Micro-frontends Architecture

@lucamezzalira
Ciao :)  
Luca Mezzalira  
VP of Architecture at DAZN  
Google Developer Expert  
London Javascript Community Manager
1. From monolith to micro…
2. What is a Micro-frontend?
3. Technical implementations
This presentation could mine many of your believes on *software development*, please be **open minded** and **think at scale**
From monolith to micro... EVERYTHING!

Moving towards the micro-world!
Scaling our applications

- Monolith
  - DB
  - SPA

- Microservice
  - DB
  - Microservice
  - API Gateway
  - SPA

- Microservice
  - DB
  - Microservice
  - ???

- Microservice
  - DB
  - Microservice
  - ???
Impact within our teams

- Frontend uses the same codebase for all the teams
- Communication overhead for managing different part of the UI
- Infrastructure, Microservices and DBs almost autonomous
Scaling our applications with micro-frontends
Scaling our applications with micro-frontends

On the edge, Server side or Client side
Impact within our teams

- End to end autonomy within a business domain
- Freedom and responsibility
- Innovation without affecting the entire application
Impact within our organization

Cross-functional team + Product team = Business Domain
2. What is a Micro-frontend?

Let’s connect the dots...
Micro-frontends are the technical representation of a business subdomain, they allow independent implementations with same or different technology choices, finally they avoid sharing logic with other subdomains and they are owned by a single team.

From Domain Driven Design (DDD)
Domain and Subdomains (DDD)
Domain and Subdomains

The **domain is the problem to be addressed** with a software effort.

A domain can be decomposed into subdomains which typically **reflect some organizational structure**.

A common example of a sub-domain is Product Catalog
Subdomains

There are 3 different types of subdomains:

- Core subdomain
- Supporting subdomain
- Generic subdomain

Share nothing... and I mean **NOTHING**

Avoid to share components or code across different bounded contexts, abstraction could make our code more complex to maintain in the long run, the communication overhead could become a bottleneck for our organizations.
3. Technical implementation

The journey of a thousand miles begins with one step
Let's review some potential approaches

**Components**

OpenTable developers experience team created Open Components, an open source project composed by a registry of components where frontend and backend logics are wrapped inside small, self-contained units usable inside any view of their website.

OpenComponent is providing tools for quickly create new components like a CLI, more info: opencomponents.github.io

**Iframes**

An iframes composition is the choice made by Spotify with an event bus for coordinating the events across different iframes.

The desktop application mixes Web technologies with C++ codebase for the low-level operations.

**Server-side composition**

Zalando was one of the pioneers on microfrontends with Mosaic9 (www.mosaic9.org) in particular we need to highlight Tailor.js, an open source system for assembling the components on-demand on a backend layer written in Go.

At the end of 2018 Zalando moves to a server-side include system called “Interface framework”
Bootstrap is responsible for:
- application startup
- I/O operations
- routing between micro-frontends
- sharing configurations across multiple micro-frontends
Each **Micro-frontend** represents a **subdomain** matching the business structure.

- It’s **technology/framework agnostic**
- A **Micro-frontend** is **AUTONOMOUS**
- Inside a **Micro-frontend** the team can share components, code, styles or any other asset
- Independent building systems
- **1 Micro-frontend loaded per time**
Micro-frontend structure

- index.html
- app.js
- vendor.js
- style.css
How **Bootstrap** works

- Bootstrap loaded as first element and always available
- Tiny JS layer responsible to load micro-frontends
- Exposes APIs for different micro-frontends
How **bootstrap** works

```html
<html>
  <head>
    <script src='./bootstrap.js'></script>
    <style type='text/css'>...</style>
  </head>
  <body>
    <div>
      ...
    </div>
    <script src='./catalogue.js'></script>
    <script src='./cat-vendor.js'></script>
  </body>
</html>
```
How bootstrap works

Window.DAZN = {
    Lifecycle: {
        OnLoad: function(){...}
        Onunload: function(){...}
    },
    Localstorage: {...},
    ...
}

- DAZN object exposes methods and properties for all the micro-frontends
- Each micro-frontend has lifecycle callbacks available
- This object abstracts the platform exposing common APIs
Components

- Components available on NPM private repos
- Components need to work with any framework
- They expose a contract for the micro-frontend to interact with
- Components are owned by specific teams
- They can be shared within the same team
Based on some scenarios I can redirect the user to a version or another

Don’t need to do a big bang deployment

Canary releases or Blue Green deployment on Frontend!

5 teams
Onboarded on the same projects in 3 weeks

Over 100 developers
Working on the same project

100% Innovation, freedom and responsibility for each team!!!
The main challenges with scaling frontend applications are scaling the teams, reducing the communication overhead and innovate!
Microfrontends frameworks

Single-SPA
single-spa.js.org

FrintJS
frint.js.org
Single SPA

- Each SPA reacts to lifecycle methods (mount/unmount and bootstrap)
- Framework agnostic, helpers available for major frameworks
- Single-spa-config with method for registering different SPAs
Frint.js

Root App

Region

App 1

App 2

frnt.js.org
Frint.js

- React focused
- Very very opinionated
- Component-based architecture
- Reactive framework with Rx.JS
- Lazy loading of modules inside the Root app
- Provides a set of libraries to use in conjunction with the core library (routing or state management for instance)
- Routing is happening at URL level
Decompose by subdomain
https://bit.ly/2DUTQ1v

Subdomains and bounded context in DDD
https://bit.ly/1BPZfIW
Learning, testing and sharing

- Micro-frontends **FREE** report for **O’Reilly** (~80 pages)

- 28th February 3 hours online workshop on Safari Books Online: [bit.ly/2BjfhFw](bit.ly/2BjfhFw)

#OReillySACon
Rate today’s session

Effective enterprise architecture

Eben Hewitt (Sabre)
1:15pm-2:05pm Wednesday, February 6, 2019
Enterprise architecture
Location: Grand Ballroom West
Level: Intermediate
Secondary topic: Best Practice, Framework-focused

Who is this presentation for?
- Architects of all stripes, tech leads, senior developers, and managers

Prerequisite knowledge
- Experience as a senior developer or architect working on software projects

What you’ll learn
- Learn a holistic approach to architecture that explains how to bring business architecture, information architecture, data architecture, application (software) architecture together to have the best chance for your system’s success
- Explore a practical set of architecture practices to create winning technical architectural guidance
- Understand how architecture works effectively with development teams, management, and product management teams through the value chain
- Get usable templates you can start incorporating into your teams immediately

Session page on oreilysacon.com/ny

O’Reilly Events App
Thank YOU

You can find me at:
@lucamezzalira
luca.mezzalira@dazn.com