Types of Repos

Core Library

@fubo/core

The core's job is to provide shared dependencies and utilities to all the projects in our micro-fe ecosystem.

It ensures everyone is using the same version of React or the design system, for example.

It can provide common configurations for our workspaces like standard TypeScript or ESLint settings.

We can also use it to orchestrate our local environments for development, since there might potentially be many servers that need to run locally.

It is a single package, distributed as a library.

Design System

@fubo/fds

Feature Repos

@fubo/fan-view

@fubo/stac

@fubo/users

These guys are app and platform agnostic (provided we are using JS / web technologies).

They each contain a sandbox app for local development, and provide the flexibility to distribute many packages, which can also be exposed as federated modules.

Because these repos distribute many packages, they can depend on each other (no more circular dependency errors).

Platform Repos

@fubo/smarttv

@fubo/web

These are similar to feature repos, except that they are more focused on specific platforms (but not apps!).

You can think of these like libraries of components (or anything else) that are tailored for a particular experience.

SmartTV for example would consume feature components from feature repos and compose them into page components which would then be consumed by top-level apps.

Like feature repos, these can also distribute many packages - up to you!

App Repos

@fubo/app/tizen

@fubo/app/xbox

@fubo/app/vizio

@fubo/app/web

These are platform-specific apps that should aim to be deployed as seldomly as possible.

The idea here is that apps are just entry-level routers that set up anything we need globally - like state providers.

They should fetch what they need at runtime (as needed), using federated modules. This can include root-level route configs, prefetching / state-hydrating services, whatever.

Anatomy of an App

Build Time (package)

Run Time (federated)

dependency (package.json)

@fubo/app/tizen

@fubo/app/xbox

```
<Root>
    <AppProvider>
        <FederatedRoutes />
    </AppProvider>
</Root>
```

@fubo/core

@fubo/smarttv

```
<Router>
    <Route path="/" element={<FederatedHomePage />}>
    <Route path="/search" element={<FederatedSearchPage />}>
</Router>
```

```
<HomePage>
    <CurrentUserInfo>
    <FederatedSTACCarousel />
</HomePage>
```

This package lives inside the @fubo/users repo

@fubo/users__components__current-user-info

@fubo/stac

```
<CurrentUserInfo>
    <Avatar />
    <DisplayName />
</CurrentUserInfo>
```

@fubo/fds

Developer Experience

- Each repo (except the core) will have a standalone server which can be thought of as a sandbox to render / test components (or other packages).
- When developing a package, use relative path imports (package.json and any files that import this package).
- BUT make sure that once you're done working on a package that it is always imported as a package (from node_modules).
- When consuming a federated module, you still need to import the package to use types.
- In local development, you will need to run servers for any federated modules you want to consume (we can probably orchestrate this..).