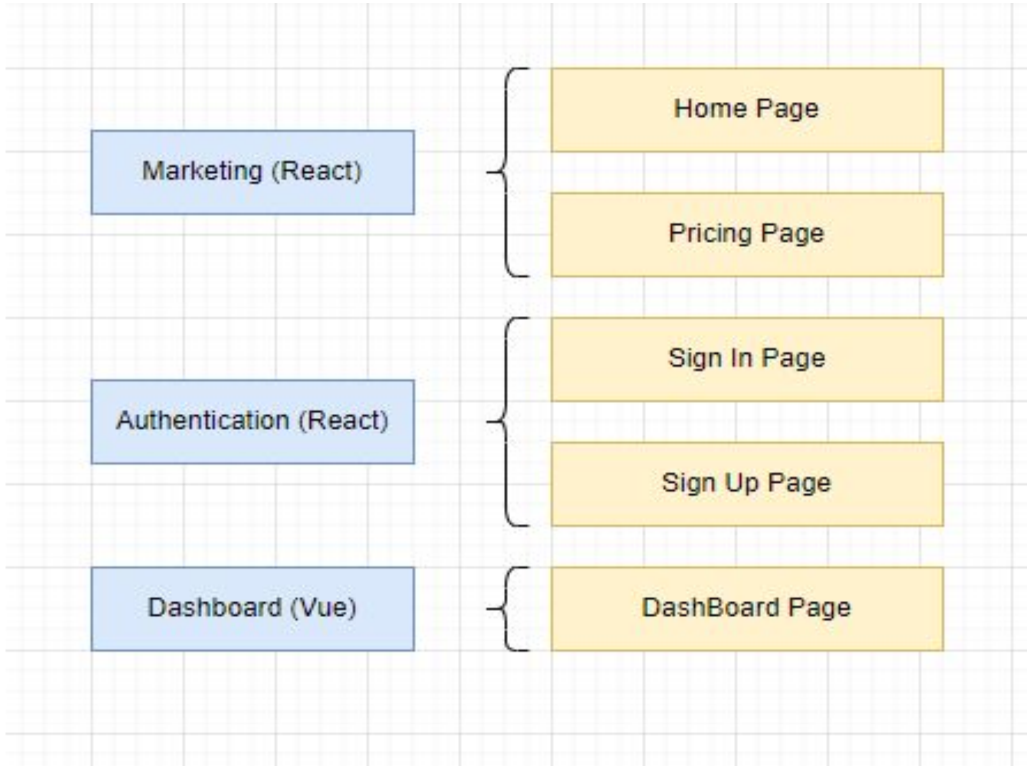
Microfrontends - Intermediate Level

Alfio Martini

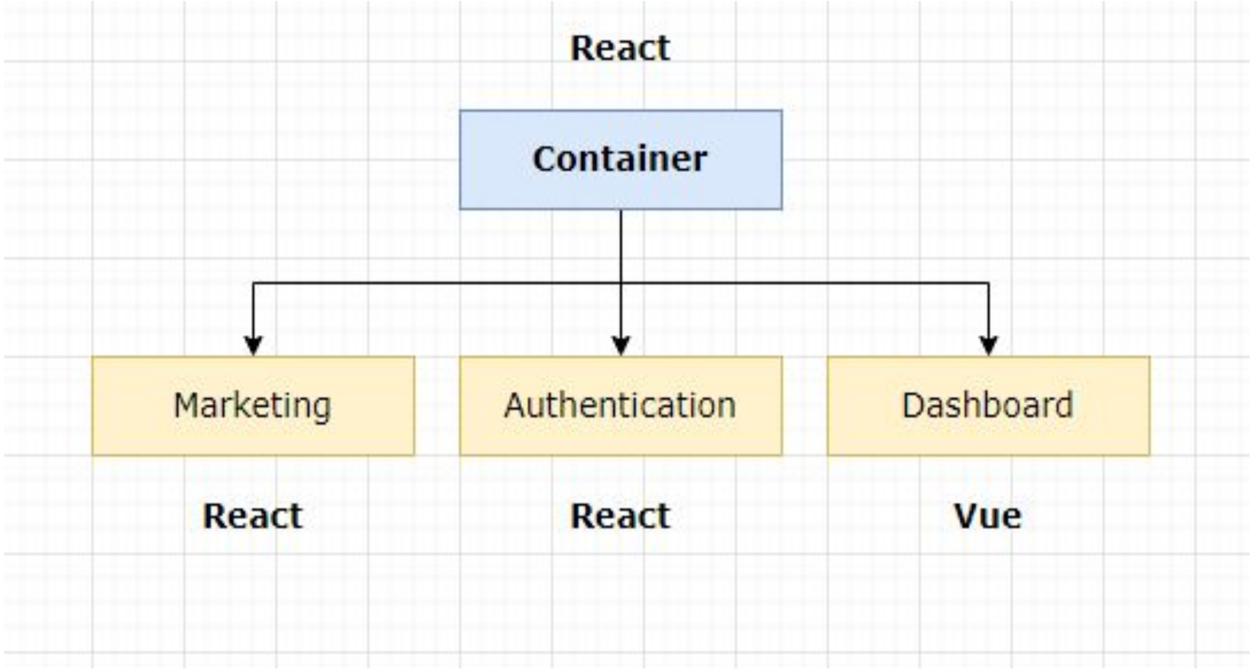
Simplifying Assumptions

- *No actual data*
- *No API*
- *No database*
- *No real authentication*
