PART 1: WHY ANGULAR IS AWESOME

Angular is the best development platform for building mobile and desktop web applications

Angular is Fast

Code generation
Angular turns your templates into code that's highly optimized for today's JavaScript virtual machines, giving you all the benefits of hand-written code with the productivity of a framework.

Universal
Serve the first view of your application on node.js, .NET, PHP and other servers for near-instant rendering in just HTML and CSS. Also paves the way for sites that optimize for SEO.

Code Splitting
Angular apps load quickly with the new Component Router which delivers automatic code-splitting so users only load code required to render the view they request.

Angular is Cross-Platform

Progressive Web Apps
Use modern web platform capabilities to deliver app-like experiences. High performance, offline and zero-step installation.

Native
Build native mobile apps with strategies from Ionic Framework, NativeScript and React Native.

Desktop
Create desktop-installed apps across Mac, Windows and Linux using the same Angular methods you've learned for the web, plus the ability to access native OS APIs.

PART 1: WHY ANGULAR IS AWESOME

Be more productive

Faster Development Lifecycle

Templates
Quickly create UI views with simple and powerful template syntax.

Angular CLI
Command line tools: start building fast, add components and tests, then instantly deploy.

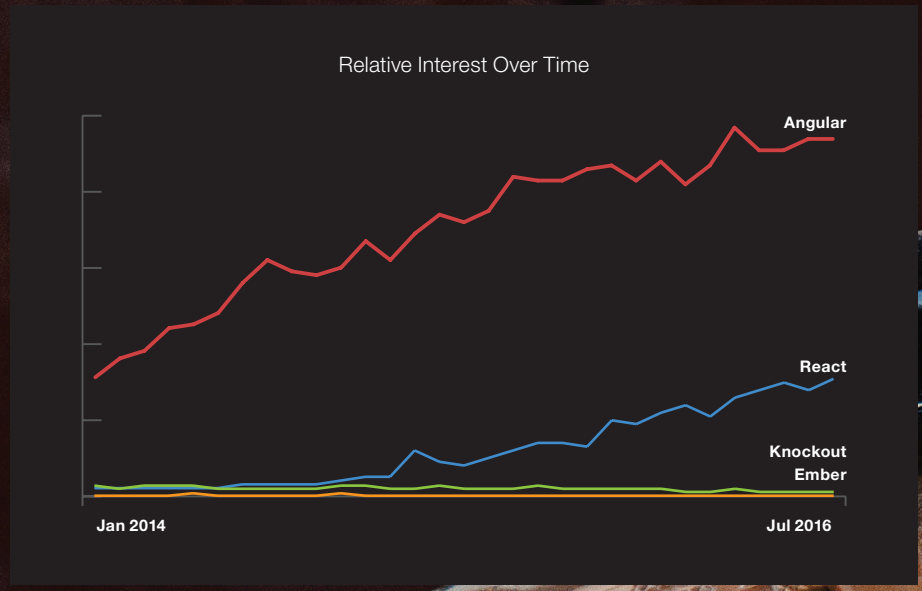
IDEs
With TypeScript, you get intelligent code completion, instant errors and other feedback in popular editors and IDEs.

Testing
With Karma for unit tests, you can know if you've broken things every time you save. And Protractor makes your scenario tests run faster and stably.

Animation
Create high performance complex choreographies and animation timelines with very little code through Angular's intuitive API.

Accessibility
Create accessible applications with ARIA-enabled components, developer guides, and built-in a11y test infrastructure.

Angular 2 is going to be huge!



