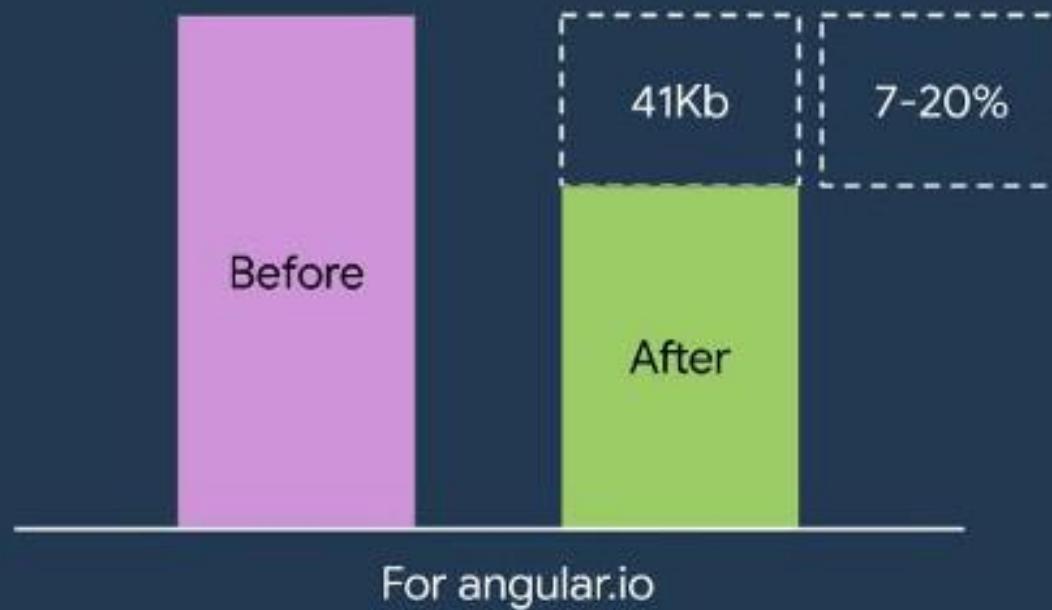
Smaller apps with Differential Loading





A

Differential Loading Savings





Angular 8

- Ivy Renderer
- According to a source, 63% of all US traffic comes from smartphones and tablets. It is further forecasted that the number will increase to 80% by the end of this year.
- One of the biggest challenges for a front end developer is increasing the loading speed of the website. Unfortunately, mobile devices always stay behind in this race either due to slow or bad internet connectivity thus making it more challenging for the developers.
- But we never run out of solutions. We can use CDN, PWA, and others for loading the application faster. But if you want to have some out of the box solution, then reducing the bundle size is the ultimate solution and thus, IVY comes into the picture.
- IVY is meant to build a next-generation rendering pipeline for Angular 8.0
- Ivy is an angular renderer that uses incremental DOM. Ivy modifies the working of the framework without any changes to the Angular applications. On completion of IVY, the angular applications become small, simple, and faster.

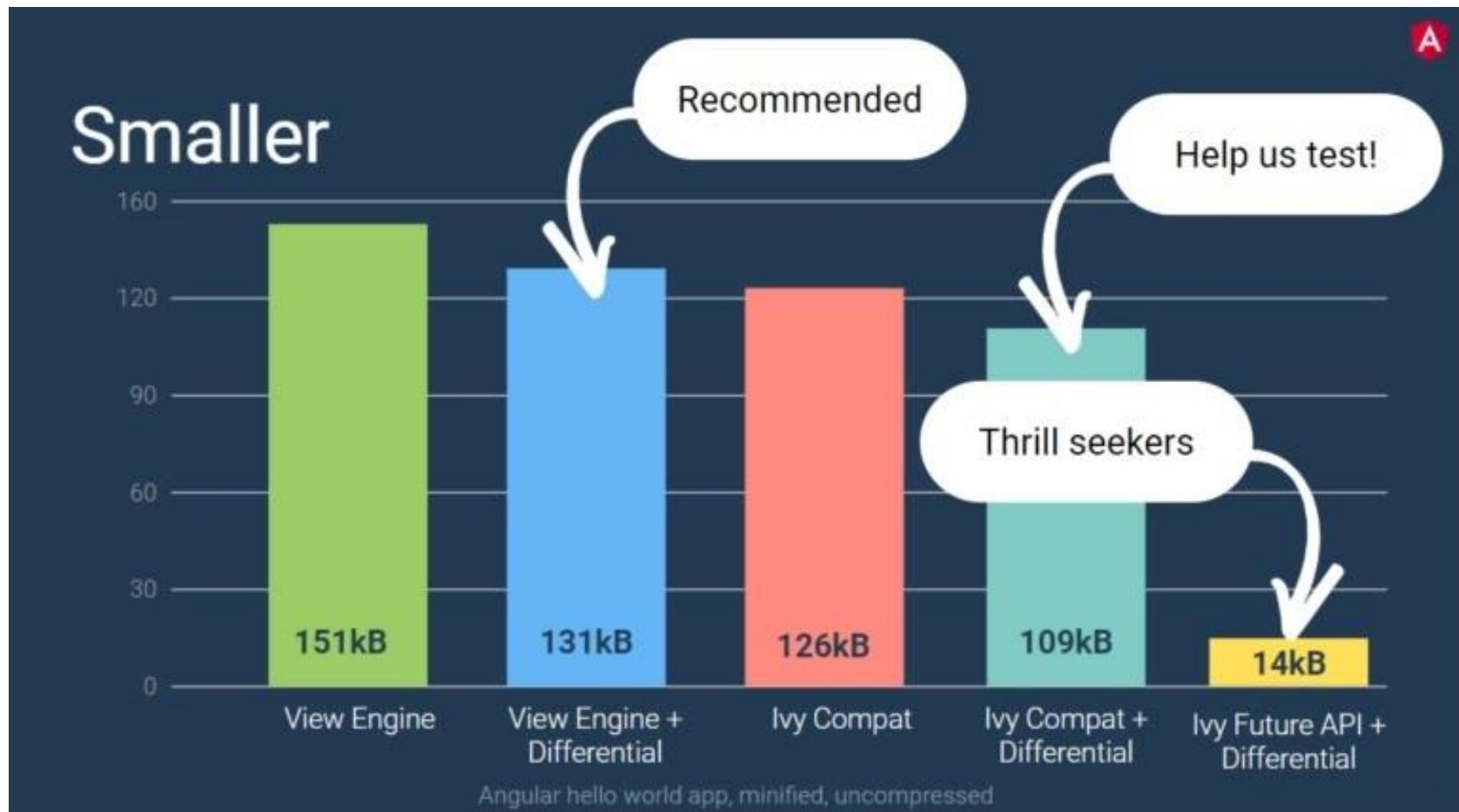


Angular 8

- IVY consists of two main concepts:
- 1.Tree Shakable 2. **Low Memory Footprint**:
- Tree shakable: To focus only on the code in use, the unused code is removed. This results in faster runtime and smaller bundles.
- Local: For a faster compilation, the changing components are recompiled.
- The benefits of Ivy are:
 - Bundles are smaller
 - Templates are debuggable
 - Tests are faster
 - Builds are faster
 - Lots of bugs fixed



Angular 8



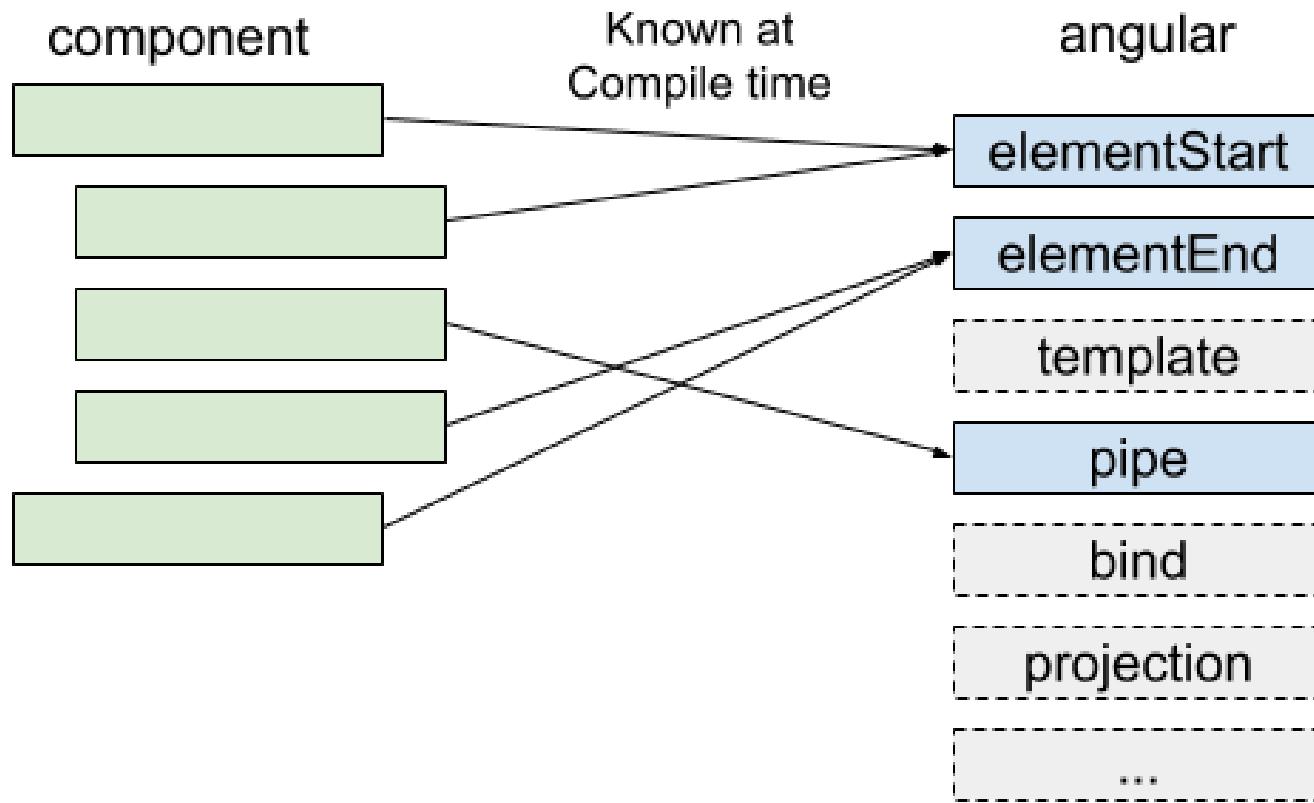


Angular 8

- It rewrites the Angular compiler and runtime code to reach:
- Better compatibility with tree-shaking
- Improved build times
- Improvised build sizes
- Loaded with features like lazy loading of components rather than modules.



Angular 8



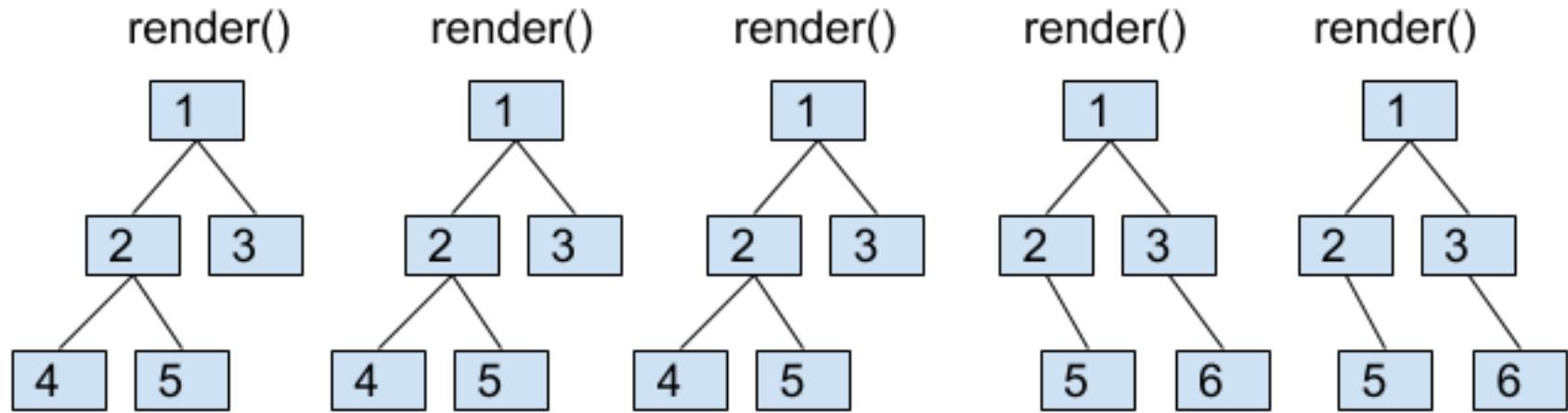


Angular 8

- **Low Memory Footprint:**
- Incremental DOM doesn't need any memory to rerender the view if it doesn't change the DOM so it allocates the memory when the DOM nodes are added or removed.
- since most of render/template calls don't change anything result in huge memory savings.

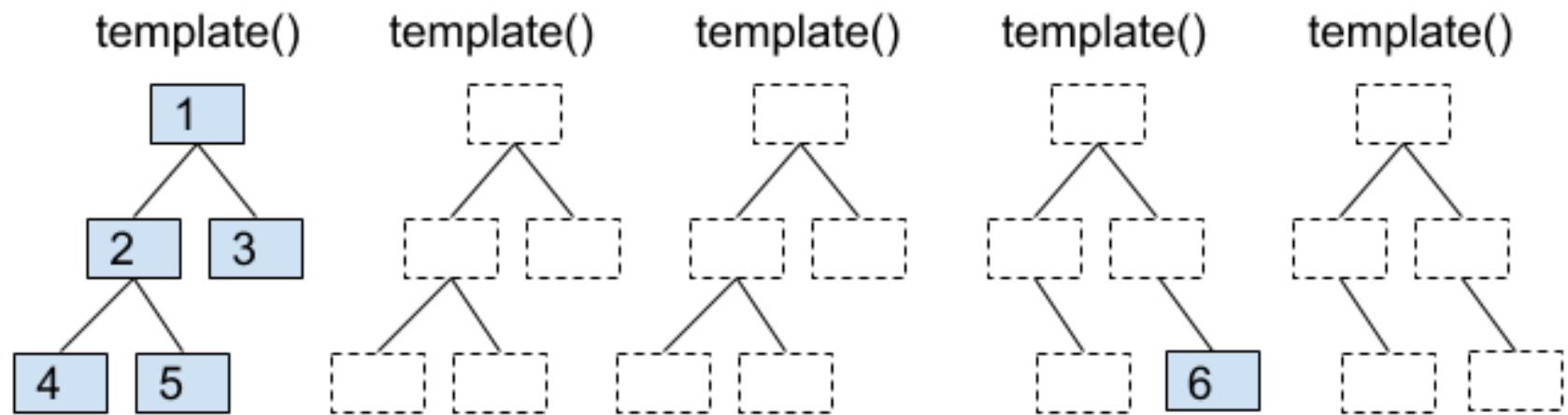


Virtual DOM





Why Incremental DOM Has Low Memory Footprint?





Angular 8

- If we want to include the Ivy for a new application, we can create new projects with enable-ivy options
- `ng new ivy-project --enable-ivy`
- This command intimate Angular CLI store the following configuration in the `tsconfig.json` file.
- `"angularCompilerOptions": {`
- `"enableIvy": true`
- `}`
- The above configuration can manually add in any existing angular application after the upgrade that application into Angular 8.
- One recommendation that if we want to use Ivy for our application, then run the application in debug mode with AOT compilation mode.
- `ng serve --aot`



Angular 8

- Web Workers
- With the newest release of Angular v8, web workers can now be easily integrated with Angular.
- “Web workers is an asynchronous system, or protocol, for web pages to execute tasks in the background, independently from the main thread and website UI.”
- It is an isolated environment that is insulated from the window object, the document object, direct internet access and is best suited for long-running or demanding computational tasks.”
- Have you built an application that includes a lot of calculations on UI? Are you experiencing the UI to be slow?
- Having heavy calculations, data table manipulations, and other complex computations results in a laggy UI.
- JavaScript running on the main thread is not the only thing. Other things like calculations also run on it thus resulting in a bad user experience. Thus, web workers come into the picture to resolve this issue.



Angular 8

- Therefore you can say if your application is unresponsive while processing data, web workers are helpful.
- Due to JavaScript being single-threaded, there is a possibility of asynchronous data calls to take place.
- Facilitating to run the CPU intensive computations in the background thread, Web workers are used.
- This is achieved by freeing the main thread and updating the user interface.
- Put simply, web workers are useful if heavy computations are offloaded to another thread.



Angular 8

- Lazy Loading
- Lazy loading helps in bringing down the size of large files. The required files are lazily loaded.
- Previously in the older versions of Angular, `@loadChildren` property was used by the route configuration.
- This property accepts a string. If any typo occurred or any module name has been recorded wrong, Angular doesn't consider this as wrong.
- It accepts the value that was there until we try building it.
- To overcome this, dynamic imports in router configuration is added in Angular 8 thus enabling the usage of import statement for lazy loading the module.
- Thus, errors will be easily recognized and we can put a stop on waiting till the build time to recognize the errors in the code.

