New Technology: Angular 2

# Findings

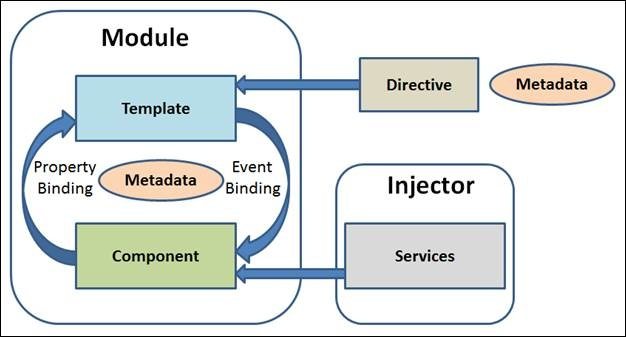
Description

Angular 2 or Ng2, is a front-end open-source framework written in JavaScript more so Typescript, a superset of JavaScript that works with ES6 modules. The framework is often referred to as a MV\* or MVW (Model-View-Whatever) because of its flexibility to separate the logic to present your business logic and application state. Traditionally this was MVC or MVVM, but because of the mixed usages of both structures it was renamed.

Angular 2 is mainly being updated by Google and a community of contributors on GitHub. The latest version of Angular is called Angular 4 which is basically the same as Angular 2 except for some better bug fix alerts, a newer Typescript version and some small syntax changes which increase the performance. Generally it’s entirely the same and irrelevant to mention.

Building the Angular Application

Angular 2 uses Typescript and ES6 standardization and can therefore be split up in a few blocks which you’ll always be able to find in an Ng2 application.



* **Module**
  + A module is basically what we would call a class in programming. It’s basically a modular class which is reusable in the entire application or other applications and is often referred to as modularity in programming. The main module of an Ng 2 app is referred to as NgModule.
* **Template**
  + The template is basically the visualization or display of the values or expressions of the logic and state of an application written in HTML. This is what we would call the V(iew) in our MVW.
* **Component**
  + Everything is a component in Angular 2, components are basically the main way to build and specify elements and logic on the page, through both custom elements and attributes that add functionality to existing components. This is what we would call the W(hatever) in our MVW.
* **Metadata**
  + Metadata is primarily used to extend and define the fuctionality of a class (Component, Service, …) through a decorator such as @Component for components and @Inject for services.
* **Services**
  + Services are designed to fulfill one type of functionality in a module. Usually this is some type of data sharing such connecting to an external or internal API and making these calls accessible throughout the application.
* **Directive**
  + Directives are custom attributes to extend the functionality of HTML and is initialized with a @Directive decorator. A common example of such a directive could be a modal trigger.
* **Data-binding**
  + In Angular 2 we have multiple ways of binding our data. One-way binding/property binding, interpolation, event binding and two-way binding. Angular 2 leans hard towards a unidirectional data flow which means that data can flow from the whole component tree up until the root of the application.
* **Dependency Injection**
  + Injecting dependencies in Angular 2 is done in the constructor of the component. Typical dependency injection would be when you would want to use a service or an angular module in the component, this makes the component easier to test and reuse.

Angular 1 vs Angular 2

Before Angular 2 there was of course Angular 1. But why this new version and what are the key differences between both those versions? The longer time goes on the more everything progresses and the more possibilities there are to increase performance.

1. Plain JS vs Typescript: Angular 1 is written in Plain JavaScript which was starting to get counterproductive seeing how certain new standards and architectures evolved and arose such as Typescript which is used in Angular 2.
2. Data-binding: In Ng2 we still have all the different kinds of data-binding as in Ng1 but with small syntax changes. For interpolation {{vm.priceList.price}}, we can now omit the vm and simply have {{pricelist.price}}; for one-way binding ng-bind is now gone because it was basically doing the same as interpolation and is now replaced by square brackets properties such as <p [style.backgroundColor]=”pricelist.color”></p>; for event binding ng-click has simply become (click) and lastly for two-way binding ng-model has now become [(ngModel)].
3. Controllers have been replaced by components and are based on the ES6 standardization.
4. Every directive, filter or service you need must be imported in Ng2, this wasn’t always the case in Ng1 where a lot of unnecessary and unused code was being loaded in which would severely decrease the performance.

# Sources

<http://www.angular2.com/>

<https://www.pluralsight.com/>

<https://jcoop.io/angular-2-practice-exercises/>

<https://angular.io/>

<https://thinkster.io/tutorials/differences-between-angular-1-and-2>

<https://angular-2-training-book.rangle.io/handout/why_angular_2.html>

<https://johnpapa.net/introducing-angular-modules-routing-module/>

<https://www.themarketingtechnologist.co/introduction-to-data-binding-in-angular-2-versus-angular-1/>

# Log

**30-07**

* Start Pluralsight course: Angular Fundamentals
* Learning about SystemJS Module Loading
* Creating Components/Building Main Module Loader
* Angular CLI (main component, tests, …)
* Data-binding Components
* External Templates
* Parent <-> Child Comms via Input and Output Events
* Template Reference Variables
* CSS Template Styling (Encapsulation)
* Interpolation, Expressions, Property Bindings
* Event Binding/Statements
* ngFor, ngIf, ngSwitch Structural Directive
* Handling Null/Undefined Values in Binding
* Hidden Attribute
* ngClass, ngStyle
* Services with 3rd party

**31-07**

* Continue Pluralsight Angular Fundamentals
* Basic Routing, Routing Params, …
* Route Guarding (canDeactivate, canActivate)
* Lazy Loading Components with Observables
* Styling Active Nav Links
* Lazy Loading Modules
* Setting up Barrels for Clean Import
* Using Models for Type Safety
* Template-based Form, Using Data and Validation
* Reactive Form, Custom Form Validation
* One-way and Two-way Binding

**01-08**

* Passing Data into/out Child Components
* (Multiple) Content Projection
* (Custom) Piping
* Sorting/Filtering Data without Pipes
* Dependency Injection Problems & Advanced
* OpaqueToken to Register 3rd Party APIs and @Inject
* UseClass, UseExisting, UseFactory Providers
* Adding Jquery
* Directives, Modals, ID Binding
* ViewChild Decorator
* Advanced Custom Validators (multi-field)
* Advanced Input/Output Setters
* HTTP, Observables, Rx with CRUD
* Authentication

**06-08**

* Write down findings of Angular 2
* Pointing out key differences between Ng1 and Ng2
* What is MVW/MV\*