What we do
Go to overview
Customer Experience, Product and Design
Data and AI
Digital Transformation and Operations
Enterprise Modernization, Platforms and Cloud
Who we work with
Go to overview
Automotive
Healthcare
Public Sector
Cleantech, Energy and Utilities
Media and Publishing
Retail and E-commerce
Financial Services and Insurance
Not-for-profit
Travel and Transport
Insights
Go to overview
Featured
Technology
An in-depth exploration of enterprise technology and engineering excellence
Business
Keep up to date with the latest business and industry insights for digital leaders
Culture
The place for career-building content and tips, and our view on social justice and inclusivity
Digital Publications and Tools

/thoughtworks Menu

Perspectives
A publication for digital leaders
>
Digital Fluency Model
A model for prioritizing the digital capabilities needed to navigate uncertainty
>
Decoder
The business execs' A-Z guide to technology
>
Looking Glass
Bringing the tech-led business changes into focus
>
All Insights
Articles
Expert insights to help your business grow
>
Blogs
Personal perspectives from Thoughtworkers around the globe
>
Books
Explore our extensive library
>
Podcasts
Captivating conversations on the latest in business and tech
>
Careers >
Go to overview >
Application process
What to expect as you interview with us
>
Consultant life
Learn what life is like as a Thoughtworker
>
Grads and career changers

Micro frontends

Last updated : May 19, 2020

NOT ON THE CURRENT EDITION

This blip is not on the current edition of the Radar. If it was on one of the last few editions, it is likely that it is still relevant. If the blip is older, it might no longer be relevant and our assessment might be different today. Unfortunately, we simply don't have the bandwidth to continuously review blips from previous editions of the Radar.

Understand more ›

MAY
2020

Adopt ?

We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end services — largely neutralizing the benefits of microservices. **Micro frontends** have continued to gain in popularity since they were first introduced. We've seen many teams adopt some form of this architecture as a way to manage the complexity of multiple developers and teams contributing to the same user experience. In June of last year, one of the originators of this technique published an introductory article that serves as a reference for micro frontends. It shows how this style can be implemented using various

NOV
2019

Adopt ?

We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end services — largely neutralizing the benefits of microservices. **Micro frontends** have continued to gain in popularity since they were first introduced. We've seen many teams adopt some form of this architecture as a way to manage the complexity of multiple developers and teams contributing to the same user experience. In June of this year, one of the originators of this technique published an introductory article that serves as a reference for micro frontends. It shows how this style can be implemented using various web programming mechanisms and builds out an example application using React.js. We're confident this style will grow in popularity as larger organizations try to decompose UI development across multiple teams.

APR
2019

Adopt ?

We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a frontend monolith — a large, entangled browser application that sits on top of the backend services — largely neutralizing the benefits of microservices. Since we first described **micro frontends** as a technique to address this issue, we've had almost universally positive experiences with the approach and have found a number of patterns to use micro frontends even as more and more code shifts from the server to the web browser. So far, web components have been elusive in this field, though.

MAY
2018

Trial ?

We've seen significant benefits from introducing microservices architectures, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create front-end down into its features, and each feature is owned, frontend to backend, by a different team. This ensures that every feature is developed, tested and deployed independently from other features. Multiple techniques exist to recombine the features — sometimes as pages, sometimes as components — into a cohesive user experience.

MAR 2017

Assess ?

We've seen significant benefit from introducing microservice architectures, which have allowed teams to scale delivery of independently deployed and maintained services. However, teams have often struggled to avoid the creation of front-end monoliths—large and sprawling browser applications that are as difficult to maintain and evolve as the monolithic server-side applications we've abandoned. We're seeing an approach emerge that our teams call **micro frontends**. In this approach, a web application is broken up by its pages and features, with each feature being owned end-to-end by a single team. Multiple techniques exist to bring the application features—some old and some new—together as a cohesive user experience, but the goal remains to allow each feature to be developed, tested and deployed independently from others. The BFF - backend for frontends approach works well here, with each team developing a BFF to support its set of application features.

NOV 2016

Assess ?

We've seen significant benefit from introducing microservice architectures, which have allowed teams to scale delivery of independently deployed and maintained services. However, teams have often struggled to avoid the creation of front-end monoliths—large and sprawling browser applications that are as difficult to maintain and evolve as the monolithic server-side applications we've abandoned. We're seeing an approach emerge that our teams call **micro frontends**. In this approach, a web application is broken up by its pages and features, with each feature being owned end-to-end by a single team. Multiple techniques exist to bring the application features—some old and some new—together as a cohesive user experience, but the goal remains to allow each feature to be developed, tested and deployed independently from others. The BFF - backend for frontends approach works well here, with each team developing a BFF to support its set of application features.

Published : Nov 07, 2016

Download Technology Radar Volume 26

English | Español | Português | 中文

Stay informed about technology

Subscribe now

Visit our archive to read previous volumes

Go to archive

Company

Insights

Site info

Connect with us