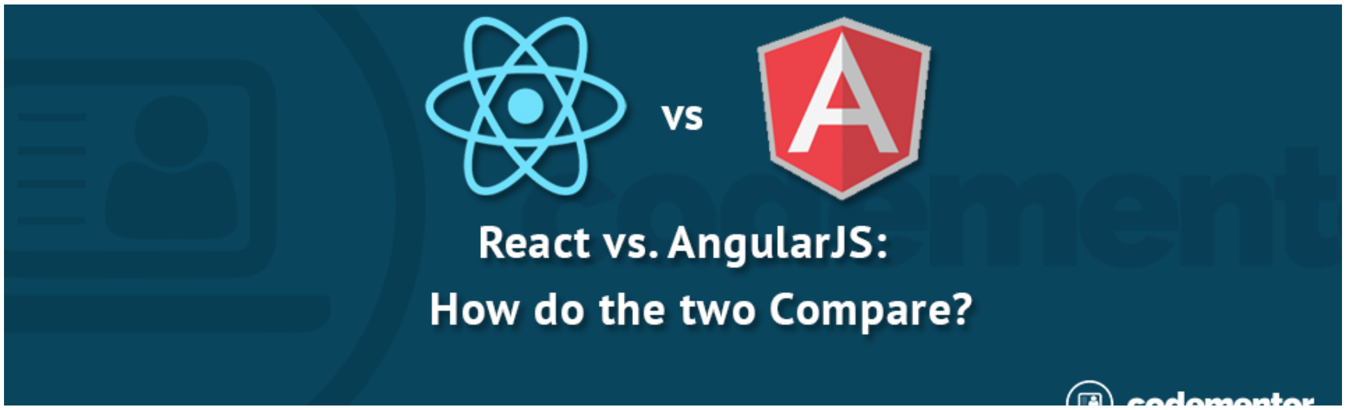
Chris Harrington FOLLOW

React vs AngularJS – How the two Compare

Published Dec 29, 2014



The purpose of this article is to go over building some simple functionality using Angular and React to give you an understanding as to what it would take to get an application off the ground. Angular is the big dog in this fight, as it's been around for a while longer than React has, but React brings increased rendering performance to the table. This post is not meant as an in-depth tutorial on either of the frameworks. As such, I'm not going to go into great detail for a lot of this stuff.

If you ever feel as though you need more clarification on a topic, either throw me a question via Twitter or comment below. The documentation for Angular and React is also pretty good. I'm hoping that after you read this, you'll have a

greater understanding as to which framework is right for you to use on your next project. At the very least, you should have a grasp on how JavaScript UI

8 73

UI controls, building up our little app as we go.

Hello, world!

Angular JS

Angular has been around for a while now, with its initial release in 2009. Since then, it's come a long way, and they're up to version 1.3.8 now, with semi-regular releases. Lately, there's been some buzz about Angular 2.0, which is going to change a lot about the framework, but it's a long way off, and is scheduled for release sometime in 2016. The current Angular version uses various attributes and custom HTML elements to provide functionality on various DOM elements. These are called **directives** and are very versatile.

XJjWze
A PEN BY chrisharrington

The set up code for Angular is a little verbose, as it requires an **application** and a **controller** , as well as definitions for any **directives** you're building. The snippet above shows how to set up an initial Angular app, then define a

controller and finally build a hello world directive. Here's a breakdown of what we're doing with the JavaScript.

8 73

dependencies, we pass in an empty array.

2. Next, we're initializing the `testController` Angular controller. It's not doing anything at the moment as our directive is handling all the "heavy" lifting.
3. Finally, the directive is defined. All it's doing is writing out a `div` containing the hello world text.

An Angular app requires that the `ng-app` and `ng-controller` directives be set on various elements. Here's a rundown.

1. The `html` tag has the aforementioned `ng-app` directive applied to it, indicating that the page reflects an Angular application named "testApp".
2. The `body` tag is annotated with the `ng-controller` directive, which tells Angular that the controller's scope should reside within it.
3. Finally, we're rendering our `hello-world` directive as the only code in the body.

React JS

If you've never heard of React JS, it's a performance-minded view renderer made by the Facebook guys. A lot of the heavyweight contenders for MVVM frameworks have a hard time rendering large amounts of data, like in lists and such. React doesn't have that problem, as it renders only what's changed. For example, if a user is viewing a list of 100 items rendered with React, and he or

she changes the third one down somehow, only that item gets rerendered, leaving the other 99 items unchanged.

8 73

creating a patch. React uses JSX files (optionally) to write views, which means that JavaScript and HTML can live in a single file. It's a bit of a paradigm shift and takes a little getting used to. You don't have to use JSX to write React views, but it's much easier than spelling out which components to render, so I'd suggest using it. If you'd like a little more information on React, I've written an in-depth tutorial for it here. It also goes over the Flux data architecture which I won't be covering here.