Google Trends - Web Search interest Worldwide, 2014 - present

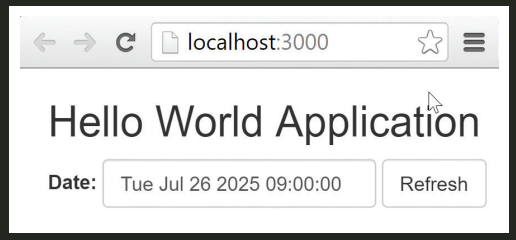
PART 2: FOR THE DEVS

hello-world.component.ts

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

@Component({
    selector: 'hello-world',
    template:
    `
    <h2>{{appTitle}}</h2>
    <label>Date:</label>
    <input [(ngModel)]="currentDateTime">
    <button (click)="setCurrentTime()">Refresh</button>
    `
})
export class HelloWorldComponent {
    appTitle: string = 'Hello World Application';
    currentDateTime: Date = new Date();

    setCurrentTime() {
        this.currentDateTime = new Date();
    }
}
```



‘Hello World’ Explained

This code sample shows a simple ‘Hello World’ component with databinding, component properties and functions

1. Import the Angular 2 core file so that our component code can have access to the @Component decorator.
2. @Component decorator adds metadata that tells Angular how to create and use this component.
3. The selector specifies a simple CSS selector for an HTML element that represents the component.
4. Component’s companion template, written in an enhanced form of HTML that tells Angular how to render this component’s view. Templates can also be stored in external HTML files.
5. Double-curly braces, {{ }}, used to bind values onto the page.
6. [(ngModel)] two-way data binding syntax to both display a data property and update that property on the component model.
7. Event binding syntax consists of a target event within parentheses on the left of an equal sign, and a quoted template statement on the right that often refers to a function on the component class.
8. An Angular class responsible for exposing data to a view and handling most of the view’s display and user-interaction logic.
9. Set public appTitle property as type ‘string’ and initialize its value as ‘Hello World Application’.
10. setCurrentTime( ) is a JavaScript function using the ES6 shorthand syntax.

PART 2: FOR THE DEVS

'Todo' App component tree

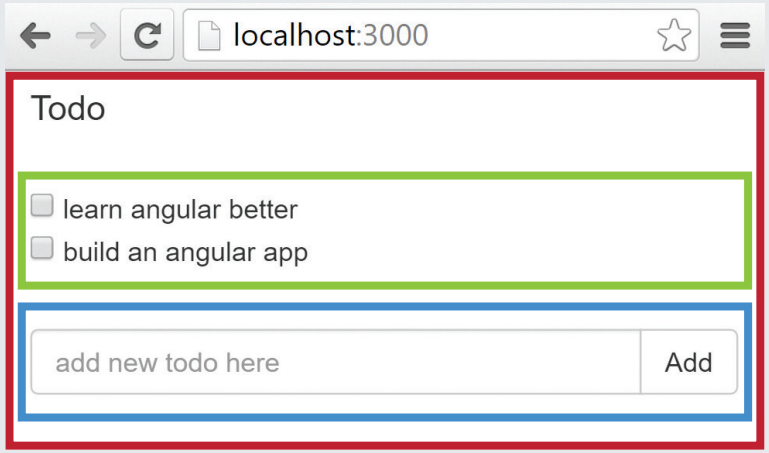
Let's move beyond Hello World and introduce something closer to the real world. We'll begin working on a new Angular 2 component that supports a Todo list.

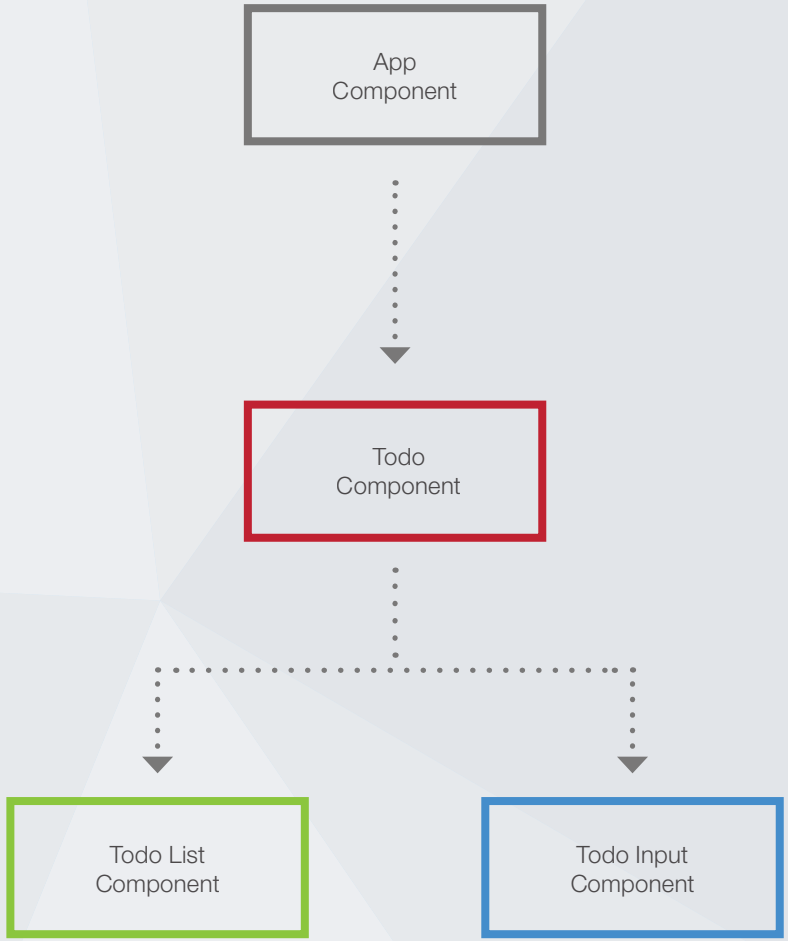
An Angular 2 application is a tree of components.

The top level component contains several child components, and each of those in turn is a parent to their child components. The App Component is the parent of the Todo Component, which is the parent of the Todo List Component and the Todo Input Component.

The benefits of the component tree:

- Composability
- Fast change detection strategies
- Component communication between parent and child





PART 2: FOR THE DEVS

todo.component.ts

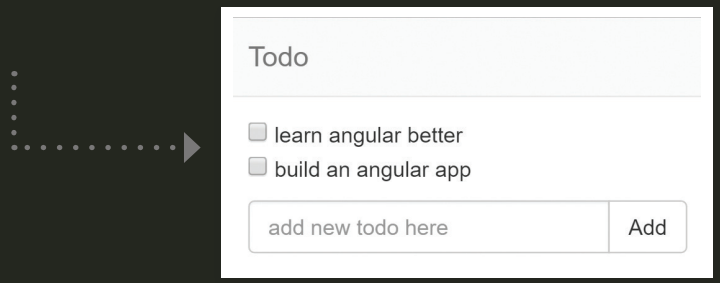
```
import {Component, OnInit} from '@angular/core';
import {Todo} from './todo.model';

