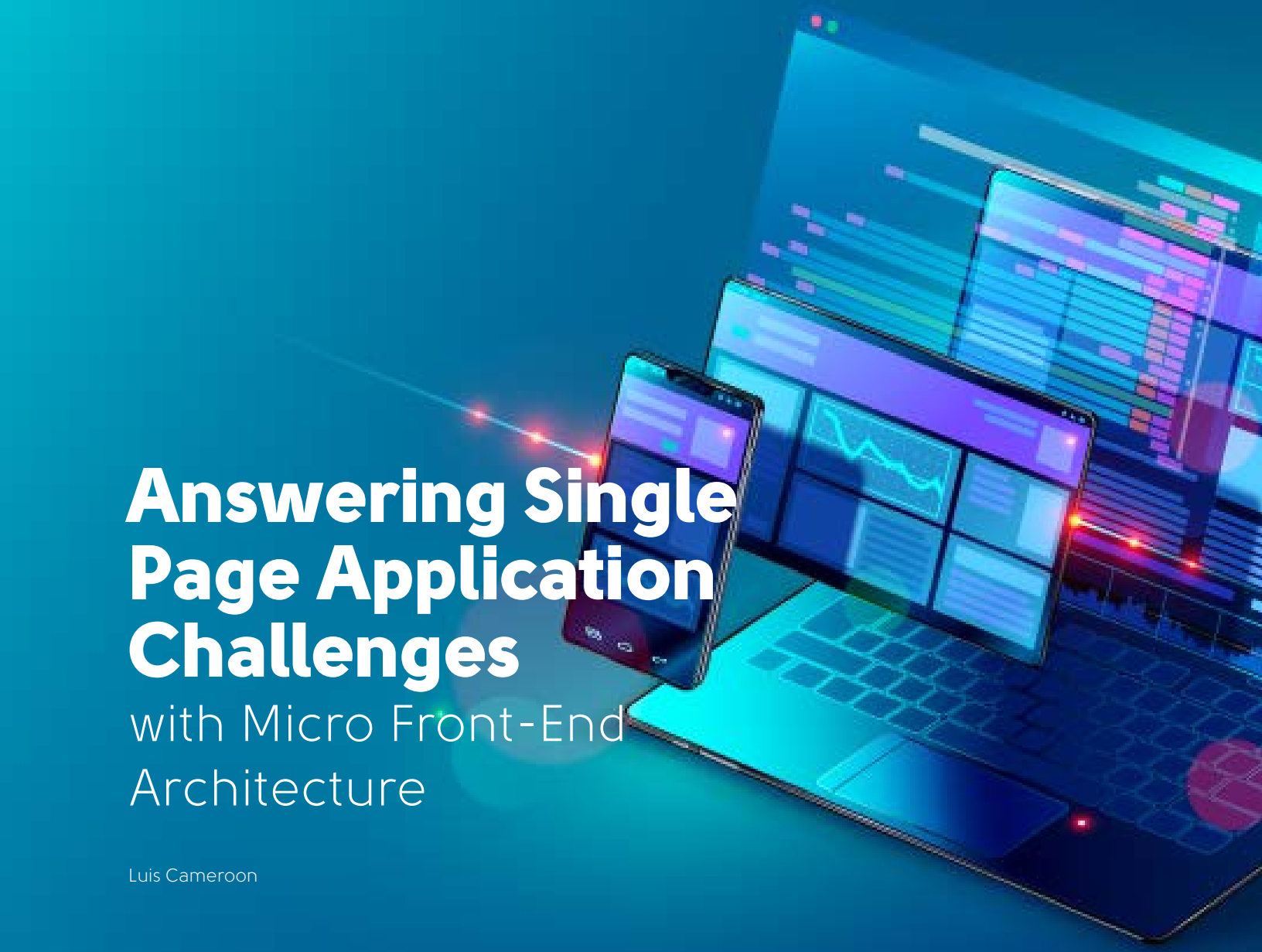
Luxoft
A DXC Technology Company



Answering Single Page Application Challenges

with Micro Front-End Architecture

Luis Cameroon

DXC TECHNOLOGY

Contents

Luxoft
A DXC Technology Company

Introduction

Micro front-ends apply the principles of microservices to the front-end. Unlike a single page application, this approach promotes agility and scalability, allowing the project team to deploy autonomously and enabling continuous delivery for web front-ends.

