Building Micro-frontends

O'REILLY®

@lucamezzalira

Ciao :)

Google Developers Experts

DAZN

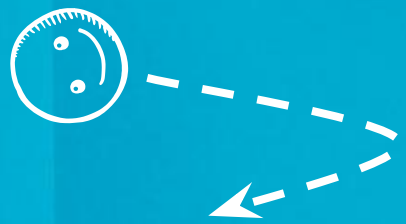
Luca Mezzalira

VP of Architecture at **DAZN**

Google Developer Expert

London Javascript Community Manager

Agenda

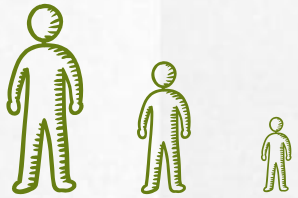


1. Introduction to **micro-frontends**
2. **Architectural** implementation
3. **Automation** for micro-frontends
4. Technical **implementations**
5. **Teams** organizations

1.

Introduction to MICRO-FRONTENDS

Moving towards the micro-world!



It's time to make a choice...

SPAs vs SSR

An hard choice to make

Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app.

https://bit.ly/1ON1SSo

SPAs PROs

. Unique codebase downloaded once

. Usually the client is intelligent (fat client)

. No need to reload the page at any point

. “Native experience” on certain devices

SPAs CONs

. If the application is complex could take long the first user interaction

. difficult to be indexed by crawlers

. complex handling a unique codebase in large and/or distributed teams in particular for multiple targets

Isomorphic applications (also called universal applications) are running the **code that usually runs on a client, in a server, pre-rendering the final result** so the client will be facilitated in the interpretation of that code

SSR PROs

. Less roundtrips to the server

. Optimised for crawlers

. Technology freedom (server and client side)

. Easier to optimise than a SPA

SSR CONs

. if content is not cachable the system could have scalability challenges on the server side

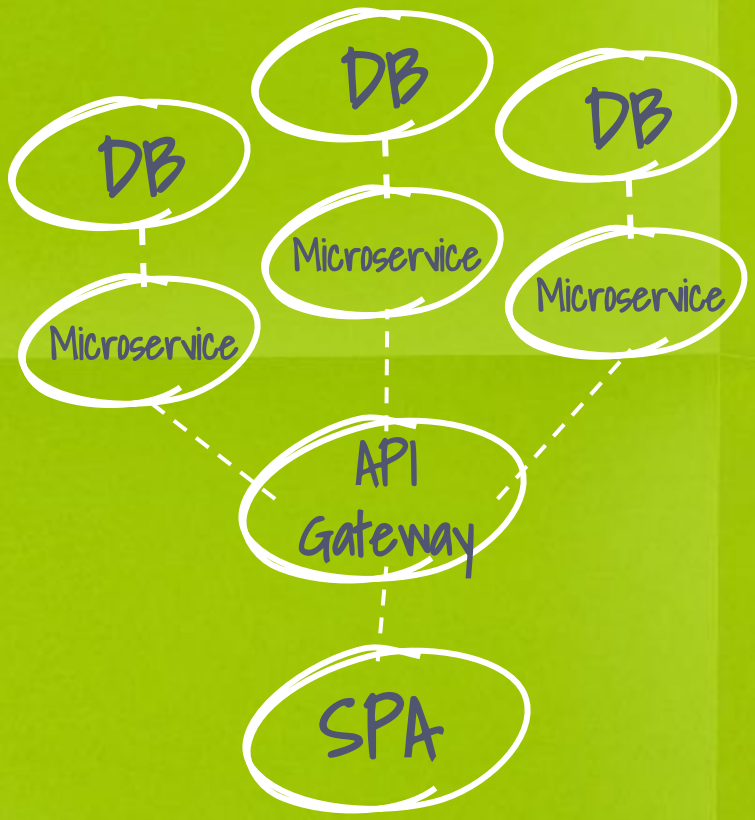
. Larger document size compared to SPAs

. slower time to first byte because the servers have to do more work

Can we have the best from those worlds?

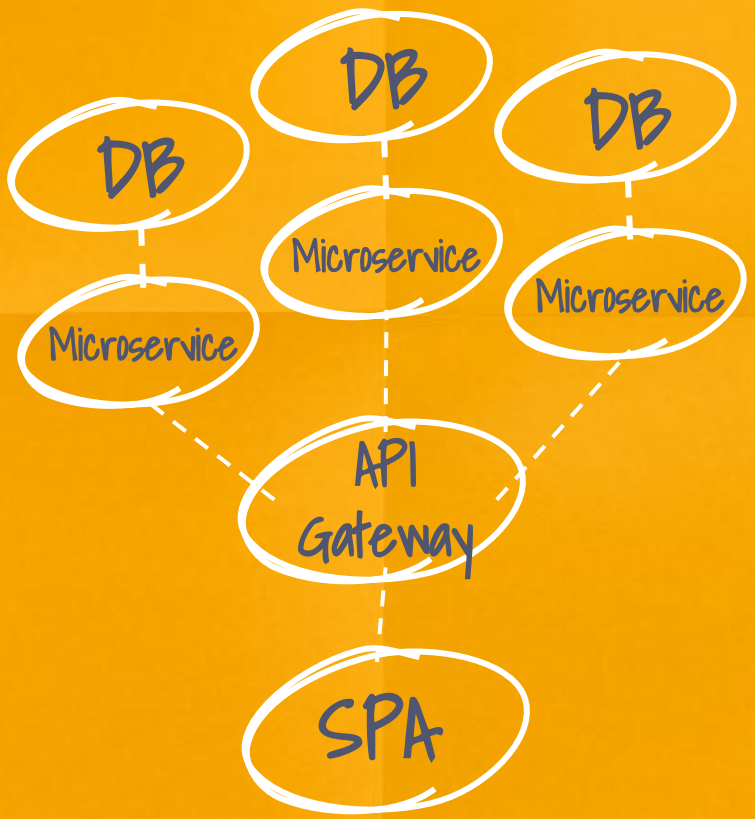
Scaling our applications

Impact within our teams

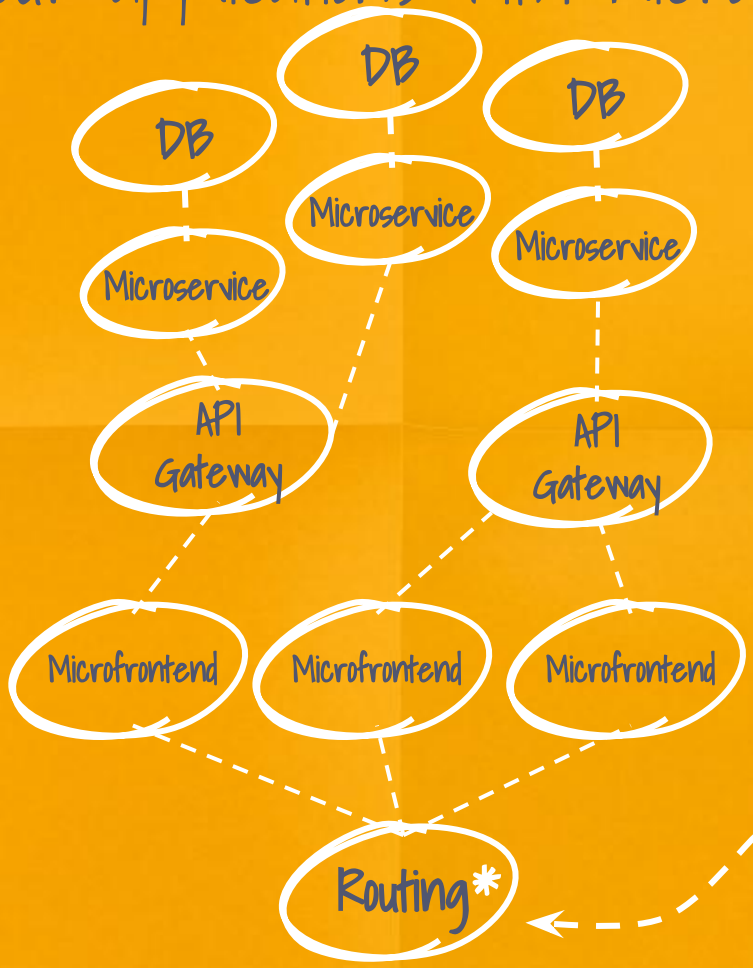


- Frontend uses the same codebase for all the teams
- Communication overhead for managing different part of the UI
- Infrastructure, Microservices and DBs almost autonomous

Scaling our applications with microfrontends

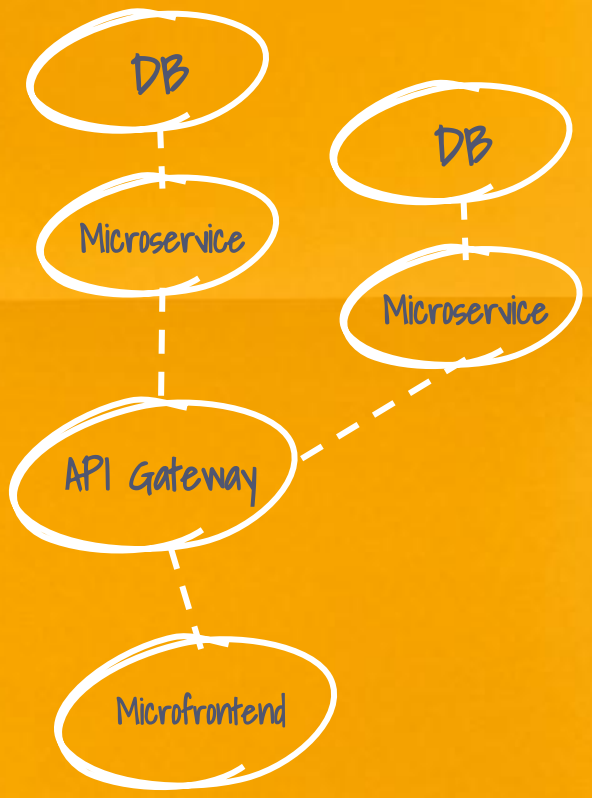


Scaling our applications with microfrontends



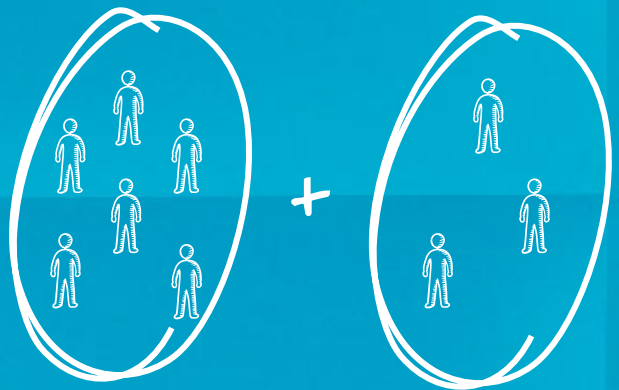
On the **edge**,
Server side
or **Client** side

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

\+

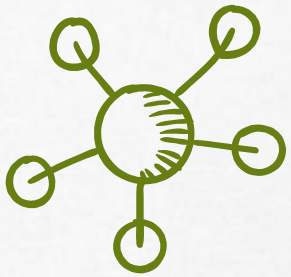
Product team

=

Business Domain

What is a **Micro-frontend**?

Let's connect the dots...



From Domain Driven Design (DDD)

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 own by a single team

Domain and Subdomains

(DDD)

Domain and Subdomains

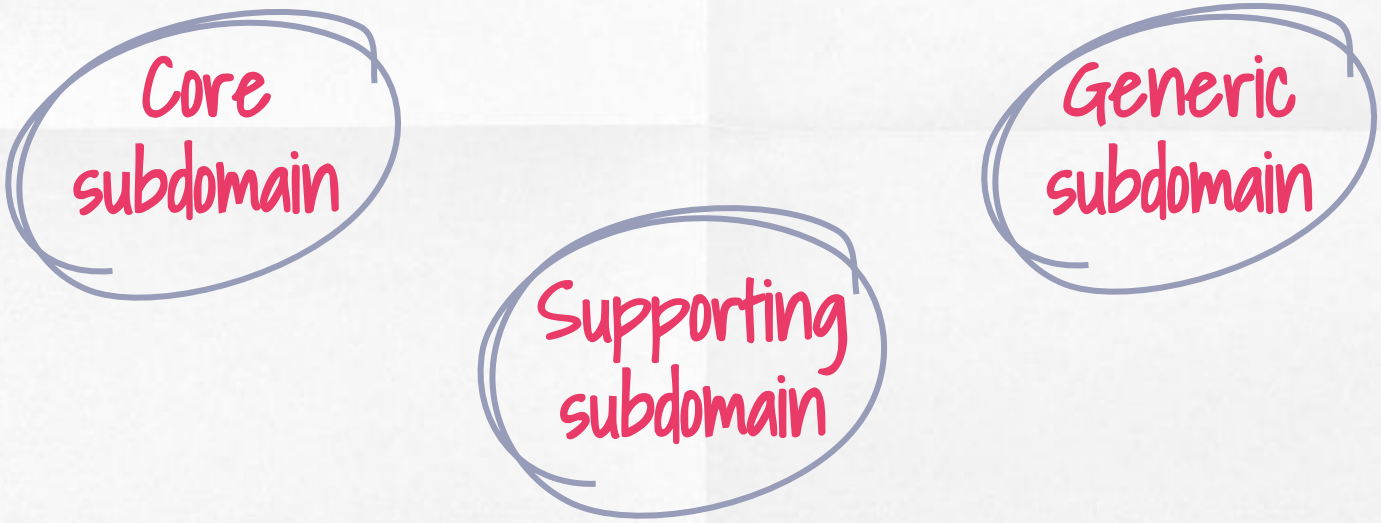
The **domain is the problem to be addressed** with a software effort.

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 type of subdomains:



https://bit.ly/2Au5Lio

How to identify your subdomains

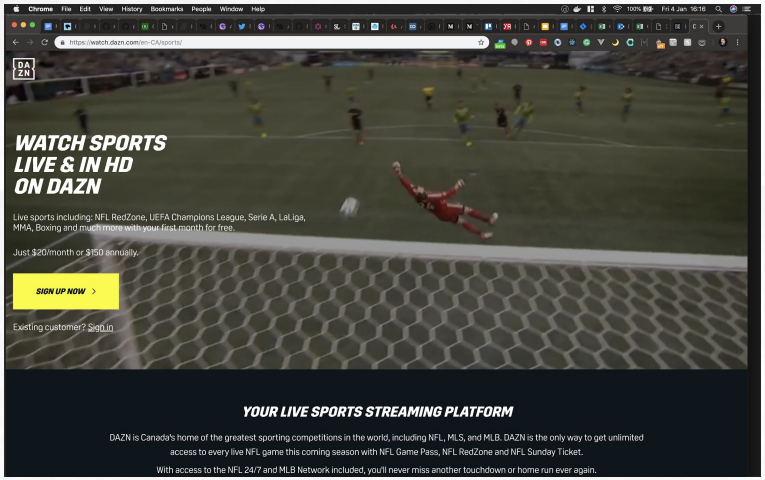
Subdomains are easily identifiable in existing applications via data

In green-field or new features projects subdomains are harder to recognise but not impossible

Subdomains evolves alongside your business, iterate on them with the product teams or your customers

The DAZN way

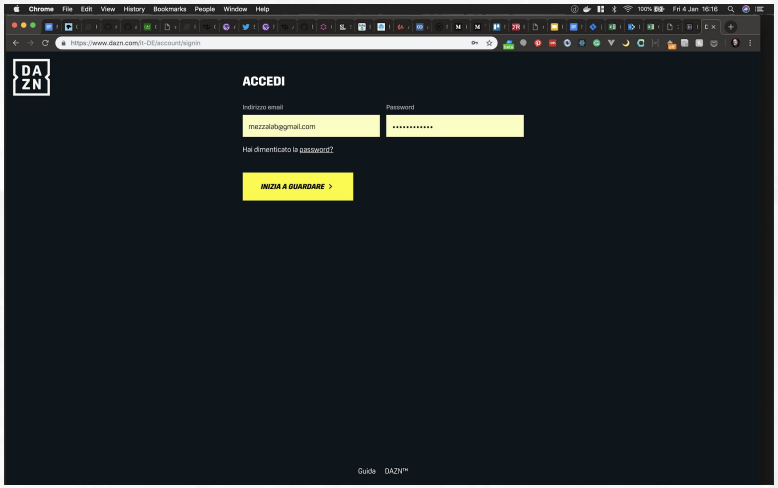
DAZN 2.0 - subdomains



Landing page

. Generic subdomain
. Single page, generated by a CMS
. Highly cacheable with tiny payload
. A lot of traffic stops here