@Component({
  selector: 'app-todo',
  template:
  `
  <div>
    <h2>Todo</h2>
    <todo-list [todos]="todos"></todo-list>
    <todo-form (newTodoAdded)="addTodo($event)"></todo-form>
  </div>
  `,
})
export class TodoComponent implements OnInit {
  todos: Todo[];

  constructor(private _todoService: TodoService) { }

  ngOnInit() {
    this.todos = this._todoService.get()
  }

  addTodo(todo: Todo) {
    this._todoService.push(todo);
  }
}
```



‘Todo’ App Explained

This code sample shows a typical component with injected dependencies and nested components.

1. Import Component, OnInit, Todo interface, TodoListComponent, TodoFormComponent and the TodoService to be available in this file.
2. @Component is a decorator that allows us to associate metadata with the component class. The metadata tells Angular how to create and use this component.
3. [ ] square bracket property binding syntax on the left side of the equals sign represent an input of a property into a component/ directive.
4. ( ) parenthesis on the left side of the equals sign represent an output of an event from a component/directive.
5. Class constructor defining a component’s injected dependencies. Angular has its own dependency injection framework. We utilize TypeScript’s constructor syntax for declaring parameters and properties simultaneously.
6. Components have a lifecycle managed by Angular as it creates, updates and destroys them. Here we use ngOnInit to run state initialization logic.
7. addTodo(todo: Todo) is a JavaScript function using the ES6 shorthand syntax.

PART 2: FOR THE DEVS

Module Loader

Angular 2 embraces the new ECMA Script 6 module loading syntax, allowing for better optimized bundles of code to be shipped to the browser.

A module is simply a JavaScript file written with module syntax. Modules export values, which can then be imported by other modules.

A module loader provides the ability to dynamically load modules, and also keeps track of all loaded modules in a module registry.

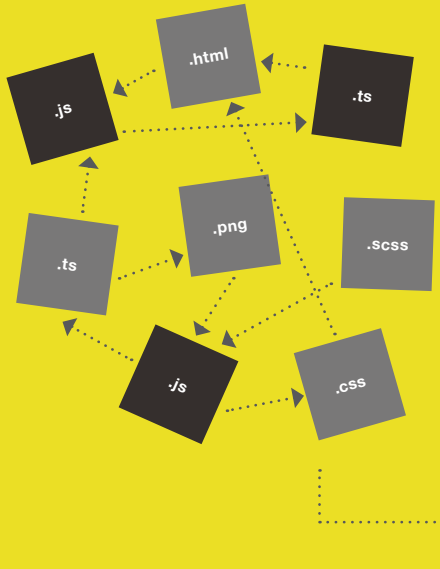
math.ts

```
export function sum (x, y) {
  return x + y;
}
```

calculator.ts

```
import {sum} from './math';