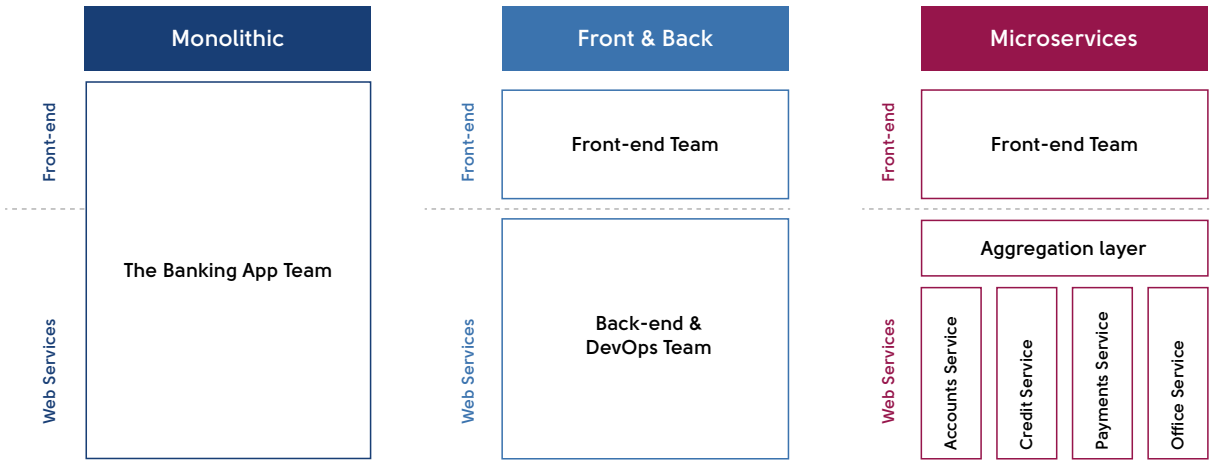
Luxoft
A DXC Technology Company

How Do Micro Front-End and Single Page Application Monoliths Differ?

A monolith front-end is the result of building a powerful and feature-rich web application, which sits on top of the microservice architecture. Over time, the front-end part of the application becomes huge and complex. Developed by a separate team, it gets more difficult to maintain. This type of application was favored before the advent of micro front-ends.

Micro front-end is a microservice approach to front-end web development. The idea is to break down the web application into smaller units based on screens representing domain-specific functionality, instead of writing a large, monolithic front-end application.

The micro front-end application is a combination of features owned by different independent, cross-functional teams that have the ability to develop end-to-end features from user interface to database. It gives the same level of flexibility, testability, and velocity as microservices.



DXC TECHNOLOGY

Luxoft
A DXC Technology Company

Problems with Front-End Monolith

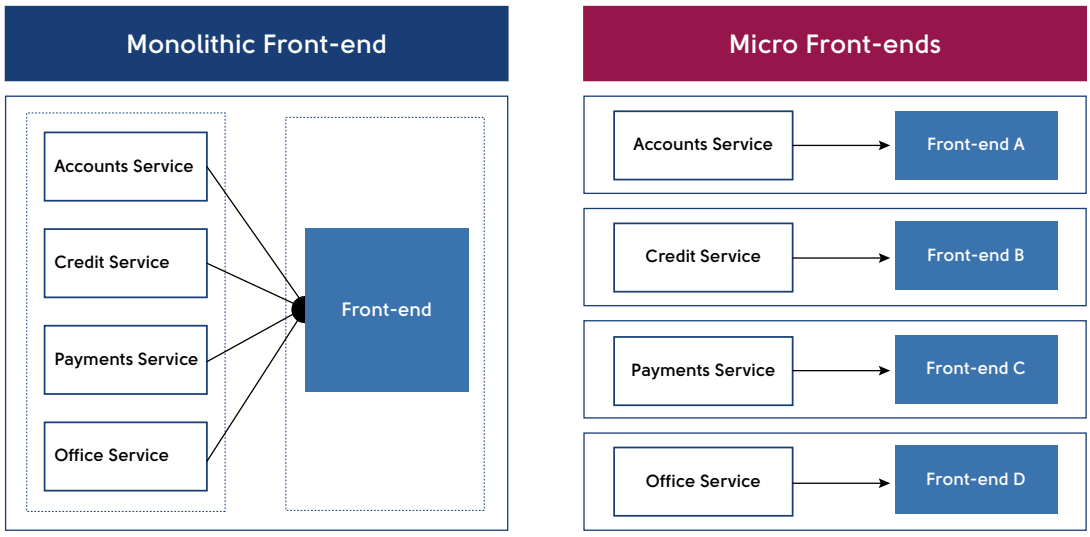
The flexibility promised by microservices can't be scaled across teams (i.e., the back-end team can't deliver business value without the front-end being updated).