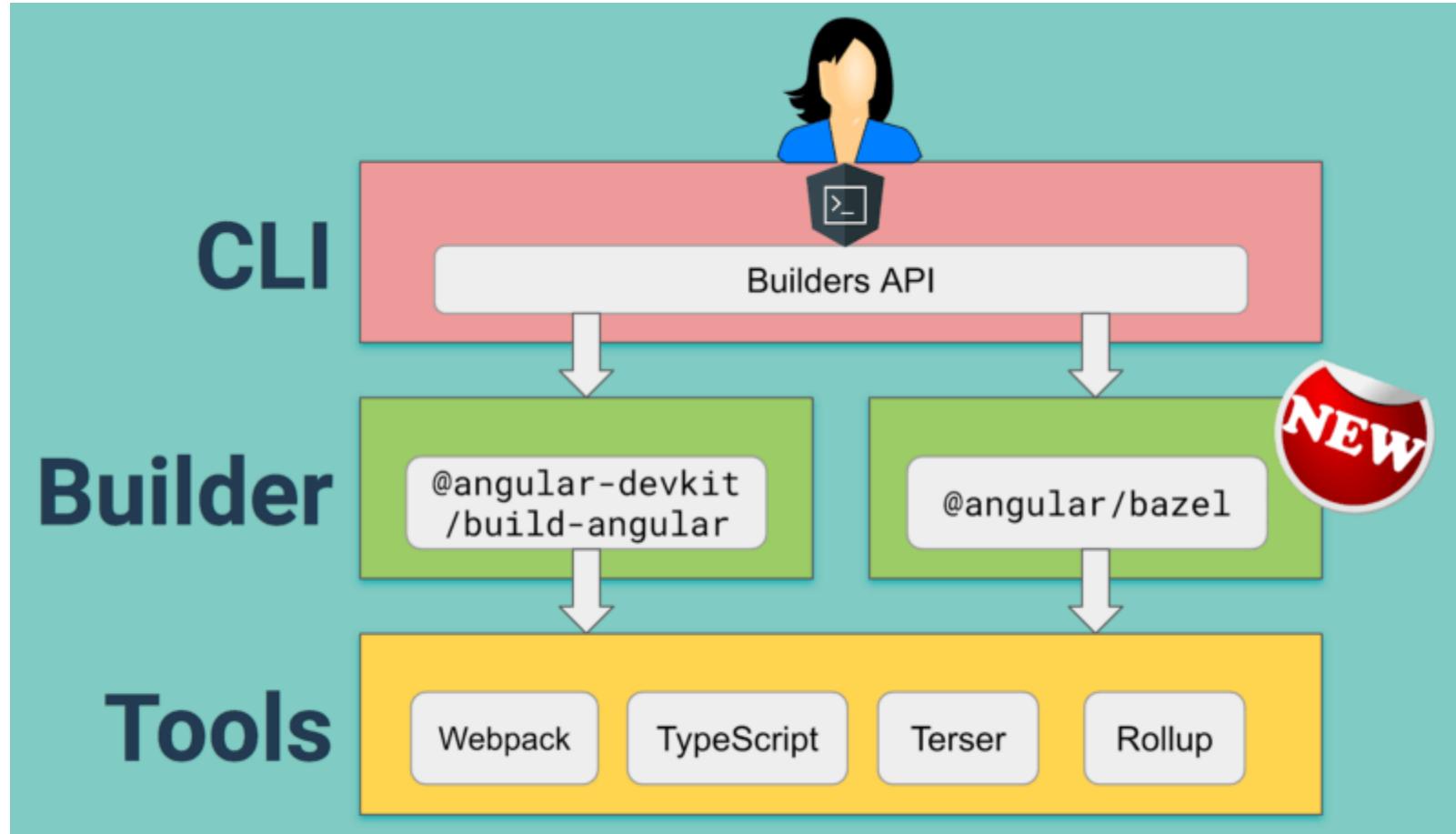


Angular 8

- Bazel Support
- Now increase the possibilities to build your CLI application with Bazel.
- The Angular framework is built with Bazel. Since it is expected to be included in version 9, it is available as opt-in in Angular 8.
- The main advantages of Bazel are:
- Using the same tool in building backends and frontends.
- The build time is faster
- Incremental build for the modified part.
- Ejection of hidden Bazel files that are hidden by default.
- Cache on the build farm
- Dynamic imports for the lazy loaded modules.



Angular 8





Bazel

- To opt-in an existing application, run
- `ng add @angular/bazel`
- To use Bazel in a new application, first install `@angular/bazel` globally
- `npm install -g @angular/bazel`
- then create the new application with
- `ng new --collection=@angular/bazel`
- `ng build --leaveBazelFilesOnDisk`



Angular 8

- CLI Improvements
- With continuous improvement in the Angular CLI, the ng build, ng test and ng run has accorded themselves by 3rd-party libraries and tools.
- For example, with the help of a deploy command, the new capabilities are already in use by AngularFire.
- Angular CLI is equipped with commands such as ng new, serve, test, build and add for quicker development experience.
- With the new ng deploy in the house, developers can deploy their final app to the cloud. Just a few clicks from their command-line interface and the work is done.
- One important thing to remember, add a builder as it accords your project's deployment capability to a specific hosting provider. But, this has to be done before using the command.
- With Angular 8, it has added new features to ngUpgrade. This new feature will make the life of developers easier for upgrading their Angular.js apps to Angular 8.



Angular 8

- TypeScript 3.4
- The most important part is TypeScript 3.4 as it is required to run your Angular 8 project.
- A new flag is introduced in TypeScript 3.4 known as – incremental.
- From the last compilation, the TypeScript is asked to save the information of the project graph by the incremental.
- Every time –incremental invokes TypeScript, it will utilize the information for detecting the cheapest way of type-checking and emit changes to your project.



Angular 8

- Dart-sass for Sass file
- To create the Sass file, Angular has finally discarded node-sass for dart-sass. Dart is perceived to be superfast by many experts and it's now all set to replace Ruby – the legend for the reference implementation
- The generated Sass file would usually remain unchanged; however, there are possibilities of compiler becoming a bit strict. Moreover, with the fibers in place the speed could double too.
- Dart is the default now, which is a big change in itself. However, you can still use node-sass given that you install it explicitly.



Angular production Deployment

```
Maven Repository: org.projectlombok | React App | localhost:6070/getuserbyname/u | Angular - Deployment | Insuranceapp + - _ X
← lite-server
}

F:\daimlerang2019\insuranceapp>lite-server --baseDir="dist/insuranceapp" --port 7070
Did not detect a `bs-config.json` or `bs-config.js` override file. Using lite-server defaults...
** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: 'dist/insuranceapp',
    middleware: [ [Function], [Function] ]
  }
}
[Browsersync] Access URLs:
-----
      Local: http://localhost:3002
      External: http://172.29.53.177:3002
-----
        UI: http://localhost:3003
UI External: http://localhost:3003
-----
[Browsersync] Serving files from: dist/insuranceapp
[Browsersync] Watching files...
19.11.11 21:06:38 200 GET /index.html
19.11.11 21:06:48 200 GET /runtime-es2015.js
19.11.11 21:06:50 200 GET /main-es2015.js
Type here to search  ↪  ⌂  P  F  📁  📱  🚙  WS  ⚡  ENG  21:07  11/11/2019
```



Angular production Deployment – Lazy Loading

```
typescript - Angular fails routing x | Router navigation with lazy loadi x A Loanapp x +  
Administrator: Node.js command prompt - serve -s dist/loanapp  
Terminate batch job (Y/N)? y  
  
F:\boaangular2019\loanapp>ng build --prod --aot=false --build-optimizer=false  
  
chunk {0} runtime.29aec890e643299bc23d.js (runtime) 1.46 kB [entry] [rendered]  
chunk {1} main.912971f0dd802e03f334.js (main) 1.49 MB [initial] [rendered]  
chunk {2} polyfills.8714ab24c2d3b182390c.js (polyfills) 118 kB [initial] [rendered]  
chunk {3} polyfills-es5.c737fda53a8badc0ce78.js (polyfills-es5) 176 kB [initial] [rendered]  
chunk {4} styles.fb20035854434f902908.css (styles) 225 kB [initial] [rendered]  
Date: 2019-12-06T04:12:42.375Z - Hash: 1bb21304c0e8745b46c3 - Time: 34422ms  
  
WARNING in budgets, maximum exceeded for initial. Budget 2 MB was exceeded by 1.09 kB.  
  
F:\boaangular2019\loanapp>serve -s dist/loanapp  
  
Serving!  
- Local: http://localhost:5000  
- On Your Network: http://172.29.53.177:5000  
  
Copied local address to clipboard!
```



Angular production Deployment

```
G angular production deploy ng se x | A Insuranceapp x Insuranceapp x +  
Administrator: Node.js command prompt - serve -s dist/insuranceapp  
Your environment has been set up for using Node.js 12.13.0 (x64) and npm.  
C:\Windows\System32>cd F:\daimlerang2019\insuranceapp  
C:\Windows\System32>f:  
F:\daimlerang2019\insuranceapp>serve -s dist/insuranceapp
```

Serving!

- Local: http://localhost:5000
- On Your Network: http://172.29.53.177:5000

Copied local address to clipboard!

```
Windows Type here to search 91 Microsoft Edge WS 09:56 ENG 02/12/2019 [33]
```



```
F:\aspireangular2020\mvpapp>npm install -g firebase-tools
C:\Users\Balasubramaniam\AppData\Roaming\npm\firebase -> C:\Users\Balasubramaniam\AppData\Roaming\npm\node_modules\fire
base-tools\lib\bin\firebase.js

> protobufjs@6.8.8 postinstall C:\Users\Balasubramaniam\AppData\Roaming\npm\node_modules\firebase-tools\node_modules\p
rotobufjs
> node scripts/postinstall

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\firebase-tools\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"
any"} (current: {"os":"win32","arch":"x64"})

+ firebase-tools@7.12.1
added 515 packages from 321 contributors in 64.928s

F:\aspireangular2020\mvpapp>firebase login
i  Firebase optionally collects CLI usage and error reporting information to help improve our products. Data is colle
cted in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to identify you
.

? Allow Firebase to collect CLI usage and error reporting information? Yes
i  To change your data collection preference at any time, run `firebase logout` and log in again.

Visit this URL on this device to log in:
```



```
F:\aspireangular2020\mvpapp>firebase login
i  Firebase optionally collects CLI usage and error reporting information to help improve our products. Data is collected in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to identify you
.

? Allow Firebase to collect CLI usage and error reporting information? No

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client\_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response\_type=code&state=178659043&redirect\_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...
+  Success! Logged in as parameswaribala@gmail.com

F:\aspireangular2020\mvpapp>
```



Angular Universal

- A normal Angular application executes within the browser, rendering pages within the DOM in response to user actions.
- Angular Universal executes on the server, generating static application pages that later get bootstrapped on the consumer.
- This implies that the appliance usually renders additional quickly, giving users an opportunity to look at the appliance layout before it becomes totally interactive.



Walmart.com

SSR

Home Page



CSR



Category Page



Search Page





Administrator: Node.js command prompt

Terminate batch job (Y/N)? y

```
F:\daimlerang2019\insuranceuniversalapp>ng add @nguniversal/express-engine --clientProject insuranceapp
Skipping installation: Package already installed
CREATE src/main.server.ts (361 bytes)
CREATE src/app/app.server.module.ts (427 bytes)
CREATE tsconfig.server.json (273 bytes)
CREATE webpack.server.config.js (1466 bytes)
CREATE server.ts (1980 bytes)
UPDATE package.json (2064 bytes)
UPDATE angular.json (4713 bytes)
UPDATE src/main.ts (451 bytes)
UPDATE src/app/app.module.ts (3325 bytes)
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/animations@8.2.14 but none is installed. You must
install peer dependencies yourself.
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/common@8.2.14 but none is installed. You must ins
tall peer dependencies yourself.
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/compiler@8.2.14 but none is installed. You must i
nstall peer dependencies yourself.
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/core@8.2.14 but none is installed. You must insta
ll peer dependencies yourself.
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/platform-browser@8.2.14 but none is installed. Yo
u must install peer dependencies yourself.
npm WARN @angular/platform-server@8.2.14 requires a peer of @angular/platform-browser-dynamic@8.2.14 but none is insta
lled. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\webpack-dev-server\node_modules\fsevents)
:
```



Type here to search



00:09
03/12/2019



Administrator: Node.js command prompt

npm ERR! C:\Users\Balasubramaniam\AppData\Roaming\npm-cache_logs\2019-12-02T18_50_27_706Z-debug.log

F:\daimlerang2019\insuranceuniversalapp>npm run build:ssr

```
> insuranceuniversalapp@0.0.0 build:ssr F:\daimlerang2019\insuranceuniversalapp
> npm run build:client-and-server-bundles && npm run compile:server
```

```
> insuranceuniversalapp@0.0.0 build:client-and-server-bundles F:\daimlerang2019\insuranceuniversalapp
> ng build --prod && ng run insuranceapp:server:production --bundleDependencies all
```

Generating ES5 bundles for differential loading...

ES5 bundle generation complete.

```
chunk {2} polyfills-es2015.2987770fde9daa1d8a2e.js (polyfills) 36.4 kB [initial] [rendered]
chunk {3} polyfills-es5.6696c533341b95a3d617.js (polyfills-es5) 122 kB [initial] [rendered]
chunk {0} runtime-es2015.edb2fcf2778e7bf1d426.js (runtime) 1.45 kB [entry] [rendered]
chunk {0} runtime-es5.edb2fcf2778e7bf1d426.js (runtime) 1.45 kB [entry] [rendered]
chunk {1} main-es2015.8c0c5c281b25bdbd5035.js (main) 1.47 MB [initial] [rendered]
chunk {1} main-es5.8c0c5c281b25bdbd5035.js (main) 1.57 MB [initial] [rendered]
chunk {4} styles.f8bfe2e61b3f4fbc4a5c.css (styles) 226 kB [initial] [rendered]
Date: 2019-12-02T18:51:55.010Z - Hash: c8bc909a93cd5ce795f8 - Time: 27327ms
```

WARNING in budgets, maximum exceeded for initial. Budget 2 MB was exceeded by 550 kB.

Hash: ae0c42b588269b42af2

Time: 14770ms

Built at: 12/03/2019 12:22:12 AM



00:22
03/12/2019 ENG 20



Administrator: Node.js command prompt - node dist/server.js

F:\daimlerang2019\insuranceuniversalapp>node dist/server.js

Node Express server listening on http://localhost:4000

Angular is running in the development mode. Call enableProdMode() to enable the production mode.





OAuth2





Okta Account

- rpsconsultingeswaribala1995.okta.com
 - Okta organization name: rpsconsulting-org-374070
Okta
homepage: <https://rpsconsultingeswaribala1995.okta.com>
- Okta username: parameswari.bala@rpsconsulting.in
Temporary password: G99pOsNM Sign-in
here: <https://rpsconsultingeswaribala1995.okta.com>
This password can only be used once within 7 days.



OKTA ODIC CONNECTION

```
F:\boaangular2019\loanapp>cd..
```

```
F:\boaangular2019>cd loanappoauth
```

```
F:\boaangular2019\loanappoauth>npm i angular-oauth2-oidc@5.0.2
npm WARN axobject-query@2.1.1 requires a peer of eslint@^5 || ^6 but none is installed. You must install peer dependencies yourself.
npm WARN angular-oauth2-oidc@5.0.2 requires a peer of @angular/common@>=6.0.0 < 8.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN angular-oauth2-oidc@5.0.2 requires a peer of @angular/core@>=6.0.0 < 8.0.0 but none is installed. You must install peer dependencies yourself.
npm [WARN optional] SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\webpack-dev-server\node_modules\fsevents)
:
npm [WARN notsup] SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm [WARN optional] SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\watchpack\node_modules\fsevents):
npm [WARN notsup] SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm [WARN optional] SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\karma\node_modules\fsevents):
npm [WARN notsup] SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm [WARN optional] SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\@angular\compiler-cli\node_modules\fsevents):
npm [WARN notsup] SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm [WARN optional] SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fsevents):
```



Alternative approach OKTA ODIC CONNECTION

Administrator: Node.js command prompt

```
F:\boaangular2019\loanappoauth>npm i @okta(okta-angular@1.0.7
npm WARN angular-oauth2-oidc@5.0.2 requires a peer of @angular/common@>=6.0.0 < 8.0.0 but none is installed. You must
install peer dependencies yourself.
npm WARN angular-oauth2-oidc@5.0.2 requires a peer of @angular/core@>=6.0.0 < 8.0.0 but none is installed. You must in
stall peer dependencies yourself.
npm WARN axobject-query@2.1.1 requires a peer of eslint@^5 || ^6 but none is installed. You must install peer depen
dencies yourself.
npm WARN @okta(okta-angular@1.0.7 requires a peer of @angular/common@>=4 <7 but none is installed. You must install pe
er dependencies yourself.
npm WARN @okta(okta-angular@1.0.7 requires a peer of @angular/core@>=4 <7 but none is installed. You must install peer
dependencies yourself.
npm WARN @okta(okta-angular@1.0.7 requires a peer of @angular/platform-browser@>=4 <7 but none is installed. You must
install peer dependencies yourself.
npm WARN @okta(okta-angular@1.0.7 requires a peer of @angular/router@>=4 <7 but none is installed. You must install pe
er dependencies yourself.
npm [WARN] optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\webpack-dev-server\node_modules\fsevents)
:
npm [WARN] notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"
any"} (current: {"os":"win32","arch":"x64"})
npm [WARN] optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\watchpack\node_modules\fsevents):
npm [WARN] notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"
any"} (current: {"os":"win32","arch":"x64"})
npm [WARN] optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\karma\node_modules\fsevents):
npm [WARN] notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"
any"} (current: {"os":"win32","arch":"x64"})
```





AOT

- Faster rendering With AOT, the browser downloads a pre-compiled version of the application. The browser loads executable code so it can render the application immediately, without waiting to compile the app first.
- Fewer asynchronous requests The compiler inlines external HTML templates and CSS style sheets within the application JavaScript, eliminating separate ajax requests for those source files.
- Smaller Angular framework download size There's no need to download the Angular compiler if the app is already compiled. The compiler is roughly half of Angular itself, so omitting it dramatically reduces the application payload.
- Detect template errors earlier The AOT compiler detects and reports template binding errors during the build step before users can see them.
- Better security AOT compiles HTML templates and components into JavaScript files long before they are served to the client. With no templates to read and no risky client-side HTML or JavaScript evaluation, there are fewer opportunities for injection attacks.



Choosing Compiler

- Choosing a compiler
- Angular offers two ways to compile your application:
 - Just-in-Time (JIT), which compiles your app in the browser at runtime.
 - Ahead-of-Time (AOT), which compiles your app at build time.



Choosing Compiler

- JIT compilation is the default when you run the ng build (build only) or ng serve (build and serve locally) CLI commands:

- ng build
 - ng serve

For AOT compilation, include the --aot option with the ng build or ng serve command:

- ng build --aot
 - ng serve --aot



PWA

- ng add @angular/pwa --project onlinebanking
- ng build --prod
- http-server -p 8080 -c10 dist/<project-name>
- use Location Strategy
- lite-server --baseDir=".dist/onlinebanking"
- no change required



PWA

```
http-server
F:\boancrfeb2020\onlinebanking>http-server -p 7070 -c10 dist/onlinebanking
Starting up http-server, serving dist/onlinebanking
Available on:
  http://172.29.53.177:7070
  http://172.22.224.1:7070
  http://169.254.131.111:7070
  http://192.168.99.1:7070
  http://192.168.58.37:7070
  http://127.0.0.1:7070
Hit CTRL-C to stop the server
```



Type here to search



19:05
06/02/2020



PWA

Ginger - An IHCL Brand | Book Di Multicasting Observables Using A Onlinebanking

localhost:7070#/Login

Apps Gmail YouTube Maps



Bank of America

Elements Console Sources Network Performance Memory Application Security Audits Augury

Application

- Manifest
- Service Workers**
- Clear storage

Service Workers

Offline Update on reload Bypass for network

<http://localhost:7070/> [Update](#) [Unregister](#)

Source [ngsw-worker.js](#) Received 2/6/2020, 2:29:26 PM

Status ● #42 activated and is running [stop](#)

Clients <http://localhost:7070#/Login> [focus](#)

Push [Push](#)

Sync [Sync](#)

Storage

- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies

Cache

- Cache Storage



PWA

A Onlinebanking × +

localhost:7070/#/Login Incognito

Gmail YouTube Maps

Nexus 5X ▾ 412 × 732 100% ▾ Online ▾

Elements Console Sources Network Performance Memory Application Security Audits

⚠ 23

+ 7:11:55 PM - localhost:7070 ▾

http://localhost:7070/#/Login

100 72 93 82

Performance Accessibility Best Practices SEO Progressive Web App

0-49 50-89 90-100



How AOT Works

- The Angular AOT compiler extracts metadata to interpret the parts of the application that Angular is supposed to manage.
- You can specify the metadata explicitly in decorators such as `@Component()` and `@Input()`, or implicitly in the constructor declarations of the decorated classes.
- The metadata tells Angular how to construct instances of your application classes and interact with them at runtime.



