MFE ARCHITECTURE

Hudson Graham – Jan 2020

BENEFITS OF MFE ARCHITECTURE

- **Independent deployments**
  - Independent release management for sub domains
  - Launch Darkly state toggling and A/B testing
  - Potentially remove regulations from sub domains

- **Incremental upgrades**
  - Upgrade sub domains to new framework incrementally (Strangler pattern)

- **Decoupled codebase**
  - Enforce decoupled architecture

THE OVERHEAD OF MFE ARCHITECTURE

- **Complex build systems**
  - Multiple CI/CD pipelines
  - Hard tracking and debug problems across the entire system
  - Need for more DevOps within squads

- **Greater squad coordination**
  - Squads need to attend/contribute/adhere to web chapter
    - Core change management
    - Version management
    - Performance
    - Coherent experience

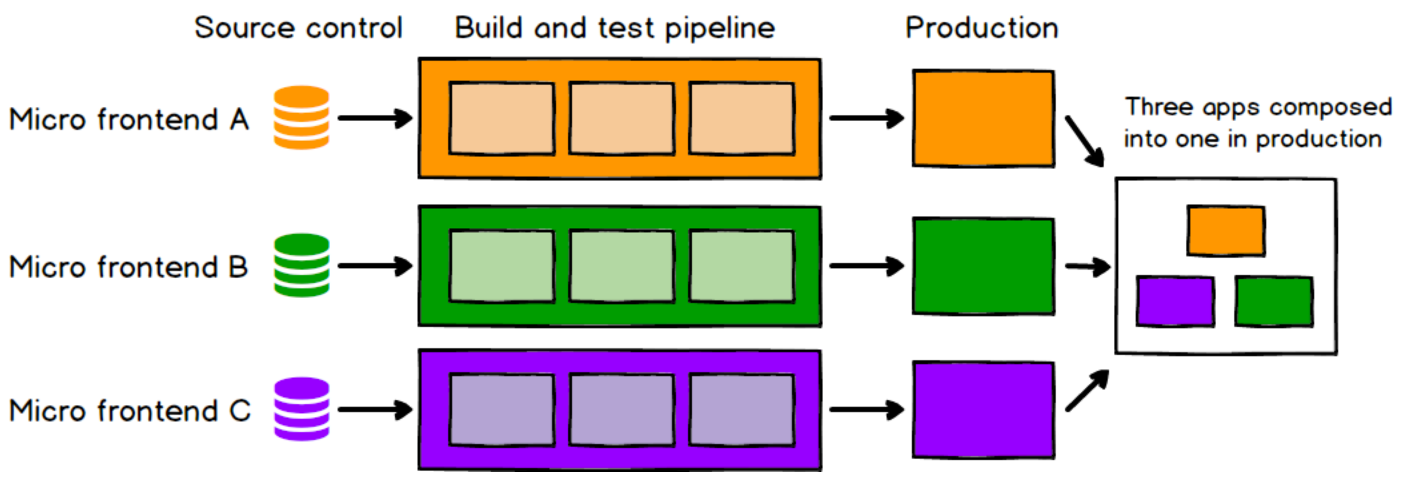
- **Requires additional work/resources for each squad**
  - Squads needs to understand and buy-in on the approach (PO, UX, DM, DL, Dev, Test)
  - Squads need to have time allocated to implement and participate more in web chapter
  - Need for integrated DevOps access/skills
  - Requires greater coverage of automated ui tests and solid integrated testing

CI/CD PIPELINE

Each MFE needs it's own CI/CD pipeline and for the POC I used Github.com > Travis-ci > AWS S3.

This setup allowed me to have full access and control of the DevOps process.