There is an overhead of a separate back-end and front-end team, which causes the entire front-end to be updated and retested for a change in the application programming interface (API) of one of the services.

In a single page application, all files are bundled together and rendered on the browser. This file size is gigantic.

As applications grow, so do the features that need support. With multiple teams contributing to a monolithic application, development and release coordination is tedious.

Newer frameworks and libraries offer considerable performance improvements and innovations for the front-end space. However, the onerous task of upgrading a monolithic application and/or making it interoperate with these new frameworks and libraries, often compromises the ability to ship new features at existing release rates.

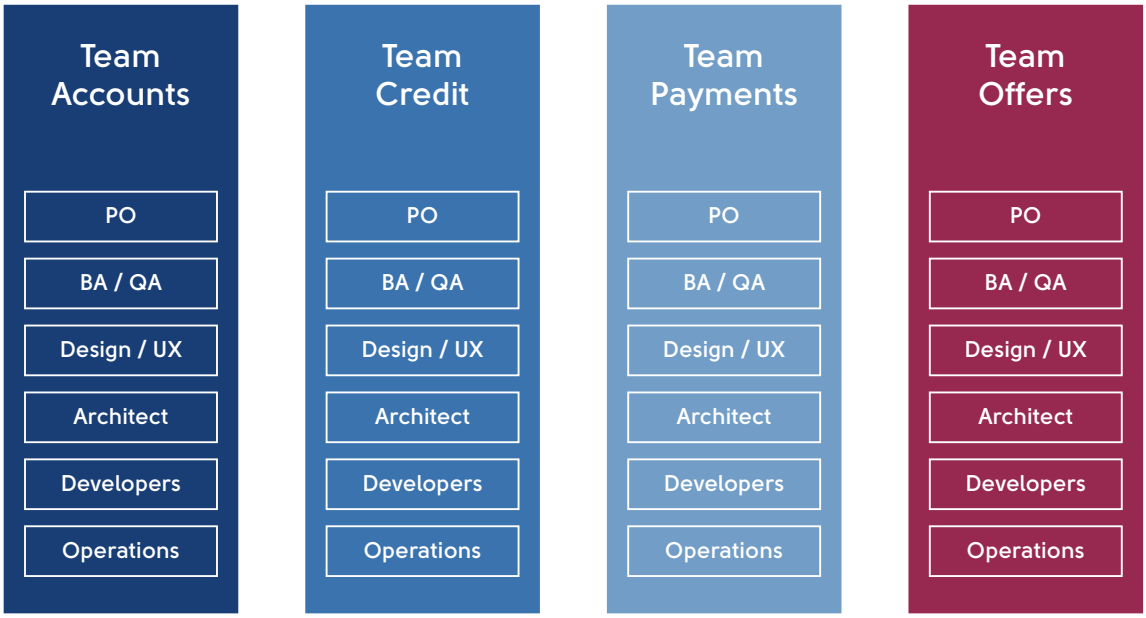


Luxoft
A DXC Technology Company

Why Micro Front-Ends Matter

Web browsers are getting increasingly powerful and front-end applications are handling more business logic than ever. This brings scalability problems to the applications as front-end teams and business requirements grow. Since maintaining the monolith application is expensive, web app developers have started taking a different approach to solving the bottleneck — the micro front-end.

End-to-End Teams with Micro Front-Ends



A cross-functional team also supports the rapid growth of individual skills across the team.

Luxoft
A DXC Technology Company

Benefits of Micro Front-Ends

The key advantages of micro front-ends architecture over a monolith are:

- **Gives back release autonomy and time to teams**
  By breaking features from the monolith into separate micro front-ends, teams enjoy increased autonomy and flexibility when releasing products and features. Teams don't need regression testing for their colleagues' changes in production. Small changes and testing become simple.

- **Independence**
  Individual development teams can choose their own technology. Not having to rely on the entire codebase reduces dependencies and scope, enabling teams to onboard and deliver faster. This creates more time for innovation without fear of breaking other teams' features.

- **Highly scalable and better performing web app**
  A loosely coupled architecture with established global standards makes it easier to add new features or spin up teams when needed. Since each app is fragmented into its own micro front-end, if a single feature (one micro front-end) on an enterprise app isn't loading fast, it has no effect on the performance of the rest of the application. It also makes it possible for certain parts of a web page to load faster, allowing users to interact with the page before all features are loaded or needed.