How AOT Works

- The Angular compiler extracts the metadata once and generates a factory for TypicalComponent.
- When it needs to create a TypicalComponent instance, Angular calls the factory, which produces a new visual element, bound to a new instance of the component class with its injected dependency.



How AOT Works

- There are three phases of AOT compilation.
- Phase 1 is code analysis. In this phase, the TypeScript compiler and AOT collector create a representation of the source. The collector does not attempt to interpret the metadata it collects. It represents the metadata as best it can and records errors when it detects a metadata syntax violation.
- Phase 2 is code generation. In this phase, the compiler's StaticReflector interprets the metadata collected in phase 1, performs additional validation of the metadata, and throws an error if it detects a metadata restriction violation.
- Phase 3 is template type checking. In this optional phase, the Angular template compiler uses the TypeScript compiler to validate the binding expressions in templates. You can enable this phase explicitly by setting the `fullTemplateTypeCheck` configuration option;



Jquery in Angular

- npm install --save jquery
- angular.json
- "scripts": [
• "node_modules/jquery/dist/jquery.min.js"
•]
- app.component.html
- <div id="elementId">
- <h1>Welcome</h1>
- </div>



Jquery in Angular

- app.component.ts
- import {Component, OnInit} from '@angular/core';
- declare var \$: any;
- @Component({
- selector: 'app-root',
- templateUrl: './app.component.html',
- styleUrls: ['./app.component.css']
- })
- export class AppComponent implements OnInit {
- ngOnInit(): void {
- \$(document).ready(() => {
- \$('#elementId').css({'background-color': 'yellow', 'font-size': '200%'});
- });
- }
- }



Angular Optimization Techniques

- #1 Angular Command Line Interface (CLI)
- #2 Tree-shaking
- #3 JIT (Just-in-Time) Compilation
- #4 AOT (Ahead-of-Time) Compilation
- #5 Prod Flag
- #6 UglifyJS and Build Optimizer Flag
- #7 Build Optimizer and Vendor-Chunk
- #8 Package.json



Angular Optimization Techniques

- #9 Third-Party Tools (gulp and grunt)
- 10 .htaccess file

What is a Reusable Component?



- Angular was designed to promote code reuse. As a result, the recommended component architecture relies on two different kinds of components:
- Container components (or “smart” components) know how to get data and work with services
- Presentation components (“dumb” or “lazy” components) that have to be fed with data

What is a Reusable Component?



- Container components are tied to the business logic of your application.
- They are not meant to be shared or reused.
- Instead, what they do is get data from the server and pass it down to presentation components.

What is a Reusable Component?



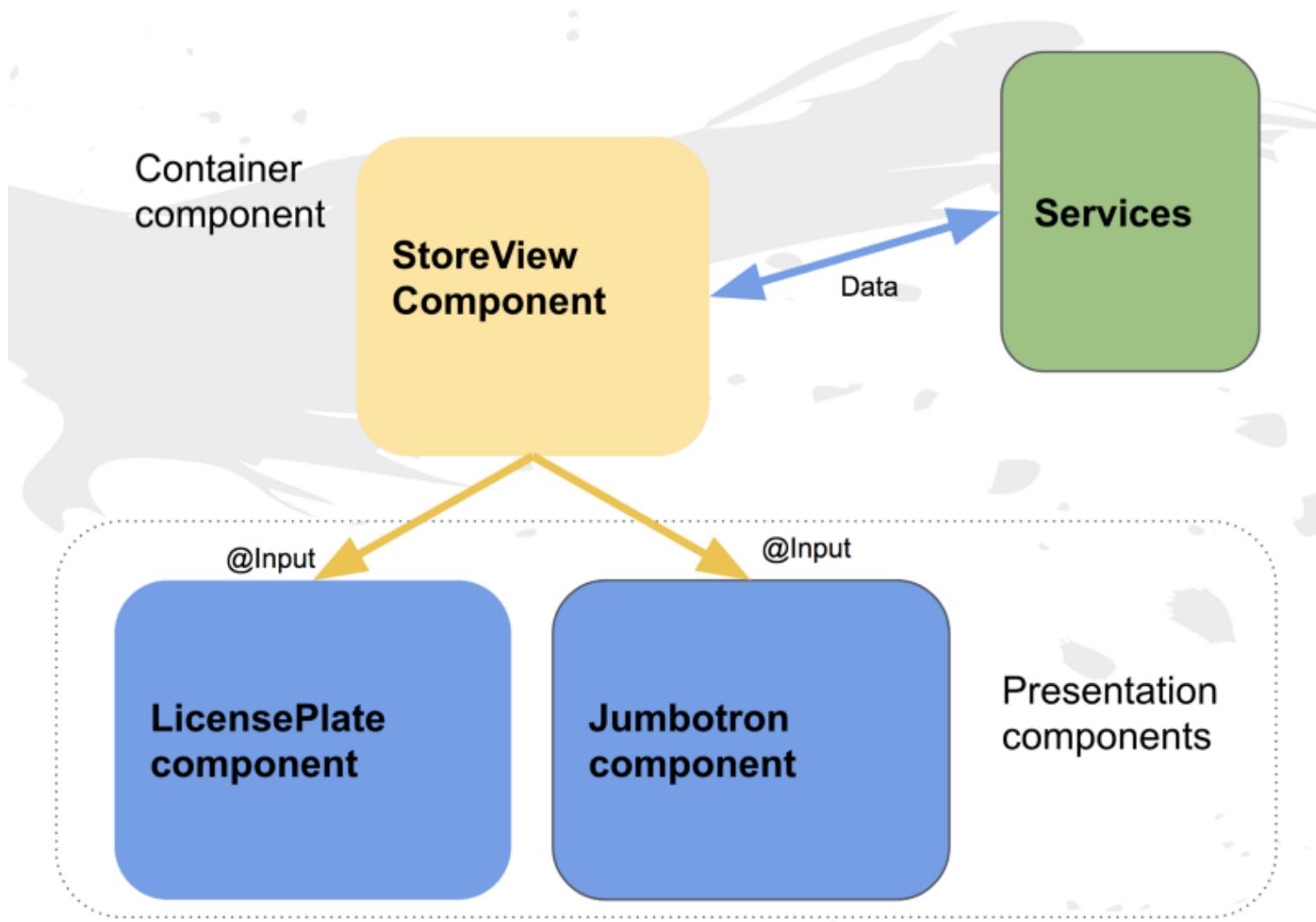
- Presentation components do not know about any service.
- They are pure, simple UI components that just need some input data to render.
- Buttons, tabs, headers, tables are all great candidates to be used as presentation components.
- As a result, such components are reusable because they are coded in a way that is not tied to any runtime context.

What is a Reusable Component?



- Presentation components are really just an HTML template that has to be filled with data.
- They can also emit events that container components would catch in order to make the actual business logic happen.

What is a Reusable Component?





Sharable Components

- ng generate library [name of the library]
- ng generate component [component name] --project [name of library]
- public API of your library is defined in projects/[name of library]/src/public_api.ts.
- Update
- ng build [name of library]
- Npm login (eswaribala70, usual password)
- cd dist/[name of library]
- npm publish
- Ensure that package.json has unique name(otherwise 403 error)



Debugging Memory Leaks in Angular | Loanapp | Successfully published bankinglib | bankinglib - npm

npmjs.com/package/bankinglib

Apps Insert title here Empire New Tab How to use Assertions Browser Automation node.js - How can I... Freelancer-dev-810... Courses New Tab hi Airtel 4G Hotspot nt8F83

bankinglib

0.0.1 • Public • Published an hour ago

Readme Admin 1 Dependency 0 Dependents 1 Versions

Bankinglib

This library was generated with [Angular CLI](#) version 8.2.14.

Code scaffolding

Run `ng generate component component-name --project bankinglib` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module --project bankinglib`.

Note: Don't forget to add `--project bankinglib` or else it will be added to the default project in your `angular.json` file.

install

```
> npm i bankinglib
```

version license
0.0.1 none

last publish
an hour ago

collaborators



This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allow us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, see our [Privacy Policy](#).

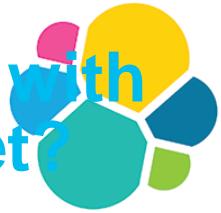
If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

Accept Decline

Type here to search

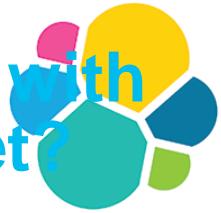
21:17 05/12/2019

What if I want to share my component locally with other applications, not publicly on the internet?



- Angular Multi-Application Project Creation
- A workspace is a set of Angular applications and libraries. The angular.json file at the root level of an Angular workspace provides workspace-wide and project-specific (application or library) configuration defaults for build and development tools.

What if I want to share my component locally with other applications, not publicly on the internet?



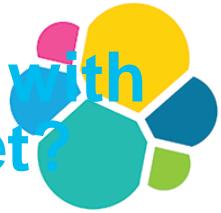
- Create a Workspace
- `ng new angularworkspace --createApplication=false --directory=frontend --interactive=false`
- `cd frontend`
- `ng generate application administration --style=css --routing=true`
- `ng generate application gatling --style=css --routing=true`

What if I want to share my component locally with other applications, not publicly on the internet?



- ng generate library tools
- ng generate library vendors
- Create a Shared Service
 - ng generate service userservice --project=tools
 - Update the AppComponent
projects/administration/src/app/app.component.ts to use this service:
 - import {userservice } from 'projects/tools/src/lib/hello-world.service';

What if I want to share my component locally with other applications, not publicly on the internet?



- Launch The Applications
- `ng serve --project=administration`

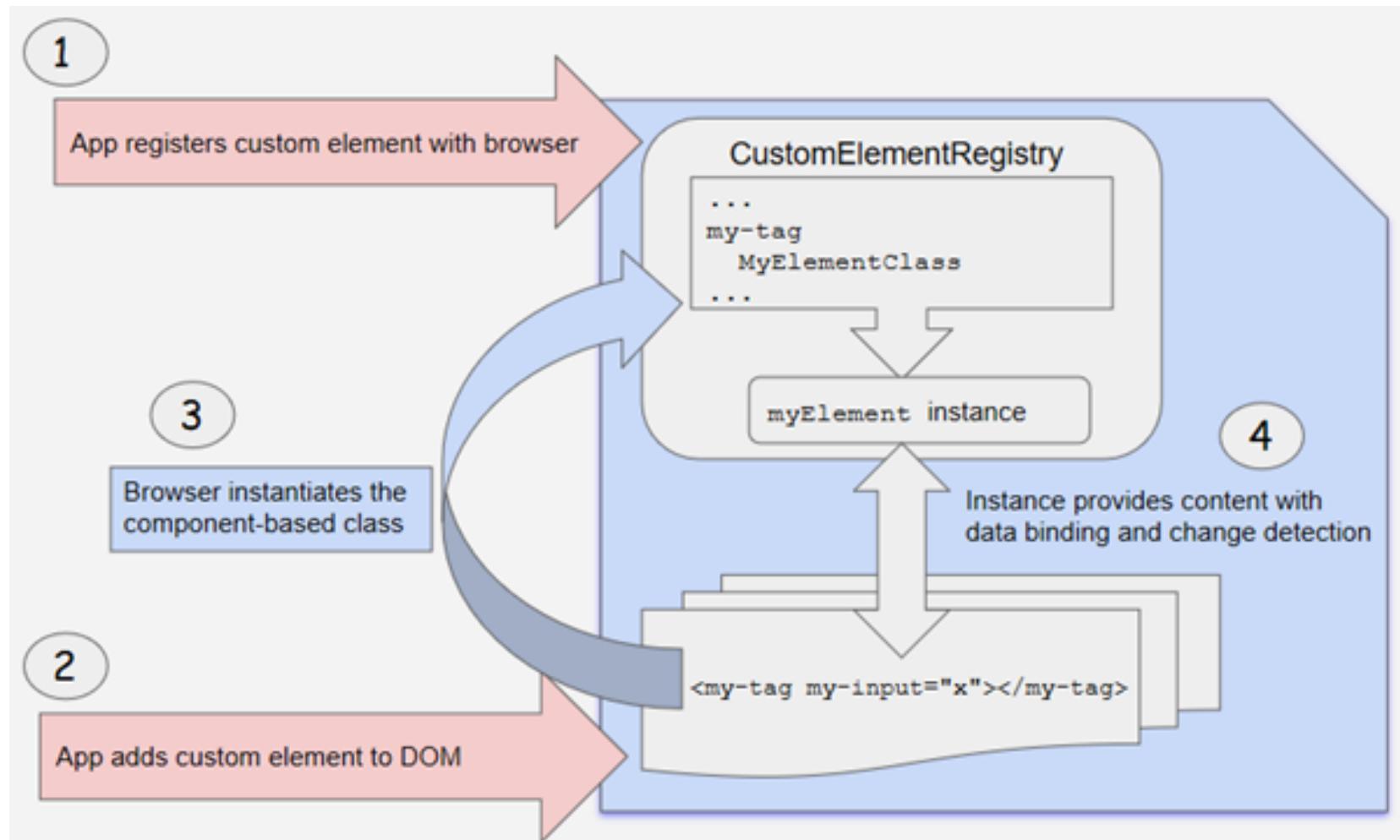


Angular Elements

- Angular Elements allow us to create reusable Angular components, which can be used outside of the Angular application.
- You can use an Angular Element in any other application such as normal HTML, React, etc.
- Essentially, Angular Elements are normal components, which are packaged as Custom Elements.

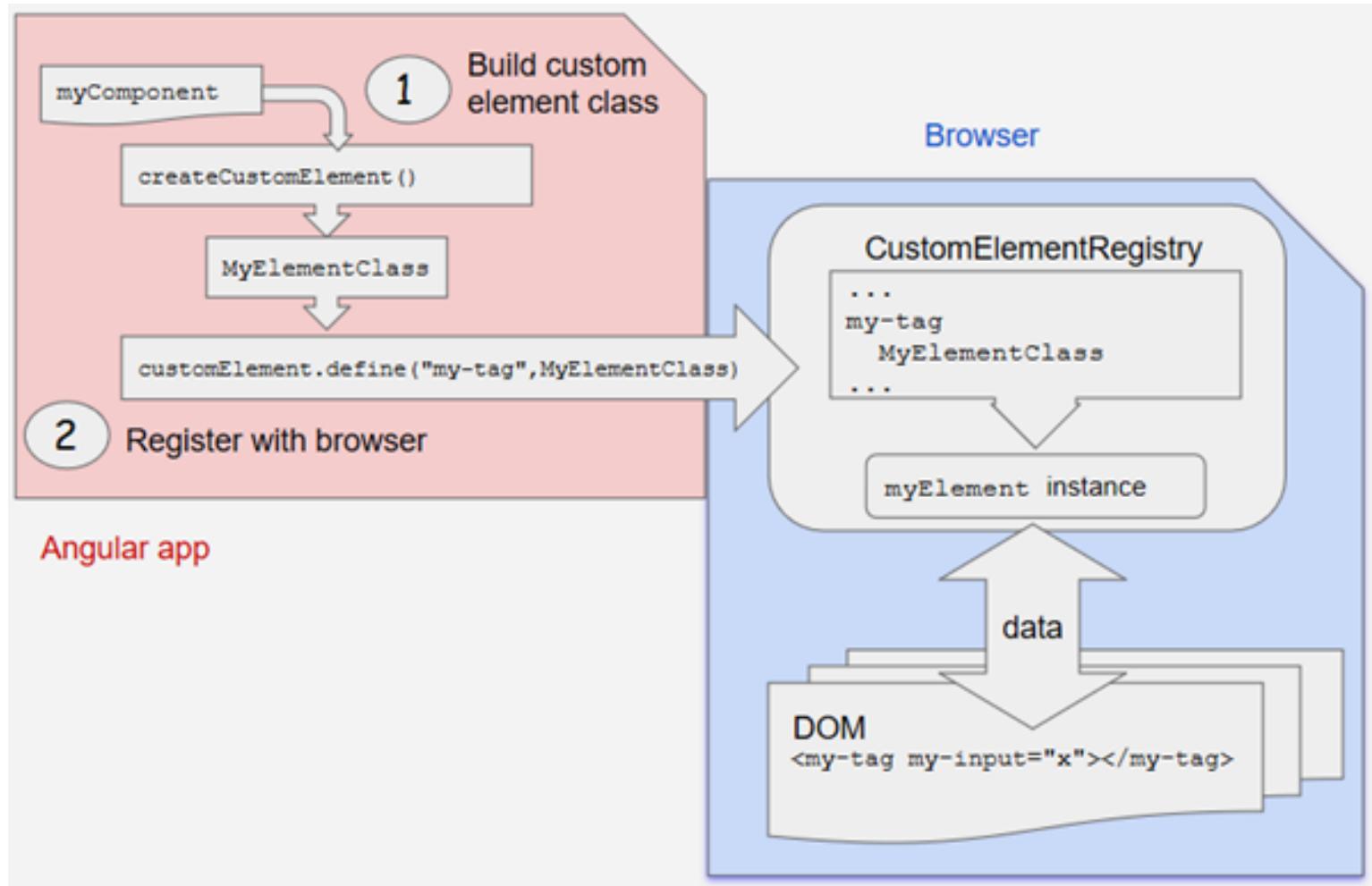


Angular Elements





Transforming Component to Custom Element





Angular Elements

- Step 1
- **npm install @angular/elements**
- To work with older browsers, we need polyfill. So, let us install that also as shown below:
- npm install @webcomponents/custom-elements
- After installing polyfill, open polyfills.ts file and add these two entries:
- import '@webcomponents/custom-elements/src/native-shim';
- import '@webcomponents/custom-elements/custom-elements.min';



Angular Elements

- Step 2:
- Create the Component
- Step 3:
- Register the Component
- Step 4:
- Create Element from the component
- Step 5:
- Use Custom Element
- `ng build --prod --output-hashing none && node build-elements.js`



Angular Interceptors

- Interceptors are the mechanism where we can operate on the outgoing request and the incoming response.
- Interceptors are the layer between our Angular application and the back-end services.
- Whenever the request is made from the application, the interceptor catches the request, transforms it, and passes to the back end.
- The same happens with the response; whenever it receives the response, we can make changes to the response and use them in the application.