GitHub Travis CI amazon web services S3



POC OVERVIEW

I created an opensource POC, implementing MFE/ObservableStore architecture called **pixelybets-app:** https://github.com/pixelypants/pixelybets-app

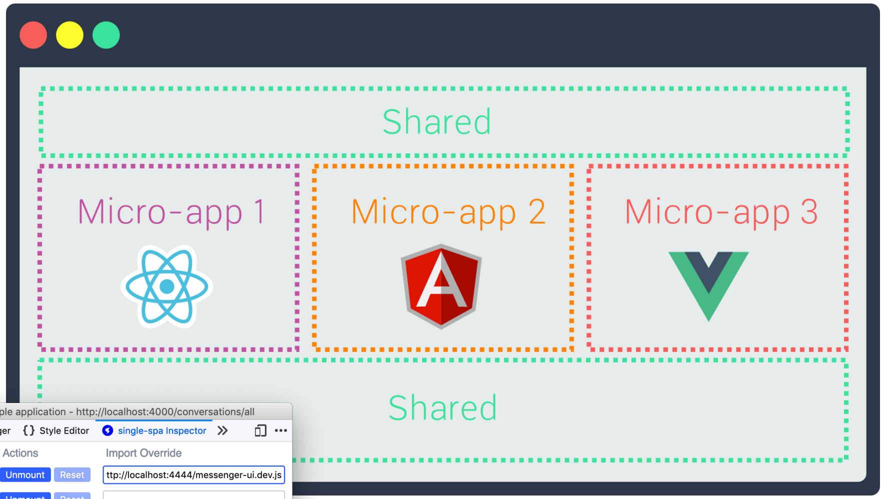
Technology used:

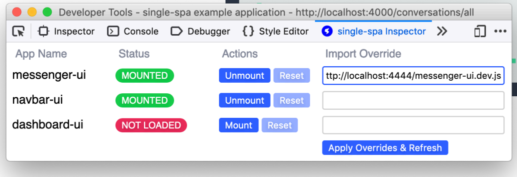
- Github.com
- Travis-ci
- AWS S3
- Webpack
- Babel
- Single-SPA
- Import Maps (SystemJS)
- RxJS (ObservableStore)
- React
- JavaScript
- TypeScript

POC TECH – SINGLE-SPA

Why I chose SINGLE-SPA

- Simple to setup and use
- Documentation and guides for migrating
- DX: import-map-overrides, Devtools, etc.
- Community and proven in production
- Slack channel support




Developer Tools - single-spa example application - http://localhost:4000/conversations/all

App Name	Status	Actions	Import Override
messenger-ui	MOUNTED	Unmount Reset	ttp://localhost:4444/messenger-ui.dev.js
navbar-ui	MOUNTED	Unmount Reset	
dashboard-ui	NOT LOADED	Mount Reset	

https://single-spa.js.org/

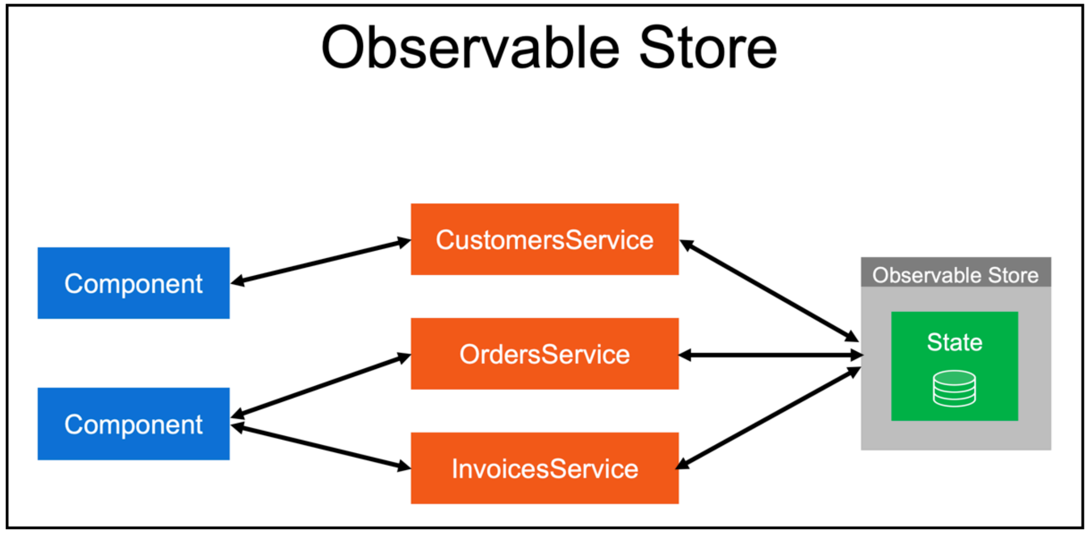
single-spa is a framework for bringing together multiple javascript microfrontends in a frontend application. Architecting your frontend using single-spa enables many benefits, such as:

- Use multiple frameworks on the same page without page refreshing (React, AngularJS, Angular, Vue, etc.)
- Write code using a new framework, without rewriting your existing app
- Lazy load code for improved initial load time.

POC TECH – OBSERVABLE STORE

Why I chose ObservableStore

- Removes Redux boilerplate
- Defines APIs between MFEs
- Consumer retrieves the data structure they want (RxJS)
- Independently deployable
- Community and proven in production

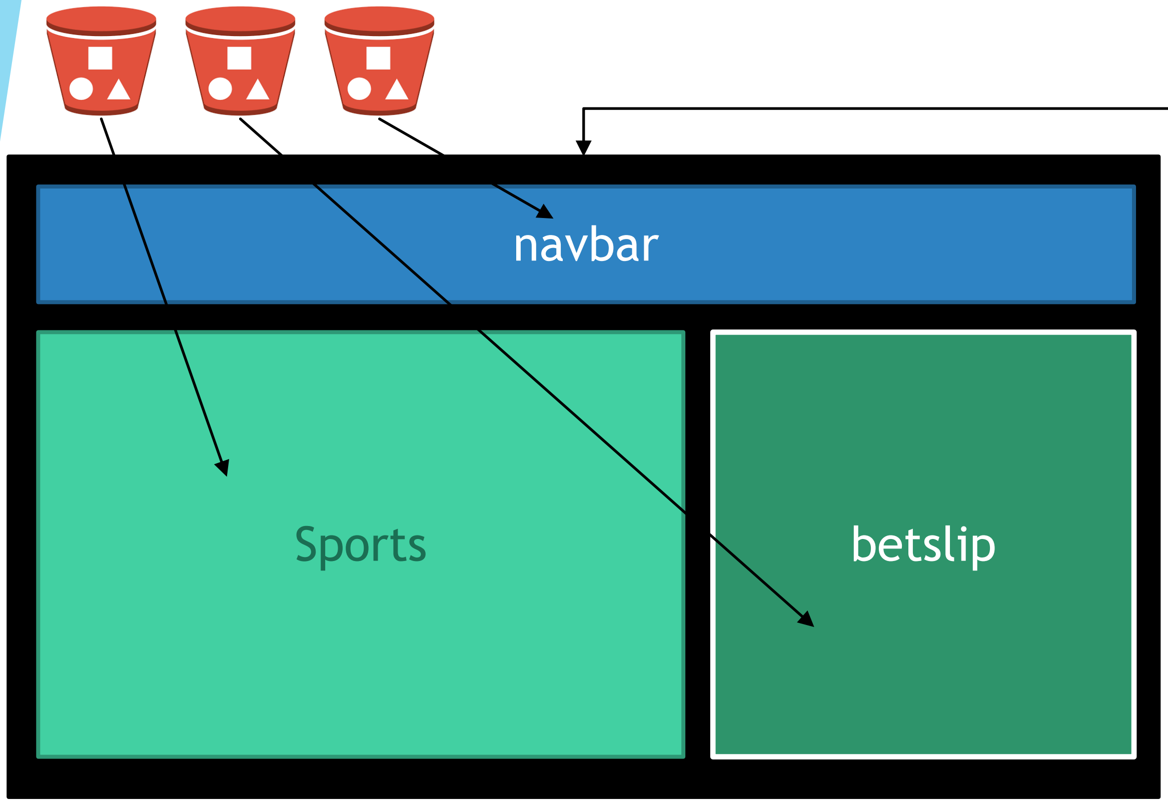


https://github.com/DanWahlin/Observable-Store

Key Goals of Observable Store:

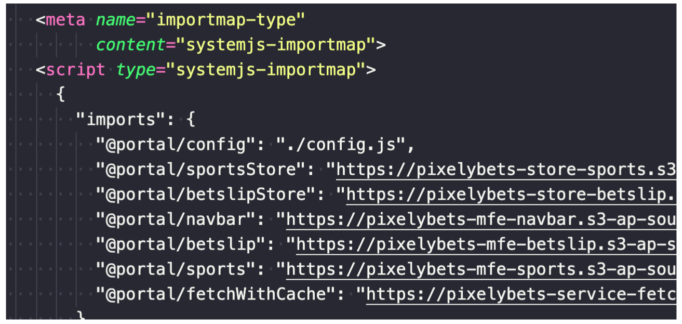
- Keep it simple!
- Single source of truth for state
- Store state is immutable
- Provide state change notifications to any subscriber
- Track state change history
- Easy to understand with a minimal amount of code required to get started
- Works with any front-end project built with JavaScript or TypeScript (Angular, React, Vue, etc.)
- Integrate with the Redux DevTools (Angular and React currently supported)

MFE ARCHITECTURE – OVERVIEW

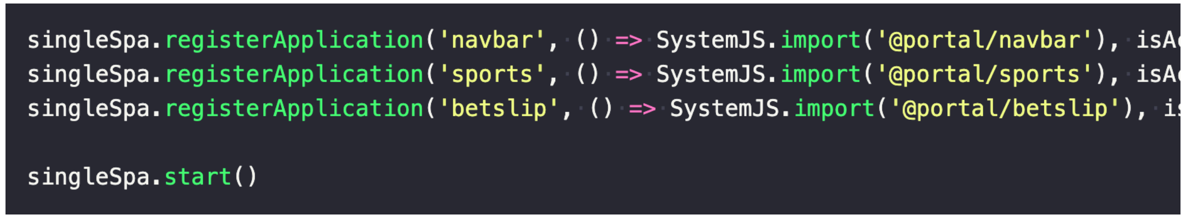


App shell – SINGLE-SPA

- Common dependencies
- Import-map (Inline/performance)
- SINGLE-SPA config



```
<meta name="importmap-type"
      content="systemjs-importmap">
<script type="systemjs-importmap">
  {
    "imports": {
      "@portal/config": "./config.js",
      "@portal/sportsStore": "https://pixelybets-store-sports.s3
      "@portal/betslipStore": "https://pixelybets-store-betslip.
      "@portal/navbar": "https://pixelybets-mfe-navbar.s3-ap-sou
      "@portal/betslip": "https://pixelybets-mfe-betslip.s3-ap-s
      "@portal/sports": "https://pixelybets-mfe-sports.s3-ap-sou
      "@portal/fetchWithCache": "https://pixelybets-service-fetc
    }
```