DAZN 2.0 - subdomains



Onboarding

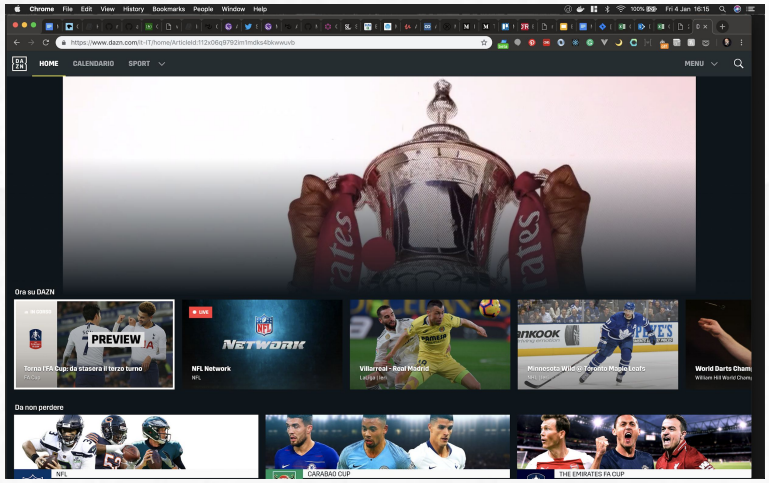
. Generic subdomain

. SPA

. Many permutations per market and device

. Sign in, Sign up, Retrieve Email, Retrieve Password and Redeem Gift Code

DAZN 2.0 - subdomains



Discovery

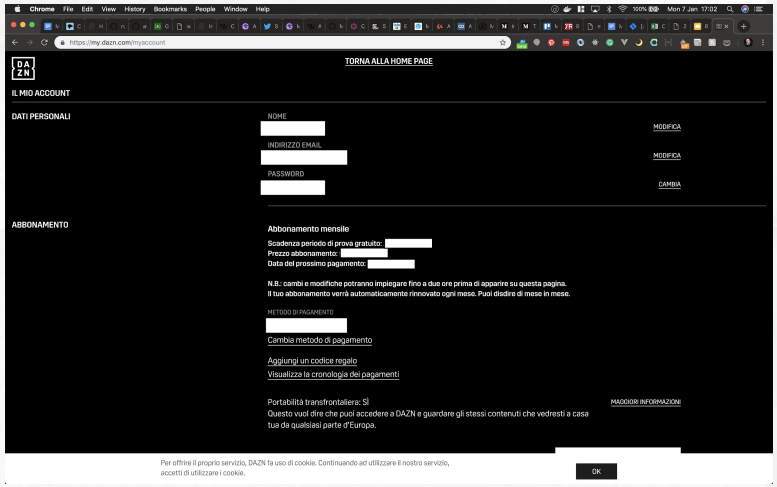
. Core subdomain

. SPA

. Video Player, Catalogue, Search, Sports Data

. Fast iterations with Product team considering it's our company's main focus

DAZN 2.0 - subdomains



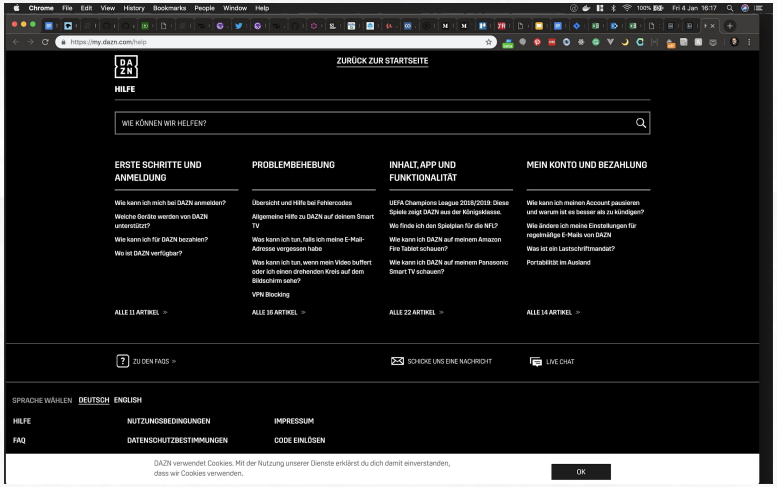
My Account

. Generic subdomain

. SPA

. Change user's details, payment methods, change application language...

DAZN 2.0 - subdomains



Help

. Generic subdomain

. SPA

. Data retrieved by different CMS compared to the landing pages

. This subdomain follows different rules than the others

From Domain Driven Design (DDD)

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 **own by a single team**

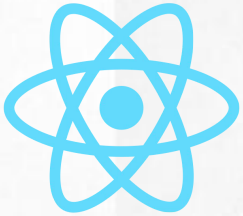
A programming language, frameworks or libraries are just

tools for expressing an intent.

The most important thing is having a

clear idea what we have to build and how do it

Technology independency



REASON

Technology independency

Tech independency is going to impact:

. best tech choice for the job
. hiring, retention and onboarding process
. building and deployment process
. company's standards
. developers morale

From Domain Driven Design (DDD)

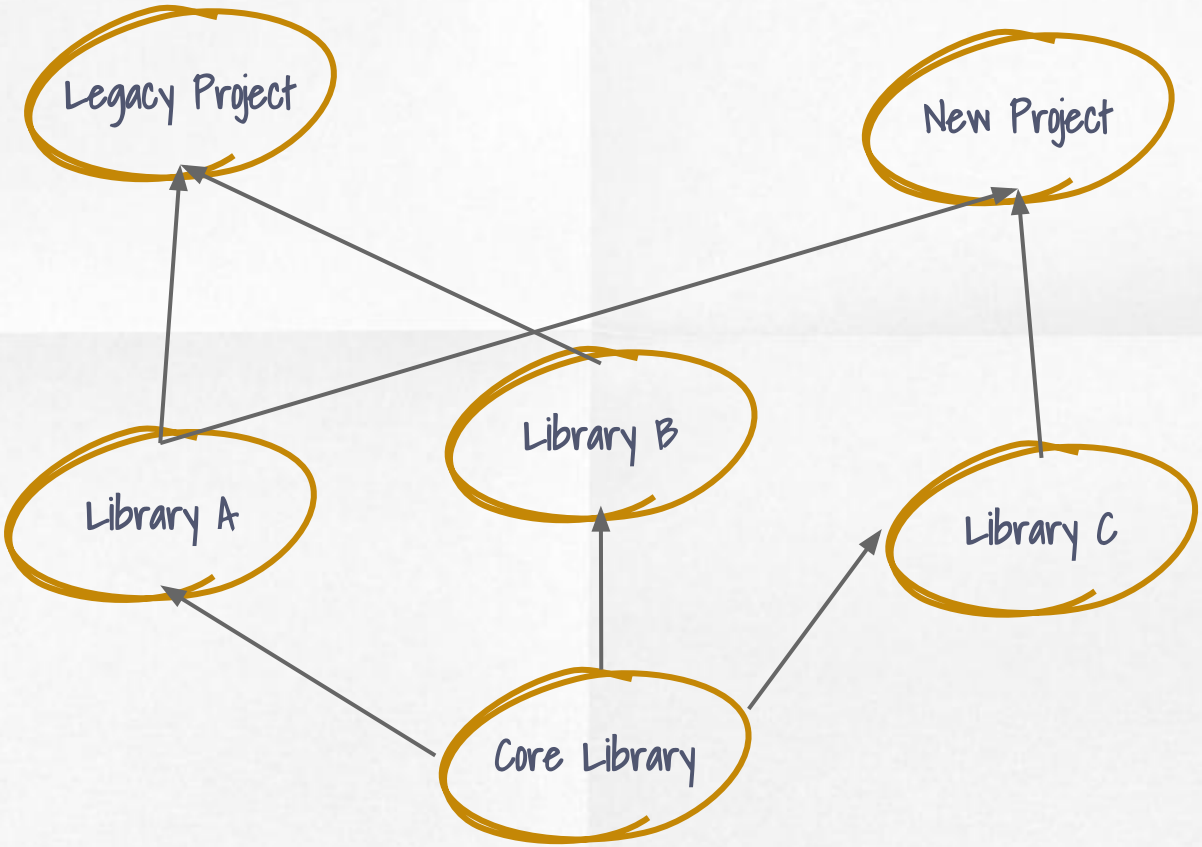
Micro-frontends are the technical **representation of a business subdomain**, they allow **independent implementations** with **same or different technology choices**, finally they **avoid sharing logic** with other subdomains and they are **own by a single team**

Share nothing... and I mean NOTHING

Avoid to share components or code across different subdomains, abstraction could make our code more complex to maintain in the long run, the communication overhead could become a bottleneck for our organizations

What about components or shared libraries?

Diamond anti-pattern



Questions questions questions

Who is responsible for the components or shared library?

How fast can we make a change without affecting anyone else?

Which technology are you going to use? Why?

How do you ensure fast iterations with distributed teams?

Web components

www.webcomponents.org

Web components at rescue

Lit-Element
https://lit-element.polymer-project.org

Stencil.js
https://stenciljs.com

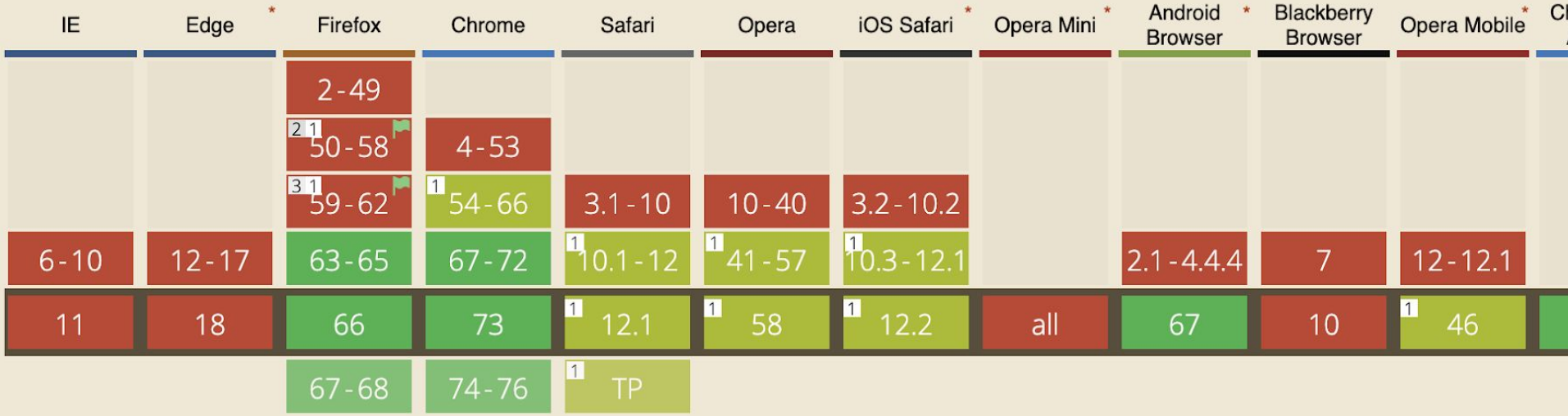
Skate.js
https://skatejs.netlify.com

Custom Elements (V1) - LS

Usage
Global

Method of defining new HTML tags.

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?



Notes | Known issues (0) | Resources (11) | Feedback

MS Edge status: In Development

1 Supports "Autonomous custom elements" but not "Customized built-in elements".

2 Enabled through the `dom.webcomponents.enabled` preference in `about:config`.

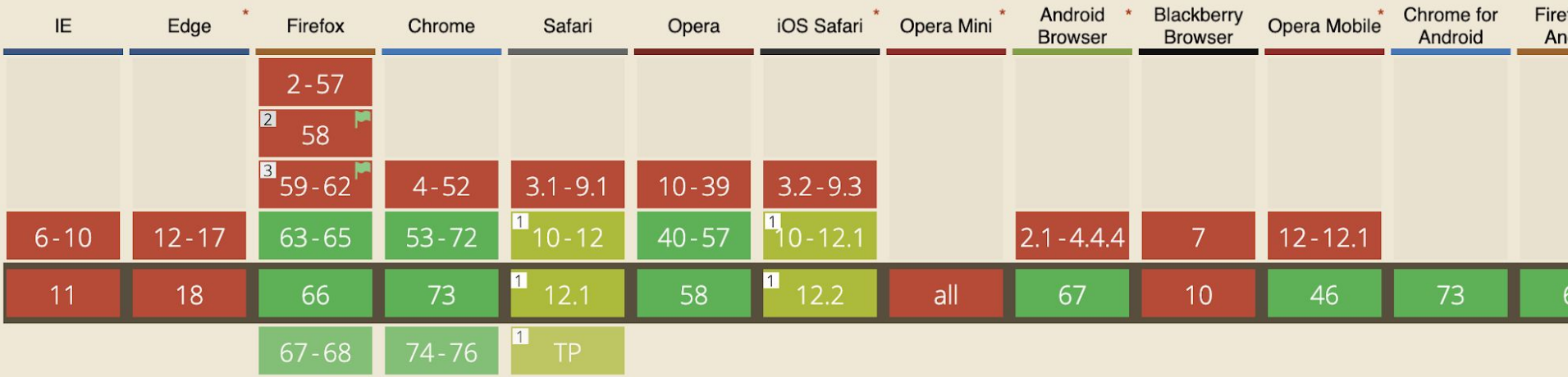
3 Enabled through the `dom.webcomponents.customelements.enabled` preference in `about:config`.

Shadow DOM (V1) - WD

Usage
Global

Method of establishing and maintaining functional boundaries between DOM trees and how these trees interact with each other within a document, thus enabling better functional encapsulation within the DOM & CSS.

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?



Notes | Known issues (0) | Resources (8) | Feedback

MS Edge status: In Development

1 Certain CSS selectors do not work (`:host > .local-child`) and styling slotted content (`::slotted`) is buggy.

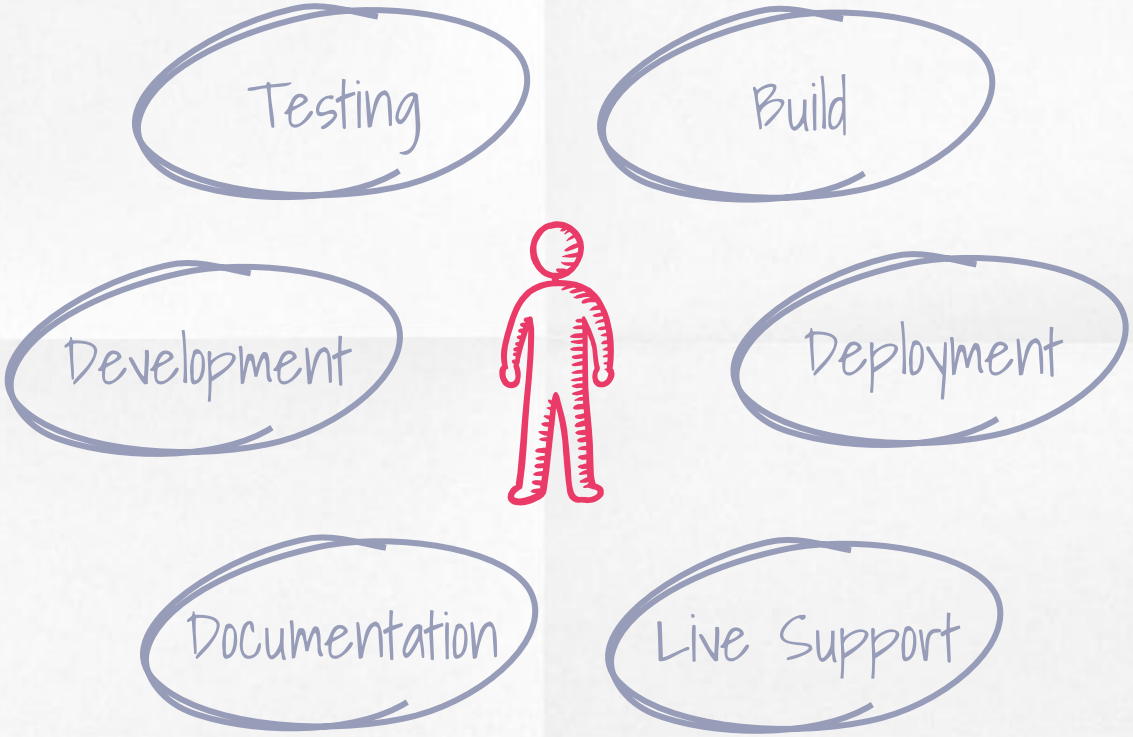
2 Enabled through the `dom.webcomponents.enabled` preference in `about:config`.