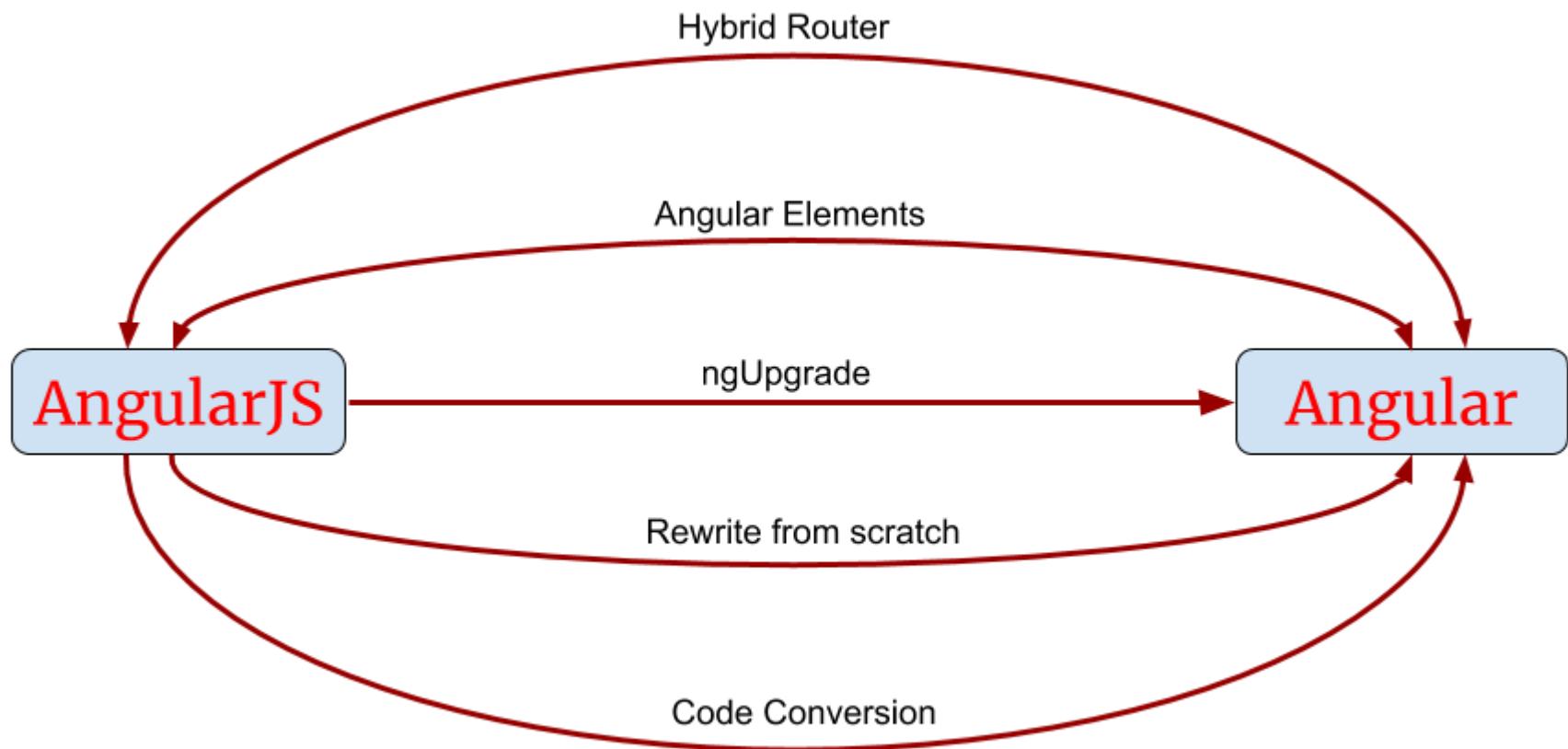


Angular Interceptors

- To set up interceptor, we need a class which will be injectable class and implementing the `HttpInterceptor`.
- When we implement this Interface, then we have a method called `intercept` which has the body like below.
- `intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {}`
- It has two parameters "req" and "next" which are `HttpRequest` and `HttpHandler` respectively.



Migration Tools





Migration Tools

- **ngMigration Assistant**
- **ngMigration Forum**



Migration Tools

- **Installing Angular & ngUpgrade**
- Step 1: Removing Bootstrap from index.html
- Step 2: Changing the AngularJS Module
- Step 3: Creating the Angular App Module
- `angular.module(MODULE_NAME, ['ngRoute'])`
- Export
- Step 4: Bootstrapping in the Angular Module



Migration Tools

- **constructor(private upgrade: UpgradeModule){**
- **}**
- **ngDoBootstrap(){**
- **}**
- **constructor(private upgrade: UpgradeModule) { }**
- **ngDoBootstrap(){**
- **this.upgrade.bootstrap(document.documentElement, [moduleName], {strictDi: true});**
- **}**



Migration Tools

- **Step5: Creating main.ts**
- Step 6: Updating Webpack
- Testing the Application

Debugging



Augury

Screenshot of the Augury browser extension interface showing the Angular application structure and component details for the 'Loanapp' component.

The browser tabs show:

- Debugging Memory Leaks in Anc
- Successfully published bankinglii
- Angular Augury
- Loanapp

The main content area displays the application's landing page with the Bank of America logo, a large "TITLE LOANS UP TO \$2500 NEED CASH?" banner, and an advertisement for "Warrah Home Loans".

The Augury toolbars at the bottom include:

- Elements, Sources, Network, Console, Memory, Performance, Application, Security, Audits, Lucid, **Augury** (selected)
- Component Tree, Router Tree, NgModules
- Angular Version: 8.2.14

The Component Tree sidebar shows the component hierarchy:

- AppComponent
 - MenuComponent
 - Menubar
 - div
 - router-outlet
 - LoginComponent
 - a
 - form
 - MatFormField
 - input
 - label
 - div
 - MatFormField
 - MatButton
 - MatButton

The Augury panel on the right shows the properties and injector graph for the MenuComponent:

- Properties**: menuService: {}, router: Object, _menuArr: Array[4]
- Injector Graph**: (\$\$el in Console)
- Change Detection**: Default
- State**: (empty)



Augury

The screenshot shows the Augury extension integrated into a browser window. The top part displays a landing page for "Bank of America" advertising "TITLE LOANS UP TO \$2500 NEED CASH?". The bottom part is the Augury interface, which includes a toolbar with various developer tools like Elements, Sources, Network, and a prominent Augury tab. Below the toolbar is a navigation bar with tabs: Component Tree (selected), Router Tree, and NgModules. To the right of the navigation bar is the text "Angular Version: 8.2.14". The main area is a hierarchical diagram of the component tree, starting from the root `AppComponent`. The tree structure is as follows:

```
graph TD; AppComponent --> HomeComponent; AppComponent --> LoanComponent; AppComponent --> AdminLazy[Admin [Lazy]]; HomeComponent --> LoginComponent; HomeComponent --> AboutloanComponent; HomeComponent --> AboutofferComponent; LoanComponent --> ApplyComponent; LoanComponent --> EligibilityComponent; LoanComponent --> StatusComponent; LoanComponent --> RegisterComponent; LoanComponent --> EmiComponent; AdminLazy --> Notification; AdminLazy --> InboxComponent; AdminLazy --> AlertComponent; AdminLazy --> ReportsComponent; AdminLazy --> AlertComponent;
```



Augury

The screenshot shows the Augury extension running in a browser. The main view displays the Angular component tree for the Bank of America homepage. The tree includes components like `AppComponent`, `TitleLoansComponent`, and `OfferCardComponent`. The `OfferCardComponent` is expanded, showing its internal structure with components such as `OfferCardHeaderComponent`, `OfferCardBodyComponent`, and `OfferCardFooterComponent`. The `OfferCardBodyComponent` is further expanded, revealing its child components: `OfferCardRateComponent`, `OfferCardOfferTextComponent`, and `OfferCardSavingsTextComponent`.

| Imports | Exports | Providers | Declarations | ProvidersInDeclarations |
|--|-------------------|---|--|-------------------------|
| <code>BrowserModule</code> <code>AppRoutingModule</code> <code>MenubarModule</code> <code>FormsModule</code> <code>ReactiveFormsModule</code> <code>HttpClientModule</code> <code>MatFormFieldModule</code> <code>MatInputModule</code> <code>BrowserAnimationsModule</code> <code>MatButtonModule</code> <code>MatIconModule</code> <code>FormsModule</code> <code>ReactiveFormsModule</code> <code>AdminModule</code> | <code>None</code> | <code>MenuService</code> <code>UserService</code> <code>LoginGuard</code> <code>AuthService</code> | <code>AppComponent</code> <code>MenuComponent</code> <code>HomeComponent</code> <code>AboutloanComponent</code> <code>AboutofferComponent</code> <code>LoanComponent</code> <code>ApplyComponent</code> <code>EligibilityComponent</code> <code>StatusComponent</code> <code>EmiComponent</code> <code>LoginComponent</code> <code>RegisterComponent</code> <code>NotificationComponent</code> <code>EqualValidator</code> <code>ParentComponent</code> <code>ChildComponent</code> | <code>None</code> |



Add debugger in ts file

Screenshot of a browser developer tools window showing the Sources tab with the app.component.ts file open. The code is paused at line 16, column 5, where a debugger; statement is highlighted.

The browser address bar shows "localhost:7070/Login". The developer tools tabs include Elements, Sources, Network, Console, Memory, Performance, Application, Security, Audits, Lucid, and Augury. The Sources tab is active, displaying the file structure and the source code.

The code in app.component.ts:

```
7 })
8 export class AppComponent {
9   title = 'loanapp';
10  //one way binding
11  private logoPath='./assets/boa.jpg';
12  private banner1='./assets/banner.jpg';
13  private banner2='./assets/banner2.jpg';
14  constructor()
15  {
16    debugger;
17  }
18 }
```

The right panel shows the "Debugger paused" status with a call stack listing:

- Debugger paused
- Watch
- Call Stack
 - AppComponent
- createClass
- createDirectiveInst...
- createViewNodes

At the bottom, the Windows taskbar shows various pinned icons and the date/time: 22:06 05/12/2019.



Console Debugging

Screenshot of the Visual Studio Code interface showing the main.ts file being edited.

The main.ts file contains the following code:

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3 import { enableDebugTools } from "@angular/platform-browser";
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6 import { ApplicationRef } from "@angular/core";
7 if (environment.production) {
8   enableProdMode();
9 }
10 //for profiling
11 platformBrowserDynamic().bootstrapModule(AppModule).then((module) => {
12   let applicationRef = module.injector.get(ApplicationRef);
13   let appComponent = applicationRef.components[0];
14   enableDebugTools(appComponent);
15 }).catch(err => console.error(err));
```

The code editor shows syntax highlighting for Angular imports and components. A blue arrow points to the line `import { ApplicationRef } from "@angular/core";` in the code.

VS Code interface elements visible:

- File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- boaangular2019 [F:\boaangular2019] - ...\\loanapp\\src\\main.ts
- Add Configuration...
- Project, Git: master
- 1: Project, 2: Structure, 2: Favorites
- File Explorer: services, assets, environments, main.ts (selected), polyfills.ts, styles.css, test.ts, typings.d.ts, .bitmap, .editorconfig, .gitignore
- Code Editor: main.ts
- Terminal: 6: TODO, 9: Version Control, Terminal
- Event Log: 1
- Bottom status bar: Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (today 20:10), 6:48, LF, UTF-8, EditorConfig, Git: master, 22:09, ENG, 05/12/2019, 26
- Taskbar icons: Search, Task View, File, Microsoft Word, Microsoft Excel, Microsoft Powerpoint, Microsoft Edge, Microsoft Outlook, Microsoft Word, Microsoft Excel



- Angular offers a precious tool: `ng.profiler`
- `ng.profiler.timeChangeDetection()`
- ran 489 change detection cycles
- 1.02 ms per check
- You can see how many change detection cycles it ran (it should be at least 5 cycles or during at least 500ms), and the time per cycle.
- You can also record the CPU profile during these checks to analyze them with `ng.profiler.timeChangeDetection({ record: true })`.



TITLE LOANS UP TO \$2500
NEED CASH?



Elements Console Sources Network Memory Performance Application Security Audits Lucid Augury

Default levels ▾ 1 hidden

[WDS] Live Reloading enabled. client:52

✖ > Uncaught TypeError: Cannot convert undefined or null to object reactTraverser.js:6

at Function.keys (<anonymous>)
at [reactTraverser.js:6](#)

> ng.profiler()

✖ > Uncaught TypeError: ng.profiler is not a function VM8937:1

at <anonymous>:1:4

> ng.profiler.timeChangeDetection()

ran 390 change detection cycles platform-browser.js:2287

1.28 ms per check platform-browser.js:2288

↳ > ChangeDetectionPerfRecord {msPerTick: 1.283935897439593, numTicks: 390} ⓘ

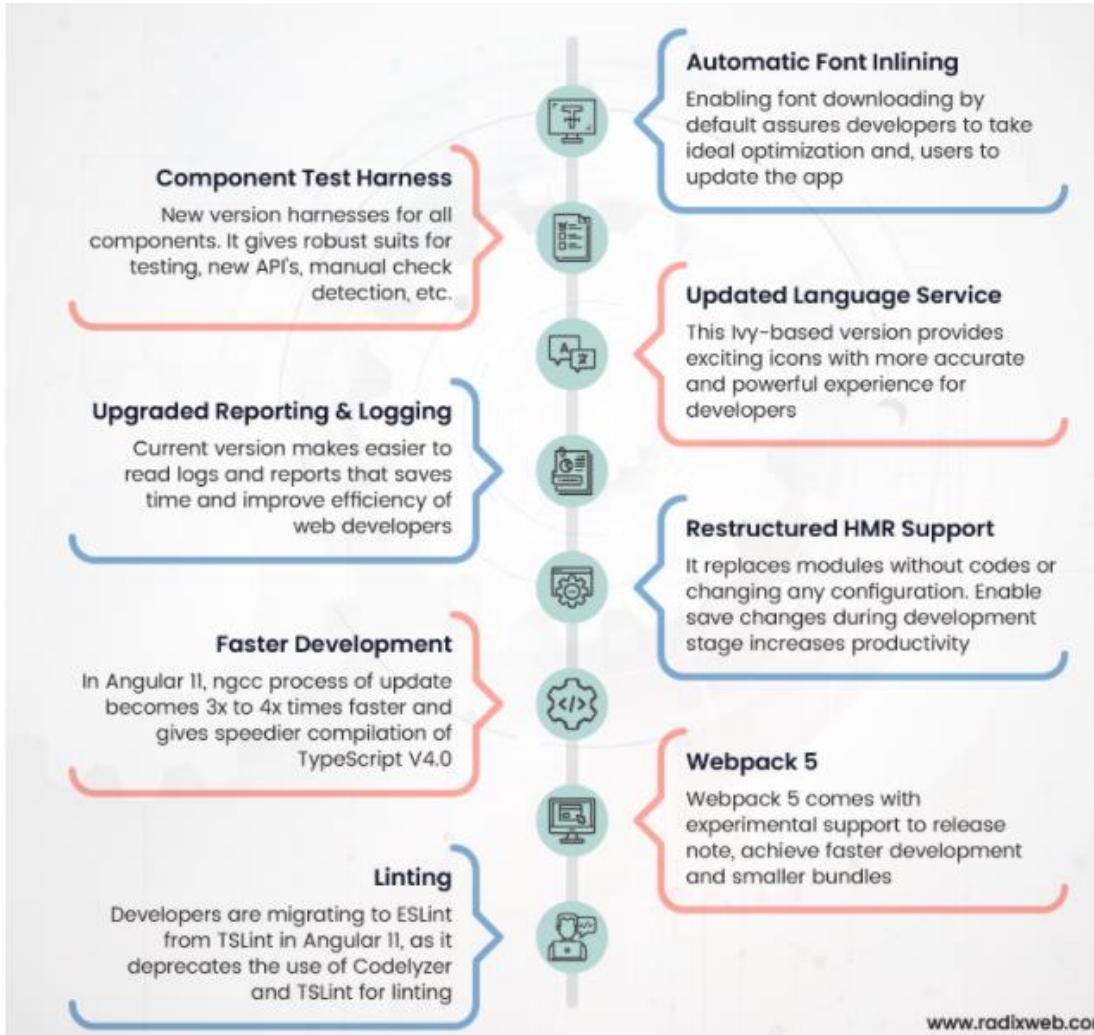
msPerTick: 1.283935897439593
numTicks: 390

▶ __proto__: Object

>



Angular 11 Features





Angular 11 Features

- Updated Hot Module Replacement (HMR) Support
 - Hot Module Replacement is a mechanism that allows the modules to be replaced without a full browser refresh.
 - `ng serve -hmr`
- Updates on Operation Byeolog
- Automatic Inlining of Fonts



Angular 11 Features

- "configurations": {
- "optimization": true
- }
- You can disable this optimisation by changing the flag to below snippets
- "configurations": {
- "optimization": {
- "fonts": false
- }
- }



Angular 11 Features

- Update to get version 11
- `ng update @angular/cli @angular/core`
- Generate E2E test with `async/wait`
- ViewEncapsulation
 - Angular 11 changelog involves the removal of the `ViewEncapsulation.Native`. Instead, in this new release, we can use `ViewEncapsulation.ShadowDom`.
 - The `ng update` will automatically update the existing pages.



Angular 11 Features

- Browser Support/cleaning
 - The support for IE 9, 10, and IE mobile in angular 11 is removed.
 - They were only deprecated in the version 10 release, but they are discontinued in this version.
 - Angular however still supports IE11 as the only version.
 - Angular 11 also has removed obsolete API and several functions has been added to its deprecated list.



Angular 11 Features

- Pipes
 - Angular recent version has fixed the typing for date and number pipe which earlier used to take any type as input.
 - In the datetime, datepipe will round off the millisecond part to the nearest millisecond provided.
 - The async pipe will not return null anymore as a value for an undefined input.
 - This can cause compilation errors which is why it is still a breaking change, however, actually uncovers invalid usages.



Angular 11 Features

- TypeScript 4.0 Support
 - The angular team in version 11 has dropped the support for TypeScript 3.9 and only supports TypeScript 4.0, only to speed up the builds.
- Forms
 - Angular 11 has made amendments to improve the typing for validators and asyncValidators that were earlier not available.



Angular 12 Features





Angular 12 Features

- Ivy Everywhere
 - Angular 12 is the step in the Angular revolution.
- Deprecating support for IE11
 - Internet Explorer is undoubtedly in its last leg as the support for IE continues to dwindle.
- Nullish coalescing
 - Angular 12 allows nullish coalescing utilization for its templates.
 - By using nullish coalescing operators (??) developers can write cleaner code.



Angular 12 Features

- {{age !== null && age !== undefined ? age : calculateAge() }}
- {{age ?? calculateAge() }}



Administrator: Node.js command prompt

```
10% building 4/4 modules 0 active
i  wds: Project is running at http://localhost:4200/webpack-dev-server/
i  wds: webpack output is served from /
i  wds: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 152 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 336 kB [initial] [rendered]
chunk {polyfills-es5} polyfills-es5.js, polyfills-es5.js.map (polyfills-es5) 655 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 733 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 7.75 MB [initial] [rendered]
Date: 2019-12-06T01:53:12.483Z - Hash: 0d56f2be39a138c9fbab - Time: 16267ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
i  wdm: Compiled successfully.
[07:23:12] I/launcher - Running 1 instances of WebDriver
[07:23:12] I/direct - Using ChromeDriver directly...
```