Note: React classes can be built in one of two ways: in raw JavaScript or in JSX, a language that combines HTML and JavaScript in one file format. I'm using JSX in the JavaScript sections of the examples below. If you'd like some more information, here's the documentation.

pvEYJv
A PEN BY chrisharrington

React requires that you build a class to define what HTML to render. That's what we're doing in the JavaScript section.

1. We call `React.createClass` to (unsurprisingly) create a React class. This class is what gets rendered later on to be appended into an HTML

8 73

3. The last line of the JSX is used to render our hello world class into the `document.body` .

The HTML for the React section isn't noteworthy at all, and won't change throughout this article, so you can stop looking at it.

Injecting Data

Angular JS

The method Angular uses to inject data into directives is via **scopes**. A scope in the Angular world is just an object which contains data for various controls, like directives and controllers. On top of that, there's a root scope object (injected with `$rootScope`) that functions as a top-level scope accessible by all other Angular components. A directive can have one of a few kinds of scope. First, there's no directive scope, which means that it uses its parent's scope (usually the controller's scope). Then, there's an isolated scope, which indicates that the directive has its own scope completely separate from the controller's scope. Typically, a controller's scope is used to inject data into a directive's isolated scope.

In my experience, the isolated scope is what gets used the most. That may have been confusing. The scope documentation should clear any of that up for you. Scope properties come in three different flavours:

- Two-way binding: Indicates that changes made on either the parent or child scope are reflected in the other. It's denoted by an equals sign (=)

Text binding: Just a string value that contains no binding information. Denoted by an at symbol (@).

LEREVv
A PEN BY chrisharrington

Here, we're using a text binding to send my name into the hello world example.

- The `hello-world` element in the HTML pane now contains a `name` attribute, which corresponds to the value defined in our isolated scope.
- Our controller now has a definition of the name variable in its scope.
- The directive template now contains a reference to the name scope variable.
- We've added a scope section to our directive definition which indicates that two things: first, that the directive should create an isolated scope, and second that it should contain only the name parameter as defined by the text binding (@). Whenever this scope section exists, an isolated scope is defined.

React JS

React injects data into its rendered views at construct time, be that when the

8 73

emdmJK
A PEN BY chrisharrington

- We've added a `{ name: "Chris Harrington" }` to the creation of our `HelloWorld` class, which gets translated into the props property inside the class.
- The JSX now includes a reference to `this.props.name` so we can render a personalized hello message.

Event Handlers

Angular JS

In Angular, event handlers are typically assigned to various portions of your view via directives. If you hadn't noticed by now, directives make up a large portion of what makes Angular great. They're used for practically everything. The `ng-click` directive is an attribute that can be placed on HTML elements that hooks up a click handler that already exists on the local scope.

ZYpExy
A PEN BY chrisharrington

8 73

- We've added a button to our directive template, which sets our greeting message on click. It has the `ng-click` directive set to fire the `greet` method.
- The `link` section to the directive has been added. This section is fired when the directive is rendered and is used to set up things like event handlers and default values, both of which are occurring here. The `greeting` property has been set to the empty string on the scope, and we've added a basic event handler (`greet`) to handle the button click.

React JS

Event handlers in React work by setting various custom attributes on the HTML elements you're building. Most of the common culprits have associated handlers.

zxKZKm
A PEN BY chrisharrington

There's a few new things we're introducing here. The big one is the idea of **state**. Each React class has its own internal state that can't be modified by isn't triggered. A good rule of thumb is that if you're setting data on the state, use the `setState` method.

- The `getInitialState` function is used to determine the initial state of the view. Without it, `this.state` will return null, potentially breaking the view. If you're going to set any state during the operation of your class, you need to add this method. In our example, we're just setting the `greeting` state property to the empty string.
- We've added a `greet` function to act as the event handler after the user has clicked on the button.
- Speaking of which, we've added a button to the HTML and are using React's `onClick` event to hook up the handler, which is the `greet` function of the context.

Nested Views

Angular JS

Nested views in Angular are accomplished using multiple directives. One directive can contain a reference to another without any problems. It's worth noting that each of these directives can have their own isolated scope, and any level of nested directives will also have their own scopes. Angular uses the idea of **transclusion** to render child elements from a directive. Transclusion is a little complicated, so here's the documentation if you need it. Here's a quick example:

zxKZGL
A PEN BY chrisharrington

Here, I've created a `wrapper` directive which (unsurprisingly) wraps our `hello-world` directive by adding some padding and a gray background colour. So, what's changed in the code?

- First, some house keeping. I've moved the templates into `script` tags in our HTML to keep things neat. In our directive initialization, I'm now grabbing the template HTML from these script tags.
- I've added the `wrapper` directive, too. It's very simplistic, containing only a div styled with a darker background and some padding on the HTML/CSS side. On the JavaScript side, the directive initialization for the `wrapper` contains one property that's not in our `hello-world` directive: the transclude property. This property indicates that any child elements of the directive should be rendered into the template via a built-in Angular directive called `ng-transclude`. You can see it in the HTML in the `wrapper` template. You'll see in the `hello-world` directive HTML that the `wrapper` contains children that show our greeting. Those children get rendered into the `ng-transclude` directive in our `wrapper` directive.
- There's now a CSS class to set the style of the wrapper. Just a background and some padding; nothing fancy.