sum(1, 2);
```



Typically, in production, the module registry would be populated by an initial compiled bundle of modules, which is what module bundlers like webpack and SystemJS/JSPM can do.

Later in the page state, it may become necessary to dynamically load a new module. This module can then share dependencies with the initial page bundle without having to reload any dependencies.

Module Bundler

JS ES6 MODULES

TypeScript

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. Any browser. Any host. Any OS. Open source.

TypeScript is the default language for Angular 2, although you can use JavaScript or Dart. TypeScript is not only the language of choice but most third party libraries, docs, blog posts and the actual Angular 2 code base is written in TypeScript.

- **Compile Time Checking**
  Errors in TypeScript are shown in the IDE as you write your code. The TypeScript compiler will fail with detailed error messages without having to leave your IDE.

- **Great Tooling**
  TypeScript's type system allows for advanced autocompletion, navigation, and refactoring. These are almost a requirement for large projects to reduce fear of refactoring a large code base.

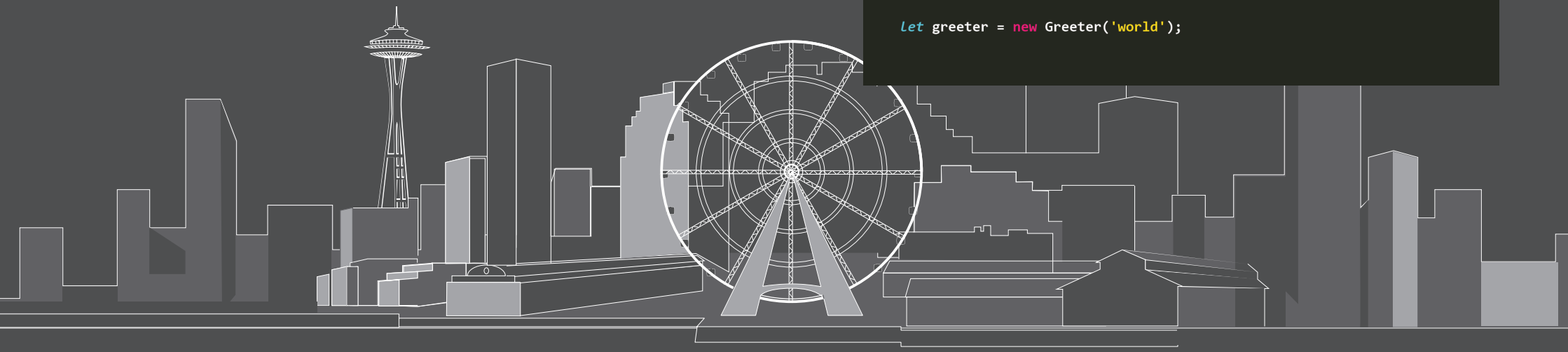
- **Easier Code To Read**
  With interfaces, typed function parameters and return types, TypeScript is much more explicit code, making it easy to understand.

- **Built In Compiler**
  If you use the latest ES6 JavaScript language features you need to compile your JavaScript to ES5 browser compatible code. TypeScript's compiler by default can compile down to ES3, ES5 or ES6.

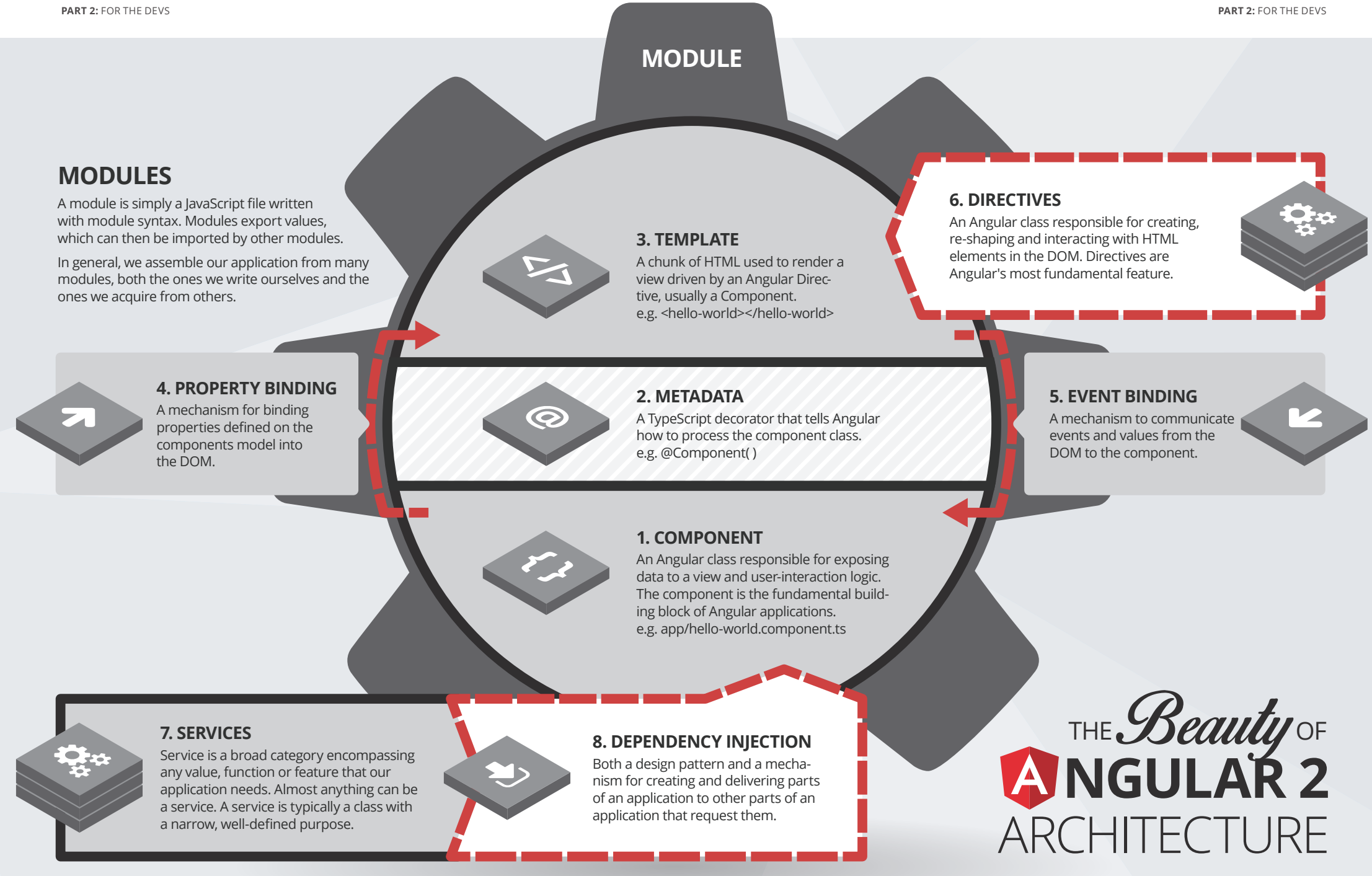
greeter.ts

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet() {
    return 'Hello, ' + this.greeting;
  }
}

let greeter = new Greeter('world');
```