```
DevTools listening on ws://127.0.0.1:64842/devtools/browser/ef029325-9f3e-47d3-a08c-abc92bc024ef
Jasmine started
```

```
workspace-project App
  ✓ should display welcome message
```

```
Executed 1 of 1 spec SUCCESS in 3 secs.
[07:23:23] I/launcher - 0 instance(s) of WebDriver still running
[07:23:23] I/launcher - chrome #01 passed
```

```
F:\boaangular2019\loanapp>
```



07:23 06/12/2019 ENG 25



Angular 13 Features

- Type Script 4.4 Support Enabled
- 100% Ivy and No More Support for View Engine specific metadata
- Angular CLI Enhancements
- Ergonomic APIs
- Enhancements to Angular Tests
- Improvements in Angular Package Format (APF)
- Component API Updates
- No more support for Internet Explorer 11
- Improvements in Localization
- Changes in Framework and Dependency Updates
- Accessibility Update in Angular Material
- New and Improved Forms
- Inline Support for Adobe fonts
- Wondering how to Upgrade to Angular 13?
- <https://www.dsebastien.net/2021-11-05-angular-13-in-depth/>



Angular 13 Features

- Angular released v13 on November 4, 2021, which is now openly available as an active version till May 04, 2022.
- Significant changes and features are observed in this version, making the development process more efficient.
- With recent advancements in typescript, NodeJs & Angular CLI versions;
- The removal of Internet Explorer 11 to improve browser compatibility for users is one major update that sets it apart from its older counterpart.
- The releases of Angular have been eagerly awaited by many web developers as it is one the most streamlined.
- This new version of the wildly popular framework offers an exciting mix of features and improvements.



Angular 13 Features

- **Type Script 4.4 Support Enabled**
 - The new version: Angular 13 supports Typescript 4.4.
 - Node.js versions older than V12.20 are no longer supported.
 - Versions prior to TypeScript 4.4.2 are no longer supported in the core.
 - The significant highlights of TypeScript 4.4 are:
 - Improved detection of type guards.
 - Default catch variables.
 - Faster incremental builds.
 - The control flow of conditions can be analyzed.
 - Symbol and template string pattern index signatures.



Angular 13 Features

- **Control Flow Analysis of Aliased Conditions and Discriminants**

Before

```
function foo(arg: unknown) {  
  if (typeof arg === "string") {  
    console.log(arg.toUpperCase());  
  }  
}
```

Now

```
function foo(arg: unknown) {  
  const argIsString = typeof arg === "string";  
  if (argIsString) {  
    console.log(arg.toUpperCase());  
  }  
}
```



Angular 13 Features

- **Control Flow Analysis of Aliased Conditions and Discriminants**

```
C:\Windows\System32>f:  
F:\Typescript4\Features>tsc TypeGuard.ts  
F:\Typescript4\Features>node TypeGuard.js  
ROCK....  
F:\Typescript4\Features>
```



Angular 13 Features

- **Defaulting to unknown type in catch blocks**
- We use try-catch blocks to handle the mess that we create.
- But since the mess can be generated from the different sources, the Error parameter in the catch block must be kept loosely typed OR an implicitly-any defined parameter.



Angular 13 Features

- **Defaulting to unknown type in catch blocks**

Before

```
try {
  // Who knows what this might throw...
  executeSomeThirdPartyCode();
} catch (err) {
  // err: any
  console.error(err.message); // Allowed, because 'any'
  err.thisWillProbablyFail(); // Allowed, because 'any' :(
}
```

Once you enable **useUnknownInCatchVariables** in your TSConfig file, the Error parameter in the catch block becomes of type any. And you will start seeing below errors:

```
Property 'message' does not exist on type 'unknown'.
Property 'name' does not exist on type 'unknown'.
Property 'stack' does not exist on type 'unknown'.
```



Angular 13 Features

- **Defaulting to unknown type in catch blocks**

After

```
try {
  // Who knows what this might throw...
  executeSomeThirdPartyCode();
} catch (err) {
  if (err instanceof Error) {
    console.error(err.message);
  }
}
```



If you have a CustomError, you can create a class and make them extend with Error.



Angular 13 Features

- **Defaulting to unknown type in catch blocks**

```
F:\Typescript4\Features>tsc CatchDemo.ts  
  
F:\Typescript4\Features>node CatchDemo.js  
Infinity  
  
F:\Typescript4\Features>
```



Angular 13 Features

- Defaulting to unknown type in catch blocks

```
let amount:number;
let months:number;
amount=100000;
months=Math.random()*0;
try {
    console.log(amount/months);
} catch (err) {
    if (err instanceof DivideByZeroException) {
        console.error(err.message);
    }
}
```

Custom
Exception



Angular 13 Features

- **Exact Optional Property Types**
- With `exactOptionalPropertyTypes` enabled, TypeScript applies stricter rules around how it handles properties on type or interfaces which have a `?` prefix.
- This feature needs to be enabled in `TSConfig` using the flag `exactOptionalPropertyTypes`.
- In JavaScript, reading a missing property in an object returns `undefined`.
- Without this flag enabled, there are three values which you can set age to be: `number` and `undefined`.



Angular 13 Features

- **Exact Optional Property Types**
- Setting the value to undefined will allow most JavaScript runtime checks for the existence to fail, which is effectively falsy.
- This isn't quite accurate age: undefined is not the same as age not being defined.
- For example “age” in settings would have different behavior with undefined as the key compared to not being defined.



Angular 13 Features

• Exact Optional Property Types

Currently, if you define an interface like below:

```
interface Person {  
  name: string;  
  age?: number;  
}
```

The equivalent of the above interface was:

```
interface Person {  
  name: string;  
  age?: number | undefined;  
}
```

which meant that an object with a property **age** and its value **undefined** were totally fine.

```
const p: Person = {  
  name: "Daniel",  
  age: undefined, // This is okay by default.  
};
```



Angular 13 Features

- **Exact Optional Property Types**
- It is supposed to be interpreted as an object of type Person must contain name and another property age is optional but if it is present, age must be of type number.
- exactOptionalPropertyTypes makes TypeScript truly enforce the definition provided as an optional property:



Angular 13 Features

- **Static blocks in class**
- Static blocks in classes are a new feature of ECMAScript features that can help you write complex initialization logic for static members.

```
class Foo {  
    static count = 0;  
  
    // This is a static block:  
    static {  
        if (someCondition()) {  
            Foo.count++;  
        }  
    }  
}
```



Angular 13 Features

- **Inlay hints**
- When we have a lengthy list of parameters in a function
- When we have a tough time remembering the sequence of it.
- Inlay hints display useful information like parameter names and return types in our code.
- We can think of it as a sort of friendly “ghost text”.

```
export function getFavoriteColor(dayOfWeek: string) : Color | undefined {
  if (dayOfWeek === "Tuesday") {
    return new Color(
      kind: "rgb",
      red: 0x64,
      green: 0x95,
      blue: 0xED
    );
  }
}
```



Angular 13 Features

- **No initializers for Abstract properties**
- Abstract properties cannot have initializers.

```
F:\Typescript4\Features>tsc Account.ts
Account.ts:2:14 - error TS1267: Property 'prop' cannot have an initializer because it is marked abstract.

2     abstract prop = 1;
      ~~~~~

Found 1 error in Account.ts:2
```



Angular 13 Features

- **Symbol and Template String Pattern Index Signatures**
- TypeScript describe objects where every property must have a certain type using index signatures.
- This allows us to use these objects as dictionary-like types, where we can use string keys to index into them with square brackets.
- For example, we can write a type with an index signature that takes string keys and maps to boolean values.
- If we try to assign anything other than a boolean value, we'll get an error.



Angular 13 Features

- **Symbol and Template String Pattern Index Signatures**

```
interface BooleanDictionary {  
  [key: string]: boolean;  
}  
  
declare let myDict: BooleanDictionary;  
  
// Valid to assign boolean values  
myDict["foo"] = true;  
myDict["bar"] = false;  
  
// Error, "oops" isn't a boolean  
myDict["baz"] = "oops";  
  
Type 'string' is not assignable to type 'boolean'.
```



Angular 13 Features

- **100% Ivy and No More Support for View Engine specific metadata**
- Angular 13 no longer supports the View Engine, but all-new versions are now 100% Ivy.
- This means that there are less codebase complicity.
- The maintenance costs because older output formats will not work with this new update removing any compatibility issues caused by previous changes made in the Framework version 4.
- The framework has converted all internal tools to Ivy in advance so that this change can function seamlessly.
- Ivy assembles every component individually, which accelerates development times.



Angular 13 Features

- **Angular CLI Enhancements**
- The Angular CLI is an essential tool for modern web developers.
- It makes the workflow easier and smoother by doing most of what we need to do in one place, saving time on multiple steps throughout our development process.
- Angular CLI basically does everything you need it.
- With the new version 13, angular developers can now enable a persistent build-cache by default.
- This feature will cause up to 68% enhancement in development speed for existing apps and is an added benefit that caches build on disk which speeds things right along.



Angular 13 Features

- **Angular CLI Enhancements**
- Way 1: Update Angular CLI:
 - We can directly upgrade our angular cli version globally by using update command of ng. so you can do it by bellow command.
 - `ng update @angular/cli @angular/core`
 - Now you can see ng version by following command:
 - `ng version`



Angular 13 Features

- **Angular CLI Enhancements**

```
ng update @angular/cli @angular/core
```

For material/cdk upgrade

```
ng update @angular/cdk
```

if you are using angular material

```
ng update @angular/material
```

If you got any errors you can append --allow-dirty and --force option

```
ng update @angular/cli @angular/core --allow-dirty --force
```

```
ng update @angular/material --allow-dirty --force
```

```
ng update @angular/cdk --allow-dirty --force
```

if your project has any outdated dependencies, you can find out using the below command.

```
npm outdated
```



Angular 13 Features

- **Ergonomic APIs**
- The release of Angular v13 has reduced the load time using ergonomic code-splitting APIs and granular disruption at a component level.
- ESBUILD also brings an enhancement in performance with this new update.
- The ESBUILD JS Bundle is a highly-efficient way to pack your website with minimal coding.
- Now, it works in tandem with Terser for optimized global scripts that work across all frameworks including Vue, Elm, or Svelte depending on what you need.



Angular 13 Features

- **Enhancements to Angular Tests**
- The Angular team has made a few changes to Test Bed that currently functions properly for the test environment and modules to be torn down after every single one of our tests.
- With DOM cleaning now possible, expecting developers will have more optimized less interdependent applications with lower memory requirements as well quicker runs on their battery life.



Angular 13 Features

- **Improvements in Angular Package Format (APF)**
- The Angular Package Format is an industry-standard package format for storing libraries used in web app development.
- It provides great guidelines and structures every third-party library into its own unique repository, which makes it easy to find what you need when working on projects with multiple developers or yourself.



Angular 13 Features

- **Improvements in Angular Package Format (APF)**
- Angular 13 brings a new version of the APF, and we find some significant changes in it:
- There is no more production of UMD bundles.
- It produces ES2020 output.
- It creates Ivy partial compilation output.
- The node command-line tool exports all its available outputs to files with a sub path pattern, making them easy for developers to use multiple versions or paths in their projects.



Angular 13 Features

- **Component API Updates**
- The View Container Ref now offers an improved API to create dynamic components with less boilerplate code.
- If we use Angular 13, there is no need for Component Factory Resolver anymore.

Here's an example of creating dynamic components using the previous versions of Angular:-

```
@Directive({ ... })
export class MyDirective {
    constructor(private viewContainerRef: ViewContainerRef,
               private componentFactoryResolver:
                   ComponentFactoryResolver) {}
    createMyComponent() {
        const componentFactory = this.componentFactoryResolver.
            resolveComponentFactory(MyComponent);

        this.viewContainerRef.createComponent(componentFactory);
    }
}
```



Angular 13 Features

- **Component API Updates**

In Angular 13, this code will appear when the new API is used:-

```
@Directive({ ... })
export class MyDirective {
    constructor(private viewContainerRef: ViewContainerRef) {}
    createMyComponent() {
        this.viewContainerRef.createComponent(MyComponent);
    }
}
```



Angular 13 Features

- **No more support for Internet Explorer 11**
- The Angular framework has discontinued support for Internet Explorer 11, which means we can no longer utilize the APIs and features of modern browsers like CSS variables and web animations in this browser.
- There are no IE-specific polyfills or need for separate loading, resulting in a quicker load and smaller bundle size for apps.
- Some organizations or authorities that are still using IE 11 and haven't yet migrated to Microsoft Edge or other modern browsers will notice a significant difference.



Angular 13 Features

- **Improvements in Localization**
- \$localize is an efficient API that offers developers the perfect way to produce for internationalization (i18n) and tag messages in code or templates.



Angular 13 Features

- **Changes in Framework and Dependency Updates**
- The latest version of Angular 13 also comes with some crucial changes and updates.
- Now, RxJS v7 is the default for new applications built using the ng module;
- Current users must update manually by installing rxjs@7 during their build process if they want access to its features such as observables or reactive requests (which replaces Subject).



Angular 13 Features

- **Accessibility Update in Angular Material**
- With the new version of Angular, Google has introduced an accessibility enhancement in their Material Design Components.
- They have assessed and checked all MDCs for improved A11y features like larger touch sizes on radio buttons or checkboxes that feature higher contrast modes to make them easier to use when someone has a cognitive disability.



Angular 13 Features

- **New and Improvised Forms**
- The introduction of Angular v13 introduces a new kind of form: FormControlStatus. It gathers all status messages for form controls:
- For example, the AbstractControl.status is now FormControlStatus instead of a string.
- The type of StatusChanges is now Observable<FormControlStatus> in place of Observable<any>.



Angular 13 Features

- **Inline Support for Adobe fonts**
- The introduction of Angular 13 has introduced an exciting change that allows users to use Adobe fonts by default.
- These enhanced apps are able to make FCP (First Contentful Paint), which is significant for optimizing app performance and user experience!
- This new update means you don't need any extra modules or libraries anymore- just ng update will do it all.



Angular 14 Features



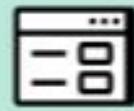
STANDALONE
COMPONENTS



STREAMLINED PAGE TITLE
ACCESSIBILITY



EXTENDED DEVELOPER
DIAGNOSTICS



STRICTLY TYPED
FORMS



ANGULAR CLI
AUTO-COMPLETION



OPTIONAL
INJECTORS



Angular 14 Features

- Bind to protected component members

Now you can bind protected component members directly from the template. Refer to the following code example.

 Copy

