Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A Comparative Analysis

Mohamed Sultan
Opencast Software
Email: mohamed.sultan1581@gmail.com

***Abstract*—Recently, numerous new mobile and web-based application frameworks have been released and adopted in both communities of software development industry and research. For example, there are currently the KnockoutJS, BackboneJS and ReactJS frameworks competing together with the different (entirely different) versions of 'Angular' frameworks. While some of these new frameworks are more popular than others, some are specialised in certain types of applications, and others have specific advanced features or outstanding capabilities that set them above others. Moreover, with the increase usage and demand of mobile applications, the need for cross-platform frameworks has significantly increased as well. In this paper, we discuss the different criteria which identifies the strengths and weaknesses of using each framework in developing mobile and web-applications. We highlight and discuss 12 different features of latest application frameworks as 'points of comparison'. Then we compare 5 of the trending frameworks (KnockoutJS, BackboneJS, AngularJS, React and Angular2) in a thorough analysis based on our earlier defined points of comparisons i.e. features. Finally, we focus more deeply on the newly released Angular2 framework showing the eminent capabilities and values added over different trending frameworks and over its own earlier versions. Overall, our comparative analysis results in a few interesting findings regarding different current frameworks, leaving us to believe that a new generation might soon emerge from the exponential path of MVC, MV*/MVW and MVVM.**

Keywords*—*Angular; mobile and web-based applications; front-end; JavaScript frameworks

I. Introduction

Both mobile and web-based application platforms evolve over time to better serve the needs and requirements of the growing business markets and their demands. Mobile and web-based applications consist of two main components: 1) client-side, also known as front-end, which includes the browser and the mobile web app client; and 2) server-side that includes all back-end functionalities (database, validations, authorizations and authentications). In this paper, we focus on the front-end that includes various dependencies affecting the service performance as well as the user satisfaction with the final software solution. Therefore, there is also a growing area of development in terms of new frameworks, platforms and IDE tools, availing different combinations of the basic and essential software development functionalities such as bundling, logging, compiling, packing, debugging and testing.

The challenge is choosing the best framework technology to that fits the desired solution and easily integrates with other systems resulting in the best user experience. Currently, there are several cross-platform client-side frameworks. Most of which include an array of libraries and responsive user interactions, often referred to as 'single-page applications' [1].

For instance, if we consider the simplicity and accessibility of jQuery as a web-development platform, it would be difficult to build and manage a medium to a large-sized solution. Although jQuery is manageable and straightforward to use, it is too low-level as far as application development is concerned. It even gets more complicated and inefficient as scale and complexity of the project increases.

In this paper, we compare five of the latest trending technologies of emerging frameworks for developing mobile and web applications from different aspects. These recent efficient frameworks: Knockout, Backbone, AngularJS, React and Angular (Angular2). Such a comparative study will contribute and assist in directing future development of the wide spreading platforms we are to witness in the near future. Our points of comparison include Data Binding, Templating, Extensibility, Variable Observation, Routing, Testing and other advanced features such as Dependency Injection, Form Validation and Browser Support. We then discuss each framework from these different aspects to analyse its usage, strengths and limitations from different angles versus other similar frameworks.

II. Background

In this section, we will give a review of the current emerging frameworks (Knockout, Backbone, AngularJS, React and Angular2) for web and mobile development to shed light on each of them. Then we pave the way to a thorough comparison between the Angular framework and other frameworks showing the strengths and weaknesses of each. Afterwards, we discuss the key features of Angular2 over AngularJS.

A. Knockout

Knockout (also known as KnockoutJS) was developed by Steve Sanderson and was initially released in July 2010. It is relatively an easy and a quick framework for beginners. However, it is rather difficult to link the system dynamic and personalised output with system data (i.e. data binding) when the model gets more complex. Yet, Knockout has enabled automating about 70% of the manual programming that previously had to be done by developers. Knockout provides a simple and easy to use model bindings between the HTML and the 'model'. But it does not provide guidance nor rules to follow, therefore mainly depending on the developers to structure their code from their point of view. This often leads non-experienced developers to write unstructured, unreadable and non-reusable code because they do not have to consider good application structure when using Knockout. Still, it provides the functionality of listening to events, binding callbacks and DOM changing elements (add, remove or update) based on users input values. Knockout supports most of the major browsers (Internet Explorer 6+, Firefox 2+, Chrome, Opera and Safari). It is an MVVM (Model-View-ViewModel) library for creating dynamic UIs [2].

B. Backbone

Backbone (also known as BackboneJS) was developed by Jeremy Ashkenas and initially released in October 2010 for implementing single-page web applications. Backbone serves as an effective ORM (Object Relational Mapping) for RESTful APIs. It is based on the MVP (Model-View-Presenter) application design paradigm and is a light-weight framework that depends only on the JavaScript Library [3]. BitTorrent.com, Airbnb, Hulu, LinkedIn Mobile, Pinterest and Reddit are built using Backbone. The downside of Backbone is that it requires writing an extensive amount of code that is mostly for integrating components, that are ready-made in Knockout and Angular with a few attributes. This can be considered as a limitation from the development point of view. However, it can be considered as an advantage where there will be no hidden configurations behind the scenes, therefore being more understandable and explicitly coded. Moreover, Backbone is easier in debugging and fixing several performance issues. In general, Backbone is recommended for DOM (Document Object Model i.e. HTML objects) manipulations.

