angular, 8 minutes, components, code architecture

Top 5 benefits after 6 months of component driven AngularJS development

Posted by Jurgen Van de Moere on November 22nd, 2014.

In February 2014 Google **announced** that they had published some new "**Best Practice Recommendations for Angular App Structure**".

For years I had been structuring AngularJS code in directories called `controllers`, `services`, `directives`, `filters`, etc as outlined by the initial AngularJS documentation and now this new document suddenly proposes a radically different way to structure our AngularJS applications?

In my daytime job as lead developer, it is my responsibility to make sure all (remote) team members can function optimally to deliver the best they can in the shortest amount of time with the goal of delivering the best possible value for money to clients.

So I had already invested a lot of time in creating a toolbox of **Yeoman** generators, **Grunt**/**Gulp** build processes and documentation to get new projects up and running quickly. It was a pain to maintain, but it worked, and above all it was quick.

On the other hand the new recommendations made a lot of sense and actually tackled problems that I had been repeatedly faced with during development, deployment and maintenance of previous AngularJS applications, where other team members eventually asked ***me*** to deploy ***their*** code because they weren't really sure where to put it.

Projects often ended up with a familiar set of directories like:

```
my-traditional-app/
├── public
```

```
│   ├── css
│   ├── img
│   ├── js
│   └── partials
├── src
│   ├── js
│   │   ├── controllers
│   │   ├── directives
│   │   ├── filters
│   │   └── services
│   └── less
└── test
    ├── e2e
    └── unit
        ├── controllers
        ├── directives
        ├── filters
        └── services
```

where each directory was filled with tens or evens hundreds of files as the project went on.

We really tried hard to create sensible hierarchical directory structures and to keep file names as descriptive as possible, but in reality it was very hard to grow and maintain, which becomes painfully clear when you have to update code in a project you developed yourself months ago.

If you can't quickly locate a piece of code in a project you developed yourself, how can you expect someone else to find it?

So six months ago I decided to take the plunge, dumped my current toolbox, got up daily at 4am to experiment with prototypes before work, and started applying the new recommendations to new projects.

It was a huge risk because entire teams were depending on my ***gut feeling***, but today I can tell you that the result is nothing short of **amazing**!

In the next couple of articles I will share some of the challenges we were faced with and how we overcame them while putting the new recommendations into practice.

> ***If you have no clue what component driven AngularJS development is: stay with me and I'll explain to the best of my abilities what it is and how we put it in practice.***

So what are the new Google recommendations about?

It's NOT about the actual code

The new recommendations are **NOT** about how you should write your actual code.

> ***Tremendous efforts are being made by the community to specify JavaScript/AngularJS coding guidelines:***
>
> - *"Google JavaScript Style Guide"*
> - *"AngularJS Styleguide", by @ToddMotto*
> - *"AngularJS Style Guide", by @John Papa*

It's about the file system structure

The new recommendations are a proposal to structure your application in such a way that related pieces of code are grouped into logical units called **components**.

So instead of putting files in directories by the nature of their function (views, controllers, directives, etc), you put them in the directory of the component that they are structurally part of.

So a structure like:

```
├── public
│   ├── ...
│   └── partials
│       └── login
│           ├── form.html
│           └── login.html
├── src
│   └── js
│       ├── controllers
│       │   ├── LoginCtrl.js
│       │   └── LoginFormCtrl.js
│       └── services
│           ├── auth.js
│           └── session.js
└── test
    └── unit
        └── services
            └── login
                ├── auth.spec.js
                └── session.spec.js
```

becomes a structure like:

```
└── components
    └── login
        ├── auth.service.js
        ├── auth.service.spec.js
        ├── session.service.js
        ├── session.service.spec.js
        ├── form.html
        ├── login.html
        ├── LoginCtrl.controller.html
        └── LoginFormCtrl.controller.html
```

where all files related to login are centralized in the `components/login` directory instead of scattered all over the place.

> ***Check out the original [Best Practice Recommendations for Angular App Structure] document for more examples and detailed technical motivations behind the structure.***

First impression

At first you may see no immediate value in this. I know I didn't.

Quite contrary: it almost felt to me as if the login component directory had now become a collection of functionally non-related files.

But I stubbornly kept going and restructured one of my existing web applications and the `components` directory finally looked like this:

```
└── components
    ├── data
    ├── header
    ├── homepage
    ├── footer
    ├── login
    ├── news
    └── portfolio
```

I wasn't sure whether I had done it ***right*** but it was my first working prototype.

One immediate benefit was that looking at the `components` directory provided a good impression of what was actually implemented inside the application.

But above all it proved to be easy to find pieces of code.

When I asked a designer in our team where he thought he would have to look to update the styles for the top navigation, he looked at the component list and said that he would probably have to update the header component.

He guessed it right!

That tiny moment was a major confirmation that I was heading in the right direction.

I was still a long way off from having something production worthy, but it was a promising start. Motivating enough to keep getting up at 4am!

Fast forward to the present

Since then I have literally experimented with dozens of approaches and technologies to end up with a structure that has meanwhile proved to be very succesful in production environments.

Top 5 benefits so far

Apart from the technical benefits outlined in "**Best Practice Recommendations for Angular App Structure**", here is a top 5 of the benefits we experienced so far:

Dramatically improved application design architecture

One of our application now contains more than 50 individual components and it is still easy (even for new developers that join the project) to get an

idea of what the application does and where code can be located.

Promotes code reusability

We now try to create most of our components in such a way that they can be used in multiple applications without changing a single line of code. We even use components in a Microsoft Windows 8.1 WinRT application

Top 5 benefits after 6 months of component driven AngularJS development
Top 5 benefits so far

To add one of our existing components to a new application, we just copy it to the `components` directory and we're done! It magically works! All scripts, stylesheets and static assets are available automatically.

Easy to remove components

Removing components is a piece of cake. Just delete the directory and all scripts, stylesheet rules and static assets are removed automatically. No traces are left behind so the (remaining) code is kept clean at all times.

Allows better teamwork

Because of the fact that each component is isolated in its own directory, team members can work together on the same project much more efficiently. There are a lot less merge conflicts and overall project progress becomes much more easy to track.

Conclusion

Component driven AngularJS development is still in its early stages but it

is incredibly promising.

Apart from its technical benefits, it highly increases productivity and offers a lot of practical advantages that benefit teams of any size.

As with any architectural design concept it takes a bit of time to get comfortable with it, but once you do, the benefits are amazing.

If you haven't already, I would highly recommend going through Google's "**Best Practice Recommendations for Angular App Structure**".

What's next?

Since component driven AngularJS development is an architectual topic, I have deliberately not included real code in this article.

In the next article however I will dig into some actual code and talk about the structure we currently use **inside** the individual components and how/why I got it wrong in my first attempt.

Stay tuned and have a great one!!

If you want me to notify you when the next article is available, please feel free to leave your email address below.