```
@Component({
  selector: 'app-root',
  template: '{{ title }}', // Now compiles!
})
export class SampleComponent {
  protected title: string = 'Angular 14';
}
```

This feature provides more control over the public API surface of your reusable components.



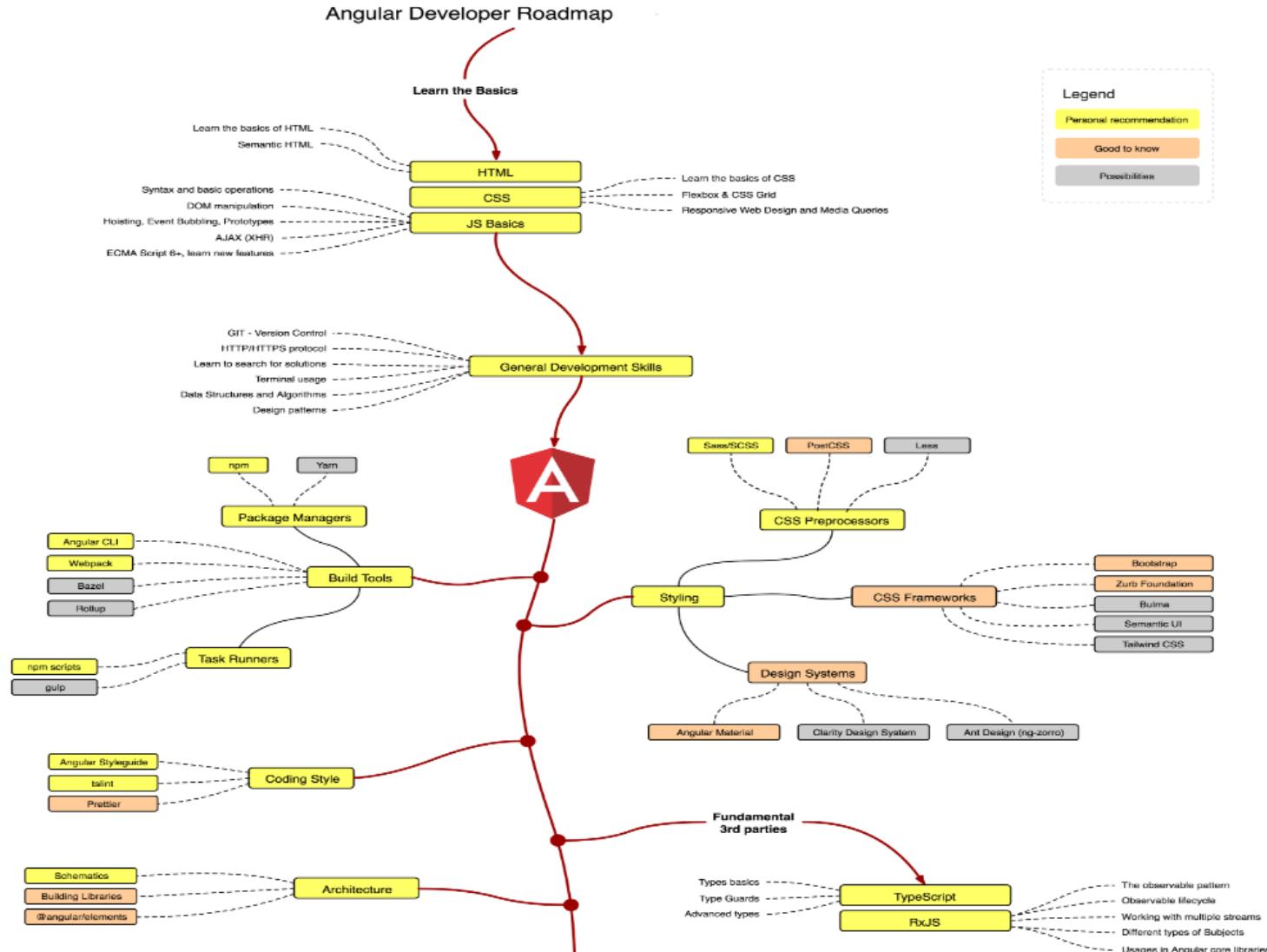
Angular 14 vs 13 Features

Comparison of Features: Angular 14 and Angular 13

| FEATURES | ANGULAR 13 | ANGULAR 14 |
|------------------------------------|------------------------------------|---|
| Release Date | Released on November 3, 2021 | Released on 2nd June 2022 |
| TypeScript Update | TypeScript 4.4 | TypeScript 4.7 |
| GitHub Issues | GitHub issue | All GitHub issues were removed |
| Page Title Accessibility | Need to add title with route.title | No additional import is required when adding titles |
| Angular CLI Auto-Completion | Not fully applicable | Enables real-time type-ahead auto-completion |
| Standalone Components | Not presented | Create a component that is not part of any NgModule |

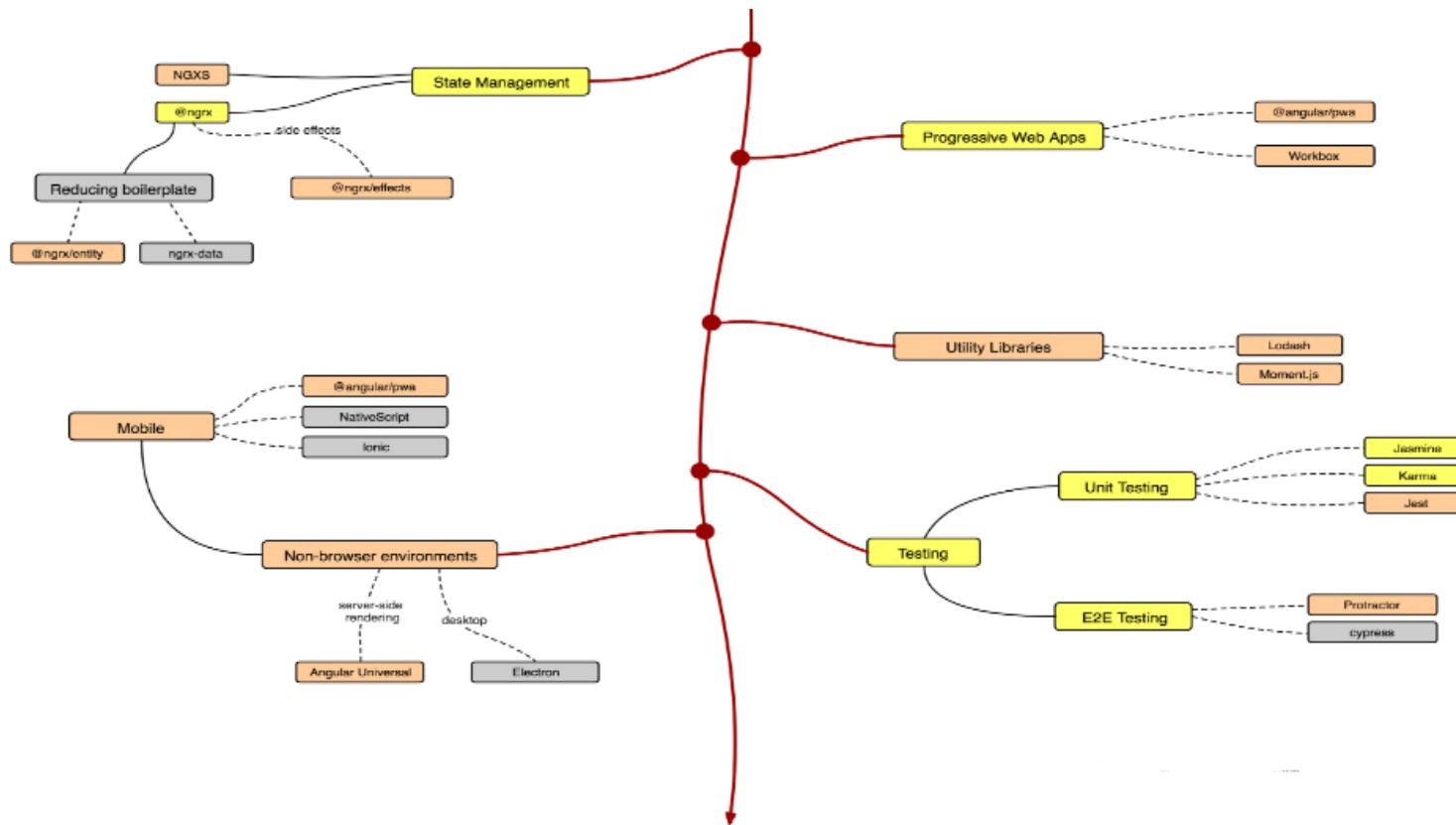


Angular Roadmap





Angular Roadmap





Angular Roadmap

ANGULAR VERSIONS



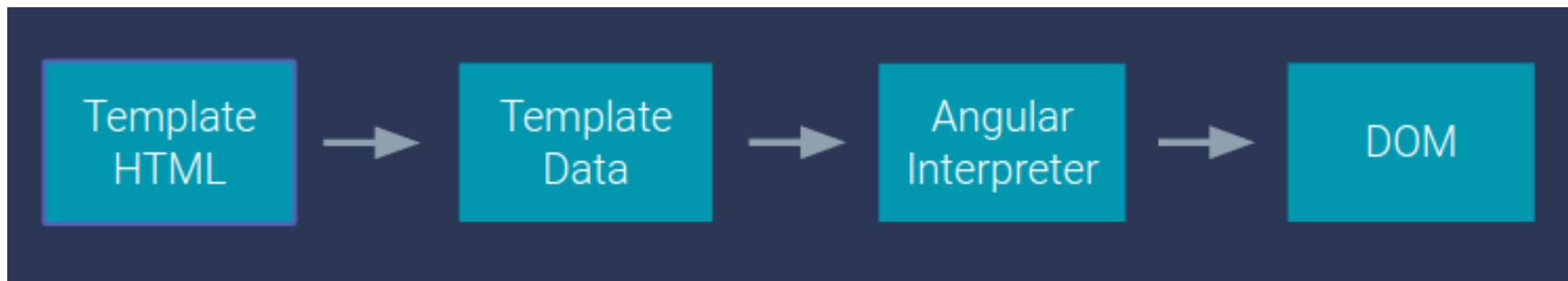


From View Engine to Ivy: Rendering in Angular

- The View Engine, called Renderer2, was released with Angular 4.
- When Angular first processes the application code, it makes a note of all associations between the HTML properties and the TypeScript (TS) properties.
- Then it goes about creating binding — a set of instructions for each HTML-TS-property association.
- Since these bindings are component-specific, it is easier to optimize them than some generic instructions.
- The resulting template data is then interpreted and transformed into DOM.



From View Engine to Ivy: Rendering in Angular





From View Engine to Ivy: Rendering in Angular

Let's have a look at the following component:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     <div>
7       <span>{{title}}</span>
8       <app-child *ngIf="show"></app-child>
9     </div>
10    `,
11   styles: []
12 })
13 export class AppComponent {
14   title = 'ivy-tree-shaking';
15   show: boolean;
16 }
```



From View Engine to Ivy: Rendering in Angular

In View Engine, it will be compiled to the following template (node_modules/.bin/ngc). Note all these “i0.eeld” — this is what the bindings look like.

```
1  /**
2   * @fileoverview This file was generated by the Angular template compiler. Do not edit.
3   *
4   * @suppress {suspiciousCode,uselessCode,missingProperties,missingOverride,checkTypes}
5   * tslint:disable
6   */
7  import * as i0 from "@angular/core";
8  import * as i1 from "./child.component.ngfactory";
9  import * as i2 from "./child.component";
10 import * as i3 from "@angular/common";
11 import * as i4 from "./app.component";
12 var styles_AppComponent = [];
13 var RenderType_AppComponent = i0.ecrt({ encapsulation: 2, styles: styles_AppComponent, data: {} });
14 export { RenderType_AppComponent as RenderType_AppComponent };
15 function View_AppComponent_1(_l) { return i0.evid(0, [(_l)(), i0.eeld(0, 0, null, null, 1, "app")]);
16 export function View_AppComponent_0(_l) {
17     return i0.evid(0, [(_l)(),
18         i0.eeld(0, 0, null, null, 4, "div", [], null, null, null, null), (_l)(),
19         i0.eeld(1, 0, null, null, 1, "span", [], null, null, null, null), (_l)(),
20         i0.eited(2, null, ["", ""])), (_l)(),
21         i0.eand(16777216, null, null, 1, null, View_AppComponent_1)),
22         i0.edid(4, 16384, null, 0, 181 | 2 tainerRef, i0.TemplateRef], { ngIf: [0, ""]),
23         function (_ck, _v) { var _co = _co || _co || _co; var currVal_1 = _co.show; _ck(_v, 4, 0, currVal_1);
24         if (_v) { _co.show = true; _ck(_v, 4, 0, currVal_1); } else { _co.show = false; _ck(_v, 4, 0, currVal_1); }
25     }
26 }
```



Ivy

- Ivy is the code name for Angular's next-generation compilation and rendering pipeline.
- With the version 9 release of Angular, the new compiler and runtime instructions are used by default instead of the older compiler and runtime, known as View Engine.
- Being not only a rendering engine, but also a compiler, Ivy brings along many new topics and optimizations across the whole framework.

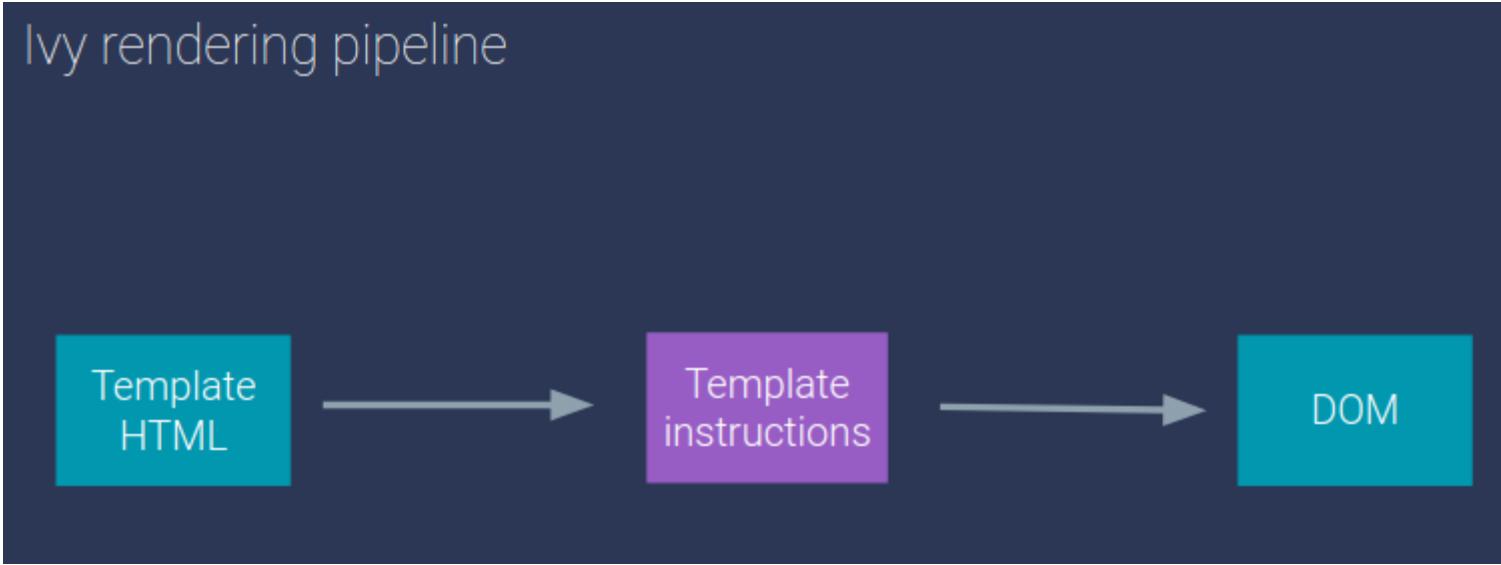


- Instead of generating template data and passing it into an interpreter that then makes decisions on which operations to run, a set of template instructions is now generated directly.
- These template instructions is where the logic that instantiates components, creates DOM nodes, and runs change detection lives now.
- There is no longer need for template interpretation, just execute it to build the DOM. How is it possible? With the help of the Incremental DOM.



Ivy

Ivy rendering pipeline

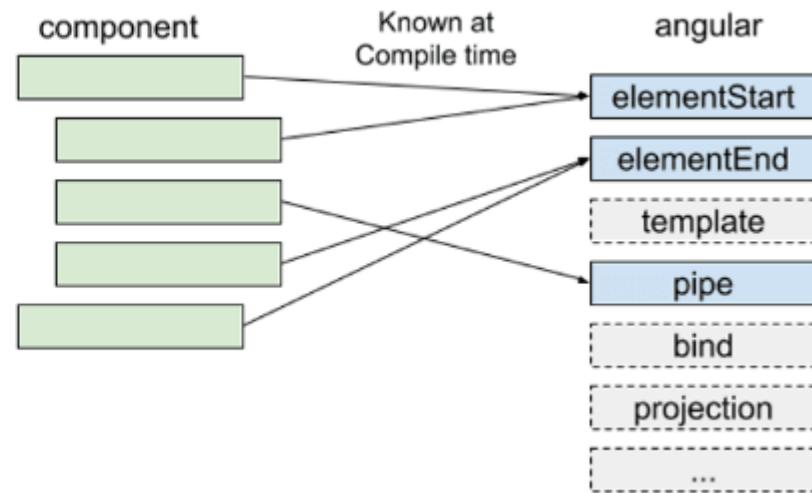




Incremental DOM

- The incremental DOM differs from Virtual DOM approaches in that a diff operation is performed incrementally (that is one node at a time) against the DOM representation itself, rather than on a Virtual DOM tree (a copy of the DOM which can differ from “the real one” before reconciliation is finished).

Incremental DOM is Tree Shakable





Incremental DOM vs. Virtual DOM

- Mobile browsing has largely subsumed user browsing trends.
- However, web developers cannot influence the hardware of the user's device.
- Thus, it totally makes sense to optimize for memory-constrained devices with the Incremental DOM.
- Ideally, the final bundle itself should be smaller, too.



Incremental DOM vs. Virtual DOM

- Memory efficiency is not necessarily the fastest solution.
- Rendering speed is largely impacted by 3 things:
- The browser's rendering engine, the browser's JavaScript engine, and the hardware of the device.
- At least Google can push on the browser's side in Chrome — the browser with the largest amount of the market share.
- As the result, they cover memory constraints and partially compensate for performance.



Incremental DOM vs. Virtual DOM

- Thus, applications that perform fewer DOM updates will benefit from the reduced memory consumption of Incremental DOM.
- While those that perform many DOM updates will benefit more from Virtual DOM's greater performance.

What exactly is Angular Ivy?



- Angular Ivy is the code-name for the next version of the Angular rendering pipeline and view engine.
- It's a complete refactor of the rendering with the intention to tackle 3 of the major issues with the current version:
 - Bundle size
 - Page loading speed
 - Debugging



Internationalization with @angular/localize

- A new package called `@angular/localize` has been introduced in Angular 9.0.
- It is used under the hood to give us the same features we had previously: translations in templates at compile time.



Internationalization with @angular/localize

```
F:\BOATypescript4\abcbank\src\app>ng add @angular/localize
i  Using package manager: npm
✓ Found compatible package version: @angular/localize@14.2.12.
✓ Package information loaded.

The package @angular/localize@14.2.12 will be installed and executed.
Would you like to proceed? Yes
✓ Packages successfully installed.
UPDATE src/polyfills.ts (2567 bytes)

F:\BOATypescript4\abcbank\src\app>ng extract-i18n
✓ Browser application bundle generation complete.

Initial Chunk Files
main.js
polyfills.js
runtime.js
Names          | Raw Size
main           | 3.37 MB
polyfills      | 983.70 kB
runtime        | 12.15 kB
Initial Total   | 4.35 MB

Lazy Chunk Files
src_app_systemaccount_systemaccount_module_ts.js
Names          | Raw Size
systemaccount-systemaccount-module | 10.92 kB

Build at: 2023-03-13T17:28:44.631Z - Hash: fffa218414af9933 - Time: 27716ms

F:\BOATypescript4\abcbank\src\app>
```



Internationalization with @angular/localize

```
ng serve --configuration=fr

F:\BOATtypescript4\abcbank>npm run start:fr
> abcbankapp@0.0.0 start:fr
> ng serve --configuration=fr
*****
This is a simple server for use in testing or debugging Angular applications locally.
It hasn't been reviewed for security issues.

DON'T USE IT FOR PRODUCTION!
*****
✓ Browser application bundle generation complete.

Initial Chunk Files      | Names          | Raw Size | Estimated Transfer Size
main.js                  | main           | 605.58 kB | 149.89 kB
styles.css, styles.js    | styles         | 547.23 kB | 71.05 kB
polyfills.js             | polyfills      | 151.42 kB | 39.40 kB
runtime.js               | runtime        | 2.73 kB   | 1.27 kB
                           | Initial Total | 1.28 MB   | 261.61 kB

Lazy Chunk Files         | Names          | Raw Size | Estimated Transfer Size
897.js                  | systemaccount-systemaccount-module | 1.54 kB   | 414 bytes

Build at: 2023-03-14T01:23:53.113Z - Hash: 43200ad639152520 - Time: 74525ms

Warning: 1 rules skipped due to selector errors:
legend+* -> Cannot read properties of undefined (reading 'type')

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

24°C Mostly sunny          Q Search          = 28  WS  P  ENG IN  06:55  14/03/2023
```



Standalone Components

- The components are not dependent on any other components and are referred to as Angular 14 standalone components.
- The components of Angular 13 and the previous versions depended on the ngModules.
- So, for example, if we want to include or use any additional module, pipes, services, or even directories in the component, we need to import all such things in the module file.
- But in Angular 14, the scenario is entirely different.
- First, we will create one component, “HomeComponent”, as a standalone by using the command.



Typed Angular Forms

- As we know, Angular typically consists of Template-Driven and Reactive Forms.
- So far, in Angular 13's reactive forms. Let's say you define a type of Reactive Form and you want to update its value to another.
- It's simple, but it's not a good practice in Angular as it strictly uses TypeScript programming language.
- To make it easy for the developers, Angular 14 introduced Typed Angular Forms.
- It uses strictly Typing and strongly eliminates the implementation of strict typing for the Angular Reactive Forms Package.
- Angular 14's Typed forms aim to ensure that the values inside form controls, groups, and arrays are type-safe across the API surface.
- It helps Angular js developers to enable safer forms, especially for deeply nested complex cases.
- **The excellent part of Angular 14 is that it allows the developers to use “UntypedFormGroup” if they do not want to use Typed Forms.**



Rxjs 7.5

- <https://rxjs.dev/6-to-7-change-summary>



Subjects

- A Subject is a special type of Observable that allows values to be multicasted to many Observers.
- The subjects are also observers because they can subscribe to another observable.
- They get value from it, which it will multicast to all its subscribers.
- Basically, a subject can act as both observable & an observer.



Subjects

Angular Subjects

Observable

`subscribe()`

Subject

`subscribe()`

`next()`

`error()`

`complete()`

`observers[]`

Observer

`next()`

`error()`

`complete()`



Subjects

- Subject is hot Observable
- Observables are classified into two groups. Cold & Hot
- Cold observable
 - The cold observable does not activate the producer until there is a subscriber.
 - This is usually the case when the observable itself produces the data.
 - Producer is part of the observable.