C. AngularJS

As part of a commercial product, AngularJS was first launched in 2009 under the name of 'GetAngluar' which was later sponsored by Google due to its great perceived benefits. For instance, a web application using GetAngular that consists of around 1,000 lines and 3 weeks of development, would take -without GetAngular- up to 17,000 lines and around 6 months of development. Later, Google re-introduced it as the currently known open-source AngularJS [4]. Similar to Knockout, AngularJS leaves HTML readable by supporting data binding into the DOM. In addition, AngularJS supported integrated data validation and introduced an array of controllers (and services) for building single-page applications. AngularJS combined the ideas of Backbone and Knockout and brought custom components via custom directives which made Angular much popular. It also brought common pattern from OOP which is the 'Dependency Injection' container. Moreover, AngularJS introduced the concept of MV* or MVW (Model View Whatever) [5] due to the fact that AngularJS combined the 'Controller' and the 'View' into one single container that handles both functionalities.

However, over time and due to many refactoring and API improvements, its now closer to MVVM (Model View ViewModel) paradigm [6] where the $scope object could be considered as the ViewModel that is being decorated by a function known as a 'Controller'. Angular's modular structure, strict development guidelines and ability to simply bind directly to plain objects, all enhance the efficiency of the coding by preventing many issues, and providing a strong architectural foundation for the application. Therefore, we believe that these reasons have enforced the position of AngularJS as one of the most recent readable and maintainable web developing frameworks. Moreover, AngularJS provides an unpreceded strong community support to respond to users' enquiries, bug detection and problem-solving which offers vast opportunities for bug fixes within next releases.

D. React

React was released by Facebook in March 2013 as a light-weight application platform. However, React should not be considered as a framework [1] but should be rather considered as a library. One of its key features is that it is quite efficient in rendering UI (User Interface). React is often paired with AngularJS to deliver applications of high-speed performance and richer presented content. Plus, React code supports cross-platform with mobile applications, through using extra libraries. Another feature is React's documentation which is useful and rich of information, facilitating coding.

Most of React projects were written in ES2015 which is ahead of browsers supporting it. ES2015 is a pseudonym for the latest version of the JavaScript programming language to be approved by ECMA International, the standards group for vetting and approving different versions of the language. The term ES2015 was given because the latest version of JavaScript is identified as the 2015 version of ECMAScript (an alternative term for JavaScript). Up until recently, ES2015 was previously referred to as ES6.

In comparison with AngularJS, React needs more lines to do what AngularJS can handle in much fewer lines. React has a key feature which is the 'virtual DOM'. The virtual DOM is a light approach for updating DOM with new changes without affecting the performance. This is done by working with a light-weight copy of the original DOM then comparing with the re-rendered what should be changed optimally (using 'dirty checking' or 'observable'). Also, React does not need a browser to test as it does not interact with the real DOM [7].

React uses JSX [8] as the programming language which is an extension to JavaScript that has XML-like syntax. Accordingly, it helps to implement the business logic and functions of a component all in the same file where the HTML template is. This approach makes React components completely isolated and reusable [9]. Since React is a library, it is essential to use a number of more libraries with it to fill the gaps and build a complete application with all necessary operations (such as routing, enforcing unidirectional flows, web API calls, dependency management and testing). For example, Redux [10] can be used to provide a single-state container that all such components in the application can access and manipulate [11].

E. Angular

Released in September 2016, 'Angular2' or 'Angular 2+' is commonly referred to as Angular. Angular is a revolutionary version and a complete rewrite of AngularJS 1.x frameworks. Being much less complicated than Angular1, Angular uses ES2015 and ES2016 in addition to 'Typescript' as a compiler, which supports classes and module loaders. When first launched, Angular was perceived as too revolutionary for its time, where ES6 was close to standardisation and two-way data binding was needed for data over forms applications. Although Angular requires significant effort for setting up the required development project than React, it is worth as it provides a clear understanding of how components interact with each other.

III. Points of Comparison

Before we are able to systematically compare between different trending frameworks we need to define and clarify the points of comparison for our analysis. We have identified 12 different features that qualify and distinguish different frameworks from one another, allowing us to perform correct and efficient selection to the framework that is suitable for developing a certain application. Below we state and simplify each of these features:

- Data Binding: This is the process of connecting the business logic data with the application front-end User Interface (UI). It is important to assure that the proper data is presented to assure the correct service delivery. Also, it should be dynamically updatable to reflect user interaction behaviour outcome. When a UI field variable is updated the content gets updated by the underlying model variable and vice versa.
- Templating: This is used to divide the View into small chunks then display them when needed. Templating supports the code maintainability and readability as well. For example: change the entire 'page' based on user state such as user selected to view details for an item. Or displaying 'items' and 'objects' as a complex HTML structure i.e. widgets.
- Extensibility: This is the ability to extend the current HTML capabilities by embedding tags that represent further functionalities. It is known as directives or custom bindings. It is known as 'directives' or 'custom bindings'.
- Variable Observation: This is used to monitor the changes in any variable properties. Hence, notify the registered action to take the necessary actions.
- Routing: This is a feature to allow storing the application states. States are bounds to views which can easily be configured to allow further features depending on the user's state defined in the business logic.
- Testing: Embedded testing capabilities became a mandatory feature in web and mobile applications. Testing evaluates the system to assure that it satisfies its specified requirement at early stages. Herein, we refer to the evolved and mature sense of embedded testing feature (that can be independently functioning) in different frameworks. In this sense, testing creates scenarios with expected results then compares it with the resultant output from the software.
- Advanced Features: These include Modular Forms, Dependency Injection, Filters, Form Validations, Browser Support and Community Support. Developing an application in a 'Modular Form' facilitates code review, code reuse and code expectation versus best practices. Sometimes, it is asked in interviews for the best practices in solving a certain situation. 'Dependency Injection' (also known as Inversion of Control) is one of the most used design patterns that demonstrates how to create 'loosely coupled' classes [12] where each class is stand-alone exporting only its methods without the need of any other classes to complete its functionality. This is very useful when changing the implementation of a class, where there will be no need to update all other classes that depend on the new one. Moreover, it facilitates testing and liberates the tidiness of classes together. Therefore, we can reuse the independent classes easily in many other functionalities/projects. 'Filters' (also known as pipelines) are used to enable passing the output of an expression as an input to a second function. Accordingly, filtering is a useful development feature in a framework that enables it and is mostly used in formatting. 'Form Validation' used in case of user invalid input, form and controls provide the needed validations to notify the user for his invalid input (e.g. password and email validations). The framework can collect all input control validation into a single object validator to be checked for the validation of the whole form. It can be used to enable a submit button as an example. 'Browser Support' is the ability to integrate with different browsers, give edges to easy Front-End framework over others as it gives the user the flexibility to use the most comfortable browser according to the user needs. 'Community Support' is relatively a newly adopted feature. Frameworks are developed by large-sized organisations like Facebook or Google relying on a huge number of developer base and continues to grow. Therefore, it is very useful and advantageous for developers to use a framework that supports this feature.

IV. The Comparison

In this section, we will use our defined key features of different web development frameworks mentioned earlier in the previous section. Our main points of comparison are Data Binding, Templating, Extensibility, Variable Observation, Routing, Testing and other Advanced Features such as Dependency Injection, Form Validation and Browser Support. We compare different frameworks with respect to each of these different features to analyse their strengths, weaknesses and best usage in developing different types of web and mobile applications.

A. Data Binding

If we have data to be displayed in knockout (as shown in Fig. 1) we need to replace the model properties with ko.observable in order to perform the two-way binding (as shown in Fig. 2).

```
var model = {
    firstName: 'john',
    lastName: 'Doe'
};
```

Fig. 1. Basic model represented in JSON [13] format.

```
var model = {
    firstName: ko.observable('john'),
    lastName: ko.observable('Doe')
};
```

Fig. 2. Example of data binding in Knockout.

The results for HTML with data binding using KnockoutJS is shown in Fig. 3. However, specifying all data properties as observables needs additional effort. In addition, Knockout can only apply bindings once and will through error(s) in case of an attempt to re-apply bindings.

```
<input type="text" data-bind="value: firstName" />
<input type="text" data-bind="value: lastName" />
```

Fig. 3. Example of data binding in Knockout - view implementation.

Backbone can provide data binding functionality using 'Backbone.ModelBinder' plugin. To achieve this, we need to create a model and a view. Then, create the model binder by passing the view and the model into the constructor of the class. If there is a need to define multiple bindings in one action, then 'modelBinder.watch(bindings)' can be used, as shown in Fig. 4. As an example, to achieve the HTML binding, Fig. 5 presents the code to be used, but appears to be somewhat complicated and hard to understand. Likewise, other bindings (text, value, area, select, checked, etc.) can be used in the same approach. Moreover, more options for different types of bindings. To achieve that, it is extensively long and complicated.

```
var view = new Backbone.View(), model = new
Backbone.Model();
var modelBinder = new Backbone.ModelBinder(view, model);
```

Fig. 4. Example of data binding in Backbone - controller side.

```
<output name="html-content"></output>
modelBinder.watch('html: html-content', {
   selector: '[name="html-content"]'
   });
```

Fig. 5. Example of data binding in Backbone - HTML side.

