I DON’T UNDERSTAND MICRO-FRONTENDS

@lucamezzalira
I don't understand micro-frontends.

Dan Abramov
@dan_abramov

4:48 AM · May 26, 2019 · Twitter Web App

130 Retweets 859 Likes

Dan Abramov @dan_abramov · May 26
Repeating to @dan_abramov

The problems they're supposed to solve sound to me like they're already solved by a good component model. So is this solving an organizational issue rather than a technical one? Such as if two teams can't agree on anything, even shared infra.

Dan Abramov @dan_abramov · May 26

OK, I just realized from reading the original doc that this seems to (at least partially?) be a Web Components architecture thing. Didn't mean to enter this debate 😳
I don’t understand micro-frontends.

Yesterday, after coming back from the walk with my dogs, I have seen a few notifications on Twitter where people tagged me asking to share my thoughts on the thread started by Dan Abramov regarding micro-frontends:

If you are following me, you know I’m very passionate about micro-frontends and I’m working with them for a while, I’m also keeping an open mind analysing different approaches and understanding their PROs and CONs.

If you don’t follow me and you are curious about the topic from a technical point of view just check my Medium page, otherwise, there are many other resources on micro-frontends, just searching on Medium or using your favourite search engine.
Luca Mezzalira
VP of Architecture at DAZN
Google Developer Expert
London JavaScript Community Manager
Architecture evolution

DB

Application Server

Single-Page Application
Architecture evolution

- DB
- Microservice
- DB
- Microservice
- DB
- Microservice
- API Gateway
- Single-Page Application
Who is using micro-frontends

fiverr
Lululemon
DAZN
Spotify
Hello Fresh
Starbucks

OpenTable
IKEA
Elsevier
edX
Cloud Factory

zalando
Microsoft
SAP
Smapio

illimity
New Relic
entando
Skyscanner
MICRO-FRONTENDS

DEFINITION
“Micro-frontends are the technical representation of a business subdomain, they allow independent implementations with the same or different technology choices. Finally they should avoid sharing logic with other subdomains and they are own by a single team.”
What is a micro-frontend?
MICRO-FRONTENDS

PRINCIPLES
Model around business domain

- Core Subdomain
- Supporting Subdomain
- Generic Subdomain
Model around business domain

Landing page
Sign in
Sign up
Catalog
Schedule
Search

Generic
Generic
Core
Model around business domain

Landing page  →  Landing page MFE

Sign in  →  Auth MFE

Sign up

Catalog

Schedule

Search  →  Discovery MFE
Model around business domain

London
Core Subdomain

Leeds
Supporting Subdomain

Katowice
Supporting Subdomain

Amsterdam
Generic Subdomain
Decentralisation

Tech Leads
Architecture

Team Mars

Team Moon

Team Venus
Decentralisation

1. Autonomous teams
2. Using the right tool for the right job
3. Components library
4. Standardization where needed
5. Code duplication vs shared code
Culture of automation
Independent deployment

- Landing Page 1.1.0
- Catalog 2.0.0
- My Account 2.4.1
- Catalog 1.9.2
Hide implementation details

PayPal Component

CONTRACT

Subscription
Micro-Frontend
MICRO-FRONTENDS DECISIONS FRAMEWORK
Key micro-frontends decisions

1. Define
2. Compose
3. Route
4. Communicate
Define a micro-frontends

- header
- image
- article description
- related articles
- footer
- landing page
- sign in
- sign up
- catalog
Horizontal VS Vertical split

1. Up front investment
2. Teams structure
3. Great for SEO
4. Testing challenges
5. Scalability challenges
6. Dependency management

1. Traditional development
2. Embracing JavaScript ecosystem
3. Dynamic Rendering for SEO
Compose micro-frontends
Route micro-frontends

ORIGIN

CDN

CLIENT
Micro-frontends communication

mysite.com/section?id=123
Isolate failure
THERE IS NO RIGHT OR WRONG
BUT ONLY THE RIGHT APPROACH BASED ON THE CONTEXT
MICRO-FRONTENDS @DAZN
Micro-Frontend @ DAZN

- HTML
- App.js
- Vendor.js
- CSS
DAZN implementation

1. Application startup
2. I/O operations
3. Micro-Frontends lifecycle
4. Communication between Micro-Frontends
DAZN implementation

1. A Micro-Frontend represents a business domain
2. A Micro-Frontend is autonomous
3. One Micro-Frontend loaded per time
4. No sharing between micro-frontends (*)
5. Technology agnostic

(*) with some exceptions
How bootstrap works

1. Call Startup service
2. Understand user status
3. Load and Mount a Micro-Frontend

CSS
App.js
Vendor.js
HTML
Components

1. Development time integration
2. APIs first
3. Own by a single team
MICRO-FRONTENDS
BEST PRACTICES
APIs first design principles

1. APIs are the first user interface of your application
2. APIs come first, then the implementation
3. APIs are self-descriptive
# Design System and Governance

<table>
<thead>
<tr>
<th>Design Tokens</th>
<th>Components Library</th>
<th>Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fonts</td>
<td>Button</td>
<td>Contract Testing</td>
</tr>
<tr>
<td>Colors</td>
<td>List</td>
<td>Visual Regression</td>
</tr>
<tr>
<td>Icons</td>
<td>Accordion</td>
<td>Performance Budget</td>
</tr>
<tr>
<td></td>
<td>Checkbox</td>
<td>NPM Private Registry</td>
</tr>
</tbody>
</table>
Duplication over abstraction
Duplication over abstraction

Core Library
- Lib A
- Lib B
- Lib C

Project Saturn
Project Moon
Wrong abstraction is more expensive than code duplication.
Performance

1. What is true for SPA is also true for Micro-Frontends
2. Pick the right approach to fulfill your needs
3. A user may download less code than a SPA
Business Evolution

Auth MFE

- Sign in MFE
- Subscribe MFE
SEO

Server Side Rendering

Dynamic Rendering
MICRO-FRONTENDS

AUTOMATION
Version of Control

Monorepo

- Improve code quality
- Always working with the latest libraries versions
- Continuous scaling investment
- Faster onboarding
- Trunk based development

Polyrepo

- Embrace conventional development methods
- Focus on part of the code
- Multiple branching strategies available
- Freedom to define CI/CD
Continuous Integration

1. Investment upfront
2. Fast feedback
3. Parallel and Sequential tasks execution
4. Fitness functions for testing architectural concerns
Fitness Function is a function that provides an objective integrity assessment of some architectural characteristics.
Continuous Integration

- Clone Git repo
- Run tests
- Static Analysis
- Analyse Permutations
- Build and Optimize
- Module swapping
- Bundle size check
- Artifacts saved
Components Deployment

- Button
- List
- Accordion
- Checkbox

Diagram:

- header
- image
- article description
- related articles
- footer
Strangler Pattern and Canary Releases

Lambda@Edge in DAZN: bit.ly/2SZ0Y24
Micro-frontends summary

- Orchestrate
- Compose
- Communicate
THANK YOU

t: @lucamezzalira
w: lucamezzalira.com
e: mezzalab@gmail.com