3 Enabled through the `dom.webcomponents.shadowdom.enabled` preference in `about:config`.

From Domain Driven Design (DDD)

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 **own by a single team**

Teams ownership



Teams ownership

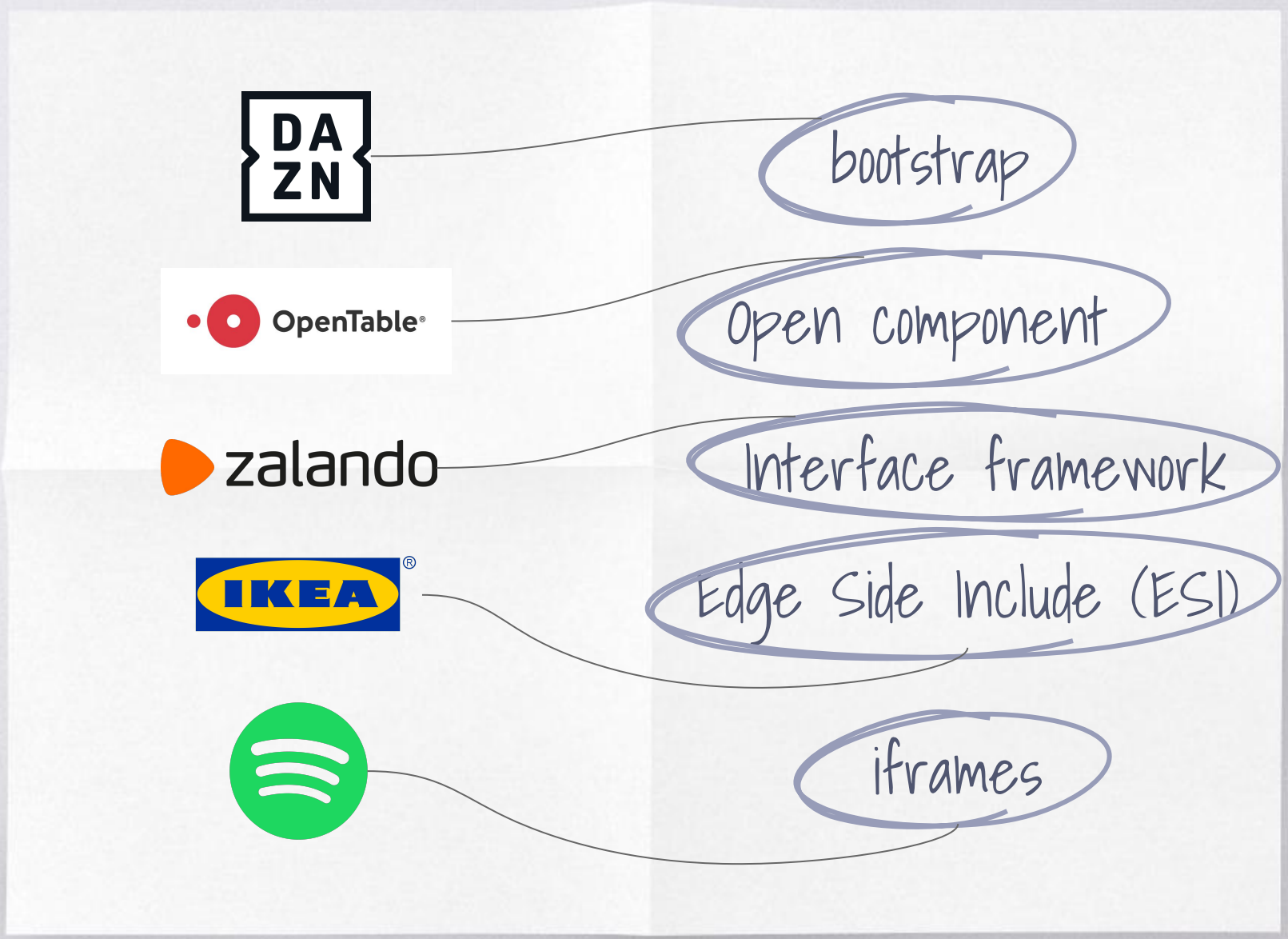
. A team can own 1 or multiple subdomains

. Teams could contribute to others team's micro-frontends

. Sharing experience between teams can help the company's growth

Which are the companies using **Micro-frontends** today?

Reality check time ;D


DAZN
bootstrap
OpenTable
Open component
zalando
Interface framework
IKEA
Edge Side Include (ESI)
iframes

Commonalities between those companies

. Large organizations

. Distributed teams

. Fast iterations and releases

. Looking for technical teams independency

. Massive growth

. Aiming for the best technical quality

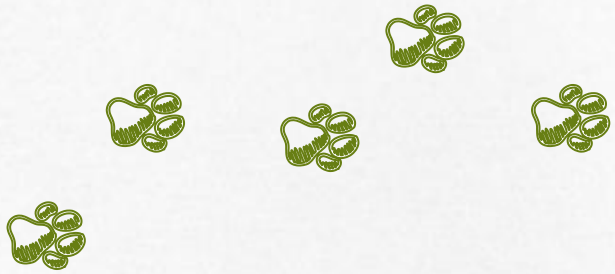
Micro-frontends are the technical representation of a business subdomain, they allow independent implementations with same or different technology choices, finally they avoid sharing logic with other subdomains and they are own by a single team


QnA

2.

Architectural implementation

The journey of a thousand miles begins with one step



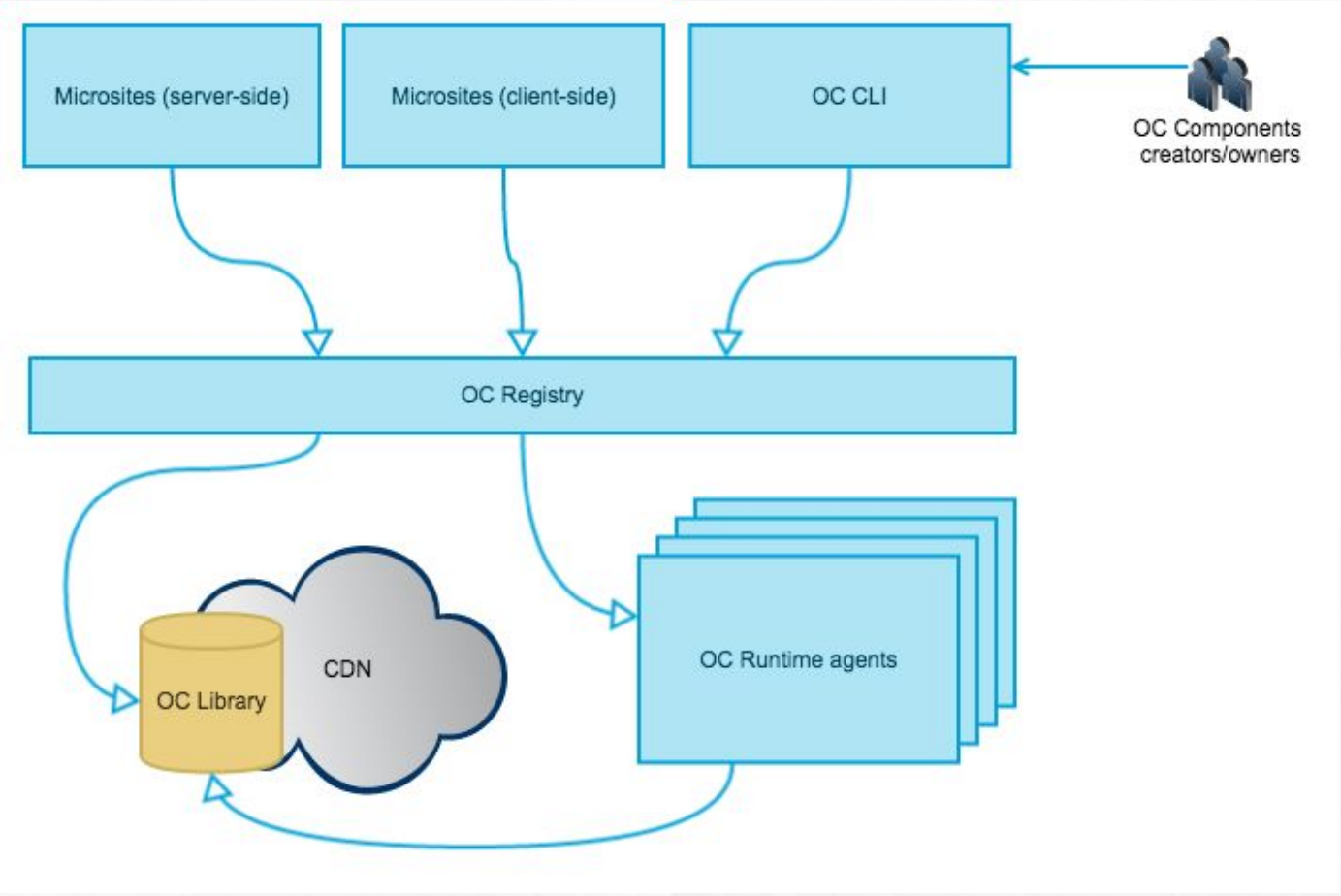
OpenTable

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

Open Components



Open Components

```
{
  "name": "base-component-handlebars",
  "description": "",
  "version": "1.0.0",
  "oc": {
    "files": {
      "data": "server.js",
      "static": ["img"],
      "template": {
        "src": "template.hbs",
        "type": "oc-template-handlebars"
      }
    },
    "parameters": {
      "name": {
        "default": "Jane Doe",
        "description": "Your name",
        "example": "Jane Doe",
        "mandatory": false,
        "type": "string"
      }
    }
  },
  "devDependencies": {
    "oc-template-handlebars-compiler": "6.0.8"
  }
}
```

Spotify

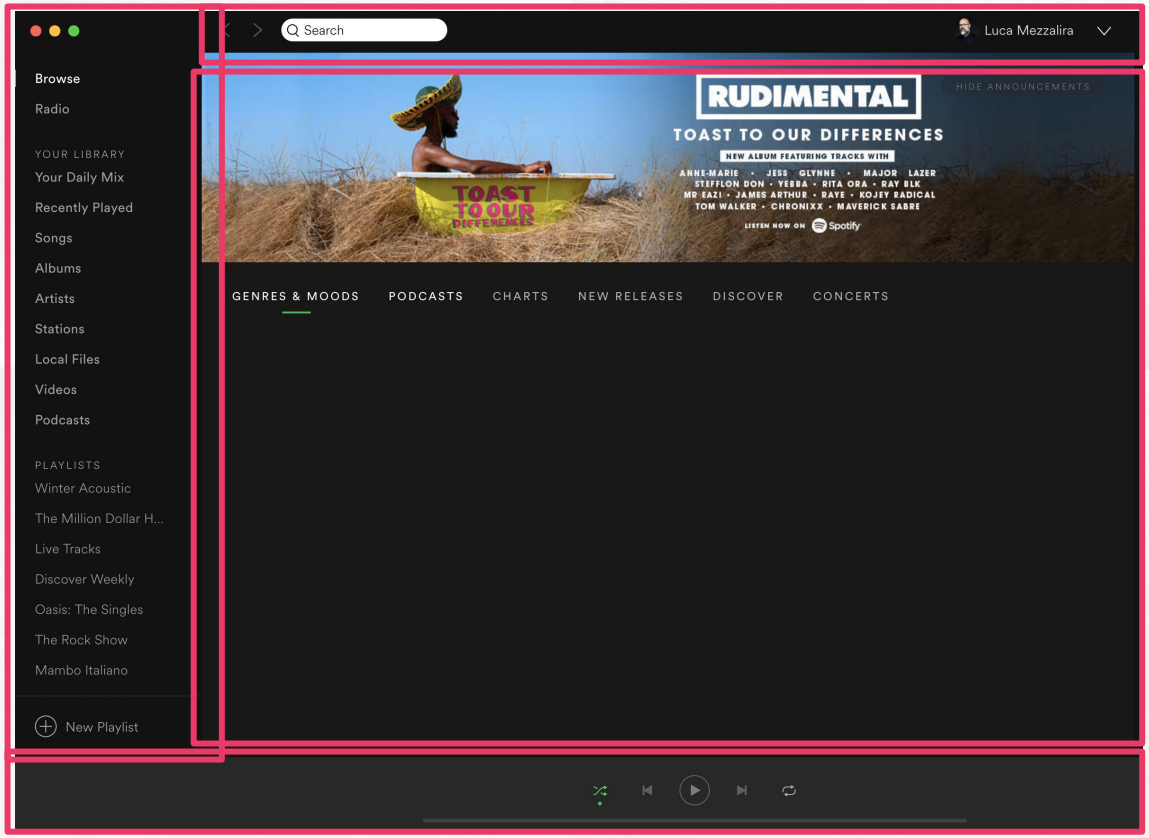
Iframes

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

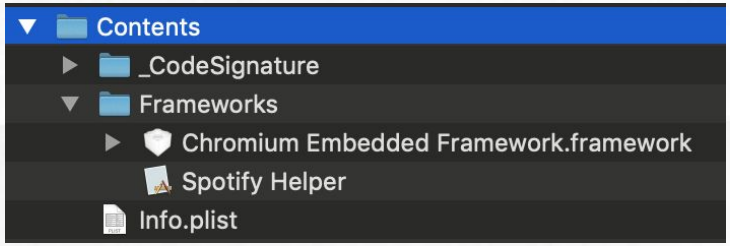
The desktop application mixes Web technologies with C++ codebase for the low-level operations

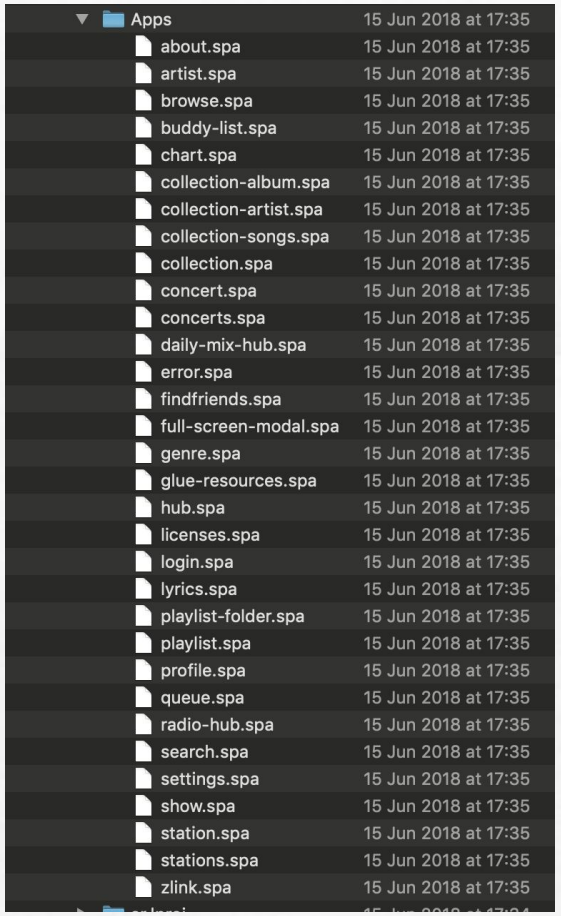
UPDATE (03/19) Spotify moved away from this implementation for the web application **https://bit.ly/2OGVPIe**

Iframes

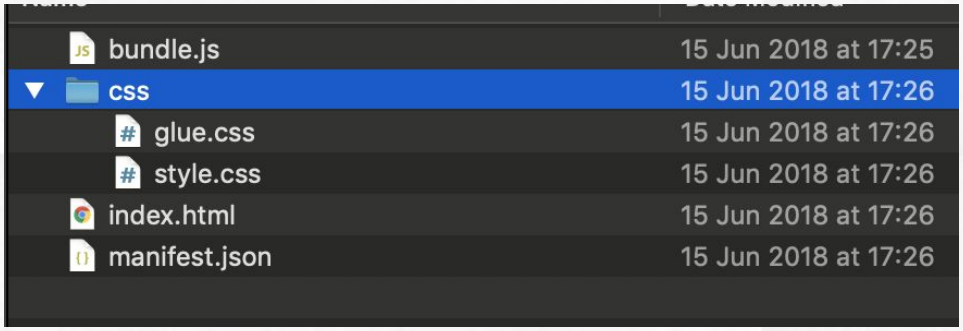

Search
Luca Mezzalira
Browse
Radio
YOUR LIBRARY
Your Daily Mix
Recently Played
Songs
Albums
Artists
Stations
Local Files
Videos
Podcasts
PLAYLISTS
Winter Acoustic
The Million Dollar H...
Live Tracks
Discover Weekly
Oasis: The Singles
The Rock Show
Mambo Italiano
New Playlist
RUDIMENTAL
TOAST TO OUR DIFFERENCES
HIDE ANNOUNCEMENTS
NEW ALBUM FEATURING TRACKS WITH
ANNE-MARIE · JESS GLYNNE · MAJOR LAZER
STEFFLON DON · YEBBA · RITA ORA · RAY BLK
MR EAZI · JAMES ARTHUR · RAYE · KOJEY RADICAL
TOM WALKER · CHRONIXX · MAVERICK SABRE
LISTEN NOW ON Spotify
GENRES & MOODS
PODCASTS
CHARTS
NEW RELEASES
DISCOVER
CONCERTS

Iframes


Contents
_CodeSignature
Frameworks
Chromium Embedded Framework.framework
Spotify Helper
Info.plist