Alternatively, AngularJS provides a simple and an easy two-way binding even for non-developers, where binding expressions are surrounded by double curly braces, as shown in Fig. 6. React also supports Data Binding but different than Angular. Angular puts JS in HTML whereas React puts HTML into JavaScript [14] as shown in Fig. 7. Apparently, it is totally acceptable and valid to implement both cases and indeed depends on personal preference of the developing team/individual.

```
<ul>
    <li *ngFor="let item of items; let i = index">
        {{ i }} {{ item }}
    </li>
</ul>
```

Fig. 6. Example of data binding in AngularJS.

```
let List = function({ items })
return (
          <ul>
           {items.map(item =>
                    <li key= {item.id}> { item.name}</li>
            )}
          </ul>
);
```

Fig. 7. Example of data binding in React.

B. Variable Observation

In fact, nothing happens literally instantaneously. For example, for fetching data from the backend, we need to interrogate an endpoint, wait for a reply, parse it, then render the view with the new data received. For a better UX (User Experience) we may add a loading GIF image while disabling other buttons to show the user that the application is intending to perform an action and that it is not recommended to apply any changes. To accomplish this process we need to monitor variable changes either in the view HTML or in the Controller.

Knockout uses the observable pattern to notify the user if and when they are changed. However, the variable needs to be subscribed first as shown in Fig. 8.

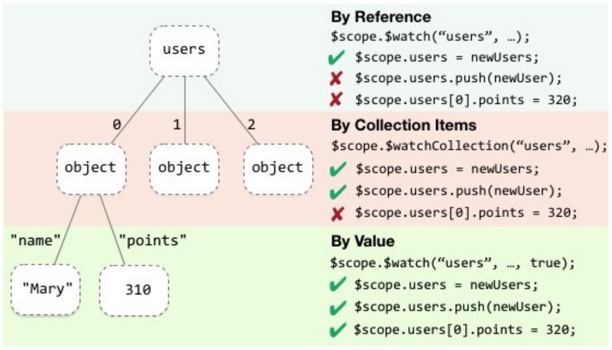
```
var name = ko.observable();
name.subscribe(function(newValue) {
    // Do something when value changes ...
})'
```

Fig. 8. Example of observe in Knockout.

Alternatively, Backbone has the functionality to bind with events (such as: showing a spinning GIF when calling external APIs). This is done by following a number of configuration steps to bind the model to the event and then bind the event with the action function. In order to monitor view changes, Backbone needs to bind with the view on the event 'change' for instance. Then, it is required to add the action (function) to be fired (triggered) on this change. It is quite straightforward and does not need extra third party libraries to take care of this functionality [15].

Since the $scope object presents the translator between the view and AngularJS. The $scope represents the view. All the variables within the scope object can be accessed by the 'View' as well as the 'Controller'. The $scope has its inner function $watch that easily binds to the variables that need to be observed and hence call the appropriate functions. The $watch provides three strategies:

1) Watching by reference: where it detects the changes on the object as a whole. If a value is changed within an array object, it is not detected.
2) Watching collection contents: detects changes that occur inside an array or an object. But, it does not include nested collections. Watching collection contents is more expensive than watching by reference because copies of the collection contents need to be maintained.
3) Watching by value: detects any change in an arbitrarily nested data structure. It is the most powerful change detection strategy, but also the most expensive [16] as shown in Fig. 9.



By Reference
```
$scope.$watch("users", …);
✓ $scope.users = newUsers;
✗ $scope.users.push(newUser);
✗ $scope.users[0].points = 320;
```
By Collection Items
```
$scope.$watchCollection("users", …);
✓ $scope.users = newUsers;
✓ $scope.users.push(newUser);
✗ $scope.users[0].points = 320;
```
By Value
```
$scope.$watch("users", …, true);
✓ $scope.users = newUsers;
✓ $scope.users.push(newUser);
✓ $scope.users[0].points = 320;
```

Fig. 9. Different level of watching an object in AngularJS.

C. Templating

Knockout supports templating which helps to break code/HTML into small pieces. However, Knockout does not support template to be stored into external HTML. On the other hand, Backbone does have the templating functionality. Nevertheless, with the use of JQuery functions, custom templates can be built and used internally within the code [15] as seen in Fig. 10. Backbone can easily integrate with many third-party template engines. The famous one is Underscore template [17] which is one of Backbone dependencies. Accordingly, it can be easily used in the project pages. It helps in adding the templating functionality without the need to add new dependencies. However, Backbone does not support creating templates in a separate file but as a 'work around' we can create a new JavaScript file that includes the template (as a variable) to be referenced within other components (which is not the ideal case) as shown in Fig. 10.

```
var appleHomeTpl = 'Apple data:
    <ul class="apples-list">
    </u1>
<div class="cart-box"></div>'
```

Fig. 10. Example of templating in Backbone.

In AngularJS, templates are written in an HTML format and can include AngularJS elements. AngularJS combines all information provided from controllers, directives, filters then renders them into a dynamic view of the user, as shown in Fig. 11.

```
<html ng-app>
<!-- Body tag augmented with ngController directive -->
<body ng-controller="MyController">
    <input ng-model="foo" value="bar">
    <!-- Button tag with ngClick directive, and
         string expression 'buttonText'
         wrapped in "{{ }}" markup -->
     <button ng-click="changeFoo()">{{buttonText}}</button>
     <script src="angular.js">
  </body>
</html>
```

Fig. 11. Example of data binding and templating in AngularJS.