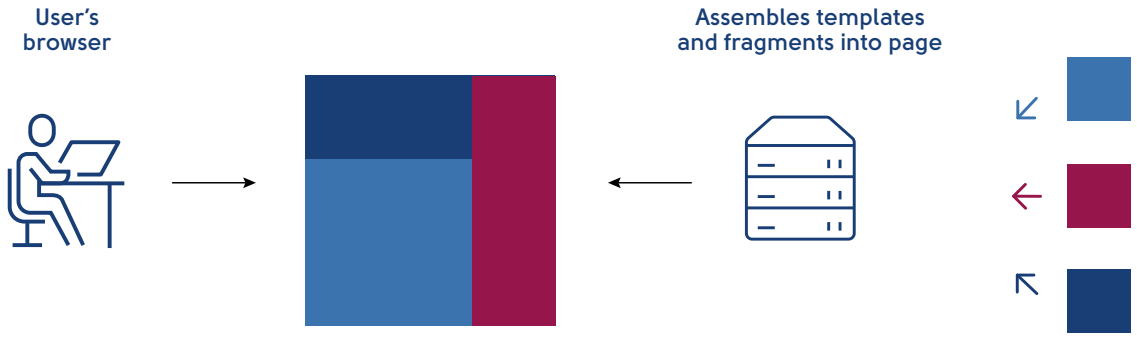
Luxoft
A DXC Technology Company

Which Technique Is Best for Micro Front-End Integration?

The options to be considered are: server-side integration, build-time integration, run-time integration via iframes, and run-time integration via script.

1. Server-Side Integration

Server-side integration is the weapon of choice for anything dynamic that should also be server-side rendered. This method excels in perceived performance.



There are various ways of realizing server-side composed micro front-ends. Using a layout engine like Podium presents a scaling approach without too much trouble. However, the dynamics of micro front-ends may be difficult to tame with a central layout engine. Here, approaches such as using a reverse proxy could be more successful.

The challenge of using a reverse proxy is that the local development setup becomes rather complicated. Quite often, the only possibility of actually seeing the integration live, is to deploy the micro front-end in question, or provide some hot loading capability for sustaining a local development environment.
The server-side integration works well for content-heavy sites like webshops.

Pros

- Best performance
- Dynamic loading
- SEO

Cons

- Framework integration
- Micro front-end isolation
- Development environment

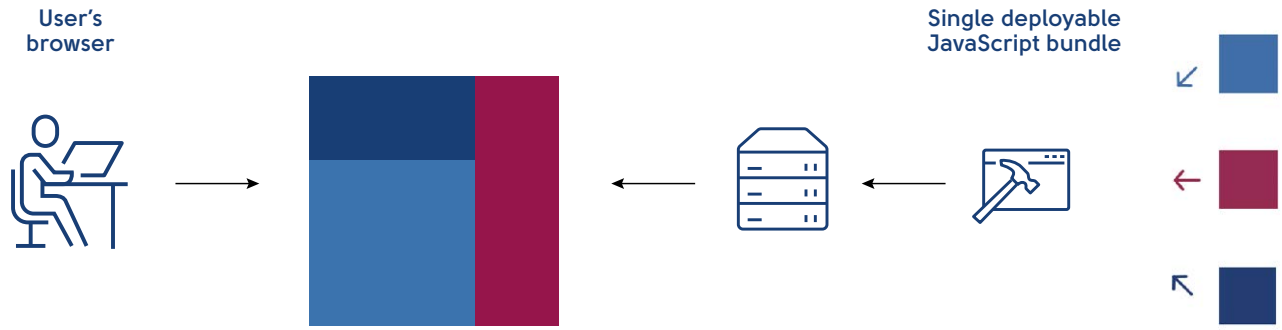
Example of libraries and solutions:

- Project Mosaic
- Podium

Luxoft
A DXC Technology Company

2. Build-Time Integration

The simplest and most reliable technique is build-time integration. This method is reliable because at build-time, it's already clear how everything works and how to join the different pieces to get a single deliverable.



This kind of mechanism is as old as writing software. Often different pieces are developed independently at various locations, then brought together for final assembly. Automation is vital here. The process works best when it triggers autonomously as soon as a piece changes.

For instance, when a single micro front-end changes, the whole application should be rebuilt. The number of micro front-ends might grow indefinitely, increasing the stress on the build server. Even if it doesn't, constant refreshing of the whole application could prevent caching and boost single page application performance.

Build-time integration works well in combination with server-side integration, or for smaller applications where only some well-defined parts are outsourced. One possible solution here is to use Webpack with the module federation plug-in.