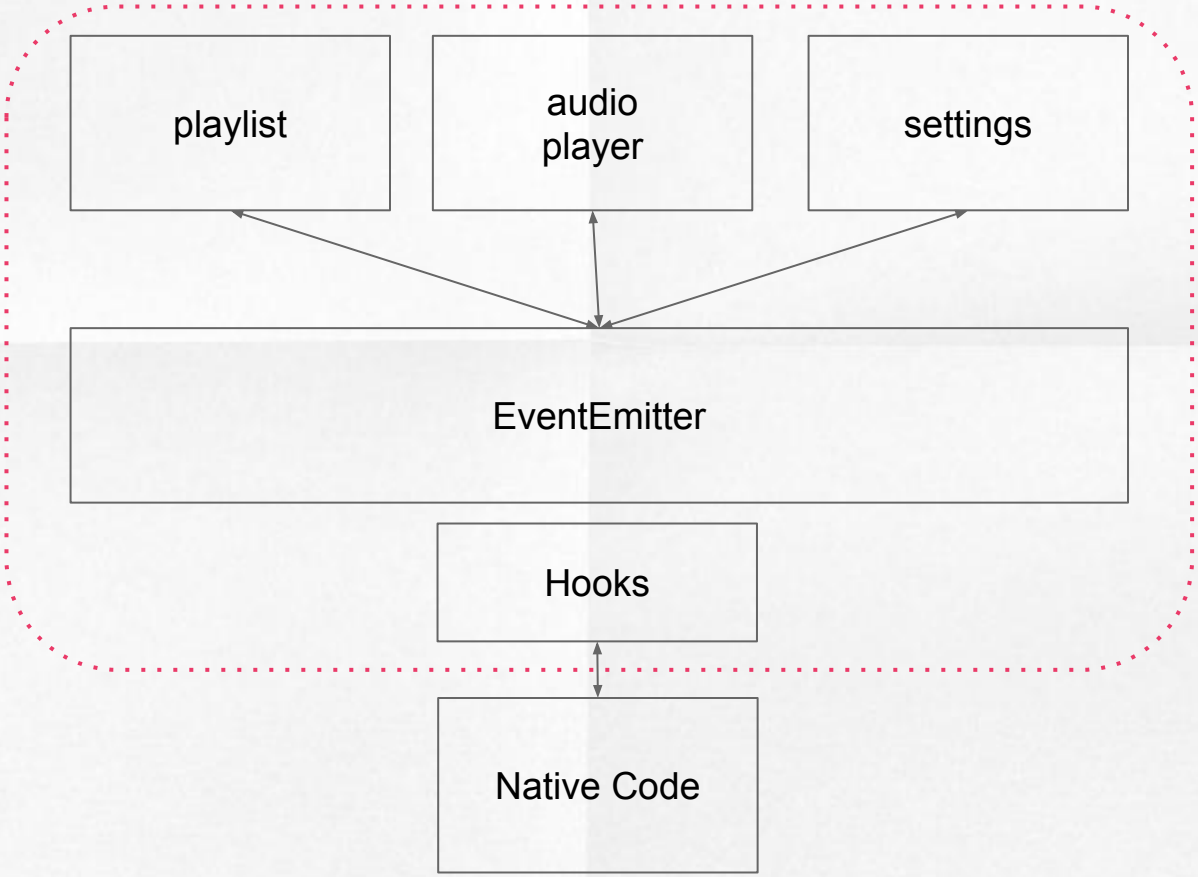

Apps 15 Jun 2018 at 17:35
about.spa 15 Jun 2018 at 17:35
artist.spa 15 Jun 2018 at 17:35
browse.spa 15 Jun 2018 at 17:35
buddy-list.spa 15 Jun 2018 at 17:35
chart.spa 15 Jun 2018 at 17:35
collection-album.spa 15 Jun 2018 at 17:35
collection-artist.spa 15 Jun 2018 at 17:35
collection-songs.spa 15 Jun 2018 at 17:35
collection.spa 15 Jun 2018 at 17:35
concert.spa 15 Jun 2018 at 17:35
concerts.spa 15 Jun 2018 at 17:35
daily-mix-hub.spa 15 Jun 2018 at 17:35
error.spa 15 Jun 2018 at 17:35
findfriends.spa 15 Jun 2018 at 17:35
full-screen-modal.spa 15 Jun 2018 at 17:35
genre.spa 15 Jun 2018 at 17:35
glue-resources.spa 15 Jun 2018 at 17:35
hub.spa 15 Jun 2018 at 17:35
licenses.spa 15 Jun 2018 at 17:35
login.spa 15 Jun 2018 at 17:35
lyrics.spa 15 Jun 2018 at 17:35
playlist-folder.spa 15 Jun 2018 at 17:35
playlist.spa 15 Jun 2018 at 17:35
profile.spa 15 Jun 2018 at 17:35
queue.spa 15 Jun 2018 at 17:35
radio-hub.spa 15 Jun 2018 at 17:35
search.spa 15 Jun 2018 at 17:35
settings.spa 15 Jun 2018 at 17:35
show.spa 15 Jun 2018 at 17:35
station.spa 15 Jun 2018 at 17:35
stations.spa 15 Jun 2018 at 17:35
zlink.spa 15 Jun 2018 at 17:35

Iframes




```
"domready": "^1.0.8",
"escape-html": "~1.0.1",
"events": "^2.0.0",
"findandreplacedomtext": "0.4.6",
"glob": "^7.1.2",
"handlebars": "^4.0.11",
"hbsfy": "^2.8.1",
"highlight.js": "^9.12.0",
"humanize": "^0.0.9",
"is-jpg": "~2.0.0",
"jsonify": "file:libs/jsonify",
"knockout": "3.4.2",
"lodash": "^4.17.10",
"minimist": "^1.2.0",
"moment": "^2.22.1",
"mout": "^1.1.0",
"nlp_compromise": "^1.1.3",
"node-sass": "4.9.0",
"partialify": "^3.1.5",
"promisify-native": "^1.0.0",
"prop-types": "^15.6.1",
"quantize": "^1.0.2",
"quill": "0.19.10",
"raven-js": "^3.25.1",
"react": "^16.3.2",
"react-dom": "^16.3.2",
"react-redux": "^5.0.7",
"react-transition-group": "2.3.1",
"react-virtualized": "^9.19.0",
"redux": "^4.0.0",
"redux-thunk": "^2.2.0",
"request": "^2.86.0",
"reselect": "^3.0.1",
"shaka-player-spotify": "2.1.4-sp.3",
"slick": "^1.12.2",
"spotify-batch": "0.0.2",
"spotify-cosmos-api": "^1.7.4",
"spotify-deferred": "^0.3.0",
"spotify-eventemitter": "^3.0.1",
"spotify-inheritance": "^1.0.1",
"spotify-liburi": "^4.6.0",
"spotify-modal": "^1.1.0",
"spotify-simple-navbar": "^1.0.0",
"teamcity-service-messages": "^0.1.9",
"url": "~0.11.0",
"uuid-js": "^0.7.5",
"webhdfs": "^1.2.0",
"when": "^3.7.8"
```

Iframes


playlist
audio player
settings
EventEmitter
Hooks
Native Code

Zalando

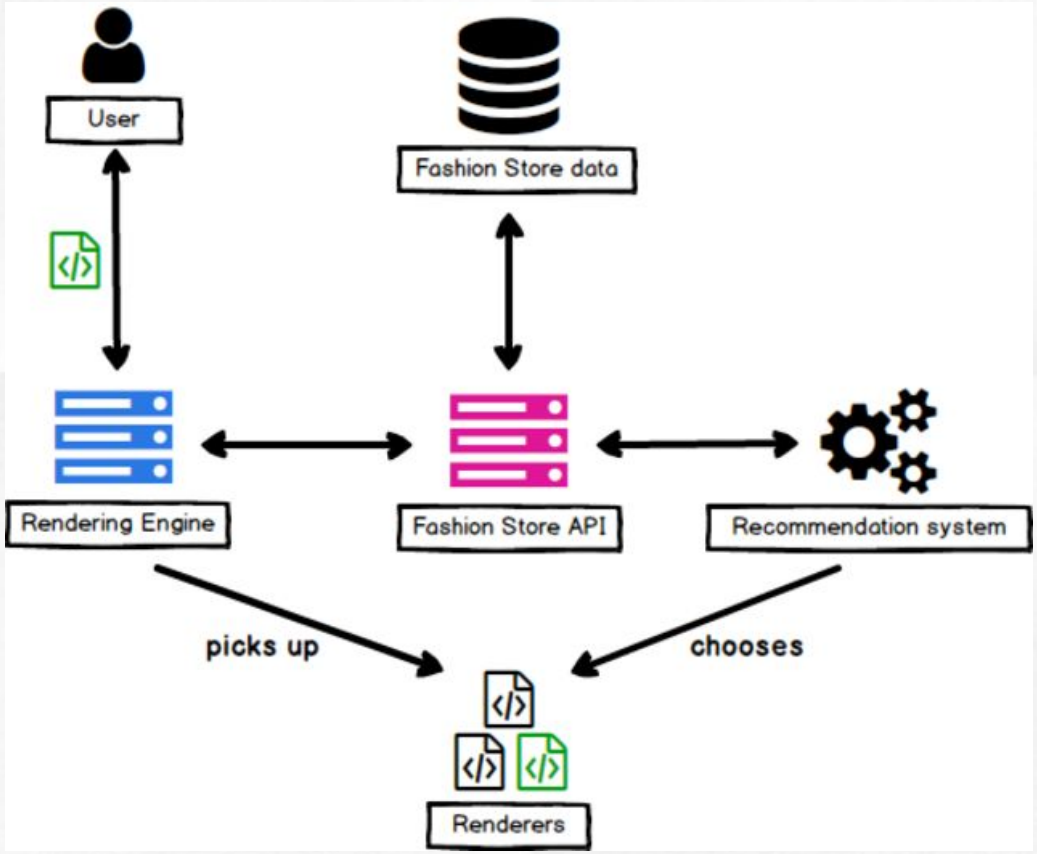
zalando

Server-side composition

Zalando was one of the pioneers on micro-frontends 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 they are moving their implementation to a server-side include system called "**Interface framework**"

Interface Framework



https://zln.do/2CS4JNJ

IKEA

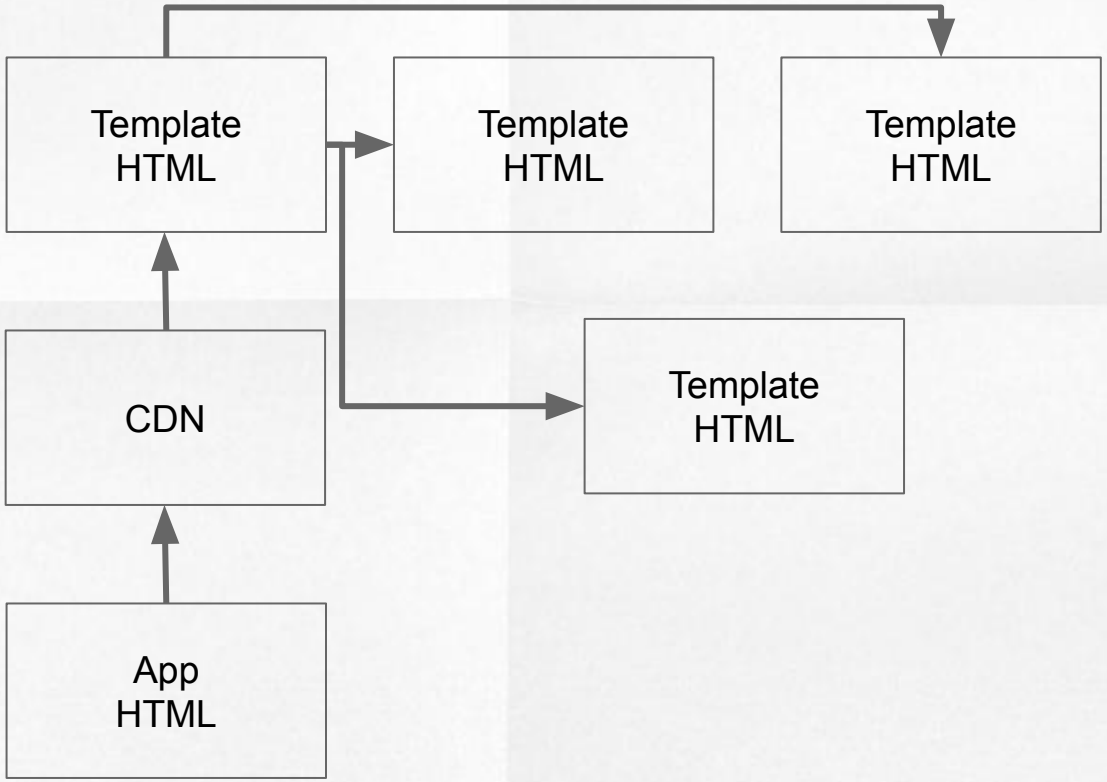
Edge Side Includes (ESI)

Ikea embraces Micro-frontends via transclusion mechanism via Edge Side Includes (aka ESI)

ESI is a proposal made by several CDNs companies in order to standardise the way you can compose frontend pages via transclusion. In this case the HTML page is composed on the edge allowing a dynamic way for composing pages

More info about ESI from Kotte's post

Edge Side Includes (ESI)



Edge Side Includes (ESI)

```
<html>
  <head>
      <title>My Ecommerce Site</title>
  </head>
  <body>
    <div id="content">
      <div id="header">
        <img src="/images/logo.jpg" />
        <esi:include src="/shopping_cart" />
      </div>
      <div id="items">
        <span id="item_1">...</span>
        <span id="item_2">...</span>
        ...
        <span id="item_n">...</span>
      </div>
    </div>
  </body>
</html>
```

https://bit.ly/2BRFGuw

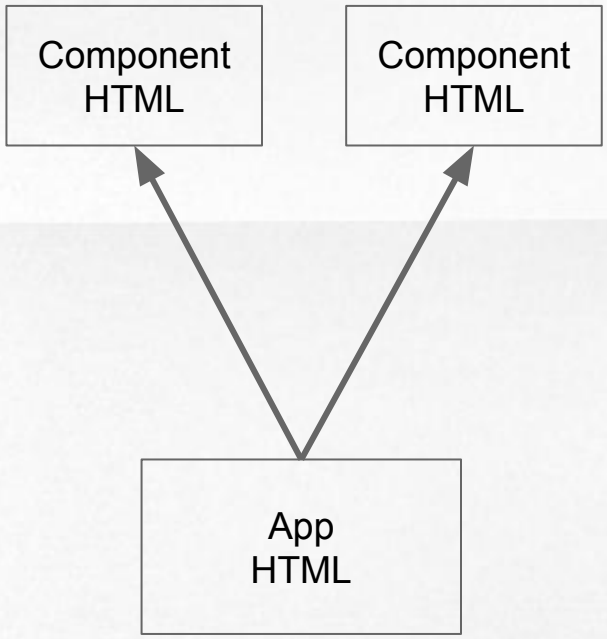
Edge Side Includes (ESI)

- The page is composed server-side transcluding HTML tags with external components
- It's used at CDN level without a proper standard

Edge Side Includes (ESI)

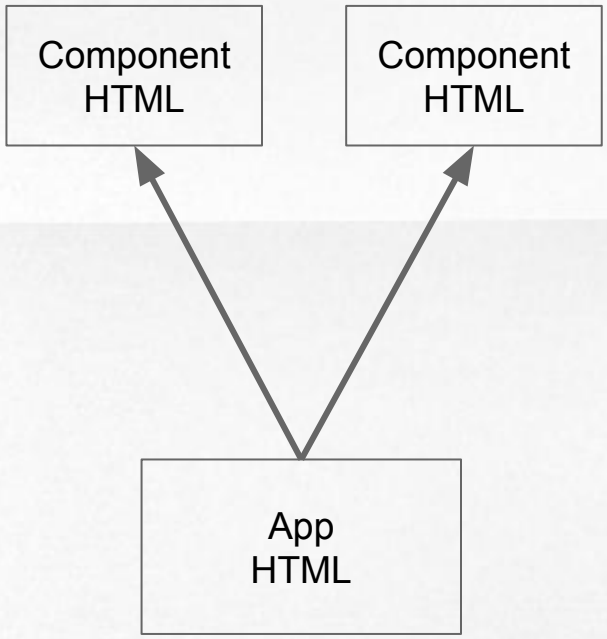
- **PRO**: page created on the server at runtime and then cached if the content is not dynamic
- **CONS**: with dynamic content is more difficult to cache the page response
- **CONS**: if some elements are hanging due to network issues, the response time could increase considerably

Client Side Includes (CSI)



- Parsing the App DOM looking for elements to substitute
- Lazy loading the component parsing an HTML document and replacing the content in App HTML
- It can be used in combination with ESI

Client Side Includes (CSI)



- **PRO**: Frictionless for developers
- **PRO**: Devs have full control on how to load part of the application
- **CONS**: harder to avoid bumping layouts
- **CONS**: network failure strategy to implement

Client Side Includes (CSI)

```
<hx:include src="/shopping-cart/component">
  <a href="/shopping-cart/component">Shopping cart</a>
</hx:include>
```

```
<hx:include src="/shopping-cart/component">
  <div>You have 0 items in your <a href="/shopping-cart">shopping cart</a></div>
</hx:include>
```