Subjects

- Hot observable
- The hot observable does not wait for a subscriber to emit the data.
- It can start emitting the values right away.
- This happens when the producer is outside the observable.
- We have created an observable using subject.
- In the ngOnInit, we emit the values & close the Subject. That is without any subscribers.



Subjects are Multicast

- More than one subscriber can subscribe to a subject.
 - They will share the same instance of the observable.
 - This means that all of them receive the same event when the subject emits it.
-
- Multiple observers of an observable, on the other hand, will receive a separate instance of the observable.



BehaviorSubject

- Behavior Subject requires an initial value and stores the current value and emits it to the new subscribers.



ReplaySubject

- Replay Subject replays old values to new subscribers when they first subscribe.
- The Replay Subject will store every value it emits in a buffer.
- It will emit them to the new subscribers in the order it received them.
- You can configure the buffer using the arguments buffer Size and window Time
- Buffer Size: No of items that Replay Subject will keep in its buffer. It defaults to infinity.
- Window Time: The amount of time to keep the value in the buffer. Defaults to infinity.



Async Subject

- Async Subject only emits the latest value only when it completes. If it errors out then it will emit an error, but will not emit any values.



toPromise Operator is being deprecated

Before:

```
const numbers$ = of(1,2,3,4,5)

// toPromise() will return the last value of the observable
numbers$.toPromise().then(n => console.log("toPromise(): " + n))
```

After:

```
const numbers$ = of(1,2,3,4,5)

// lastValueFrom() will return the last value just like to promise
lastValueFrom(numbers$).then(n => console.log("lastValueFrom(): " + n))

// firstValueFrom() will return the first value of the observable
firstValueFrom(numbers$).then(n => console.log("firstValueFrom(): " + n))
```



Consuming AsyncIterable Support

```
const range = {
  from: 1,
  to: 5,

  [Symbol.asyncIterator]() { // (1)
    return {
      current: this.from,
      last: this.to,

      async next() { // (2)
        await wait(1000);
        if (this.current <= this.last) {
          return { done: false, value: this.current++ };
        } else {
          return { done: true };
        }
      }
    };
  }
};
```



Consuming AsyncIterable Support

We can consume the above AsyncIterable object as follows:

```
from(range).subscribe(x => console.log(x))
```



Memory Leak Handling

- Unsubscribe
- Using unsubscribe is the basic method, the most known.
- The principle is quite simple: we store our subscribes in objects of type Subscription.
- And we destroy them when we want to stop listening to them.
- Most commonly, the destruction is done in the ngOnDestroy method (which is the last step of an Angular component lifecycle).



Memory Leak Handling

```
subscription = new Subscription()

constructor(){
  this.subscription = interval(200).subscribe( value) => console.log(value)
}

ngOnDestroy(){ this.subscription.unsubscribe() }
```

It is possible to use the add method of the Subscription class to chain several subscriptions.

Doing so, if we destroy the first observable of the chain, all those that follow will be destroyed.

This avoids writing the unsubscribe line several times.

Of course, this method destroys all observables at once, so it is not to be used in all situations.



Memory Leak Handling

- Take, first, etc
- Another method to manage subscriptions in a controlled way is the use of RxJs functions in the pipe() of the observable.
- There are several of them that allow managing each observable in a unitary way: first(), take(), takeUntil() and takeWhile().
- First: it allows retrieving only the first value that comes from an observable that respects the indicated condition (otherwise, it only takes the first one).
- This is very useful when you want to retrieve, for example, data about an authentication.
- We will create an observable to manage the asynchronism for the time it takes to check a role and as soon as the information is returned, we will destroy the observable.
- WARNING: if you put a predicate and there is no match, an exception is thrown!



Memory Leak Handling

- Take: it allows retrieving the first x values of an observable before destroying it.
- It is often recommended to use a `take(1)` rather than a `first()` if you want the first value emitted by the observable.
- Indeed, `first()` is preferred when you have some logic (other than the order of emission) in the selection, and it is more likely to cause mistakes.



Memory Leak Handling

- TakeUntil: it is certainly one of the most useful tools to manage observables efficiently in RxJs operators.
- It allows putting a destruction condition in the form of a Subject.
- In complex applications where the observables have the same life cycle, this solution is really to be recommended compared to unsubscribe, by setting the Subject to true when the component is destroyed.



Memory Leak Handling

- TakeWhile: it works as a filter on the values of the observable.
- It is rarely used in practice, except if you want to filter the returns of an observable.
- For example, we launch a call on a Meteo France API to retrieve the temperatures of August, but we want to retrieve the temperatures until the first one below 15 °C.



Memory Leak Handling

```
interval(200).pipe(first((val) => val > 5))  
    .subscribe( (value) => console.log(value))  
  
// Output:  
// 6
```

```
interval(200).pipe(take(3)).subscribe( (value) => console.log(value))  
  
// Output:  
// 1  
// 2  
// 3
```



Memory Leak Handling

```
onDestroy = new Subject()

constructor(){
    interval(200).pipe(takeUntil(this.onDestroy))
        .subscribe( value) => console.log(value)

    setTimeout(destroyMyObservable(), 500);
}

destroyMyObservable(){
    this.onDestroy.next(true)
    this.onDestroy.complete()
}

// Output:
// 1
// 2
```



Memory Leak Handling

- Angular Async Pipe
- The recommended approach for handling Observables in Angular is by using the Angular async pipe.
- The async pipe subscribes to an Observable and returns the latest value it has emitted.
- The async pipe marks the component to be checked for changes every time a new value is emitted.
- When the component is destroyed there is no need to unsubscribe because the async pipe automatically unsubscribes preventing memory leaks.
- In addition, the async pipe makes the code more readable.



Memory Leak Handling

```
myObservable$ = interval(200)
```

component.ts

```
<div> {{ myObservable$ | async }} </div>
```



Memory Leak Handling

- Conclusion
- Observables are powerful tools for a dynamic website.
- But using them in an uncontrolled way could greatly decrease the performance of your website.
- You can anticipate memory leak problems by using different functions:
 - if you use it locally in the HTML: | async
 - if you have more than one observable with the same life cycle: takeUntil
 - if you only want the first value emitted or the nth ones: take
 - if you have a specific life cycle logic: first, takeWhile, takeUntil



Why Memory Leaks are tricky

- The trickiest aspect of memory leaks is that they are hard bugs to spot.
- Unlike CPU usage issues, where you can see the UI lagging, memory leaks are much a more subtle sort of problem.
- If an application is leaking memory, the user will at some point realize that it is slower, sluggish, and will likely pause very frequently.



Why Memory Leaks are tricky

- Monitoring Memory with the Performance Monitor tool
- The first thing to do is to open the Chrome Dev Tools, open the panel on the right and click on More Tools > Performance Monitor.
- The memory of our application is displayed in the blue graph.
- Whenever I switch page, the memory jumps up almost 20mb!
- 154MB and 99% CPU? Clearly, something's wrong



Why Memory Leaks are tricky

- Starting the debugging process: Memory Snapshots
- The first thing I do when debugging is to record memory snapshots in two stages:
- at initial load, as soon as the app becomes stable, and all the elements have been loaded
- a second time once the initial data is replaced by other data.
- Make sure your app is not actually adding additional resources, unless of course if that is a bug.
- For example, you could be switching page or forcing some elements to show/hide



Why Memory Leaks are tricky

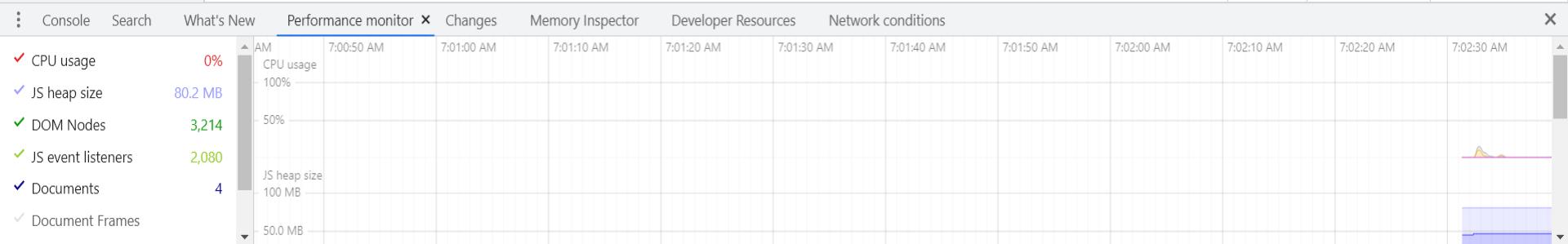
- In order to take a Memory Snapshot, open the Dev Tools->Memory, select “Heap Snapshot” and then click on the button “Take Snapshot”.
- The profiles are listed on the left-hand side, and you can compare them with each other to visualize which objects have been retained in memory.



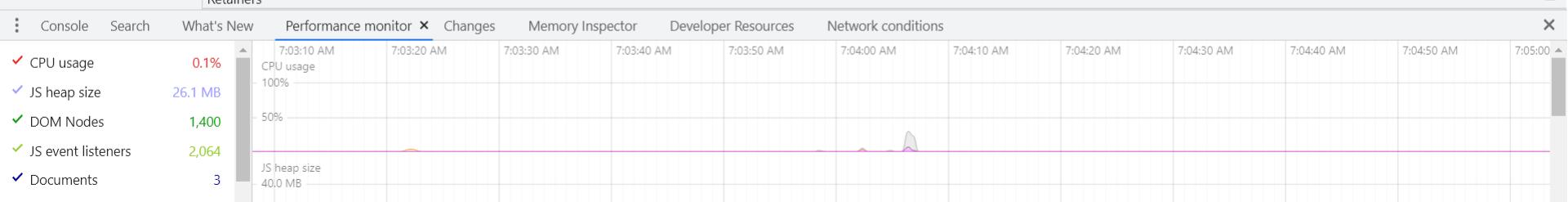
Why Memory Leaks are tricky

Elements Console Recorder Performance insights Network Sources Performance Memory Application Security Lighthouse Augury

| Profiles | Summary | Class filter | All objects | Constructor | Distance | Shallow Size | Retained Size |
|----------------|-----------|--------------|-------------|-------------|----------|--------------|---------------|
| HEAP SNAPSHTOS | Retainers | | | | | | |
| | Object | | | | Distance | Shallow Size | Retained Size |
| | | | | | | | |



| Profiles | Summary | Class filter | All objects | Constructor | Distance | Shallow Size | Retained Size |
|----------------|--------------------|--------------|--------------------------------|-------------|-----------|--------------|----------------|
| HEAP SNAPSHTOS | Snapshot 1 18.6 MB | Save | ► (compiled code) ×122951 | - | 6 922 040 | 37 % | 7 675 664 41 % |
| | | | ► (closure) ×49794 | - | 1 512 972 | 8 % | 6 188 260 33 % |
| | | | ► Object ×11367 | 2 | 283 324 | 2 % | 4 621 116 25 % |
| | | | ► system / Context ×11066 | 3 | 365 636 | 2 % | 4 494 252 24 % |
| | | | ▼(array) ×14961 | 2 | 2 401 704 | 13 % | 3 008 428 16 % |
| | | | (internal array)[] @1659 | - | 131 080 | 1 % | 131 080 1 % |
| | | | ►(object properties)[] @135489 | 13 | 49 180 | 0 % | 63 232 0 % |
| | | | ►(object properties)[] @135493 | 13 | 49 180 | 0 % | 49 180 0 % |
| | | | ►(internal array)[] @753349 | 3 | 4 104 | 0 % | 42 608 0 % |
| | | | Retainers | | | | |





Why Memory Leaks are tricky

- When the initial snapshot is taken, the tools will show you the summary of the current snapshot, but you can compare two snapshots by choosing “Comparison” from the dropdown above the objects.



Why Memory Leaks are tricky

- Another common cause of memory leaks is DOM events that are never unregistered.
- Angular's Renderer may take care of it, but that is only the cause if the events are defined in the template, just as with the async pipe.



Why Memory Leaks are tricky

```
@Component({...})  
export class ScrollComponent {  
  constructor(private renderer: Renderer2) {}  
  
  ngOnInit() {  
    this.renderer.listen(document.body, 'scroll', () => {  
      this.updatePosition();  
    });  
  }  
  
  updatePosition() { /* implementation */ }  
}
```

This does, indeed, create a memory leak every time we instantiate ScrollComponent



Why Memory Leaks are tricky

```
@Component({...})
export class ScrollComponent {
  private listeners = [];

  constructor(private renderer: Renderer2) {}

  ngOnInit() {
    const listener = this.renderer.listen(
      document.body,
      'scroll',
      () => {
        this.updatePosition();
      });
    this.listeners.push(listener);
  }

  ngOnDestroy() {
    this.listeners.forEach(listener => listener());
  }

  updatePosition() { /* implementation */ }
}
```

Unregistering all the events prevents the component ScrollComponent to be retained in memory and will be cleaned up once destroyed, along with its children.



Why Memory Leaks are tricky

- Websocket Connections
- WebSocket connections must always be closed when unused.
- Imagine we have a component PricerComponent that subscribes to a WebSocket and displays incoming cryptocurrency prices.



Why Memory Leaks are tricky

```
@Component({
  selector: 'pricer',
  template: `
    <span>{{ id | titlecase }}:</span>
    <span>{{ ( price$ | async) || 'loading...' }}</span>
  `,
})
export class PricerComponent {
  @Input() id: string;

  public price$ = new Subject();
  private static Endpoint = 'wss://ws.coincap.io/prices/';
  private webSocket: WebSocket;

  ngOnInit() {
    this.webSocket = new WebSocket(
      this.getEndpoint(this.id)
    );
    this.webSocket.onmessage = (msg) => {
      const data = JSON.parse(msg.data);
      this.price$.next(data[this.id]);
    };
  }
  private getEndpoint(id: string) {
    return PricerComponent.Endpoint + '?assets=' + id;
  }
}
```



Why Memory Leaks are tricky

- we receive an ID as input, and we subscribe to it via WebSocket every time the component is initialized
- when is the WebSocket connection cleared? Never! Once again, we're missing the good gold `ngOnDestroy` hook to take care of it when the component gets destroyed
- This creates multiple issues: not only we're creating multiple WebSocket connections, but we're also retaining PricerComponent in memory every time it gets re-initialized



Why Memory Leaks are tricky

```
const onPriceReceived = (msg: MessageEvent) => {
  const data = JSON.parse(msg.data);
  this.price$.next(data[id]);
};

this.webSocket.onmessage = onPriceReceived;
```

This bug is easily solved by adding the following method:

```
ngOnDestroy() {
  this.webSocket.close();
}
```



Workspace Configuration – Angular.json

- `version`: The configuration-file version.
- `newProjectRoot`: Path where new projects are created. Absolute or relative to the workspace folder.
- `defaultProject`: Default project name to use in commands, where not provided as an argument. When you use `ng new` to create a new app in a new workspace, that app is the default project for the workspace until you change it here.
- `schematics` : A set of schematics that customize the `ng generate` sub-command option defaults for this workspace. See Generation schematics below.
- `projects` : Contains a subsection for each project (library or application) in the workspace, with the per-project configuration options.



Strict Mode

- When you create a new workspace or a project you have an option to create them in a strict mode using the `--strict` flag.
- Enabling this flag initializes your new workspace or project with a few new settings that improve maintainability.
- It helps you to catch bugs ahead of time, and allow the CLI to perform advanced optimizations on your application.
- Additionally, applications that use these stricter settings are easier to statically analyze, which can help the `ng update` command refactor code more safely and precisely when you are updating to future versions of Angular.



Strict Mode

- Specifically, the strict flag does the following:
 - Enables strict mode in TypeScript, as well as other strictness flags recommended by the TypeScript team.
 - Specifically, forceConsistentCasingInFileNames, noImplicitReturns, noFallthroughCasesInSwitch.
 - Turns on strict Angular compiler flags strictTemplates and strictInjectionParameters
 - Bundle size budgets have been reduced by ~75%
 - Turns on no-any tslint rule to prevent declarations of type any
 - Marks your application as side-effect free to enable more advanced tree-shaking
 - `ng new [project-name] --strict`



Project Configuration Options

```
"my-app": {  
  "root": "",  
  "sourceRoot": "src",  
  "projectType": "application",  
  "prefix": "app",  
  "schematics": {},  
  "architect": {}  
}
```



Project tool configuration options

- Default Architect builders and targets
- Angular defines default builders for use with specific CLI commands, or with the general `ng run` command.
- The JSON schemas that define the options and defaults for each of these default builders are collected in the `@angular-devkit/build-angular` package.
- The schemas configure options for the following builders.
 - `app-shell`
 - `browser`
 - `dev-server`
 - `extract-i18n`
 - `karma`
 - `protractor`
 - `server`
 - `tslint`



Configuring builder targets

- Each target object specifies the builder for that target, which is the npm package for the tool that Architect runs.
- In addition, each target has an options section that configures default options for the target, and a configurations section that names and specifies alternative configurations for the target.