PART 2: FOR THE DEVS



THE *Beauty* OF ANGULAR 2 ARCHITECTURE

PART 2: FOR THE DEVS

RxJS is an essential ingredient of Angular 2 and simplifies asynchronous code

typeahead.component.ts

```
this.albumsStream = this.searchFieldText.valueChanges
    .debounceTime(500)
    .distinctUntilChanged()
    .switchMap((searchText: string) => albumService.search(searchText));
```

RxJS

ReactiveX combines the Observer pattern with the Iterator pattern and functional programming with collections to fill the need for an ideal way of managing sequences of events.

RxJS has over one hundred operators and is why some people think of RxJS as Lodash for events. This four line typeahead example would take more lines of code, be less expressive and less performant without RxJS.

Operators used:

- **debounce(500)**
  Only emit the next search text field value every 500ms.
- **distinctUntilChanged( )**
  Only emit the next search text field value if it has changed.
- **switchMap( )**
  Map search text field values into a new observable of http search requests.

PART 2: FOR THE DEVS

The redux pattern is a way to implement a predictable state container for JavaScript apps.

Redux

The redux pattern helps you manage and simplify your application state. Redux (big R) is a library and redux (little r) is a design pattern that is completely framework agnostic.

Three principles of the redux pattern

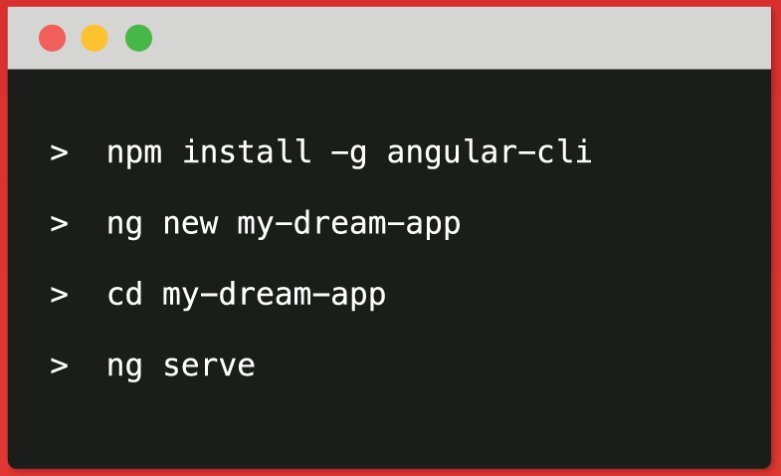
- The entire state of the application is represented in a single JavaScript object called a store.
- The store is acted upon using special functions called reducers.
- State is immutable and reducers are the only part of the application that can change state.

ngrx

ngrx builds on the concepts made popular by Redux, by supercharging it with RxJS.

The result is a tool and philosophy that will revolutionize your applications and development experience.