Pros

- Type checking
- Runtime optimizations
- Easy for migration

Cons

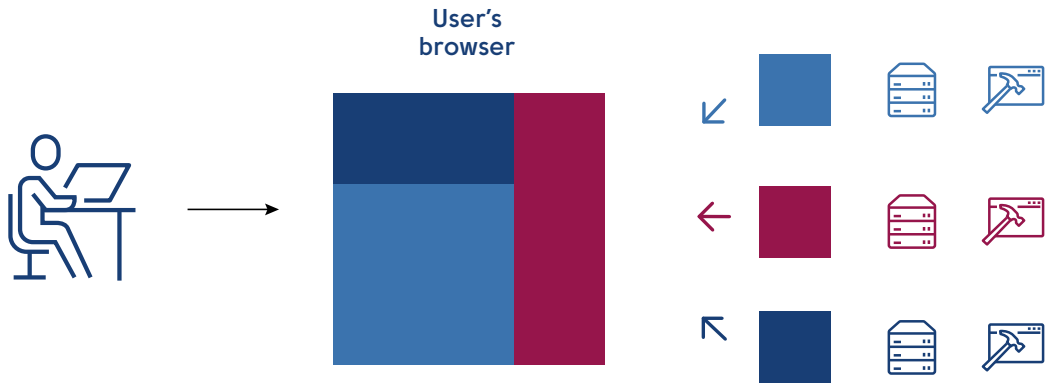
- Dynamic loading
- Build times
- Orchestration

Example of libraries and solutions:

- Lerna
- Bit
- Webpack 5 and Module Federation

Luxoft
A DXC Technology Company

3. Run-Time Integration



3.1. Via iframe

Joining micro front-ends at runtime has many advantages, but usually requires JavaScript. This presents a challenge to SEO and accessibility. While modern Google crawlers use a powerful JavaScript engine (in fact, they use a very recent version of Chrome to "see" the web), standard SEO rules still mandate a quick response and rapid rendering times. Runtime integrations often struggle here.

One exception is the inclusion of iframes. This can be prepared on the server-side, but it requires single elements (including their purpose and area of use) to be known centrally.

The best aspect of iframes is their isolation. This also beats alternatives such as shadow DOM or CSS modules, as nothing is shared with the hosting application. Since iframes come from a dynamic source, their content can be rendered server-side, too. This is necessary to some degree, as resources can't be shared and need to be loaded multiple times.

The runtime integration via iframes works well for pages using third-party content, where strong isolation is required. This technique has been in use for a long time (e.g., the first onsite PayPal integrations used it). Many chatbots and consent solutions use it because the provided boundaries shield one application from another.

Pros

- Strong isolation
- Full flexibility
- Web-native

Cons

- No sharing possible
- Difficult to sustain great UX
- Worst performance

Example of libraries and solutions:

- PostMate
- Microfronts

Luxoft
A DXC Technology Company

3.2 Via script

For the runtime integration of micro front-ends, using a plug-in mechanism enables the building of everything easily, while choosing all the right parameters centrally. The central location is usually called the application shell (app shell). It loads the scripts and evaluates their content.

While some frameworks offer great control over the distributed API, others are only script loaders or basic routing engines. However, all solutions in this space focus on developer experience.

This approach can give great flexibility but comes at a cost. Interesting applications such as VS Code have been built using a plug-in system, proving that a combination of a powerful app shell that comes with the majority of the UI, is as viable as a weak app shell that only orchestrates the different micro front-ends.

Alternatively, integration via script can bring micro front-ends in the form of web components. While this approach does have some advocates, it also comes with additional challenges — mostly to backward compatibility.

Pros

- Very dynamic
- Super flexible
- Best developer experience

Cons

- Weak isolation
- Requires JavaScript
- Efficient orchestration

Example of libraries and solutions:

- Single SPA
- Frint.js
- Luigi

Luxoft
A DXC Technology Company

Considering Micro Front-Ends

As front-end codebases continue to get bigger and more complex, there's a growing need for more scalable architectures. The ability to scale software delivery across independent, autonomous teams is important. As is the ability to draw clear boundaries that establish the right levels of coupling and cohesion between technical and domain entities.

While far from the only approach, there are many real-world cases where micro front-ends deliver these benefits. The technique is gradually being applied to legacy codebases as well as new ones. Whether or not micro front-ends are the right approach for you, we hope this will be part of a continuing trend where front-end engineering and architecture are treated with the seriousness that they deserve.

About the Author

Luis Cameroon
Principle Consultant

Luis has been designing and developing high-performing, scalable and innovative web-based applications for clients in the financial, energy and media industries for more than 15 years. He's fascinated by human-computer interaction and is a keen proponent of user-experience design.

www.luxoft.com

Luxoft
A DXC Technology Company

DXC TECHNOLOGY