Creating Effective Websites using AngularJS

Brandon Mota

Abstract

Websites provide an effective form of displaying information. However, the site itself has to be designed in such a way to accurately and effectively display the information. While a basic HTML page may be acceptable for small scale sites, html pages by itself cannot encompass the growing complexity of information being provided. For large scale applications the information may prove to be too complex for developers to develop and present the information. That's where AngularJS comes in. AngularJS will allow a developer to neatly and quickly set up web pages to more effectively present the information.

1. Introduction

AngularJS is a web application JavaScript framework. It is a framework specifically for client-side model-view-controller, or part of it. AngularJS acts as an extension to HTML by as well as JavaScript. What makes AngularJS different from other JavaScript frameworks such as Backbone.js or Ember.js is how AngularJS introduced the extra functionality -- specifically the simplification of two way data binding, of any of the other frameworks. Not only that, but it is also customizable to fit the developer's needs.

1.1 Technology and Terminology

In order to understand how AngularJS works, there are several key terms and technologies that AngularJS manipulates to achieve what it does. There are many different aspects that AngularJS touches upon on, but the focus will be on MVC, DOM, and Ajax programming.

1.1.1 MVC

The MVC, or Model-view-controller, is the architecture, or structure, of the web. As the name implies, the architecture is separated into three key areas: the model, the view, and the controller. The model is used to capture any data from internal – items already present on the view itself, or external – items from a database, or other external storage, sources (Sinha, Karim, & Gupta, 2015). The view is what is being displayed to the user (Sinha, Karim, & Gupta, 2015). The controller is what ties the model and view together. The controller retrieves the data from the model which is then displayed on the view (Sinha, Karim, & Gupta, 2015). This occurs every time a new web page is opened. Angular modifies this method, but that will be discussed when on Two Way Data Binding.

It is also important to know the reason why the separation of the model, view, and controller is essential when developing any web site. Knowing this will also help understand why Angular changes the MVC. If this separation did not exist, the code will be unstructured. Not only that, but redundancy will be rampant as simple dependencies will be reiterated on each and every web page (Sinha, Karim, & Gupta, 2015). For example, you would not want to have to hardcoded information that is always changing onto a web page. If this was the case, you would need to update the web page itself when the information changes instead of the database.

1.1.2 DOM

DOM (Data Object Model) is how the data is manipulated within the view from the MVC architecture. It is standard amongst all web browsers of how the DOM is actually modeled. First, the server provides the view. Then the browser interprets the view and the respective scripting code -- the scripting language is usually JavaScript, and creates the scope from the script to the view. This is known as the DOM tree (Scully, Dobos, & Sturm, 2015). The DOM tree is the link from the HTML page to the script and contains all references to each tag to each script, as well as the scope. One thing is that the DOM is always done by the client, and never the server. So the web browser has code on how to create the DOM.

To better understand what the DOM actually is, here is a basic example of JavaScript and HTML code.

HTML Code:

```
<div id="container"></div>
```

Javascript Code:

```
<script>
  var container = document.getElementById("container");
  container.innerHTML = "New Content!";
</script>
```

Code obtained from: Covier, C. (2013, December 29). *What is the DOM?* Retrieved October 30, 2015, from CSS-Tricks: https://css-tricks.com/dom/

When you see this example, you can see that the div tag has the id of container. The container is now part of the script. The browser creates the scope for the container for the entire div tag. The DOM tree is created by the browser. When you actually try to see the source code for the browser, you will not see that the code you see above, but instead you will see the new content is which the script has created (Covier, 2013).

This is the div tag code you will see from your browser when you request the source code:

```
<div id="container">New Content!</div>
```

Code obtained from: Covier, C. (2013, December 29). *What is the DOM?* Retrieved October 30, 2015, from CSS-Tricks: https://css-tricks.com/dom/

It is important to know how the DOM functions since Angular modifies the way the script and view is interpreted (Scully, Dobos, & Sturm, 2015). More of how Angular does so will be discussed later.

1.1.3 Ajax

Ajax programming is used to create dynamic web pages. Data is retrieved from the background without completely changing the view. Ajax is a form of client-server interaction (Sinha, Karim, & Gupta, 2015). Ajax stands for asynchronous JavaScript and XML. The asynchronous piece refers to the ability to not interfere with areas other than the area that is being modified. As stated before, Ajax allows a developer to create dynamic pages by preventing any significant view changes (Sinha, Karim, & Gupta, 2015).

How Ajax works is by always checking for a state change. As the name Ajax implies, it can be done with JavaScript. Once an event occurs that causes a state change, data is retrieved from an external source and then displayed without any noticeable interruptions. For example, if there was a button to reveal spoilers to a favorite TV show, the button can be clicked on to reveal the spoilers. When the button is pressed, the page itself does not refresh to retrieve the data; however the client calls upon data from the server to reveal the spoiler. I want to point out this is an example of how Ajax works, and it does not necessarily mean that all spoiler buttons use Ajax. Angular uses Ajax as the basic principle on the functionality it provides.

2. How AngularJS is different

The technologies mentioned are only a part of what Angular modifies. As such, it was not a comprehensive list. The technologies mentioned are the essentials to understand why Angular was originally created. Angular provides its own set of easy to use functions that goes against was originally was the web architecture.

2.1 Two -Way Data binding

Two way data binding works just like Ajax with one huge difference. With Ajax, data is always static, it cannot account for dynamic data say a user's input. If a user inputs the data, the Ajax site will need to refresh the page.

