Like rats building a maze from which later we'll try to escape

- Home
- About

- Subscribe to RSS

September 27, 2013

- by odiseo
- in angularjs, d3.js
- 3 Comments

Proper use of D3.js with Angular directives

D3.js is not only an eye-candy way to create data graphics, as a library proposes a very powerful concept: manipulate documents (read DOM) based on data. So instead of feeding a library with input data to draw a graph, D3.js creates a binding between your data source and the DOM, with all the dynamic benefits that this implies.

Although it's something completely different, AngularJS also implements the same underlying philosophy with it's all famous two-way data binding: have your models as plain JavaScript objects, and represent it directly modifying the DOM. continuously changing the view whenever the DOM changes, having only "one source of truth".

AngularJS does most of this magic via setting watcher on your JavaScript plain objects and re-rendering the DOM via $digest cycle. Hence you have to be kinda careful whenever modifying the DOM directly, and as a guideline, you shouldn't do DOM modifications in your controllers directly, and rather choose directives to do this job. From the official documentation:

> Stop trying to use jQuery to modify the DOM in controllers. Really. That includes adding elements, removing elements, retrieving their contents, showing and hiding them. Use built-in directives, or write your own where

necessary, to do your DOM manipulation.

So, I've seen many tutorial and examples using AngularJS as framework but also D3.js for data visualization that actually miss this point and do terrible things.

What would be "the Angular way" of integrating D3? I propose the following:

1. **All your D3 logic and presentation must be contained within a directive**
2. **Use HTML-declarative syntax to feed of data to your directive instances**
3. **By doing that, you can store the data in your controller, passing it to your D3 directive via two-way data binding of parameters**

Let's illustrate that with an example.

All your D3 logic and presentation must be contained within a directive

This part assumes you're familiar with AngularJS directives, if not, please watch this awesome video. First of all, think how you want to abstract the behavior of your data visualization: you might want a directive to draw D3 SVG shapes, or charts (pie, chart, maps, anything you want). In this example, let's create a D3 graphic that plots an array of data into a bars chart.

```
angular.module('myApp', []).
   //camel cased directive name
   //in your HTML, this will be named as bars-chart
   directive('barsChart', function ($parse) {
     //explicitly creating a directive definition variable
     //this may look verbose but is good for clarification purposes
     //in real life you'd want to simply return the object {...}
     var directiveDefinitionObject = {
         //We restrict its use to an element
         //as usually  <bars-chart> is semantically
         //more understandable
         restrict: 'E',
         //this is important,
         //we don't want to overwrite our directive declaration
         //in the HTML mark-up
         replace: false,
         link: function (scope, element, attrs) {
           //converting all data passed thru into an array
```

```
            var data = attrs.chartData.split(',');
            //in D3, any selection[0] contains the group
            //selection[0][0] is the DOM node
            //but we won't need that this time
            var chart = d3.select(element[0]);
            //to our original directive markup bars-chart
            //we add a div with out chart stling and bind each
            //data entry to the chart
             chart.append("div").attr("class", "chart")
              .selectAll('div')
              .data(data).enter().append("div")
              .transition().ease("elastic")
              .style("width", function(d) { return d + "%"; })
              .text(function(d) { return d + "%"; });
            //a little of magic: setting it's width based
            //on the data value (d)
            //and text all with a smooth transition
          }
      };
      return directiveDefinitionObject;
   });
```

Use HTML-declarative syntax to feed of data to your directive instances

This is why we wanted a directive in the first place. It’s one of the neatest things of AngularJS: extending HTML. So now, whenever we want to declare a new bar chart in your HTML mark-up, we’d be including something like this:

```
<bars-chart chart-data="40,100,80,15,25,60,10" ></bars-chart>
```

Notice how the HTML itself remains completely agnostic of how the chart is being draw. This time we’re using D3, but we may change the implementation in the future, or do even greater things using the latest D3 version. Whatever happens, out HTML remains the same. That’s modularity yo!

Store the data in your controller, passing it to your D3 directive via two-way data binding of parameters

We don’t really want to have a hardcoded chart-data=”40,100,80,15,25,60,10″ in your HTML, since that information is meant to be in our models: our source of truth. So the right place for it is in our controller’s $scope variable:

```
$scope.myData = [10,20,30,40,60];
```

And now we can feed this myData array to our directive view

```
<bars-chart chart-data="myData"></bars-chart>
```

Our directive will need some work, binding its scope to this new data source given by its parameter chart-data

```
scope: {data: '=chartData'},
```

See everything (HTML, JS and CSS) in my Github Gist or watch it in action below:

Result Edit in JSFiddle

See it Live in CodePen!

So now we have it, a correctly place, semantically beautiful and modular data visualization with AngularJS and D3! what are your thoughts?

More from this category

« Simple solution for local Cross Origin Requests

3 Comments Odiseo.net 1 Login

Recommend 1 Share Sort by Best

Join the discussion…

Kabby · 2 years ago
Thank you for so clearly explaining this! Other explanations I have found online only confused me...your post was easy to follow and worked on the first try!
1 · Reply · Share ›

levelsoft · 2 months ago
Great starter tutorial - thanks a lot
· Reply · Share ›

brentchow · 8 months ago
Thanks! This was great!
· Reply · Share ›

ALSO ON **ODISEO.NET** WHAT'S THIS?

Simple solution for local Cross Origin Requests
4 comments • 2 years ago
Ivan G — A different and fantastic approach is to avoid the use of http server and use script tags with id's for those needed external scripts, as shown at http://stackoverflow.com/quest...

Front-end optimizations you can start doing right now
6 comments • 2 years ago
bebraw — Good tips! Thanks for sharing. Is a contextual selector (ie. $('input.myClass', '#profile-container')) equal to $('#profile-container').find('input.myClass')? Normally the …

Understanding AngularJS transclude in directives
2 comments • 2 years ago
odiseo42 — Thanks for noticing it! Indeed Wordpress messed with me when I copied that from StackOverflow, but now it's all fixed, thanks!

5 AngularJS Power Features that are awesome in non-trivial webapps
2 comments • 2 years ago
odiseo42 — Angular has very unique features and philosophy which set it apart from other frameworks, and the result is that

fixed, thanks!

which set it apart from other frameworks, and the result is that you have to get used your mind to the "angular way". For …

Subscribe Add Disqus to your site Privacy DISQUS