D. Extensibility

Knockout uses the same keyword as of 'data binding' to extend the HTML DOM elements (data-bind) with extra functionality, which often creates some confusion for code maintainability and readability, as shown in Fig. 12.

```
<div data-bind="myAddress: location"></div>
<script type="text/javascript">
ko.bindingHandler.myAddress = {
update: function (element, valueAccessor) {
var address = ko.unwrap(valueAccessor());
$(element).text(address.street() + ',' + address.city + '' +
address.state());
        }
}
</script>
```

Fig. 12. Example of extendability in Knockout.

On the other hand, AngularJS uses the concept of 'Directives' to refer to HTML vocabulary. A specific behaviour can be added to the DOM elements to transform it. The major difference between AngularJS and Knockout is the binding method. Knockout binds to the provided model while AngularJS binds with $scope object which is a special AngularJS object that handles the binding between the controller and the DOM.

E. Routing

While Knockout does not support routing, Backbone provides an easy configuration to provide the routing functionality. This is done by defining the routing map in JSON format triggering a function named action. Moreover, Backbone supports *parameterized* parameter within the URL.

Alternatively, the routing service within AngularJS allows the organisation of service routing based on URL. Based on the configuration map between map and template, the service will behave based on state.

F. Testing

Knockout can be integrated with many testing frameworks e.g. Protractor. However, testing in Knockout is not as straight forward as in AngularJS [18]. Some coding is required to enable 'testing'. Knockout has a very low barrier of entry recommending it when a quick start for a project is required. However, if a more complicated project is needed, it is better to use AngularJS [3].

Due to its flexibility to integrate with testing frameworks, Protractor - which is a Google product as well - is a popular end-to-end test framework for AngularJS applications. Protractor runs tests with an actual browser interacting with it as a normal user which facilitates code reviews and code maintainability.

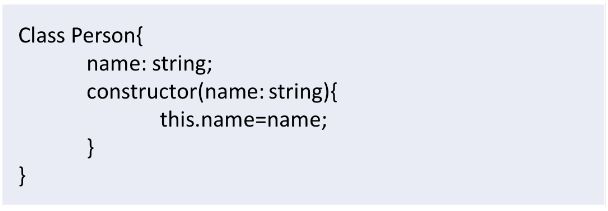
V. Angular

The same people who built AngularJS built Angular. They brought the knowledge and best practices from AngularJS to Angular. Also, they are active in the Angular community fixing bugs and providing support for such issues.

Angular is simpler than AngularJS but is not less powerful. Angular combined all in a component getting rid of $scope, ViewModels and the controllers. Just create a class accompanied with @component decorator

Angular added new features and capabilities that can not be found combined in React or any other framework.

1) Typescript: Typescript is a new feature embedded and integrated with Angular. We can say that Typescript provides the best IntelliSense developer experience available in the JavaScript world today. Typescript is the ES6 version of JavaScript plus a few other Typescript only features which Angular needs in order to work. Angular is flexible enough to be written in either Typescript, ECMAScript 6 (ES6) or even ES5 [19]. Therefore, we can conclude that ES5 is a subset of ES6 which is a subset of Typescript. As per Fig. 13 which represents a typical example of ES6 creating a class. Now classes are defined and are almost as readable as Java code[20].



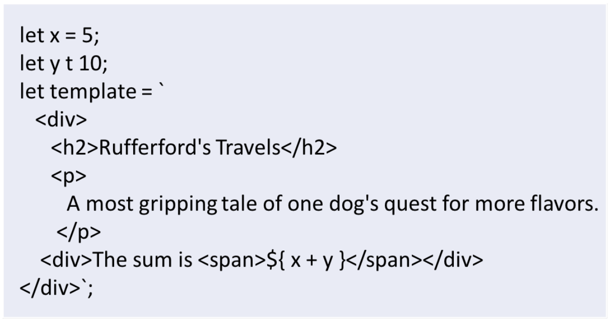
```
Class Person{
        name: string;
        constructor(name: string){
                this.name=name;
        }
}
```

Fig. 13. Example of class structure in Angular.

Typescript is an extension of ECMAScript, in fact: Typescript = ES6 + Types + Annotations. However, Angular itself is written in Typescript, it is not mandatory to develop using Angular with Typescript. Although most online examples are written in Typescript as well. Also, Typescript provides more functionalities related to Object Oriented like Generics and Lambdas expressions.

2) Data Binding: Angular uses different ways of data binding either to select one-way binding from the component(controller) to the view, vice versa or two-way binding. This means that if it is needed to change the value in the DOM, in an input field or in a text area, both the view and the model will be updated. This has a negative drawback, if we have a lot of data bindings within the view, will create - indirectly lot of observers which needs more watchers which may affect the project performance more with more increase of data bindings. This data binding approach makes it possible to create a stand-alone library accompanied with the functions and controls.