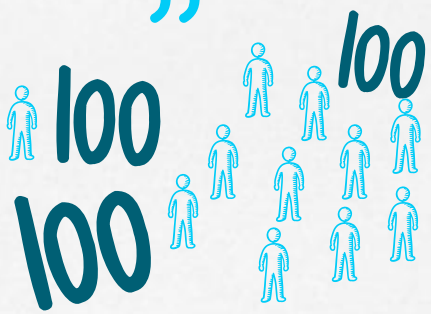
https://bit.ly/2XqWLVa

The DAZN way

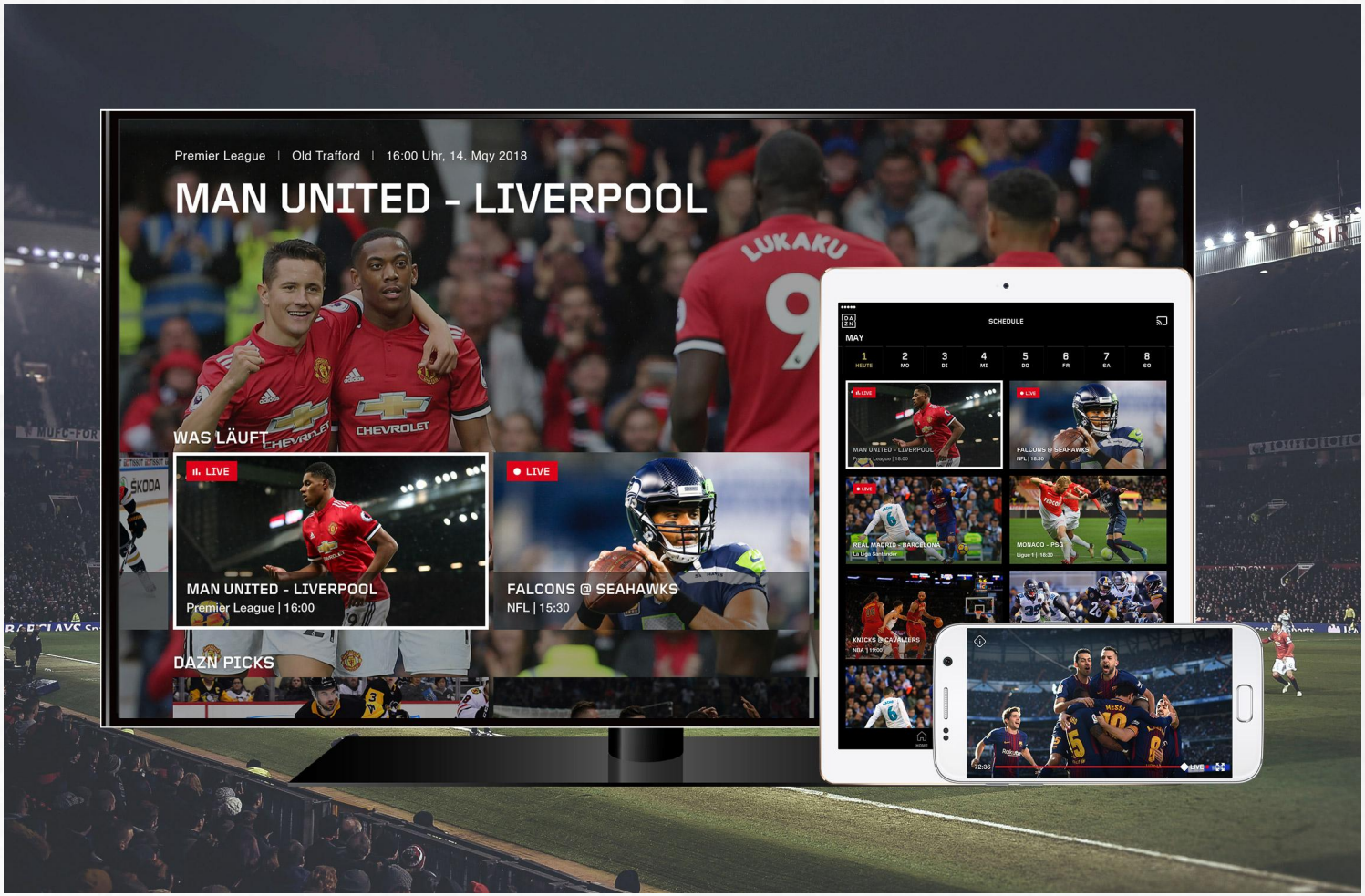
Design for hundreds of developers

"We are going to have **hundreds of developers** working on **DAZN**.

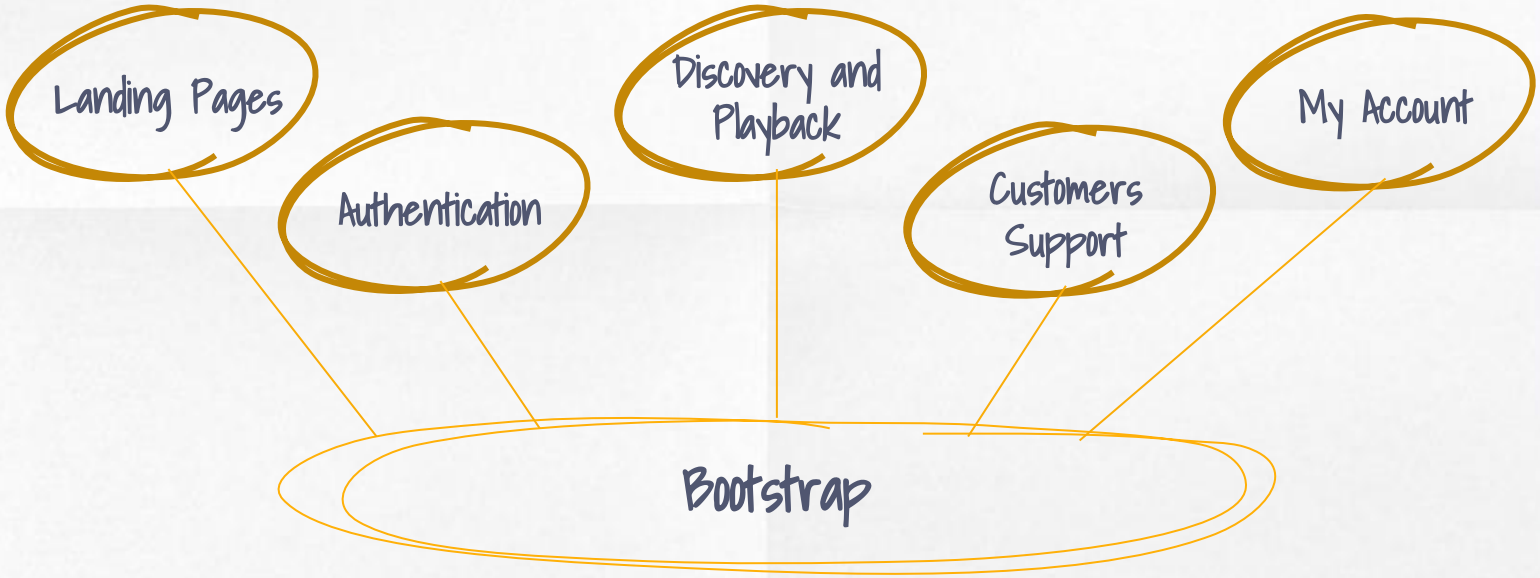
Bear in mind when you design our **new architecture**!"



Multi Device experience



DAZN implementation



DAZN implementation

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**
- **1 team own 1 Micro-frontend**

Micro-frontend structure

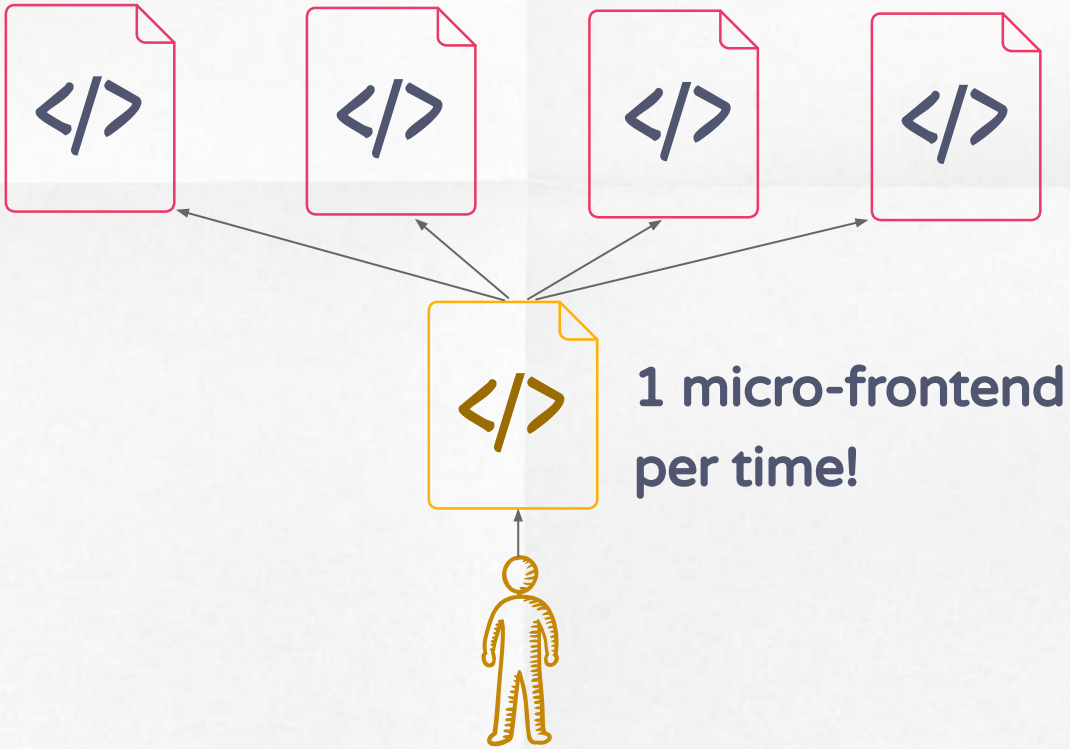
index.html

app.js

vendor.js

style.css

How bootstrap works



How bootstrap works

Bootstrap is responsible for:

. application startup
. I/O operations
. routing between micro-frontends
. sharing configurations across multiple micro-frontends

How bootstrap works

```
<html>
    <head>
        <script src="./bootstrap.js" />
    </head>
    <body>
        ...
    </body>
</html>
```

- 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

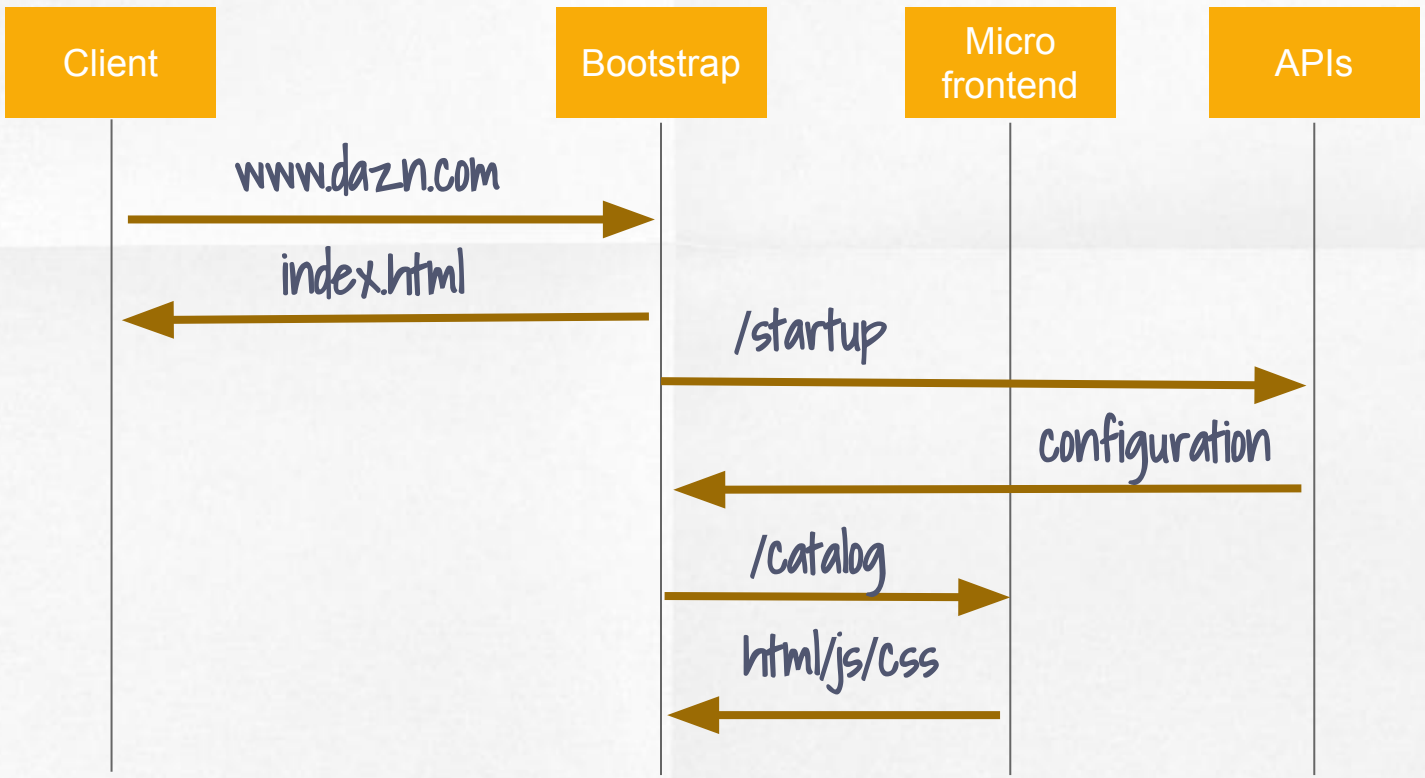
```
<html>
    <head>
        <script src="./bootstrap.js" />
        <style type="text/css">...</style>
    </head>
    <body>
        <div>
            ...
        </div>
        <script src="./catalogue.js" />
        <script src="./cat-vendor.js" />
    </body>
</html>
```

How bootstrap works

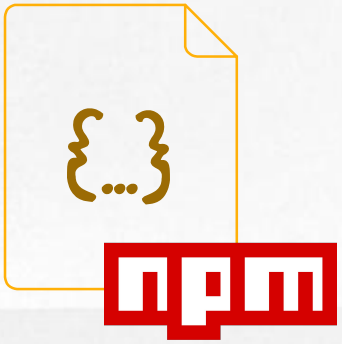
```
Window.DAZN = {
    Lifecycle: {
        onLoad: function(){...}
        onunLoad: function(){...}
    },
    Localstorage: {...},
    ...
}
```

- 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

How bootstrap works



Components



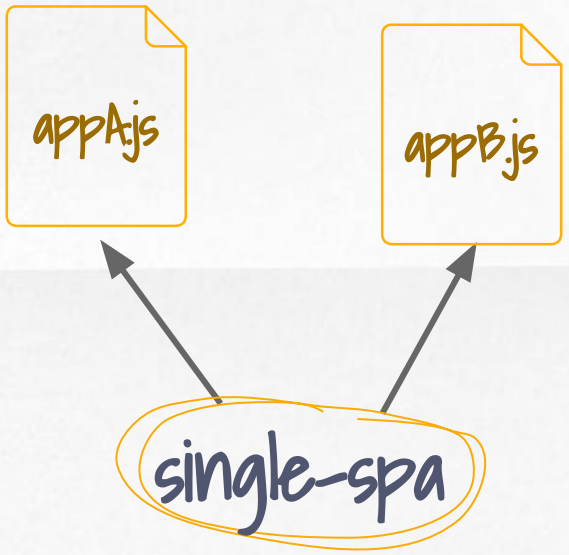
- Components available on NPM private repos
- Components need to work with any framework
- They expose a contract for the micro-frontend to interact with
- Components are own by specific teams
- They can be shared within same team

Micro-frontends frameworks

Single-SPA

single-spa.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


QnA

3.