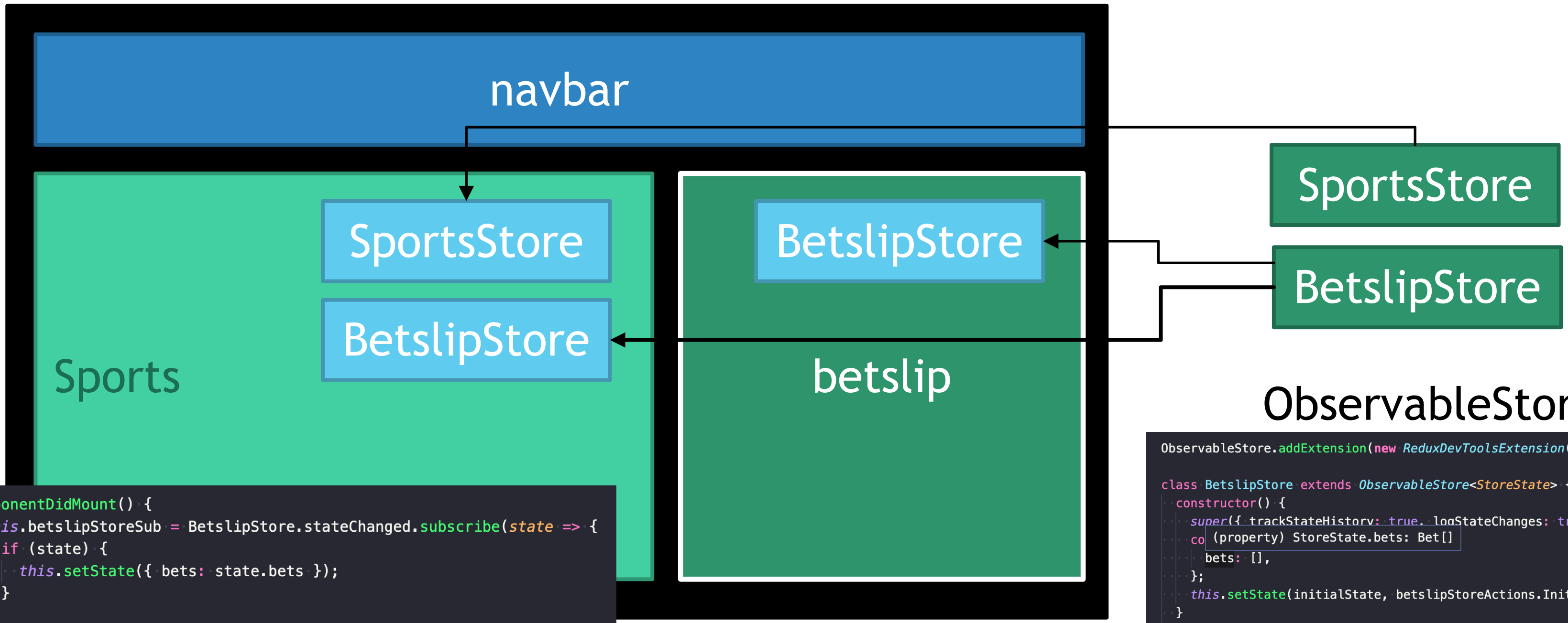


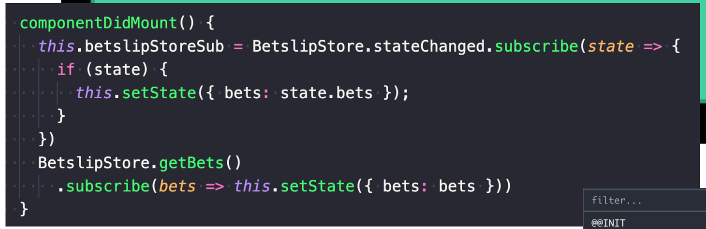
```
singleSpa.registerApplication('navbar', () => SystemJS.import('@portal/navbar'), isA
singleSpa.registerApplication('sports', () => SystemJS.import('@portal/sports'), isA
singleSpa.registerApplication('betslip', () => SystemJS.import('@portal/betslip'), i

singleSpa.start()
```

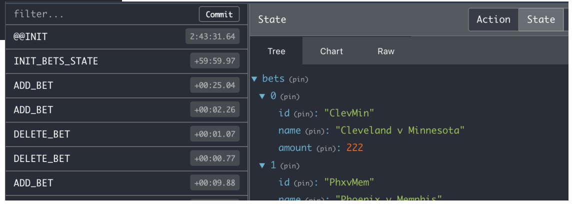
MFE ARCHITECTURE – COMMUNICATION

Observable stores allow for decoupled services (Think frontend BFF) that define public APIs for consumers.