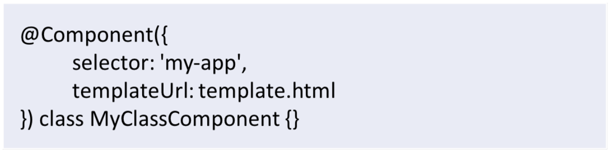
3) Templating: Beside AngularJS capabilities, ES6 added the ability to write long in line strings without having to use concatenation or other odd tricks. All is needed to do is to use backticks at the start and end of the string. Also, string interpolation using ${expression} placeholders can be used as well inside templating. Fig. 14 shows the usage of backticks and string interpolation.



```
let x = 5;
let y t 10;
let template = `
  <div>
    <h2>Rufferford's Travels</h2>
    <p>
      A most gripping tale of one dog's quest for more flavors.
     </p>
   <div>The sum is <span>${ x + y }</span></div>
</div>`;
```

Fig. 14. Example of templating in Angular.

Also, while defining a component simply we can refer to a template HTML external file presentation by adding the template Url field as shown in Fig. 15



```
@Component({
        selector: 'my-app',
        templateUrl: template.html
}) class MyClassComponent {}
```

Fig. 15. Example of referencing to HTML file used as template in Angular.

4) Extensibility: For untrained eyes, Angular templates might look like when using AngularJS (e.g. using [], (), [()], # and *), but, in fact, the entire template system is redesigned bottom-top to support web components and native elements. Moreover, this enables rendering a cross-platform native UI for iOS and Android.

5) Variable Observation: Angular uses ReactiveX [21] that provides an implementation of Observables. RxJS is the JavaScript implementation of the ReactiveX API. The API has multiple implementations in different languages, such as Java, .NET, and Pythons. Microsoft, GitHub, NETFLEX, airbnb and others are using ReactiveX. By running a single line, a variable becomes watchable and a defined function can be called once changed as per Fig. 16.

```
obs.subscribe(value => console.log("Subscriber:" + value));
```

Fig. 16. Variable subscription in Angular using RxJS.

6) Routing: With the use of router-outlet directive, components can be defined, how to reach, redirect is some cases. All via 'router-outlet' HTML tag as shown in Fig. 17.

```
const routes:Routes = [
        {path: '', redirectTo: 'home', pathMatch: 'full'},
        {path: 'find', redirectTo: 'search'},
        {path: 'home', component: HomeComponent},
        {path: 'search', component: SearchComponent},
        {path: '**', component: HomeComponent}
];
```

Fig. 17. Example of Routing map in Angular.

7) Testing: As mentioned before, Angular is written with testability in mind which facilitates integrating with Protractor in order to deliver a bug-free stable application. Also, easily, it can be integrated with Jasmine [22]. Jasmine is a behaviour-driven development (BDD) framework for testing JavaScript code. It does not depend on any other JavaScript frameworks.

8) Modularity: Developing an application in a modular form facilitates code review, code reuse and code expectation as opposed to the 'best practices'. The module can be considered as a container for server Angular components like services, filters and controllers. It is considered a packaging of components in order to be referenced in other modules or other projects as well.

9) Dependency Injection: The Dependency Injection has been improved from Angular1. The Component communication can and should be implemented in a loosely coupled manner. Each component can declare its input and output properties. To inject a child into a parent component, the parent binds its values into the child input properties and child passes its value to the parent component via Events. The child does not need to know who provided the data nor how it is implemented [23]

Angular takes care of referencing other modules or components with their dependencies dynamically. This supports eliminating components coupling together in a way that requires updating many places if we want to make a minor modification in one of the components. On the other hand, React does not explicitly use the concept of dependency injection. However, the parent object can pass its data (state or function) to the child object as properties as sometimes, It is needed to share or modify some values or states between components.

10) Filters: Also, known as a 'pipeline', a filter allows passing the output of an expression as an input to a second functionality. Therefore, such filters are mostly used in formatting the 'View'. In Angular, filters format the value resultant of an expression. They can be used in many places like DOM (templates), controllers and services. Angular also has the feature of creating a custom pipe (filter) to be used entirely in the system.

Using RxJs operator's filter, which data to be processed can be determined. For example, only valid form values will be submitted. This enhances the project performance and enhances the customer experience as well.

11) Form Validation: Angular provided an advanced version of form validation. Like data binding, form controls can be defined then linked with the DOM. Hence, many types of validations can be performed plus the form as a container can be validated as well which is needed in form submission as a bulk. Each controller has its own validation which makes it possible to build a personal library. The same is for validating errors. It is very useful in providing feedback to users with the corresponding descriptive message.

12) Community Support: Not only Angular has the largest community, it is also backed by Google. The core team is growing resulting in innovative tools and support that improves the developer productivity. Protractor, Batarang, ngmin and Zone.js are just a few of many. In addition, the team collaborate with the community in the design decisions. For example, all the design documents are shared in the community and everyone can make suggestions.