React JS

Nested elements in React are much easier to put together, at least in my opinion. It's as easy as declaring the various classes, and then referencing the

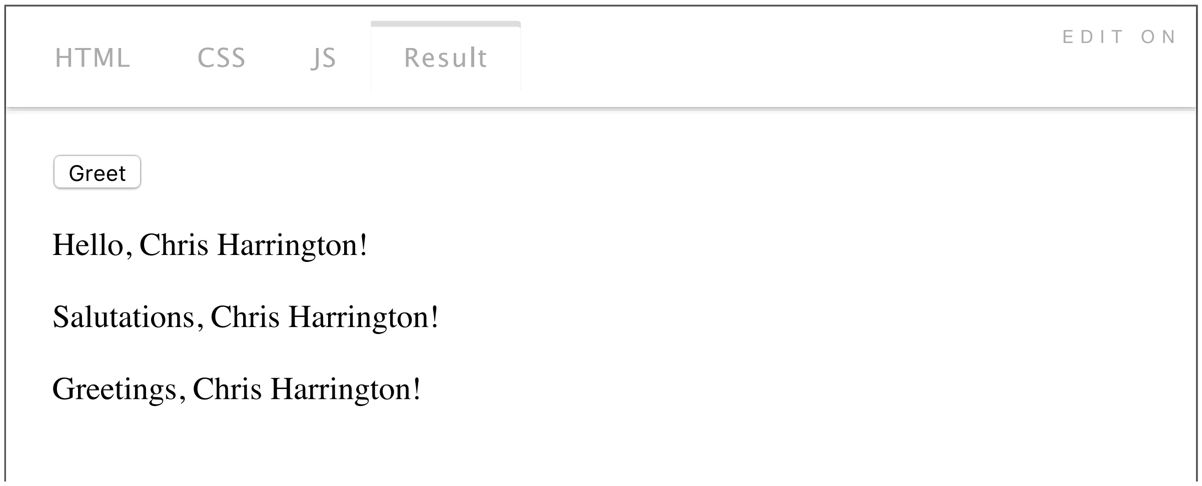
8 73

Like in the Angular example, we've wrapped up the meat of our `HelloWorld` HTML in a new `Wrapper` class, which again provides a background colour and some padding.

- The `Wrapper` class comes before the `HelloWorld` class because it's required in the `HelloWorld` 's render method. It's not necessary to do it this way, but I like it as it gives a natural order to where your classes sit. The `Wrapper` 's render method contains a single div with a CSS class specified to give our wrapper the background and padding. Inside that div is a reference to a special variable: `this.props.children` . This variable contains all inner elements of the class when rendered in another class. Our `HelloWorld` class has the important parts of our greeting HTML inside the `Wrapper` , so that's what gets rendered as a result of a call to the children props variable.
- Our `HelloWorld` class now contains a reference to the `Wrapper` class so as to provide the wrapping container.

That's it. There are no more changes for the React side of things. It's much cleaner than the Angular implementation, in my opinion. If you'd like to read some data. With Angular, it's accomplished via the built-in `ng-repeat` directive. In React, nothing special is required, as we can just loop over various HTML elements and put them into an array; a benefit of using JSX.

Angular JS



```
Hello, Chris Harrington!

Salutations, Chris Harrington!

Greetings, Chris Harrington!
```

Here, we're making use of the `ng-repeat` directive to add three greetings when clicking on the greet button.

- The `hello-world` directive now has a div after the button which uses the `ng-repeat` directive. It indicates that the marked div should be repeated for every greeting in the greetings array on the directive's scope.
- The greet method on the directive's scope now adds three separate greetings to the greetings array to be rendered on the view.

React JS

Looping in React is relatively simple, as JSX is just JavaScript, so we can write a simple JavaScript loop to render what we want.

```
Hello, Chris Harrington!

Salutations, Chris Harrington!

Greetings, Chris Harrington!
```

- The render method in the `HelloWorld` class has been refactored to reference a `renderGreetings` method, which renders all the greetings in the state. When I first started building React classes, I was under the mistaken impression that only the render method could contain HTML. That's definitely not the case.
- The `greet` method now pushes three separate greetings onto the `greetings` state variable.
- The `getInitialState` method replaces the previous `greeting` variable with a new `greetings` variable, which is set to an empty array.

Conclusion

While the differences between Angular JS and React JS are large, they can both accomplish the same thing. If I appear biased in what I've written here, it's because I'm biased, definitely toward React. Angular is a much fuller featured framework than React, which isn't really a framework at all, but that doesn't mean a lot when I don't see the need for most of the features that Angular provides. I've found that I have to write less code to do more in React, and React has better performance than Angular due to React's implementation of a virtual DOM (which I'll prove in a future post). Despite that, support for Angular is much, much better, with a larger community and following, as React is just getting started.