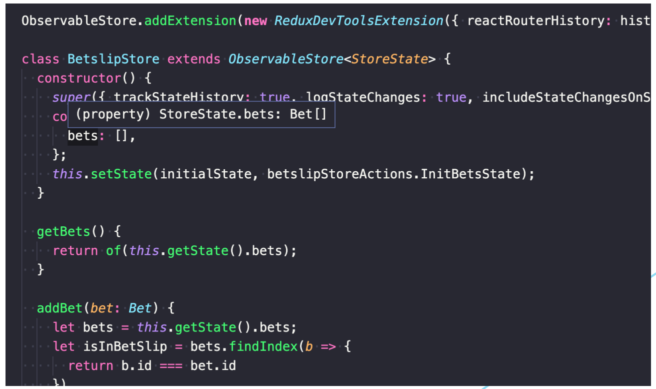
```
componentDidMount() {
  this.betslipStoreSub = BetslipStore.stateChanged.subscribe(state => {
    if (state) {
      this.setState({ bets: state.bets });
    }
  })
  BetslipStore.getBets()
    .subscribe(bets => this.setState({ bets: bets }))
}
```



```
filter...                    Commit
@@INIT                   2:43:31.64
INIT_BETS_STATE          +59:59.97
ADD_BET                  +00:25.04
ADD_BET                  +00:02.26
DELETE_BET               +00:01.07
DELETE_BET               +00:00.77
ADD_BET                  +00:09.88
```

```
State                         Action  State
Tree  Chart  Raw
▼ bets (pin)
  ▼ 0 (pin)
      id (pin): "ClevMin"
      name (pin): "Cleveland v Minnesota"
      amount (pin): 222
  ▼ 1 (pin)
      id (pin): "PhxvMem"
      name (pin): "Phoenix v Memphis"
```

ObservableStor



```
ObservableStore.addExtension(new ReduxDevToolsExtension({ reactRouterHistory: hist

class BetslipStore extends ObservableStore<StoreState> {
  constructor() {
    super({ trackStateHistory: true, logStateChanges: true, includeStateChangesOnS
    co (property) StoreState.bets: Bet[]
      bets: [],
    };
    this.setState(initialState, betslipStoreActions.InitBetsState);
  }

  getBets() {
    return of(this.getState().bets);
  }

  addBet(bet: Bet) {
    let bets = this.getState().bets;
    let isInBetSlip = bets.findIndex(b => {
      return b.id === bet.id
    })
```

DEMO – CI/CD

- **Git repos**
  - https://github.com/pixelypants/pixelybets-app

  - https://github.com/pixelypants/pixelybets-mfe-navbar
  - https://github.com/pixelypants/pixelybets-mfe-sports
  - https://github.com/pixelypants/pixelybets-mfe-betslip

  - https://github.com/pixelypants/pixelybets-store-betslip
  - https://github.com/pixelypants/pixelybets-store-sports
- **Travis-ci**
  - https://travis-ci.com/
- **AWS S3**
  - https://s3.console.aws.amazon.com/s3/home?region=ap-southeast-2

DEMO – DX

- **Run App Shell locally**
  - http://localhost:8233/
- **Show application and working from cloud**
  - Show Redux DevTool integration
- **Show local DX**
  - Override MFEs with **Import Map Overrides**
    - Allow for developing against prod
  - Show SINGLE-SPA DevTool
    - Mount/Unmount and overlays
  - Show the need for many VSCode instances
- **Show source code**
    - SINGLE-SPA config
    - Import maps
    - Store APIs
    - MFEs using Stores

React

webpack

SystemJS

RxJS

SINGLE-SPA

DEMO – WHERE TO NEXT

- **Integrate framework agnostic component lib**
    - https://github.com/pixelypants/pixely-ui

WEBCOMPONENTS

stencil