The numbers of Angular users are growing dramatically especially after the launch of Angular 2. By the time this paper was written the total number of stars and watchers as per GitHub [24] are 23K and 2.4K, respectively. The same for React, Facebook is big corporations and also have a big community. Core team are supporting developers all the time releasing bug fixes and other tools as well. Total numbers of followers are higher than which of Angular, due to its popularity for the past few years as it was the first choice for developers. By the time this paper was written the total number of stars and watchers as per GitHub [24] are 64K and 4.3K, respectively. The numbers are growing up dramatically due to its flexibility and it is becoming to be the latest technology. We cannot consider the number of followers low as the project is just released.

A. *Time Compiling*

Currently, Just in Time compiling (JiT) concept is used which is we ship out the compiler and the code to the browser then compile, parse and render the generated code to the browser. The compiler occupies about 50% of the payload. Angular supports Ahead of Time compiling (AoT) which is taking out the compilation process, hence sending the pre-compiled code to the browser for rendering. By this way, we can make the application 2.5 times faster and 3 times smaller in size [23], [11], [25]. In addition, Angular provides the Tree Shaking feature which is removing the unused piece of codes or libraries in the project which saves memory and increases the project performance as well [11], [23].

B. Mobile Capability

Angular is not limited to only working in the browser, it is the first framework that works on mobile devices as well. It has first-class support for touch events and gestures that work across all devices

C. Supporting Tools

React community recently released create-react-app which is a useful tool that automatically creates all needed configuration for a new project. This facilitates new project development without the need to spend time in applying configurations that are obvious can be automatically implemented. On the other hand, Angular released in March 2017 the first stable release of Angular CLI. Angular command line interface (CLI) is one of the most Angular important tools. It facilitates the process of generating Angular components, services, enum, pipes and classes. Hence make it easier and faster to build Angular applications. Via Angular CLI, it is easier to create a project from scratch and it will add all needed components and libraries needed for the projects including the project best practice structure (source code, environment variables, main and index.html, E2E testing locations). Plus, it can build and deploy the project either for development or production aim. The combination of Angular with Typescript provides a big opportunity in object oriented programming in the front-end. The declarative nature makes it much clearer [20].

D. Angular Expressions

Angular Expression is a powerful feature in AngularJS. It is used extensively in the View Layer. It lets the developer write complicated logic or even performs the assignment in the view templates. But, it has a drawback which it is impossible to test and makes the HTML more complicated. For example in the below code as shown in Fig. 18.

```
<button ng-click="(oldPassword &&
checkComplexity(newPassword) &&
oldPassword != newPassword) ?
(changePassword(oldPassword, newPassword) &&
(oldPassword=(newPassword=''))) :
(error Message='Please input a new password matching the
following requirements: ' +
passwordRequirements)">Click me</button>
```

Fig. 18. Example of writing expressions in HTML when using Angular.

In case the compiler faces code problems, errors or even syntax error, simply, the browser ignores them without giving any notification which makes it rather difficult to find and to regenerate for fixing [3].

E. Angular4

In March 2017, Angular4 was released. Due to internal versioning conflicts, Angular3 is skipped. Angular4 is not a complete re-write of Angular2 as what happened with AngularJS, it is just an update of Angular2 adding few more functionalities and deprecating other features. For example, Angular4 added more flexibilities in writing the if statement in the HTML, else can be added plus the usage of local references as well as shown in Fig. 19. It uses Typescript 2.2. Also, Renderer is deprecated and Renderer2 is added instead. Also, Email validator is added instead of writing the whole pattern to validate email fields, simple mention that the input field is an email.

```
<p *ngIf="showFirst; else elsePart">You can either see this
paragraph ...</p>
<ng-template #elsePart>
<p> ... or this one</p>
</ng-template>
```

Fig. 19. Example of writing if else condition in HTML with local reference in Angular4.

F. Comparison

We ran a quick survey on Google trends to measure the difference between using these terms online in the computer science field over the last five years. As shown in Fig. 20 , the search on the term Angular2 and React started by the first release of each of them, which is by end of 2014 and end of 2016, respectively maintaining the same trend of the AngularJS and Angular as a generic search. On the other hand, the search on the term Backbone is almost negligible with respect to other terms.

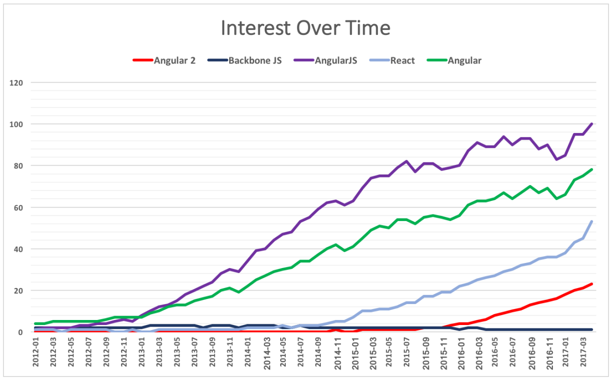


Fig. 20. The graph shows Interest over time starting from Jan-2014 till Mar-2017.

Numbers represent search interest relative to the highest point on the chart for the given region and time. A value of 100 is the peak popularity of the term. A value of 50 means that the term is half as popular. Likewise, a score of 0 means the term was less than 1% as popular as the peak.

