International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume1, Issue 4 (July '2015)

STUDY OF ANGULAR JS: A CLIENT SIDE JAVASCRIPT FRAMEWORK FOR SINGLE PAGE APPLICATIONS

Pranav Nerurkar
Department of Computer Engineering,
Thadomal Shahani Engineering College
Mumbai-400068, India

Aruna Pavate
Department of Computer Engineering,
St. Francis Institute of Technology
Mumbai, India

Abstract: The spread of internet has created a revolution in the way communication occurs among people. Since the use of internet and the number of hours spend by users online is only growing a new trend is emerging among web users. To make maximum use of the wide reach of internet most entrepreneurs have spearheaded the movement of creating a web based interface for their products or ideas by building a website. Single Page Applications (SPA) is a result of this trend. Offering desktop application like features with a view to provide the users with fluid, responsive features is the main goal of SPA. This is important as the entry barriers for creation of websites is low and number of competitors is increasing. The main intention of web developers nowadays should be to provide the users with an enriching experience so as to not only engage the site viewers but also retain existing customers. Angular JS is a upcoming technology which can enable developers to achieve this goal.

***Keywords: Single Page Applications, Angular JS, JavaScript framework*, AJAX,HTML5,CSS;**

I. INTRODUCTION

The beginning of the twenty first century saw a change in the way internet would be used. Internet is developed as a tool for exchanging information between military facilities; it found a new birth and purpose as a tool for spreading information around the world. Now days the internet is becoming a daily necessities without which even some of the routine tasks can't be accomplished. Because of such a transformation in the use of internet a new era of digital communication was dawned and a by product was developed in the form of a website. Websites became interface of an organization to the netizens. Although initially they were limited only to the elite with further developments in technology and improvements in communication systems, websites became more and more common place [1].

Websites are now everywhere. Everyone has one. From the local retailer to the multinational company everyone wants a piece of this goldmine. But this is not for no reason as these websites have the potential of transforming potential customers into actual customers. These days' websites became not only a source of broadcasting information but also a tool for interacting with web users. Companies have also changed their attitude towards this product and have worked to create a good impression in the users mind.

However there is down side to this trend. As the entry barriers of cost are low and no restrictions are placed on new entrants the web world has exploded with information making it difficult to locate it. The task of bringing a customer to your site on the web world is daunting but once he is there to ensure that his experience is good and getting him to come back again is even more daunting.

To achieve this tsk is the work of the programmer or the web designer. Most of them are engaged in creating a fluid, responsive, engaging experience for the users by innovative designs and SEO techniques. But these things are just one side of the coin the other more important question is whether the implementation can match up to the design. To answer that question the W3C and league of many other global giants as well as freelance developers have teamed up and developed some web development languages like HTML5, CSS, and JavaScript etc.

Along with these techniques the new trend emerged of having a Website that could provide a fast response to user actions by allowing client side data processing and low latency due to fewer interactions with the server. This trend then underwent a metamorphosis and the old design of having multi page websites with full refreshing capabilities was replaced with Single Page Applications.

In single page applications the concept was to load the entire webpage at one go with a single page load. Another approach was to bring data and resources dynamically as per user actions and loading them into page section without refreshing the entire page. The control of the page does not transfer at any point in time and the interaction with the web server occurs behind the scenes. Modern browsers also allow developers to shift the user interface and application logic from the server side to the client side (only for browsers that allow parsing of HTML5) [2].

To implement such an application we can make use of open source libraries which reduce the efforts so that instead of fighting technology problems concentration can be on fast response to user actions. AngularJS, Backbone, Batman,

CanJS, Ember, Meteor, Knockout, Spine are some JavaScript frameworks that allow for rich single page application development. The characteristics of a good client side framework should be ease of use, ability to extend HTML vocabulary, extensibility and achieving more with less code.

AngularJS is a client side JavaScript framework which is cross platform. It allows data binding and data injection. Data binding provides automatic updating of view if the model changes on user action also the other way round i.e. changing the model if the view updates. The main difference between traditional method of communication and AngularJS is that it maintains the state of controller and model at client side allowing new page generation without going to the server. With this it also addresses the root problem of HTML that it's not meant for dynamic web pages. Each feature can be developed to suit your unique workflow and development needs [3][4].

The contribution of this paper is:

- To understand the working of SPA.
- Features of AngularJS.
- Comparison with BackboneJS and benefits over it.

II. RELATED RESEARCH

A. Inner workings of Single Page Applications

In SPA's the user action prompts the browser to send request to the server in form of asynchronous call (AJAX). As shown below the server then returns only the data asked instead of markup. The data is in form of JSON response. The benefit of this is dual. On one hand there is saving of time as unnecessary rendering and reloading doesn't take place also there is a clear demarcation between HTML (presentation layer) and the AJAX+JSON responses (business logic).

