Gil Fink Following
Hardcore web developer, @sparXys CEO, Pro SPA Development co-author, husband, dad and a geek.
Nov 17 · 5 min read

Why I'm Betting on Web Components (and You Should Think About Using Them Too)

Web Components

For years there is a long discussion about the Web Components standards. On one side, there are developers who think that there is a broken promise in the standard. Others think otherwise and the discussion continues until today (and will probably continue in the future).

In the past 6 years I delivered numerous sessions about HTML5 and Web Components and I'm trying to help web developers and companies to adapt the standards. In this post I'll share why I think you should still give Web Components a try (if you haven't yet).

· · ·

Houston, We Have a Problem

I'm a web developer since 2002 in the days that websites were built mostly with server side rendering, and ASP.NET web forms were the new cool kid in the block. In the last decade there has been a big shift from server to client side rendering. HTML, CSS and JavaScript became the backbone of many apps and nowadays you can find JavaScript everywhere. There are awesome libraries and frameworks that we use daily such as React and Angular and web development is considered cool.

But… Houston, we have a problem…

In the last 4 years I'm a freelancer consultant and I helped numerous projects both in development and in architecture. One of the biggest problems that some of my customers struggle with is what I call the "Framework Catholic Wedding".

What is that problem you ask? At the beginning of a project you always pick a framework and you build your entire project around it. This is of course a good thing, isn't it?

Let me tell you 3 real world stories from my clients:

- One of my clients, enterprise size company, started to build a new web project one year ago. In the app design phase I set with the project architect and we talked about the front-end framework issue. In that company all the web infrastructure was built with AngularJS 1.5. They invested a lot of time and effort in building various styled components/directives and the company management forced the architect to use this infrastructure even though Angular 4's release was already around the corner. Today, the project managers have a big problem to find developers who are willing to join the project that is invested heavily in AngularJS. They are also considering migrating to the new Angular version but that means they will have to migrate the infrastructure too, which is very costly and time consuming.
- Another client, enterprise size company again, asked me to help them to make a decision on how to approach the following problem. Their company acquired another company. The second company product was implemented with Angular 4 and their product was implemented using AngularJS. Their managers asked them to merge the two apps together in the next quarter (3 months deadline…). The suggested patch solution was to use IFrame with postMessage API which helped them to gain time to start migrating their app to Angular 4 using ngUpgrade module. Again, they need to invest a lot of time and effort in order to move forward.
- One of my clients has 2 apps which are written in one with React and other with Polymer 1. Today they want to create a third app

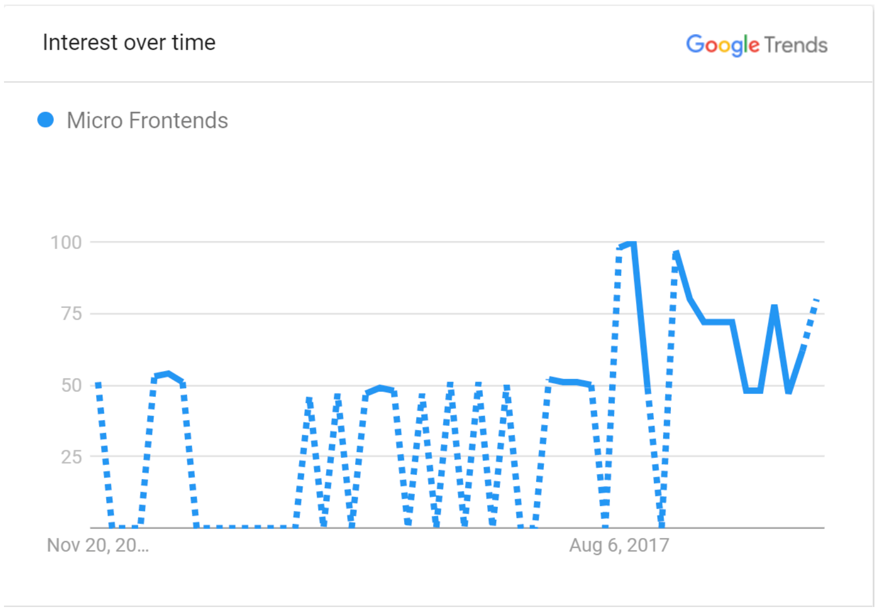
and share some of the functionality from both of those apps. Unfortunately, after I looked at the code base of both of the apps, it will be very difficult for them to reuse their code without investing time and development effort. Moreover, they will have to change their apps to support this change.