VI. Conclusion

In this paper, we have surveyed the latest technologies of web and mobile frameworks from Knockout, Backbone, and AngularJS to the recent releases of React and Angular. Through analysing each of these frameworks based on the different key features of application platforms we were able to compare them thoroughly. We identified the main points of comparison as the ability of each application platform to support Data Binding, Templating, Extensibility, Variable

Observation, Routing, Testing and other Advanced Features including Dependency Injection, Form Validation and Browser Support.

In conclusion, our survey resulted in a number of interesting findings. Although Backbone is a small-sized and lightweight framework, it is much basic in terms of capabilities than Angular that is therefore much more complicated.

Both Knockout and Angular include a massive number of libraries and declarative bindings. The key difference is that Angular deals with the project as a whole providing guideline how the project should be structured. On the other hand, KnockoutJS enables the design of the project on personal preference. This paradigm, might not be the best practice as it affects the code review, reuse, manipulations and cross-platform support. Unlike Knockout, Angular mitigates these risks and moreover supports cross-platform capabilities.

Angular and React both require a considerable amount of time to configure Webpack or Gulp for the deployment readiness of a project. However, Plunker [26], a supporting web application, can be used to workaround this matter as it handles the web server configuration and other configurations, if we are talking about a small project using Angular or React. In general, we recommend using Knockout or Backbone in small-sized applications that do not expect extendability, scalability or advanced features. In the matter of speed and performance, we say that React has the upper hand. Nevertheless, Angular has various features considering being a framework such as advanced project structure configurations and best practices that compensate for being less in speed than React.

Finally, there is no absolute winner or 'the best framework'. Each framework has its strengths and weaknesses, we have to learn different frameworks to see which is more suitable to the situation we are in and what functionality we seek. If the goal is creating a fast UI, structured project or a lightweight application needed for a quick prototype, then, the best option will be React, Angular and Knockout/Backbone, respectively. Needs to update with real data.

Still, we believe that Angular stands above other frameworks due to its efficiency, convenience, different features and cross-platform support. Although Angular can require significant time for learning (over other similar frameworks), it can be the best practice and the trending future of web and mobile applications to come.

References

[1] K. N. Ard, "Single page architecture as basis for web applications," 2015.

[2] A. MacCaw, *JavaScript Web Applications*, 2011.

[3] Codeutopia, "Knockout vs Backbone vs Angular." [Online]. Available: https://codeutopia.net/blog/2013/03/16/knockout-vs-backbone-vs-angular/

[4] U. Shaked, "AngularJS vs. BackboneJS vs. EmberJS," *https://www.airpair.com/js/javascript-framework-comparison*, vol. 10, p. 2015, 2014.

[5] J. Fujima, "Building a Meme Media Platform with a JavaScript MVC Framework and HTML5," *Communications in Computer and Information Science*, vol. 372 CCIS, pp. 79–89, 2013.

[6] E. Sørensen and M. I. Mihailesc, "Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology," *MegaByte Journal*, vol. 9, no. 4, pp. 1–19, 2010.

[7] E. Masiello and J. Friedmann, *Mastering React Native*.

[8] JSX, "JSX." [Online]. Available: https://jsx.github.io/

[9] Facebook Inc., "A JavaScript library for building user interfaces - React." [Online]. Available: https://facebook.github.io/react/

[10] Redux, "Introduction · Redux." [Online]. Available: http://redux.js.org/

[11] S. A. Mousavi, "Maintainability Evaluation of Single Page Application," Ph.D. dissertation, 2016.

[12] J. Papa, "Client Insight - Getting Started with Knockout." [Online]. Available: https://msdn.microsoft.com/en-us/magazine/hh781029.aspx

[13] JSON, "JSON." [Online]. Available: http://www.json.org/

[14] Muhammad, "codementor." [Online]. Available: https://www.codementor.io/codementorteam/react-vs-angular-2-comparison-beginners-guide-lvz5710ha

[15] A. Mardan, *Full Stack JavaScript, 2nd Edition*, 2015.

[16] Google, "Angular Scope." [Online]. Available: https://docs.angularjs.org/guide/scope

[17] Underscore, "Underscore.js."

[18] Andy Lee, "ScottLogic - END-TO-END TESTING WITH ANGULAR AND KNOCKOUT." [Online]. Available: http://blog.scottlogic.com/2014/08/06/angular-knockout-e2e-testing.html

[19] A. Hussain, *Angular 2 From Theory To Practice*.

[20] E. Muller, "DZone/Web Dev Zone." [Online]. Available: https://dzone.com/articles/typed-front-end-with-angular-2

[21] ReactiveX, "ReactiveX." [Online]. Available: http://reactivex.io/

[22] Jasmine, "Jasmine."

[23] P. Deeleman, *Learning Angular 2*, 2016.

[24] Github, "GitHub." [Online]. Available: https://github.com/

[25] Google, "Angular Docs - ts - INDEX."

[26] Plunker, "Plunker." [Online]. Available: https://plnkr.co/