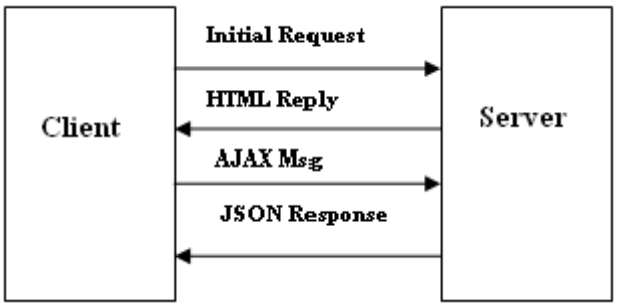

Figure 1: **SPA Lifecycle**

B. Advantages of Single Page Applications

1. SPA is extremely good for very responsive sites:

- Server-side translation is hard to implement for all the intermediate states[5].
- Single page apps are renowned by their capacity to redraw any part of the UI without requiring a server roundtrip to retrieve HTML.

2. SPA doesn't need to use extra queries to the server to download pages.

C. Qualities needed in a framework to implement SPA.

1. *Data binding.*
2. *Persistence.*
3. *Routing.*
4. *Built in helpers.*
5. *Form handling.*
6. *Keeping view in sync with model.*
7. *Full stack framework.*

D. Common Terms:

1. *Client side framework*: these are pieces of code called scripts that are loaded at the clients/users machine at page load and provide minor functionality like form validation , event handling without interacting with the server.

2. *JavaScript:* It is most commonly used as part of web browsers, whose operations allow client-side scripts to interact with the user, control the browser, interconnect asynchronously, and modify the document content that is displayed.

3. *BackboneJS:* **Backbone.js** is a JavaScript library with a RESTful JSON interface and is based on the model–view–presenter (MVP) application design paradigm.

4. *REST:* Stands for **Re**presentational **S**tate **T**ransfer. It depends on a stateless, client-server, cacheable communications protocol. REST uses HTTP for CRUD and is a light alternative for RPC, CORBA etc.

5. *AJAX:* It uses combination of JavaScript and xml to retrieve replies to information from the server. It doesn't affect display and markup information. Xml might not be used always JSON are used.

III. ANGULAR-JS

A. Benefit for XAML Developers

XAML, it is a declarative language based on XML used to instantiate object graphs and set values. A component of XAML is the support for data-binding. This agrees there to exist a stated relationship between the presentation layer and the data without creating hard dependencies between components. These same principles are reflected in the Angular framework. It enables a departure of concerns that allows a true parallel workflow between various components including the

markup for the UI itself and the underlying logic that fetches and processes data.

B. Easy data binding

To create a text property on a model that you want to bind to your UI.In Angular, this will work without any issues and immediately reflect what you type in the ng-model:

<input ng-model='firstName'/>

** {{ firstName }} **

It illustrates how easy and straightforward data-binding can be in the Angular world. There is very little involved with standing up a model that participates in data-binding. You don't have to derive from an existing object or explicitly declare your properties and dependencies.

C. Handles Dependencies

Single Page Applications use dynamic loading to present a very "native application" feel from a web-based app. These apps can grow quickly with lots of dependencies on various services and modules. Angular makes it easy to organize these and grab them as needed without worrying about things like, -- namespace and instance. Instead, Once Angular figures out what you need it goes and gets it for you and manages the lifetime of the objects for you.

D. Testing

Angular embraces this approach to building your application. AngularJS was designed from ground up to be testable. It encourages behavior-view separation, comes pre-bundled with mocks, and takes full advantage of dependency injection. It also comes with end-to-end scenario runner which eliminates test flakiness by understanding the inner workings of AngularJS. The ability to mock dependencies and inject them in Angular is very important and you can test everything from UI behaviors to business logic.

E. Enables Massively Parallel Development

Angular just took parallel development to another level. That's not to say it completely eliminates dependencies, but it certainly makes them easier to manage. In a traditional JavaScript application it could have been a merge nightmare to scale this across a large team. With Angular, however, it was straightforward to break down the various actions into their own services and sub-controllers that developers could independently test and code without crashing into each other as often[6].

F. Enables a Design <—> Development Workflow

The designer can add markup without completely breaking an application because it depends on a certain id or structure to locate an element and perform tasks. Instead, rearranging portions of code is as easy as moving elements around and the corresponding code that does the binding and filtering moves with it. Essentially the teams agree on the elements that will be displayed, then design goes and lays it out however they want while development wires in the $scope with their controllers and other logic, and the two pieces just come together in the end.