Can you spot the repeating theme in these stories?

All these examples are common and they are happening in companies as I write. These problems can happen in your company as well. Don’t get me wrong, libraries/frameworks such as React and Angular are good and vital to the web eco-system but choosing them as an infrastructure is a catholic wedding for the good and the bad.

Micro Frontends to The Rescue

In the last two years the term Micro Frontends started to be popular due to the problems I presented.



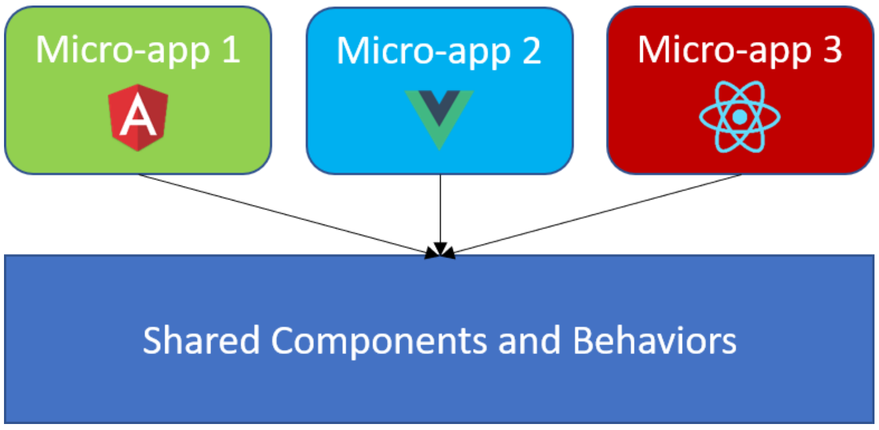
Micro Frontends in Google Trends

The term Micro Frontends is related to techniques, strategies and recipes for building a modern web app with multiple teams using different JavaScript frameworks.

Note: If you are not familiar with Micro Frontends, there is a very good introduction you can read here.

One of the options to build Micro Frontends is just to use the web platform and what it is offering. How does it work? The idea is to build

Web Components which are shared across the company development teams. The components need to be agnostic of any framework or library and can be used by any framework or library. So, how we can do it using the web platform? We can use the Custom Elements API.



Using Web Components for Building Shared Components

Creating custom elements answers the previous requirements but it forces the creation of an entire set of components. You will also have to craft data binding, component state management and more into your new components, don't you?

The Web Platform Community to The Rescue

In the last few months you can see a small Web Components renaissance. Companies such as Google (Angular Elements and Polymer 2) and Ionic Framework (Stencil) are building tools that make Web Components more approachable and performant. There are also new community ventures such as SkateJS and SlimJS that make the development of Web Components easier. All the new tools include framework goodies such as data binding and more which bridges the gap of web components usage. More then that, the community uses these tools to produce reusable web components that can be adopted in any framework.

All these projects and their effect in the development community makes me believe that there is hope for Web Components and that innovation in those web areas is still in progress and will continue.

. . .

Concluding Words Summary

In the post I tried to address some of the problems that companies face today and how Micro Frontends and Web Components might help to solve those problems.

Do you think Web Components are here to stay? I'd love to hear your thoughts.

Some resources to continue from here:

- Introduction to Micro Frontends
- Micro Frontends
- The broken promise of Web Components and Rob Dodson answer Regarding the broken promise of Web Components
- Web Components organization
- Polymer project
- Stencil—The magical, reusable web component compiler
- SkateJS project
- SlimJS project

Thanks to Uri Shaked for his valuable feedback before I published the post.

#UseThePlatform