PART 2: FOR THE DEVS



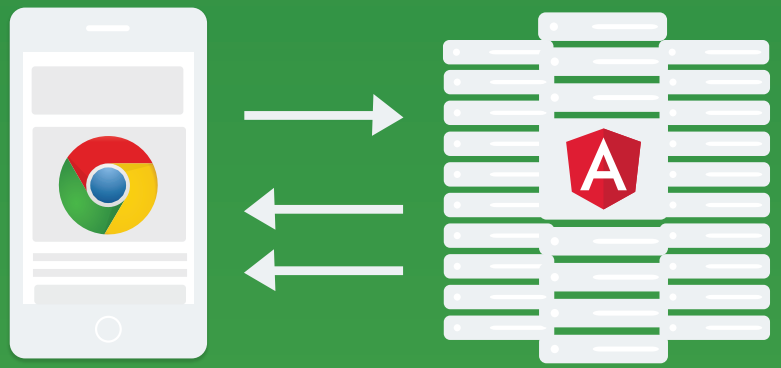
Angular CLI

ng new
The Angular 2 CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

ng generate
Generate components, routes, services and pipes with a simple command. The CLI will also create simple test shells for all of these.

ng serve
Easily put your application in production

Test, Lint, Format
Make your code really shine. Run your unit tests or your end-to-end tests with the breeze of a command. Execute the official Angular 2 linter and run clang-format.



Angular Universal

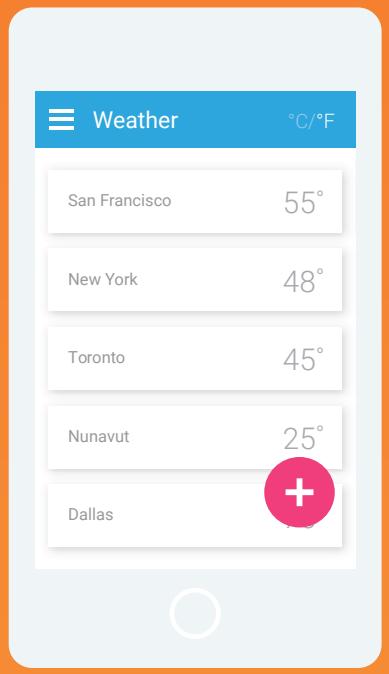
Server-side Rendering for Angular 2 apps

Better Perceived Performance
First time users of your application will instantly see a server rendered view which greatly improves perceived performance and the overall user experience. According to research at Google, the difference of just 200 milliseconds in page load performance has an impact on user behavior.

Optimized for Search Engines
Although Googlebot crawls and renders most dynamic sites, many search engines expect plain HTML. Server-side pre-rendering is a reliable, flexible and efficient way to ensure that all search engines can access your content.

PART 2: FOR THE DEVS

Angular Mobile Toolkit

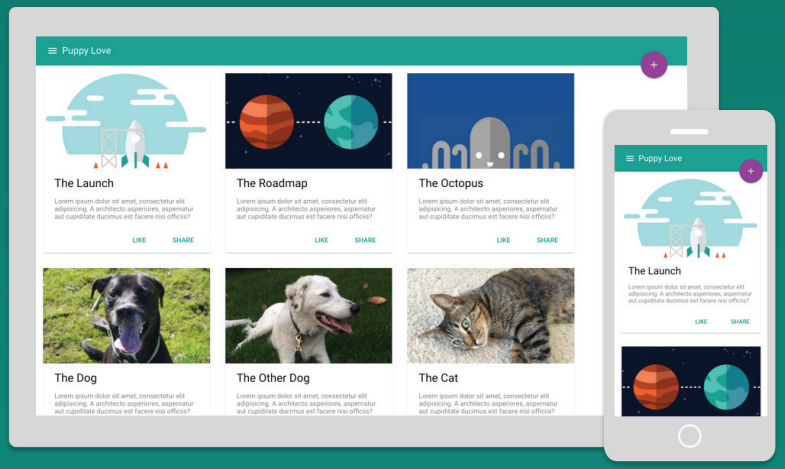


All the tools and techniques to build high performance mobile apps

Performance of native, discoverability of the Web
The Angular Mobile Toolkit makes it easy to build snappy Web apps that load instantly on any device, even without an internet connection. Take advantage of the searchability, shareability and no-install-required-ability of the Web without compromise.