Automation as first-class citizen

automate, automate, automate

Principles

- Fast feedback for developers
- Parallel jobs/steps execution
- Build an automation pipeline as code
- Easy to change and update any single step of any pipeline
- Innovation

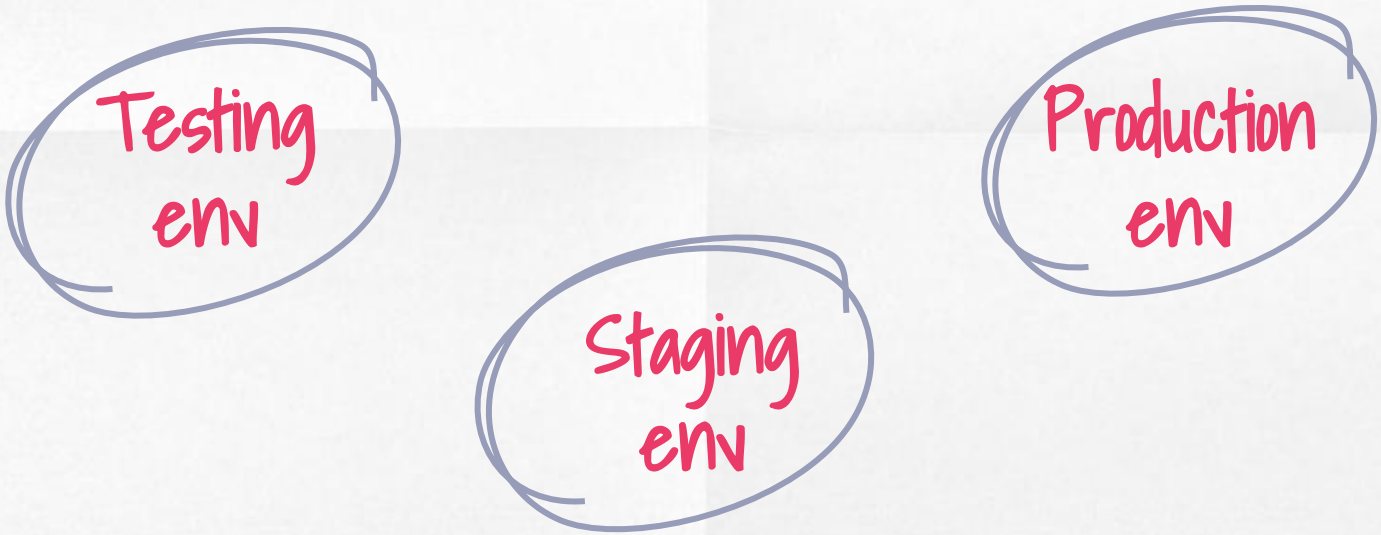
Tech decisions

- Tech stack (on-prem, cloud...)
- Version of control
- Testing strategy (unit, integration, end to end…)
- Artifacts storage
- Deployment strategy (canary releases, big-bang…)
- Additional automation checks (security, performance…)

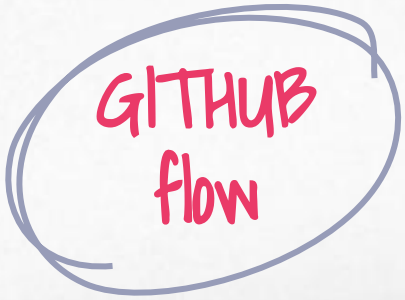
The DAZN way

Environments strategy

Multi-environments allow you to test your assumptions in a predictable way



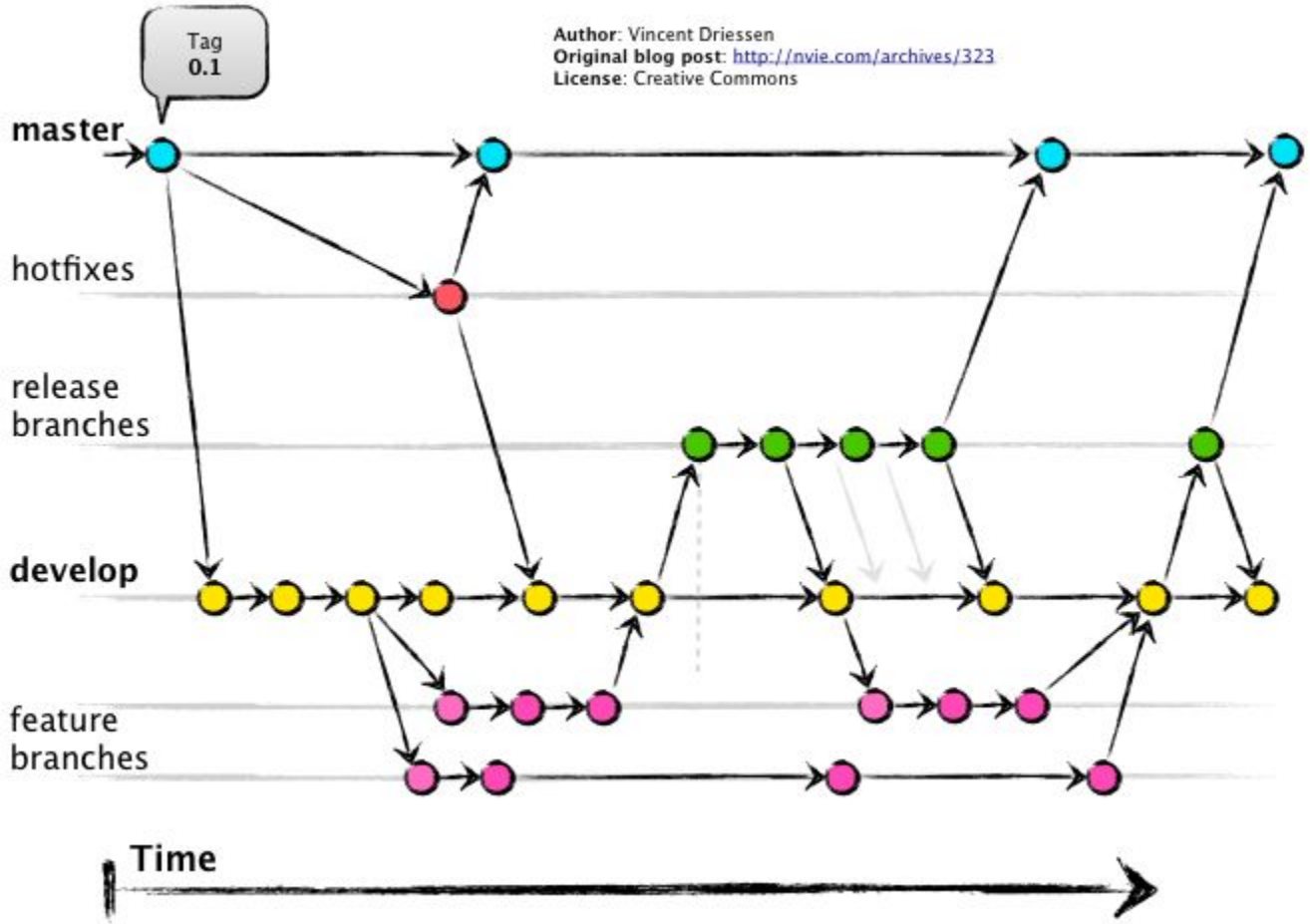
Branching strategy



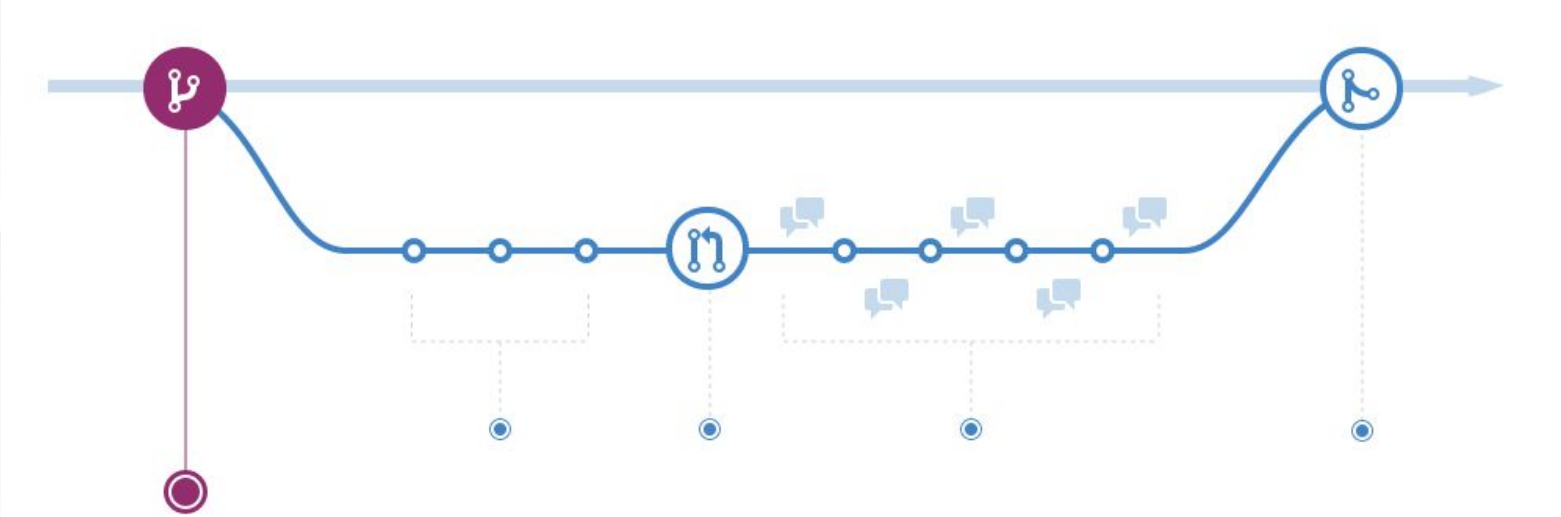
The branching strategy defines the way you are creating your build and deploy pipeline

The project and the teams skills allow you to pick the right strategy for having a successful automation

Git Flow

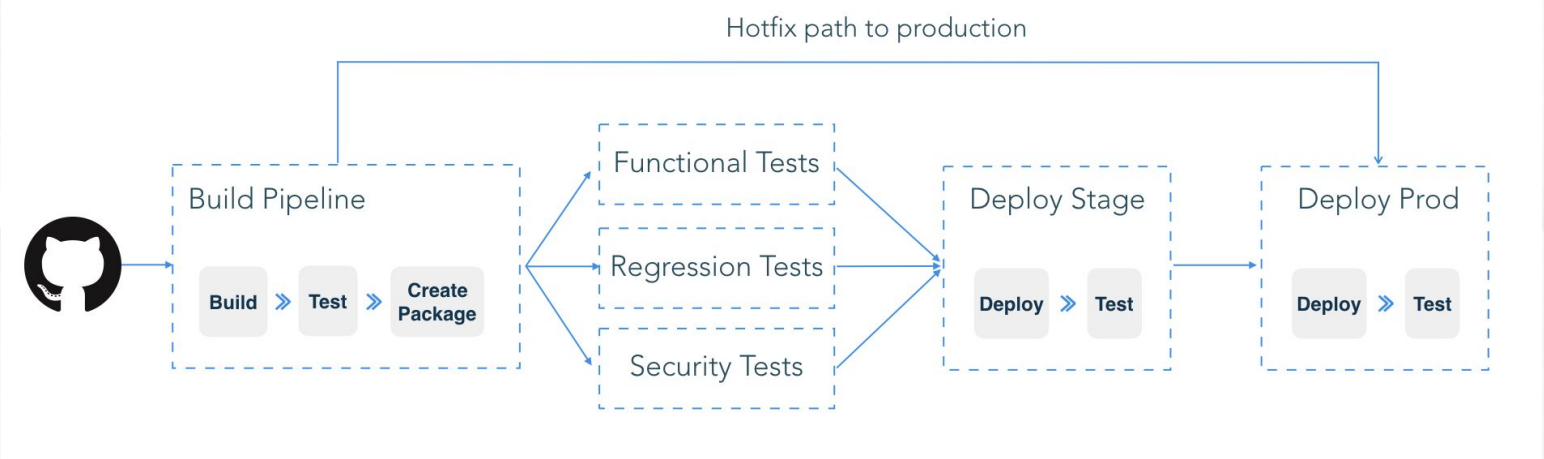

Tag
0.1
Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons
master
hotfixes
release
branches
develop
feature
branches
Time

Github Flow

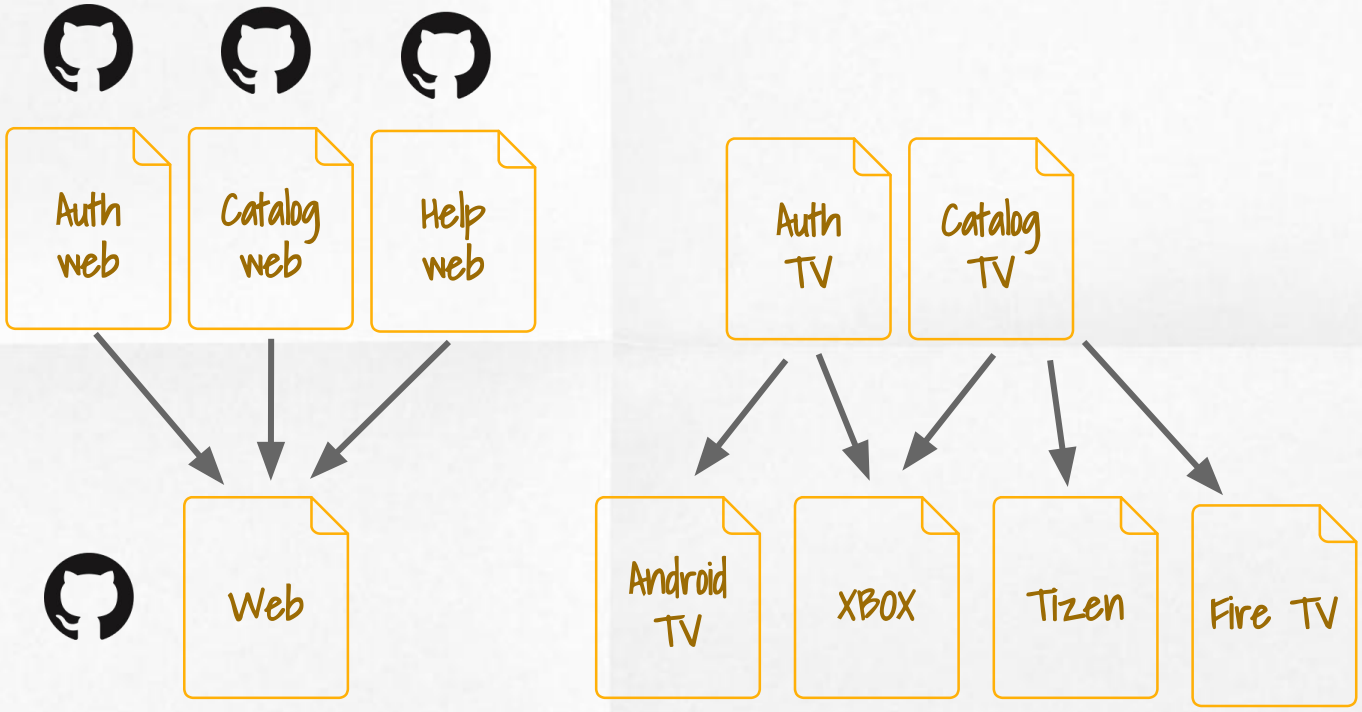


https://bit.ly/2TlmEGP

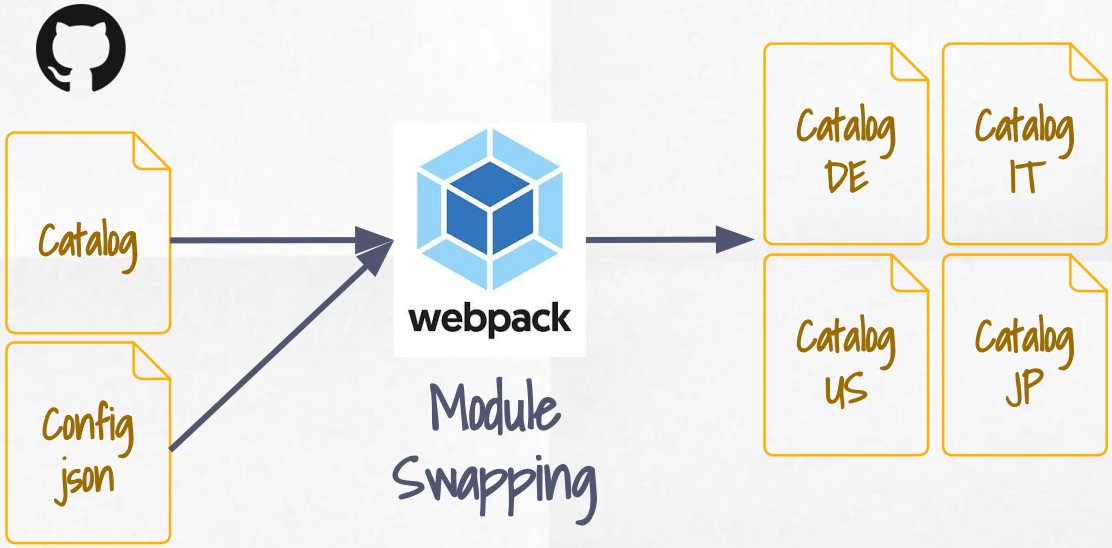
Hot Fixes


Hotfix path to production
Build Pipeline
Build
Test
Create Package
Functional Tests
Regression Tests
Security Tests
Deploy Stage
Deploy
Test
Deploy Prod
Deploy
Test