G. Gives Developers Controls

Angular enables a new scenario known as a "directive" that allows you to create new HTML elements and attributes. In the earlier example, the directive for the "ng-model" allowed data-binding to take place. This gives developers their controls – and more importantly, control over the controls.

Our project has evolved with literally dozens of directives and they all participate in previous points:

- Directives are testable
- Directives can be operated on in parallel
- Directives enable a declarative way to extend the UI, moderately than using code to wire up new constructs
- Directives reduce usual and ceremony
- Directives partake in dependency injection

H. Helps Developers Manage State

This refers to client state and how to manage properties, permissions, and other common cross-cutting concerns across the app in the browser. Angular not only knobs dependency injection, but it also manages the lifetime of the components for developers. That means they can approach code in a very different way.

I. Supports Single Page Applications.

Single Page Applications are becoming more popular for a good reason. They fill a very precise need. More functionality is being moved to the web, and the browser is finally realizing its potential as a distributed computing node. By design, SPA applications are far more responsive (even though some of that is perception). They can provide an experience that feels almost like a native app in the web. By rendering on the client they cut down load on the server as well as reduce network traffic – instead of sending a full page of markup, you can send a payload of data and turn it into markup at the client.

Angular provides the entire necessary infrastructure from routing (being able to take a URL and map it to dynamically loaded pages), templates, to journaling (deep linking and allowing users to use the built-in browser controls to navigate even though the pages are not refreshing) needed to stand up a functional SPA application, and is quite adept at enabling

developers to share all of the bits and pieces across individual areas or "mini-SPA" sections to give the user the experience of being in a single application.

IV. Comparison with other frameworks

The criteria to decide whether AngularJS is better than other frameworks were made based on the following points.

- **UI Bindings** – This talks about a declarative approach to automatically updating the view layer when the underlying model changes.
- **Composed Views** – Creation of modular code when programming UI is a necessity, also it must be able to compose views (preferably at the template layer). This should also entail the potential for a rich view component hierarchy.
- **Web Presentation Layer** – No use of native style widgets. There is also no reason for a web framework to create its own layout manager. HTML and CSS are already the richest way to do style and layout in existence, and should be used as such. The framework should be centered on this concept [7].

Feature	BackboneJS	AngularJS
Feature detection	Yes	Yes
DOM wrapped	Yes	Yes
XMLHttpRequest data retrieval	Yes	Yes
[Web Socket]	No	No
Event handling	No	No
Server push data retrieval	No	No
Other data retrieval	No	No
Drag and drop	Yes	Yes
Simple visual effects	No	Yes
Grid	Yes	Yes
Auto completion tools	No	Yes
Widgets theme able / skin able	Yes	Yes
UI Bindings	No	Yes
Composed Views	No	No
Web Presentation Layer	Yes	Yes

Figure 2: Comparison of Frameworks

V. Client side m-v-c disadvantages

Client-side processing and decoupling is detrimental to both the speed of development, and application performance (a ton of JavaScript has to be loaded and evaluated each time you fire up the app). It's better to let the server handle HTML rendering and minimize the use of JavaScript on the client. Letting the server handle most stuff doesn't mean that you have to cut back on cool front-end features and user friendliness.

Sometimes all these libraries are actually detrimental to the user experience in some ways, as they lock you into certain patterns.

VI. CONCLUSION

The need for j query frameworks is increasing as they are preferred over State-ful frameworks like EJB. The need for storing state of the client isn't needed and just simple request reply type messages are transmitted over the Internet. This creates a separation between design and implementation. Result of this are good applications, light weight and faster to load.

While choosing a proper framework for your application care should be taken to consider points like DOM wrapping, Event handling, server side communications etc.

VII. REFERENCES

[1] Angular, Ember, And Backbone: Which JavaScript Framework Is Right For You? "http://readwrite.com/2014/02/06/angular-backbone-ember-best-javascript-framework-for-you".

[2] Why you should use client side MVC frameworks "http://jster.net/blog/why-should-you-use-client-side-mvc-framework"

[3] Client side MVC is not a silver bullet. http://mir.aculo.us/2013/02/26/client-side-mvc-is-not-a-silver-bullet/Rich javascript applications –the Seven Frameworks.

[4] http://blog.stevensanderson.com/2012/08/01/rich-javascript-applications-the-seven-frameworks-throne-of-js-2012/ Top 10 javascript MVC frameworks reviewed.

[5] http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/

[6] AngularJS the superheroic javascript MVW framework.

[7] https://angularjs.org/Reason why you should use AngularJS.

[8] http://www.sitepoint.com/10-reasons-use-angularjs/