Angular has a "live view" of the data being manipulated. What this means is that anytime data is manipulated, Angular will dynamic change the view to match the new data (Sinha, Karim, & Gupta, 2015). For example, if a user wants to review a product, the user enters his or her review. This could be via a text box. Above that, the user is provided a live preview of the review being entered. As soon as there is a new character is entered, the new character is also displayed.

How Angular achieves this is by manipulated the MVC. Instead of having the model, view, and controller into separate entities, the model and view can continuously loop until the controller is needed (Developer Guide, 2015). What this means is the controller can be dynamically used within the model itself (Developer Guide, 2015). This also means the model and view work interchangeably to modify each other (Developer Guide, 2015). Instead of MVC, Angular coined the phrase MVW, or Model-View-Whatever (Developer Guide, 2015). It can also be referred to as the MVVM, or Model-View-ViewModel (Developer Guide, 2015).

2.2 Client Side Services

Client side services manipulate the DOM of any web page. One key part of the services is the dependency injection. Dependency injection allows controllers in Angular to access the scope (Developer Guide, 2015). An html tag contains its own dependencies and would no longer require the browser to interpret the scope of scripts. With Angular you can assign scripts that are assigned to each tag. If you were to view the source code of the web page you would view the actual source code that way originally written by the developer.

With Angular, if you have recurring themes in all webpages, you can simply have the recurring items be the main page. If you have a menu bar that is never changing, you can force that page to be the main page. For example, if you were creating an online retailer, a products page may be the same for all products. However, an online retailer could have a "Contact Us" page that is missing elements from the product page. One thing in common between the

"Contact Us" page and the product page is the menu bar. The menu bar never changes. So you can make the menu bar the main index page, and the other pages can be created to interact with the main page. These pages are known as partials (Developer Guide, 2015). This concept is also similar to PHP server side includes, but Angular is client side and therefore the partials are a part of the client side template for the site (Developer Guide, 2015).

3. How to use AngularJS

Knowing how Angular functions will help with understanding how Angular can create effective websites. The focus will be on using the functions first introduced by AngularJS: two-way data binding. With this example, the structure of Angular can be easily understood.

```
<div ng-app="">
<p>Name: <input type="text" ng-model="name"></p>
<p>You wrote: {{ name }}</p>
</div>
```

Code obtained from: AngularJS Examples. (2015). Retrieved November 5, 2015, from weschools.com: http://www.w3schools.com/angular/angular_examples.asp

To create a live view of data manipulation, you must first declare the AngularJS app that is being used. It is assumed that the Angular script has already been declared. Since this example is not complex a specific ng-app is not necessary. The ng-app element, however, is required to let Angular know that the following tag scope has to be completed by Angular. Notice there is an input tag; ng-model is then used to start the two-way data binding. To complete the data binding process you declare the directive "name" with two curly braces. Since Angular detects the model is exactly the same as the name as the directive, the two-way data binding is complete. Another way to complete the two-way data binding is to remove the directive and use the element ng-bind on the paragraph tag. I also want to stress how neat the code is. There are no scripts bloating the code, there are no extra tags necessary, and there is no logic in the HTML code.

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
```

```
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    }
});
</script>
```

Code obtained from: AngularJS Examples. (2015). Retrieved November 5, 2015, from weschools.com: http://www.w3schools.com/angular/angular_examples.asp

In the example above, a controller is demonstrated. Using the ng-controller on a tag will set the scope for the controller. This is how Angular changes the typical MVC architecture to a MVVM architecture by combining the controller to the model. This means that instead of having the controller run whenever a page is loaded and retrieved, the script will only run when an event occurs. In this example, the script will run whenever the input has been changed. In this case it when the “First Name” and “Last Name” is changed. Also note that Angular declared that “John” should be written in the scope of the “firstName” element. Since “firstName” is only set as the element for the first input tag, the input tag will be initialized as “John.”

The formatting for Angular follows the same structure as the examples shown above. You declare Angular by using ng-app. Then, you add angular specific directive as an element to a tag. Angular can do two-way data binding from a database, such a MySQL. Angular can use custom controllers that are specific to certain tags. With Angular may use of filters, forms, expressions, tables, client-side includes, multiple applications on a page, validation, and more. All this can be done cleanly in an HTML page. You are also not forced to use Angular directives. You may create your own directives if Angular does not have what is needed for the application.

4. Conclusion

AngularJS is an excellent JavaScript framework. While not helpful for simple sites with very little data, it is extremely useful for medium to large scale web applications. I highly recommend using Angular if you need to create a website. It is easy to use, keeps HTML code

clean, and data manipulation allows for information to be easily accessed by users. When combined with other frameworks, such as bootstrap, a website can be created fast and efficiently.

Works Cited

Anderson, B., Davila, P., & Oliver, S. (2014). CoPerformance: A Rapid Prototyping Platform for Developing Interactive Artist-audience Performances with Mobile Devices. *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services* (pp. 605-607). New York City: ACM.

AngularJS Examples. (2015). Retrieved November 5, 2015, from weschools.com: http://www.w3schools.com/angular/angular_examples.asp

Covier, C. (2013, December 29). *What is the DOM?* Retrieved October 30, 2015, from CSS-Tricks: https://css-tricks.com/dom/

Developer Guide. (2015). Retrieved November 11, 2015, from AngularJS: https://docs.angularjs.org/guide

Scully, T., Dobos, J., & Sturm, T. (2015). 3drepo.io: Building the Next Generation Web3D Repository with AngularJS and X3DOM. *Proceedings of the 20th International Conference on 3D Web Technology* (pp. 235-243). New York: ACM.

Sinha, N., Karim, R., & Gupta, M. (2015). Simplifying Web Programming. *Proceedings of the 8th India Software Engineering Conference* (pp. 80-89). New York, NY: ACM.