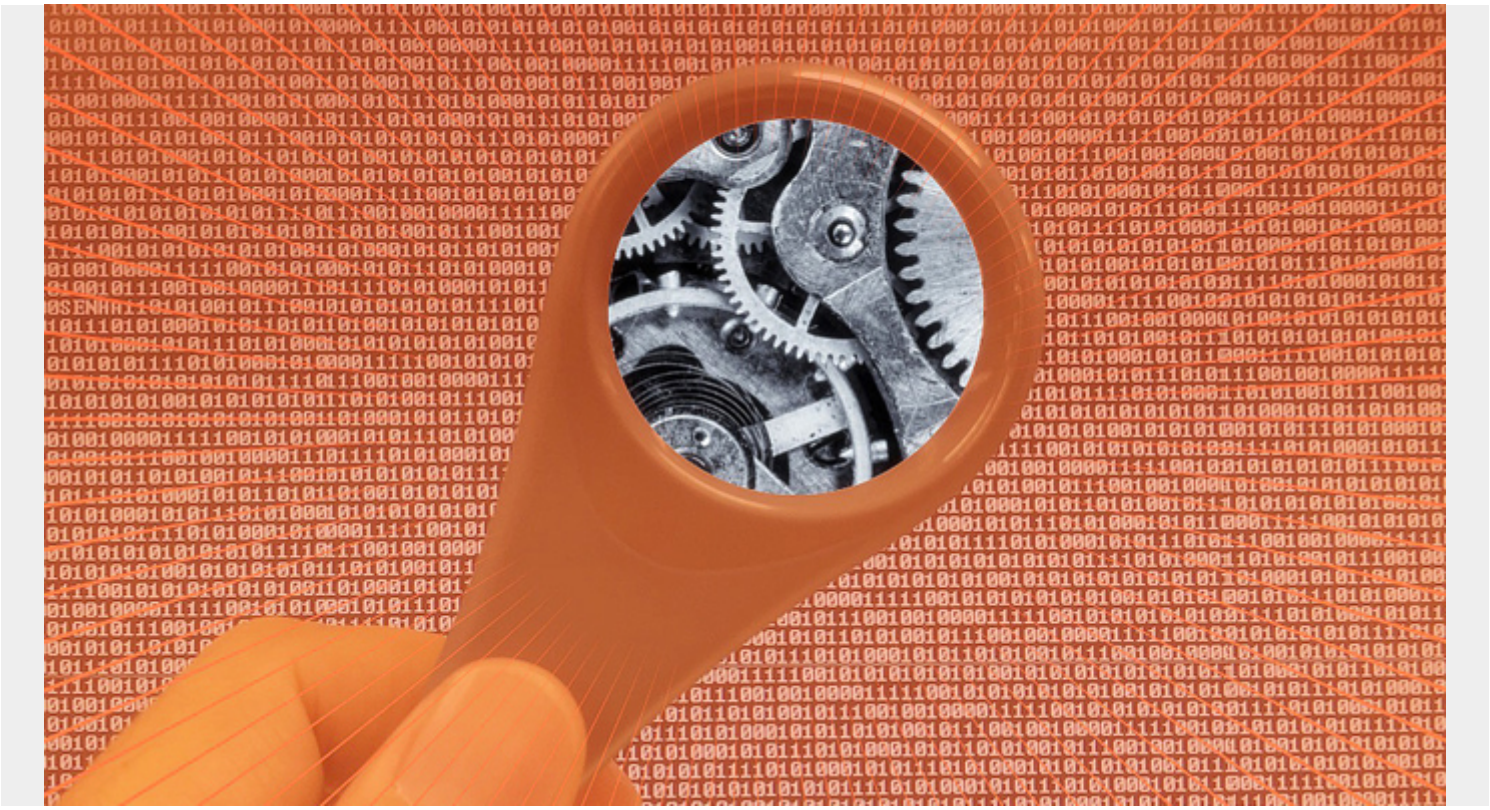
AN INTRODUCTION TO MICRO FRONTENDS



The concept of micro frontends, first introduced on the 2016 Thoughtworks Technology Radar, is becoming increasingly popular with engineers and development teams. Since its introduction, more and more teams have utilized this method of architecture to help manage the complexity that comes from having multiple teams contribute to the same frontend experience.

This concept has become particularly valuable as more organizations find themselves with large frontend apps. For most apps or websites, the frontend is continually growing, making it difficult and complex to maintain. Traditional development styles relied on frontend monolith, which just doesn't work for this type of app.

In contrast, micro frontends attempt to avoid the issues that come with frontend monolith by breaking up frontend monolith into smaller pieces. The concept of micro frontends achieves this by thinking about a site or an app as a set of features that are owned by independent teams. Each team independently develops end-to-end features.