Configuring builder targets

```
"architect": {  
  "build": { },  
  "serve": { },  
  "e2e": { },  
  "test": { },  
  "lint": { },  
  "extract-i18n": { },  
  "server": { },  
  "app-shell": { }  
}
```



Builder targets

| PROPERTY | DESCRIPTION |
|----------------|---|
| builder | The npm package for the build tool used to create this target. The default builder for an application (<code>ng build myApp</code>) is <code>@angular-devkit/build-angular:browser</code> , which uses the webpack package bundler. Note that a different builder is used for building a library (<code>ng build myLib</code>). |
| options | This section contains default build target options, used when no named alternative configuration is specified. See Default build targets below. |
| configurations | This section defines and names alternative configurations for different intended destinations. It contains a section for each named configuration, which sets the default options for that intended environment. See Alternate build configurations below. |



Complex configuration values

- Assets configuration
- Styles and scripts configuration
- Style preprocessor options
 - // src/app/app.component.scss
 - // A relative path works
 - @import '../style-paths/variables';
 - // But now this works as well
 - @import 'variables';



Build Configuration

- `ng build --configuration <config>`



Build Configuration

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  constructor() {
    console.log(environment.production); // Logs false for development environment
  }

  title = 'app works!';
}
```





Build Configuration

```
import { Injectable, Inject } from '@angular/core';
import { Environment } from '../environments/ienvironment'
import { ENV } from '../environments/environment.provider'

@Injectable()
export class UserService {

  constructor(@Inject(ENV) private env: Environment) {
    }

  save(user: User): Observable<User> {
    if (this.env.production) {
      ...
    } else {
      ...
    }
  }
}
```



Build Configuration

```
import { ENV } from '../environments/environment.provider'

describe('UserService', () => {
  describe('when in production', () => {
    beforeEach(() => {
      const env = {production: true, ...}
      // without TestBed
      const service = new UserService(env)
      // or with TestBed
      TestBed.configureTestingModule({
        providers: [
          {provide: ENV, useValue: env}
        ]
      });
    });
  });
});
```

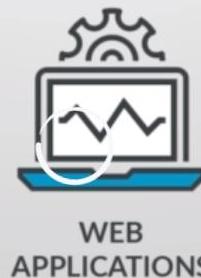


Automation Testing

Develop



Deploy



Deliver





Automation Testing



Catch **Bugs** before your application steps into production



WEB APPLICATIONS

build / lint

test

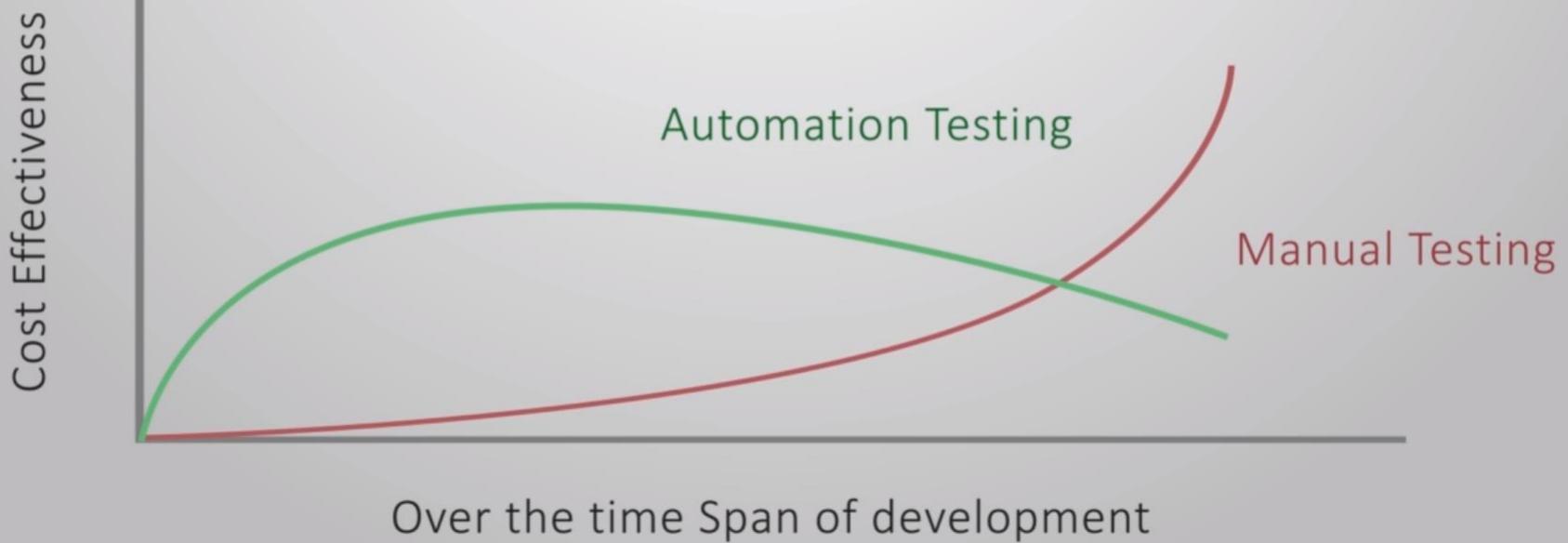
deploy





Automation Testing

Reduced Business Expenses & Time





Automation Testing

Write More **Code** . . .



Application Code

Application Code and
Testing Code



Automation Testing

START
UP!





Automation Testing

Lets Get Started





Jasmine Unit Testing

- Jasmine is one of the popular JavaScript unit testing frameworks which is capable of testing synchronous and asynchronous JavaScript code.
- It is used in BDD (behavior-driven development) programming which focuses more on the business value than on the technical details



Suite

- A Jasmine suite is a group of test cases that can be used to test a specific behavior of the JavaScript code (a JavaScript object or function).
- This begins with a call to the Jasmine global function describe with two parameters – first parameter represents the title of the test suite and second parameter represents a function that implements the test suite.
- //This is test suite
- describe("Test Suite", function() {
- //.....
- });



Spec

- A Jasmine spec represents a test case inside the test suite.
- This begins with a call to the Jasmine global function it with two parameters – first parameter represents the title of the spec and second parameter represents a function that implements the test case.
- In practice, spec contains one or more expectations.
- Each expectation represents an assertion that can be either true or false.
- In order to pass the spec, all of the expectations inside the spec have to be true. If one or more expectations inside a spec is false, the spec fails.



Spec

- //This is test suite
- describe("Test Suite", function() {
- it("test spec", function() {
- expect(expression).toEqual(true);
- });
- });



Spec

File Edit View Selection Find Packages Help

Project

simple calculator

- lib
- calculator.js
- calculator.spec.js**
- main.js
- simple-calculator.html
- spec-runner.html
- style.css

calculator.spec.js

```
1 describe('calculator.js', function() {
2     it('should add numbers to total', function() {
3         // TODO: Expectations
4     });
5
6     it('should subtract numbers from total', function() {
7         // TODO: Expectation
8     );
9
10    it('should multiply total by number', function() {
11        // TODO: Expectation
12    );
13
14    it('should divide total by number', function() {
15        // TODO: Expectation
16    );
17 });
18
```



Spec

```
simple calculator
  lib
    calculator.js
  calculator.spec.js
  main.js
  simple-calculator.html
  spec-runner.html
  style.css

13      expect(calculator.total).toBe(25);
14    });
15  );
16
17  it('should multiply total by number', function() {
18    const calculator = new Calculator();
19    calculator.total = 100;
20    calculator.multiply(2);
21
22    expect(calculator.total).toBe(200);
23  );
24
25  it('should divide total by number', function() {
26    const calculator = new Calculator();
27    calculator.total = 200;
28    calculator.divide(2);
29
30    expect(calculator.total).toBe(100);|
31  );
```



Jasmine functions

- Suite (describe):
Groups related specs.

- Spec (it):
Groups expectations.

- Expectations (expect):
Test the state of the code.

```
describe('test.js', function() {  
  it('Should be true', () => {  
    expect(true).toBe(true);  
  });  
});
```

Setup and Teardown



- For setup and tear down purpose, Jasmine provides two global functions at suite level i.e. `beforeEach()` and `afterEach()`.
- `beforeEach()`
- The `beforeEach` function is called once before each spec in the `describe()` in which it is called.
- `afterEach()`
- The `afterEach` function is called once after each spec.



What is a matcher?

- * it is just a function
- * Implements a boolean comparison between the actual value and the expected value.

```
expect(/*Actual value*/calculator.total)
  .toBe(/*expected value*/5);
```

-> boolean comparison -> true / false
-> actual value with the expected value
(calculator.total === 5) -> true / false

- * A matcher is responsible for reporting to jasmine if the expectation is true or false. -> passing or failing spec.

Jasmine Matchers



| MATCHER | PURPOSE |
|-----------------|---|
| toBe() | passed if the actual value is of the same type and value as that of the expected value. It compares with === operator |
| toEqual() | works for simple literals and variables; should work for objects too |
| toMatch() | to check whether a value matches a string or a regular expression |
| toBeDefined() | to ensure that a property or a value is defined |
| toBeUndefined() | to ensure that a property or a value is undefined |

Jasmine Matchers



| | |
|-------------------|---|
| toBeNull() | to ensure that a property or a value is null. |
| toBeTruthy() | to ensure that a property or a value is true |
| toBeFalsy() | to ensure that a property or a value is false |
| toContain() | to check whether a string or array contains a substring or an item. |
| toBeLessThan() | for mathematical comparisons of less than |
| toBeGreaterThan() | for mathematical comparisons of greater than |
| toBeCloseTo() | for precision math comparison |
| toThrow() | for testing if a function throws an exception |
| toThrowError() | for testing a <i>specific</i> thrown exception |



```
calculator.spec.js
```

```
it('should divide total by number', function() {
  const calculator = new Calculator();
  calculator.total = 200;
  calculator.divide(2);

  expect(calculator.total).toBe(100);
});

// toBe

it('should initialize the total', function() {
  const calculator = new Calculator();

  expect(calculator.total).toBe(0);
});
```



Deep equality comparison:

- * Equal keys and Equal values

```
{ __someKey: 'some value', ... }  
{ someKey: 'some value' ... }
```

=> true



```
expect(calculator.total).toBe(100);
});

it('should initialize the total', function() {
  const calculator = new Calculator();

  expect(calculator.total).toBe(0);
}

it('has constructor', function() {
  const calculator = new Calculator();
  const calculator2 = new Calculator();

  expect(calculator).toEqual(calculator2);
});
});
```

Disable Suites and Specs



```
1 Disabled spec: A spec that will report as pending  
2 and will not be executed.  
3  
4 2 Common cases!  
5  
6 1. Changes in the code.  
7 2. TDD -> Test driven development.
```



Jasmine matchers

- `toBe()`:

Expects the actual value to be equal
`(==)` to the expected value.

- `toEqual()`:

Similar to the `toBe` method but it uses
deep equality comparison.

- `toBeFalsy()`:

Expects the actual value to be falsy.

```
describe('test.js', function() {
  it('is true', function() {
    expect(true).toBe(true);
  });

  it('has same keys and values', function() {
    expect(obj1).toEqual(obj2);
  });

  it('is falsy', function() {
    expect(0).toBeFalsy();
  });
});
```



Jasmine matchers

- `toBeTruthy()`:

Expects the actual value to be truthy.

- `toBeNull()`:

Expects the actual value to be null.

- `toBeDefined()`:

Expects the actual value to be defined.

```
describe('test.js', function() {  
  it('is truthy', function() {  
    expect({}).toBeTruthy();  
  });  
  
  it('is null', function() {  
    expect(result).toBeNull();  
  });  
  
  it('is defined', function() {  
    expect(result).toBeDefined();  
  });  
});
```



Jasmine matchers

- `toThrow()`:

Expects a function to throw something.

- `toThrowError()`:

Expects a function to throw an error.

- `toMatch()`:

Expects the actual value to match a regular expression.

```
describe('test.js', function() {
  it('throws', function() {
    const err = function(message) {
      throw new Error(message);
    };
    expect(err).toThrow();
  });

  it('throws an error', function() {
    const err = function(message) {
      throw new Error();
    };
    expect(err).toThrowError();
  });

  it('matches regex', function() {
    const STR = 'my string';
    expect(STR).toMatch(/string$/);
  });
});
```



Jasmine functions

- Disabled suite (`xdescribe`):
Any specs inside the disabled suite
are skipped.
- Pending spec (`xit`):
A pending spec does not run, but it
will show up in the results as pending.

```
xdescribe('test.js', function() {  
 xit('Should be true', () => {  
    expect(true).toBe(true)  
  });  
});
```

Disable Suites and Specs



- //disable suite
- xdescribe("MathUtils", function() {
-
- });
- describe("MathUtils", function() {
- //Spec for sum operation
- //disable spec
- xit("should be able to calculate the sum of two numbers",
function() {
- expect(10).toBeSumOf(7, 3);
- });
- });

Working with Jasmine Spies



- Jasmine has test double functions called spies.
- A spy can stub any function and tracks calls to it and all arguments.
- A spy only exists in the describe or it block in which it is defined, and will be removed after each spec.
- To create a spy on any method, use `spyOn(object, 'methodName')` call.
- There are two matchers `toHaveBeenCalled` and `toHaveBeenCalledWith` which should be used with spies.

Working with Jasmine Spies



- `toHaveBeenCalled` matcher will return true if the spy was called; and `toHaveBeenCalledWith` matcher will return true if the argument list matches any of the recorded calls to the spy.

Working with Jasmine Spies



| TRACKING PROPERTY | PURPOSE |
|-----------------------|--|
| .calls.any() | returns false if the spy has not been called at all, and then true once at least one call happens. |
| .calls.count() | returns the number of times the spy was called |
| .calls.argsFor(index) | returns the arguments passed to call number index |
| .calls.allArgs() | returns the arguments to all calls |
| .calls.all() | returns the context (the this) and arguments passed all calls |
| .calls.mostRecent() | returns the context (the this) and arguments for the most recent call |
| .calls.first() | returns the context (the this) and arguments for the first call |
| .calls.reset() | clears all tracking for a spy |



Organizing your specs

- `beforeAll()`:

Runs **once** before all of the specs in the `describe` are run.

- `afterAll()`:

Runs **once** after all of the specs in the `describe` are run.

```
describe('test.js', function() {
  beforeEach(function() {
    this.boolean = true;
  });

  afterEach(function() {
    this.boolean = false;
  });

  describe('someMethod()', function() {
    it('is true', function() {
      expect(this.boolean).toBe(true);
    });
  });
});
```



Jasmine organizing your specs

- `beforeEach()`:

Runs before each of the specs in the `describe` in which it is called.

- `afterEach()`:

Runs after each of the specs in the `describe` in which it is called.

```
describe('test.js', function() {  
  describe('someMethod()', function() {  
    beforeEach(function() {  
      this.boolean = true;  
    });  
  
    afterEach(function() {  
      this.boolean = false;  
    });  
  
    it('is true', function() {  
      expect(this.boolean).toBe(true);  
    });  
  })  
});
```



Angular Test bed

- . The TestBed creates a dynamically-constructed Angular test module that emulates an Angular @NgModule
- The TestBed.configureTestingModule() method takes a metadata object that can have most of the properties of an @NgModule.

ComponentFixture



- The ComponentFixture is a test harness for interacting with the created component and its corresponding element.
- Access the component instance through the fixture and confirm it exists with a Jasmine expectation:
 - content_copy
 - const component = fixture.componentInstance;
 - expect(component).toBeDefined();



| Properties | Description |
|---------------------|---|
| componentInstance | The instance of the component class created by TestBed.createComponent. |
| <u>debugElement</u> | <p>The DebugElement associated with the root element of the component.</p> <p>The debugElement provides insight into the component and its DOM element during test and debugging. It's a critical property for testers. The most interesting members are covered below.</p> |
| nativeElement | The native DOM element at the root of the component. |
| changeDetectorRef | <p>The ChangeDetectorRef for the component.</p> <p>The ChangeDetectorRef is most valuable when testing a component that has the ChangeDetectionStrategy.OnPush method or the component's change detection is under your programmatic control.</p> |



| Methods | Description |
|-------------------|---|
| detectChanges | <p>Trigger a change detection cycle for the component.</p> <p>Call it to initialize the component (it calls <code>ngOnInit</code>) and after your test code, change the component's data bound property values. Angular can't see that you've changed <code>personComponent.name</code> and won't update the name binding until you call <code>detectChanges</code>.</p> <p>Runs <code>checkNoChanges</code> afterwards to confirm that there are no circular updates unless called as <code>detectChanges(false)</code>;</p> |
| autoDetectChanges | <p>Set this to true when you want the fixture to detect changes automatically.</p> <p>When <code>autodetect</code> is true, the test fixture calls <code>detectChanges</code> immediately after creating the component. Then it listens for pertinent zone events and calls <code>detectChanges</code> accordingly. When your test code modifies component property values directly, you probably still have to call <code>fixture.detectChanges</code> to trigger data binding updates.</p> <p>The default is false. Testers who prefer fine control over test behavior tend to keep it false.</p> |
| checkNoChanges | Do a change detection run to make sure there are no pending changes. Throws an exception if there are. |
| isStable | If the fixture is currently <i>stable</i> , returns true. If there are async tasks that have not completed, returns false. |
| whenStable | Returns a promise that resolves when the fixture is stable. To resume testing after completion of asynchronous activity or asynchronous change detection, hook that promise. See above . |
| destroy | Trigger component destruction. |



Code Coverage

- Ng test –code-coverage



Fragile Test

- `Expect(result).toContain(expectedvalue)`



TestBed with Services

- let service: ValueService;
- beforeEach(() => {
- TestBed.configureTestingModule({ providers: [ValueService] });
- });



TestBed with Services

- `beforeEach(() => {`
- `TestBed.configureTestingModule({ providers: [ValueService] });`
- `service = TestBed.get(ValueService);`
- `});`



Spec Parallel Execution

- capabilities: {
- browserName: 'chrome',
- 'shardTestFiles': true,
- 'maxInstances': 2
- },
-



Spec Parallel Execution

```
F:\aspireangular2020\mvpapp>ng e2e --devServerTarget="" --webdriver-update=false --host=localhost --port=5000  
[02:24:32] I/launcher - Running 2 instances of WebDriver  
  
DevTools listening on ws://127.0.0.1:58793/devtools/browser/7acf24ec-3414-40e7-839a-a4be87776ee7  
  
DevTools listening on ws://127.0.0.1:58799/devtools/browser/20d8948c-689d-43df-a96b-61bcc1d7e35a  
. [02:25:20] I/testLogger -  
-----  
  
[02:25:20] I/testLogger - [chrome #01-0] PID: 25644  
[chrome #01-0] Specs: F:\aspireangular2020\mvpapp\e2e\src\app.e2e-spec.ts  
[chrome #01-0]  
[chrome #01-0] [02:24:33] I/direct - Using ChromeDriver directly...  
[chrome #01-0] Jasmine started  
[chrome #01-0] Jasmine started  
[chrome #01-0] {  
[chrome #01-0]   baseUrl: 'http://localhost:5000',  
[chrome #01-0]   default: { baseUrl: 'http://localhost:5000' }  
[chrome #01-0] }  
[chrome #01-0]  
[chrome #01-0] workspace-project App  
[chrome #01-0]   ✓ should display welcome message  
[chrome #01-0]  
[chrome #01-0] workspace-project App  
[chrome #01-0]   ✓ should display welcome message  
[chrome #01-0]
```



HTML Report

Protractor Test Report

Passed: 2 **Time Elapsed: 0h 0min 8s**

1/31/2020, 11:23:00 PM

2 TestSuites

100%

Passed

2 TestCases

100%

Passed

► Environment

Browser: chrome 79.0.3945.130 Platform: Windows

TestSuite: workspace-project App 1

TestCase: should display welcome message Passed

Time elapsed: 4 seconds



Skip current project deployment with ng e2e

```
ng e2e --devServerTarget="" --webdriver-update=false --host=localhost --port=5000
```

```
F:\aspireangular2020\mvpapp>ng e2e --devServerTarget="" --webdriver-update=false --host=localhost --port=5000
[20:57:17] I/launcher - Running 1 instances of WebDriver
[20:57:18] I/direct - Using ChromeDriver directly...
```

```
DevTools listening on ws://127.0.0.1:50559/devtools/browser/d8c00351-5d76-4941-a046-45e67fc910d8
```

```
Jasmine started
```

```
{
  baseUrl: 'http://localhost:5000',
  default: { baseUrl: 'http://localhost:5000' }
}
```

```
workspace-project App
```

```
  ✓ should display welcome message
```

```
workspace-project App
```

```
  ✓ it should navigate after successful login
```

```
Executed 2 of 2 specs SUCCESS in 12 secs.
```

```
[20:58:40] I/launcher - 0 instance(s) of WebDriver still running
```

```
[20:58:40] I/launcher - chrome #01 passed
```

```
F:\aspireangular2020\mvpapp>
```



Apollo Client

- npm install --save apollo-angular \
- apollo-angular-link-http \
- apollo-link \
- apollo-client \
- apollo-cache-inmemory \
- graphql-tag \
- graphql

Questions