Artifacts

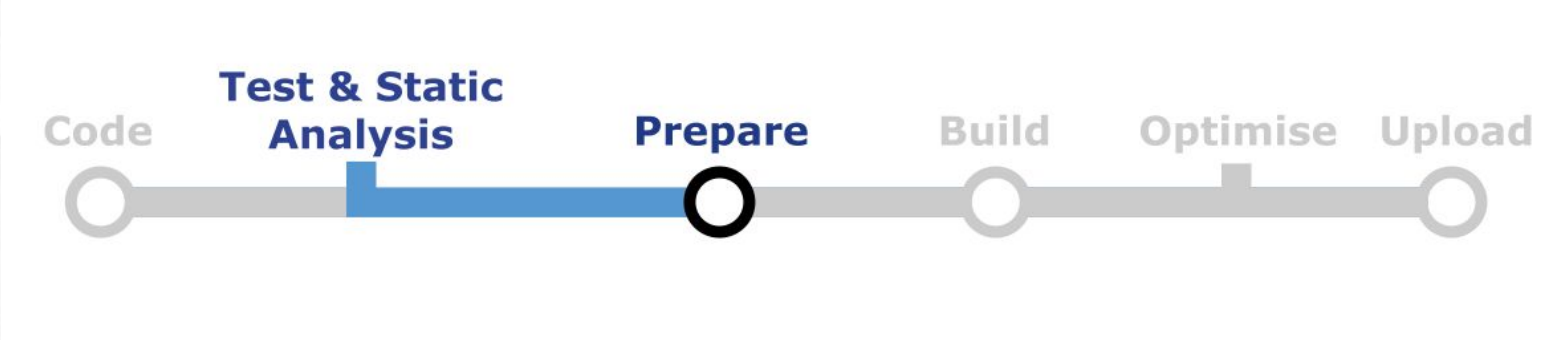

Auth web
Catalog web
Help web
Web
Auth TV
Catalog TV
Android TV
XBOX
Tizen
Fire TV

Artifacts

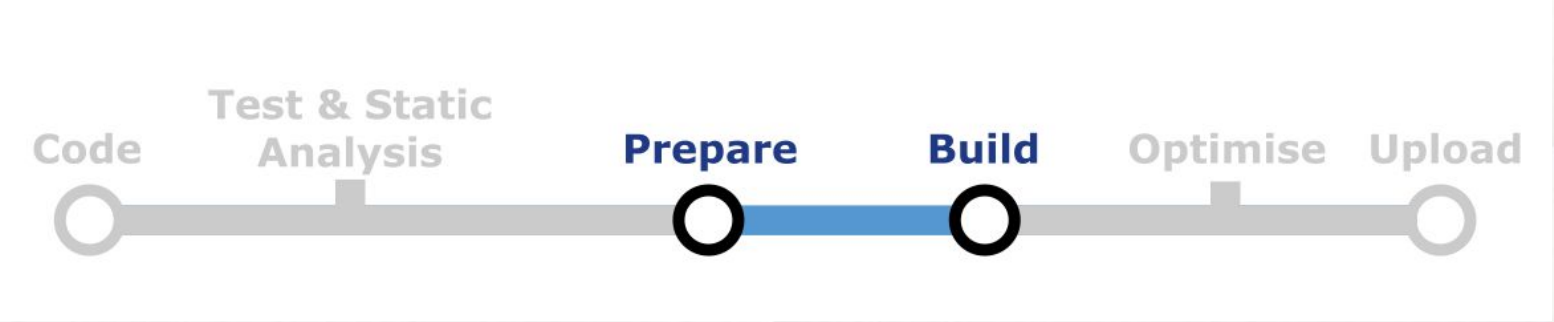

Catalog
Config
json
webpack
Module
Swapping
Catalog
DE
Catalog
IT
Catalog
US
Catalog
JP

DAZN Continuous Integration

DAZN Continuous Integration



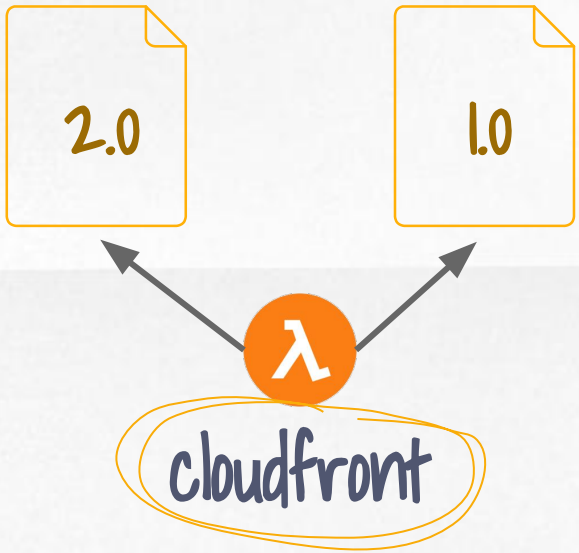
DAZN Continuous Integration



DAZN Continuous Integration

DAZN Continuous Integration

Deployment



https://bit.ly/2Afy44t

- 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! (Amplify-like)

Further improvements

https://bit.ly/2FjmNjH

- Lighthouse and webperf
- Performance checks before deploying in production
- Optimization checklist: https://bit.ly/2TBDHRm


QnA

4.

Technical challenges

Better make the horizon your goal;
it will always be ahead of you

Shift of mindset

From:

. technology decisions upfront

. reluctance of architectural changes

. global coding style decisions

. shared automation pipelines

. code reusability

. longer onboarding process

Shift of mindset

To:

. technology agnosticism

. embracing changes and accommodate new architectural requirements

. team coding style and best practices definition

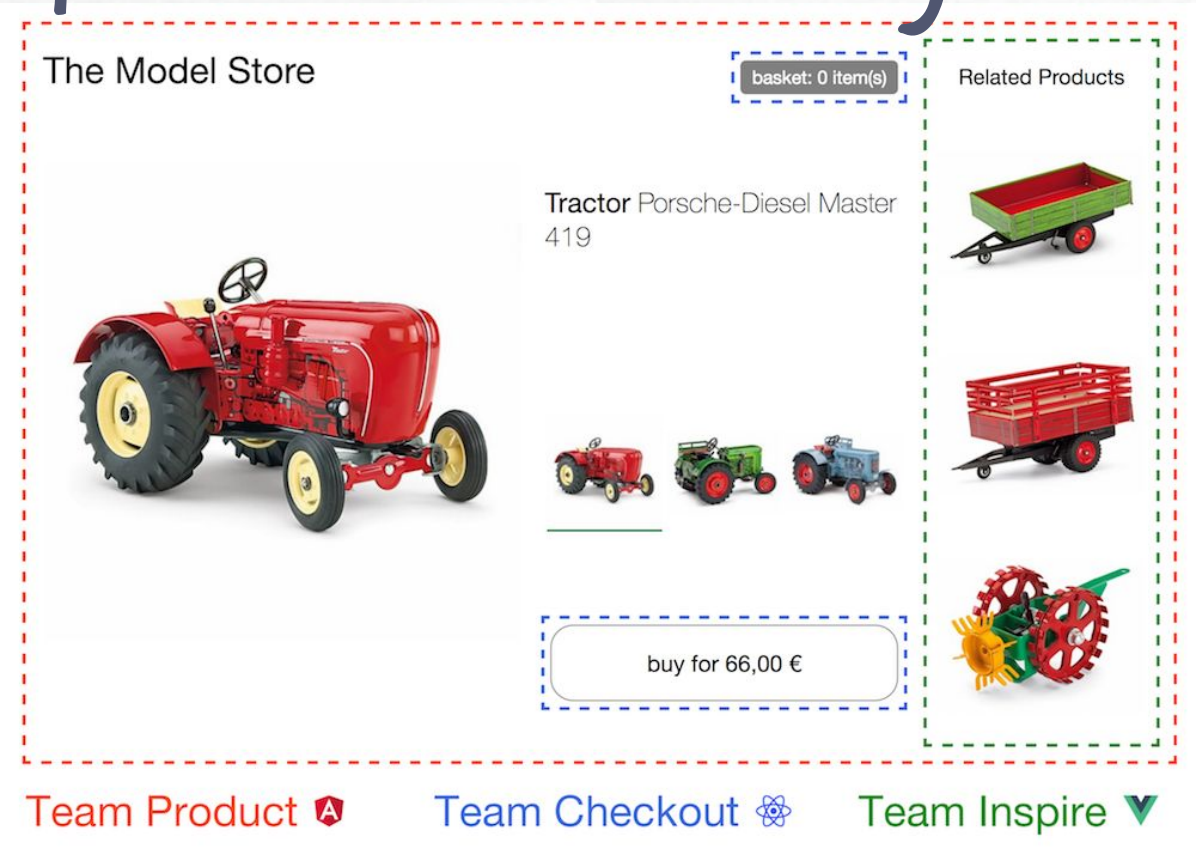
. independent automation pipelines

. acceptance of code duplication in favour of speed

. easier hiring and onboarding

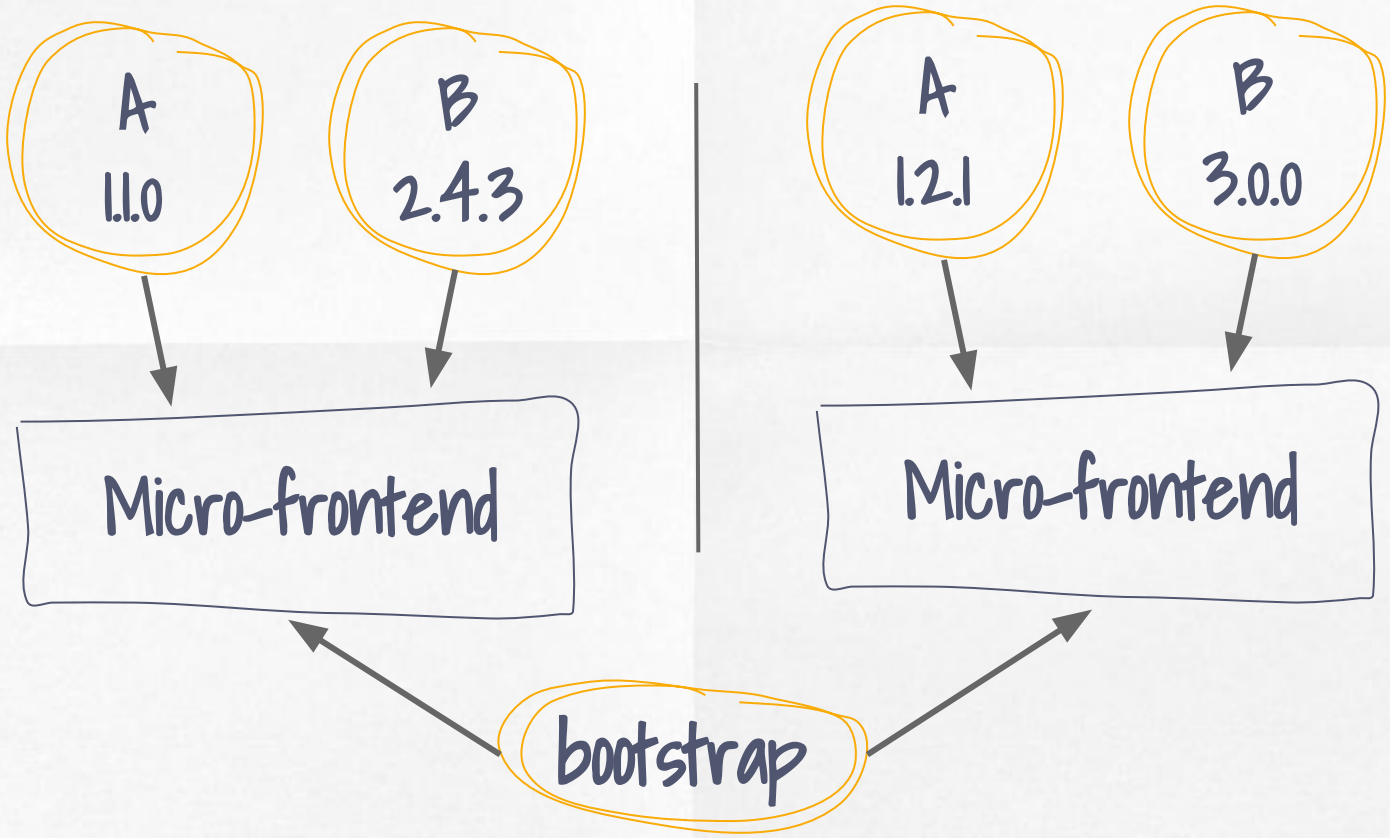
. innovation

Dependencies management

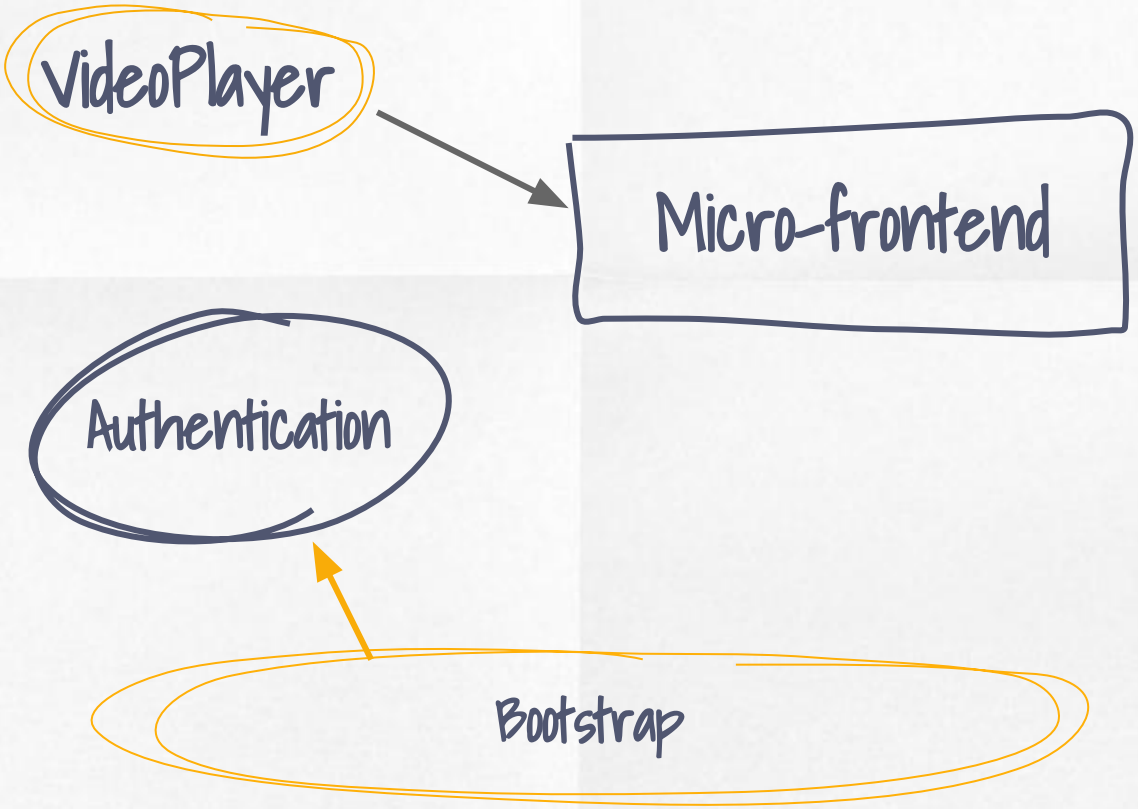


micro-frontends.org

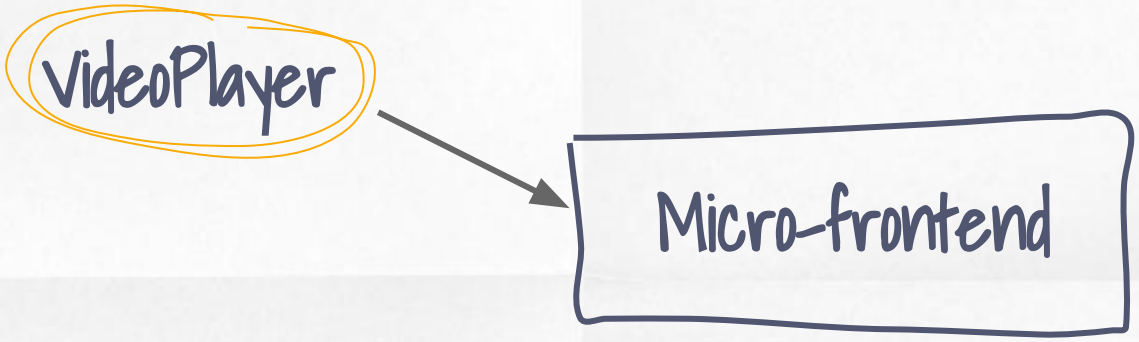
Dependencies management



Working by contracts



Working by contracts



APIs first design principles

. APIs are the first user interface of your application
. APIs come first, then the implementation
. APIs are self-descriptive


SEO

Search Engine Optimizations

Standard
implementation

Dynamic
Rendering

Standard Implementation

- Pay attention on how you render the content in your page (SSR over dynamic DOM manipulation)
- When you use VDOM libraries remember the 5s rules
- GoogleBot uses web rendering service based on Chrome 70 (recently updated)
- GoogleBot doesn't retain the state (no local or session storage available)

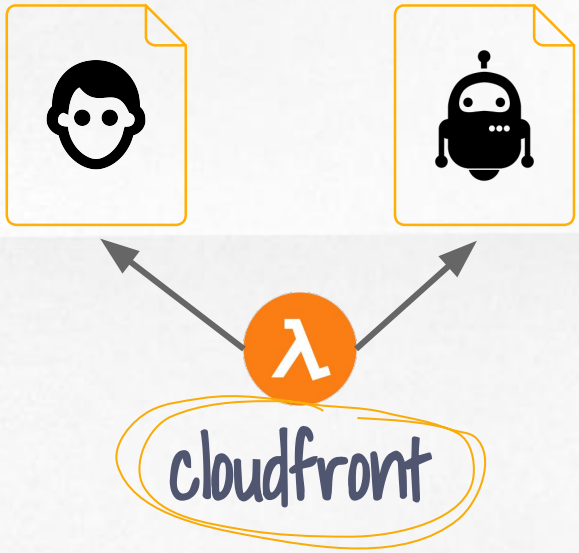
Standard Implementation

- Key dom elements in light-DOM not shadow-DOM
- Load content when visible in the viewport and lazy load the rest of your pages
- Remember, ALL the requests are coming from USA

Dynamic Rendering

- Different website for GoogleBot using SSR for all the pages (where possible)
- Show immediately relevant content that is not available without user interaction
- Content available in those pages has to be the same for human version
- All the rules on the standard implementation are applicable to dynamic rendering too

Dynamic Rendering



https://bit.ly/2Cs2eDj

- Based on the user-agent we serve the bot or human version
- All the logic is wrapped in the **lambda@edge**
- Automate the testing between different version to check the contents is the same


QnA

5.