A Rise in Microservices

While the use of micro frontends is a relatively new concept, the idea of microservices is not. Microservices is a development technique that has been widely adopted for backend services for a number of years. It's simply an architectural style that lets developers do independent deployment for certain parts of a platform without impacting other parts. A key to this design principle is having each component only being responsible for one, small task.

As more companies have utilized microservices, they've realized that this has led to bottlenecks and backups with user interfaces, which were not using the same techniques. The result was slowdowns, confusion, and organization-wide issues. Additionally, despite efforts to build cross-functional teams, there were limits to how effective these teams could be without being connected to frontend systems. It was to help address these issues, among others, that organizations began utilizing micro frontends.

The Development of Micro Frontends

Since its introduction in 2016 as a technique to assess, the concept of micro frontends has been subsequently identified by ThoughtWorks as a technique to try and, currently, as a technique to adopt. What this shifting classification means is that ThoughtWorks now sees this as a tested and proven concept that developers should use whenever appropriate.

Software engineer Michael Geers defines micro frontends well, stating that "the idea behind Micro Frontends is to think about a website or web app as a composition of features which are owned by independent teams. Each team has a distinct area of business or mission it cares about and specializes in. A team is cross-functional and develops its features end-to-end, from database to user interface."

As many teams have realized, with traditional architectural models, scaling frontend development to allow for simultaneous development across teams is complex and difficult. In fact, for apps with particularly large frontend components, it can be nearly impossible to do this well.

To help address this issue, a number of teams have adopted the architecture style of micro frontends with the objective of breaking the frontend monolith into smaller pieces that are more manageable for teams. When done correctly, this leads to increased efficiency and effectiveness for frontend teams. Perhaps one of the strongest arguments for adopting this style is the number of major companies that are currently using it, including Spotify, Upwork, Allegro, and HelloFresh.

The Benefits of Using Micro Frontends

Obviously, the key benefit of using the micro frontend architectural style is making frontend development less complex and more manageable. More specifically, though, micro frontends allow for smaller, more cohesive codebases that are easier to maintain. Additionally, this style allows for increased scalability, in large part due to having autonomous teams. It also makes it easier to update or rewrite frontend code than it is with other design styles. Plus, it makes it easier for independent teams to collaborate and to migrate from old to new apps.

While it clearly has many benefits, it's not without its downsides. Using this style can lead to duplication, an increase in the number of bytes that users have to download, and fragmentation of ways that teams work.

Nevertheless, leaders in the industry believe that the benefits outweigh the downsides, particularly in light of the trends of having increasingly large frontends and less significant backends. As a result, teams should find ways to shift to this style whenever possible.

Key Principles to Consider

It's important to note that making the transition from frontend monolith to micro frontends is not an easy one for many organizations. With that in mind, it's important to find ways to make this transition, allowing for flexibility while encouraging creative solutions to ensure that the shift is seamless and effective for the unique needs of every team.

When working to develop teams that handle features from end-to-end, there are a few things to keep in mind. First, you want each team to be able to develop and update their features without having to coordinate with other teams. Allowing for this autonomy is important for making developments, updates, and maintenance less complex. Second, you want each team to build independent, self-contained apps. As you work towards this, you should strive to have teams that don't share a runtime or rely on shared variables.

A third key principle to prioritize is developing naming conventions. You'll likely have some apps where isolation is not immediately possible. Having naming systems for CSS, local storage, and cookies will help to avoid confusion, making team ownership more clear.

Additionally, whenever possible, your teams should use browser events for communication instead of building a cross-team API. Finally, each team should ensure that their stack is resilient and performs well, even in the event that Javascript fails.

Implementation Techniques

While these key principles will help to make micro frontend development smoother and more effective, there are a number of different implementation techniques that teams can use. After all, it's a broad concept and there are many different ways to achieve the style and goals of micro frontend development.

Amidst the different approaches that teams take, there are some common trends that emerge throughout, namely a micro frontend for each page and a single container application that provides common page elements, including address authentication and navigation. These are generally utilized to bring different micro frontends together, regardless of the specific implementation approach adopted. A few of the more common implementation techniques that teams use include:

- Using single-SPA Meta Framework, which allows for a combination of multiple frameworks on the same page without having to refresh
- Having multiple single-page apps with different URLs that have components for shared functionality
- Using IFrames to isolate micro apps; IFrames use libraries and window.postmessage APIs for coordination
- Using a shared events bus to communicate with each module's unique framework
- Integrating modules with Varnish Cache
- Utilizing web components for integration

Conclusion

As more teams make the shift to micro frontends, the industry is seeing a transition in how teams function, develop new products, collaborate, and address issues. For many organizations, micro frontends help to make scalability, maintenance, and development significantly less complex. The

result is development teams that are more effective and efficient.