Automatic Progressive Web Apps
Build Progressive Web Apps with ease using Angular CLI and Angular Mobile Toolkit. Plugins and libraries for App Shell, Service Worker and Application Manifest make it easy to create installable, offline-capable, mobile-network-friendly apps in minutes.

One developer, many platforms
A single Web development team is now a cross-platform application powerhouse. Focus more energy on building better experiences, and less energy on making experiences consistent across development teams and device platforms.



Sprint from Zero to App
Hit the ground running with comprehensive, modern UI components that work across web, mobile and desktop.

Fast and Consistent
Finely tuned performance, because every millisecond counts. Fully tested across IE11 and current versions of Chrome, Edge, Firefox and Safari.

Versatile
Themable, for when you need to stay on-brand or just have a favorite color. Accessible and internationalized so that all users are welcome.

Optimized for Angular
Built by the Angular team to integrate seamlessly with Angular 2.

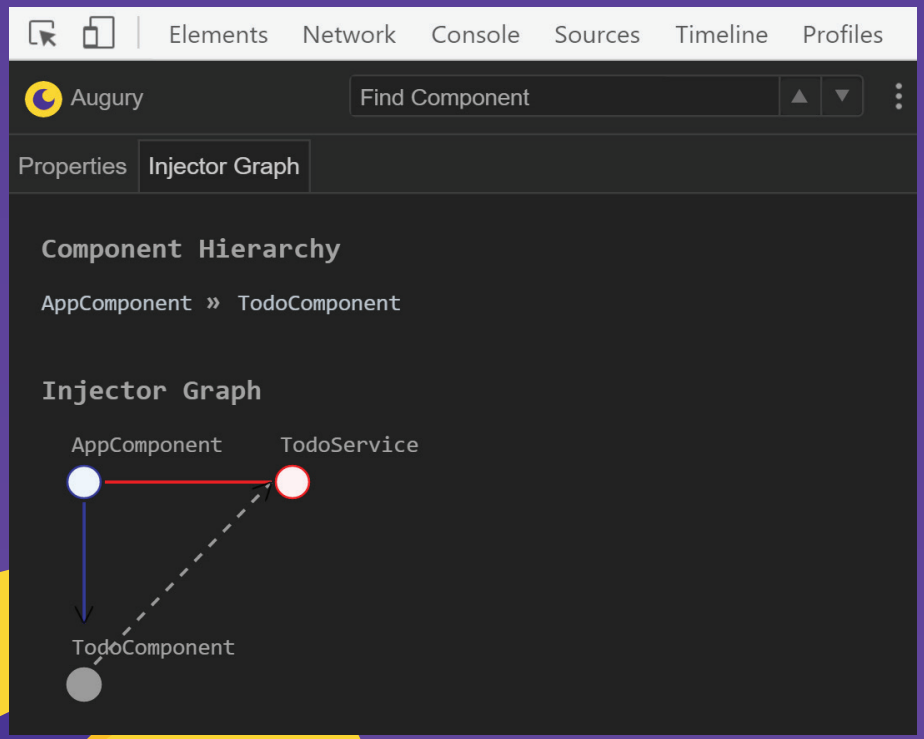
Angular Material

Material Design components for Angular 2 apps

Angular Augury

Visual Interface
Helps Angular 2.0 developers visualize the application through component trees and visual debugging tools.

An example Augury visualization tool is the Injector Graph. The blue line shows the component hierarchy. The dotted line shows injected dependencies. The red line shows registered providers with the Angular injector.



Thank You !!!!!!!