Teams organization

Great things in business are never
done by one person.
They're done by a team of people.

The main challenges with **scaling frontend applications** are **scaling the teams, reducing the communication overhead** and **innovate**!

New Relic: a case study

New Relic was “born agile”

… then growth happened

. more dependencies
. higher friction
. higher frustration

Frustration is not good for innovation

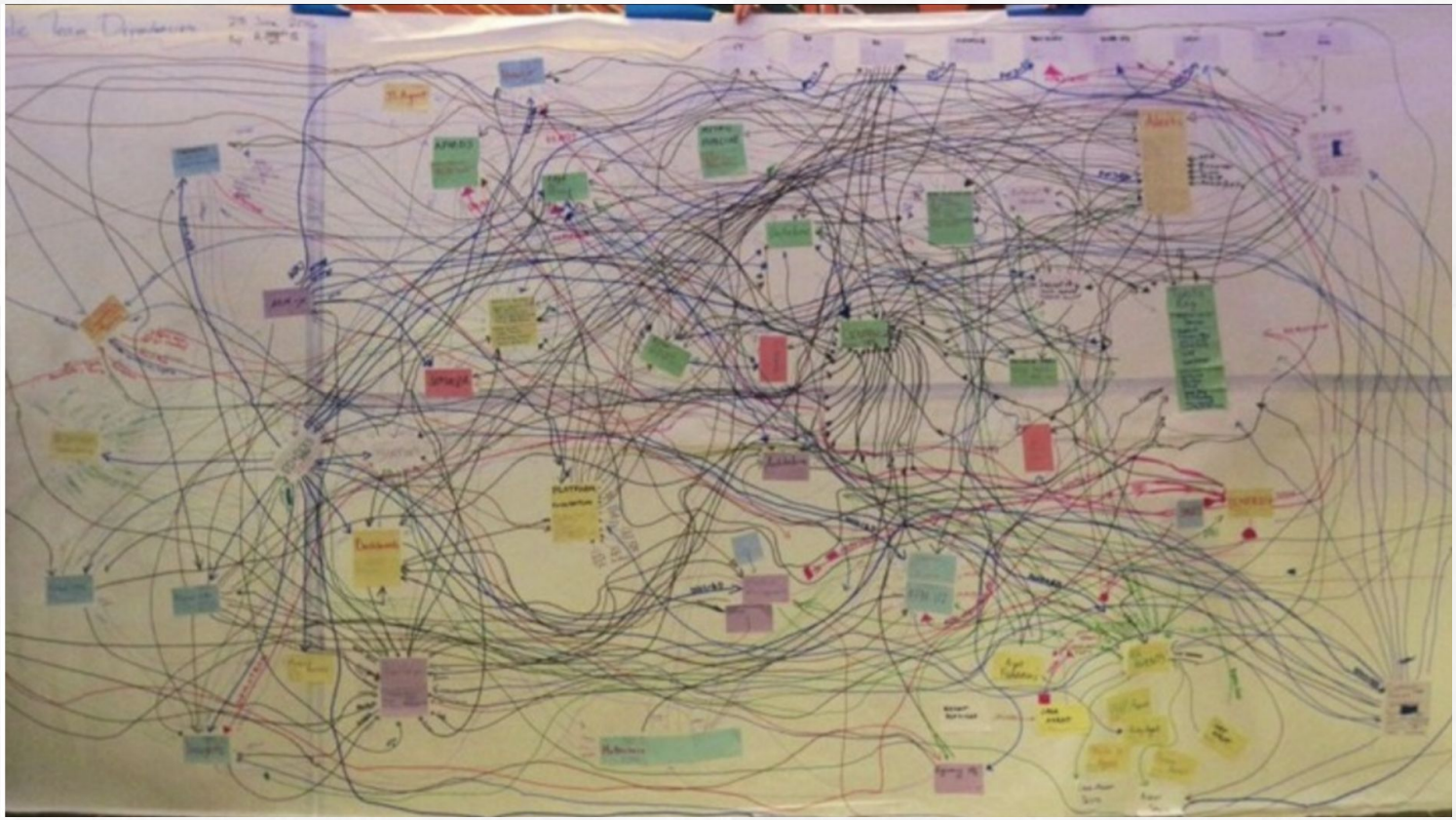
New Relic: a case study

Add process + oversight to manage complex dependencies

VS

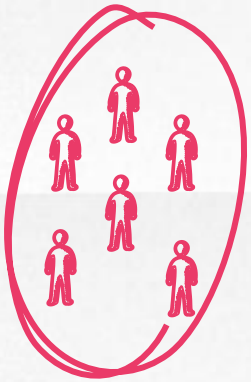
Reduce dependencies to increase team autonomy

New Relic: a case study

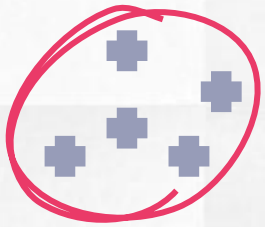


The DAZN way

DAZN teams structure



Features and Components teams



Distributed teams in Europe

Team Composition

- Product
- Architect
- Scrum master
- Developers
- QAs
- Dev in Test

DAZN teams structure

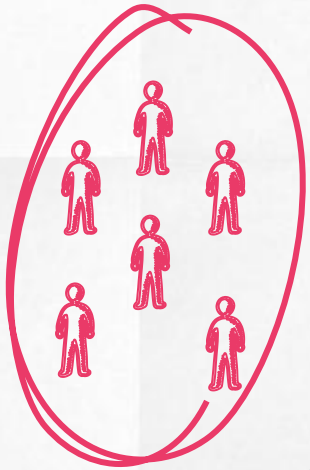
SREs

Cloud Engineers

Video Player FE

DX Team

Payments BE



Catalog FE+BE

Onboarding FE+BE

My Account team

Mobile Native teams

Help team

TVs FE only

Sports Data team

DAZN teams structure

Core

Video Player FE
Catalog FE+BE
Sports Data team

Global

DX Team
Cloud Engineers
SREs

Generic

Payments BE
My Account team
Help team
Onboarding FE+BE

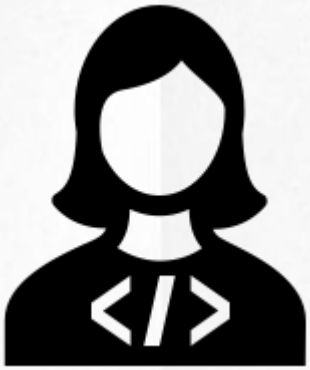
Core+Supporting

Mobile Native teams
TVs FE only

Micro-frontends impact

- Less/No dependencies shared across teams
- Speed of development
- End to end independence
- Ownership of a specific domain

Learning



Ownership

What are developers looking for?

Innovation

Retain YOUR TALENT

The cost of replacing an employee is somewhere between 90% and 200% percent of their annual salary.

https://bit.ly/2E0g3Im

Hidden churn costs

- Cost of hiring
- Cost of onboarding
- Cost of learning and mastering a domain
- Cost of time with unfilled role


QnA

Checkpoint

What are the impacts of micro-frontends on development teams?

Why is important retain your talent inside the organization?

What's the main challenge on scaling frontend applications?

6.

Wrap up

Scaling a business at speed can feel like an out of control roller-coaster.

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 own by a single team

Micro-frontends principles

1. Business subdomain representation
2. Autonomous
3. Own by one team
4. Technology agnostic

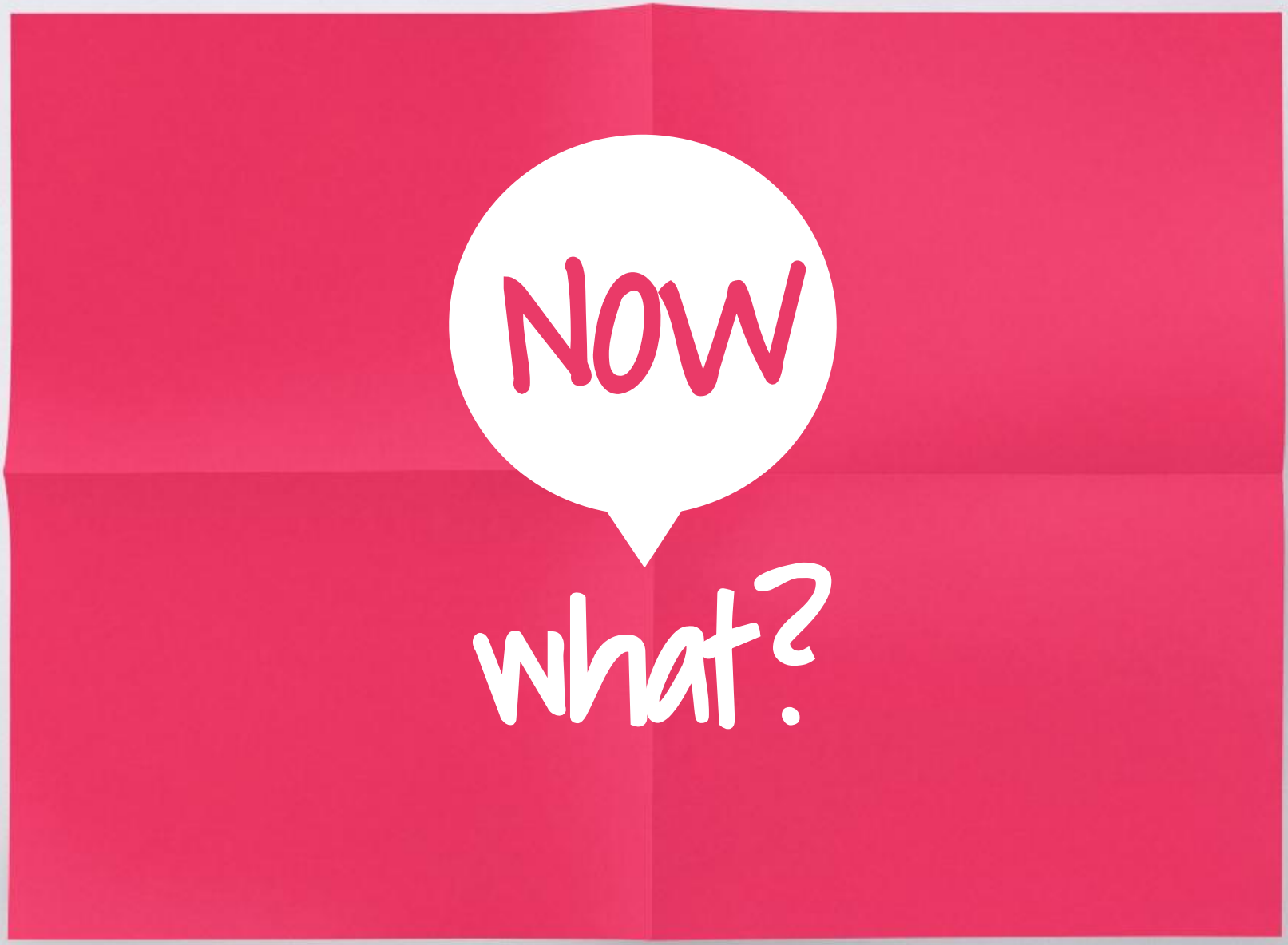
The main challenges with **scaling frontend applications** are **scaling the teams**, **reducing the communication overhead** and **innovate**!

Different Micro-frontends implementations

1. Iframes
2. Open Components
3. Interface Framework
4. ESI + CSI
5. Bootstrap + SPAs or SSR or Single Page

Automation, automation, automation

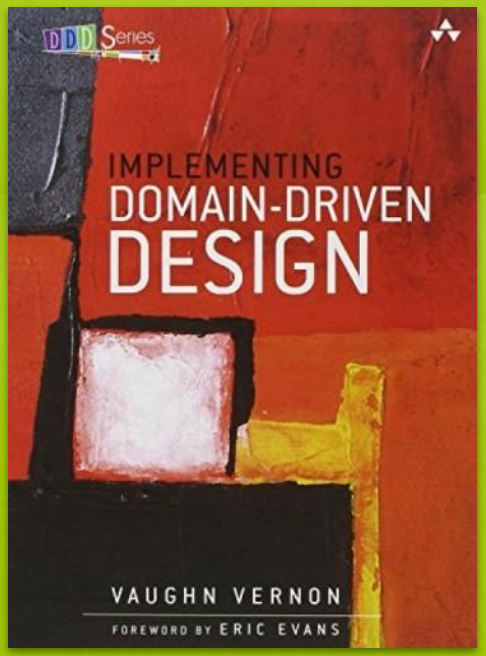
1. Automation is key for micro-frontends
2. Independent testing
3. Independent build
4. Independent deployment


NOW
what?

DDD resources

Decompose by subdomain
https://bit.ly/2DUTQ1v

Subdomains and bounded context in DDD
https://bit.ly/1BPZfIW



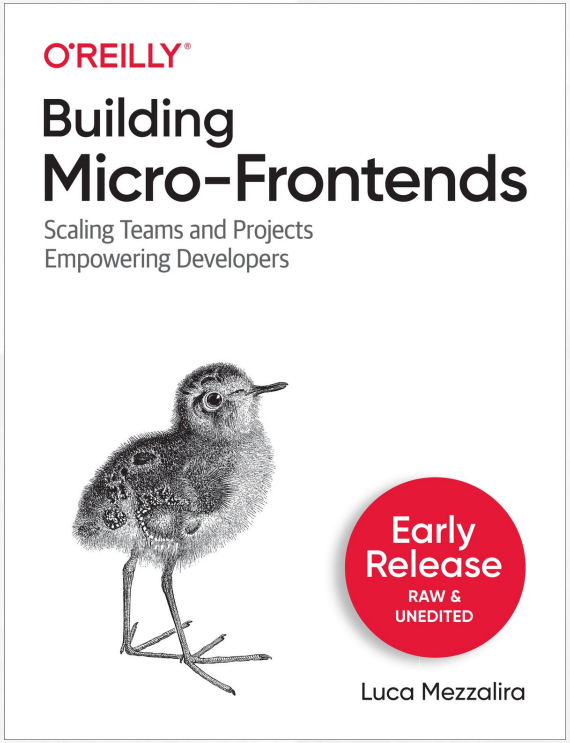
Micro-frontends... try yourself!

- Single-SPA: www.single-spa.js.org
- ESI: gustafnk.github.io/microservice-websites/
- Mosaic: www.mosaic9.org
- Open Components: github.com/opencomponents/oc
- Feature Hub: feature-hub.io

Micro-frontends sessions

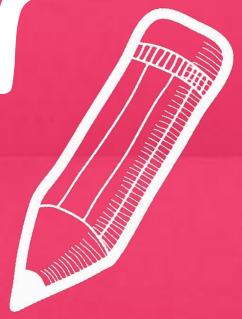
- Introduction to Micro frontends
- Micro frontends architecture
- OpenComponents as Microservices
- Painless Micro frontends delivery
- The recipe for scalable frontends
- Frint.js: build scalable reactive frontends with React

Learning, testing and **sharing**



www.buildingmicrofrontends.com

Leave your feedback!




QnA

You can find me at:

@lucamezzalira

luca.mezzalira@dazn.com