- ***Routing***
- *Focus:* ***Integration*** *of frontend frameworks* (React & Vue)

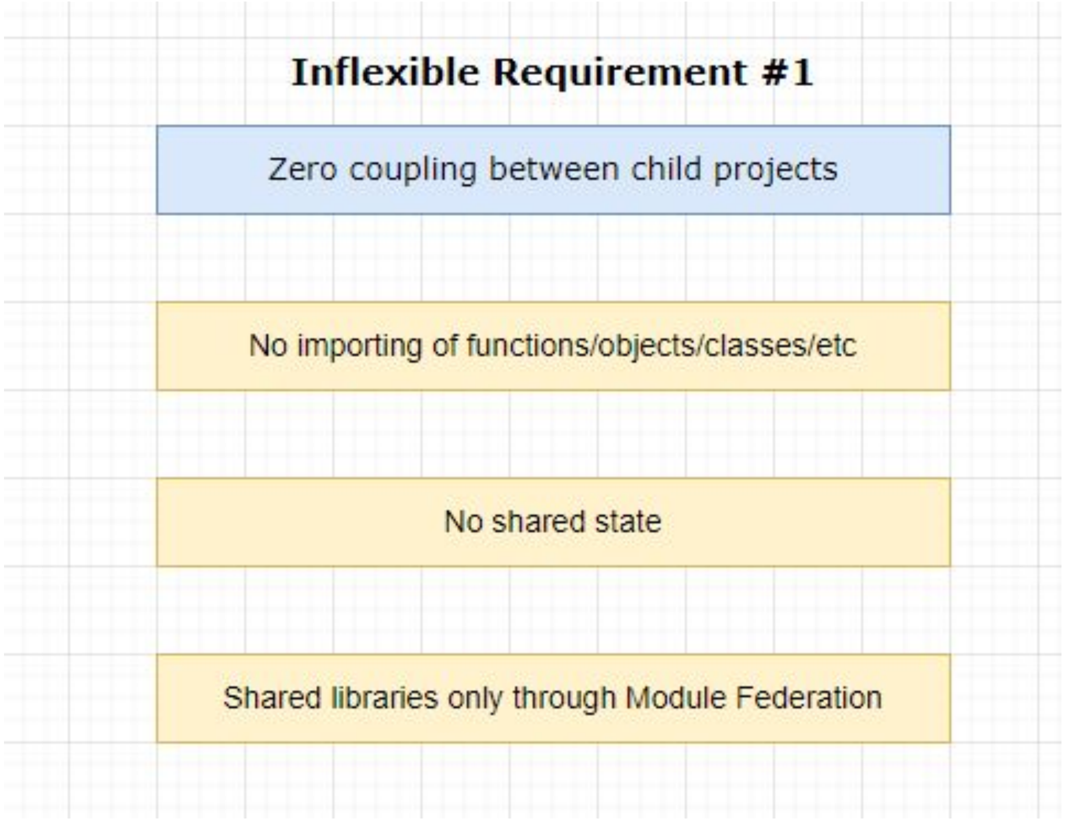
Pages and Apps



Integration



Architectural Requirements



Inflexible Requirement #1

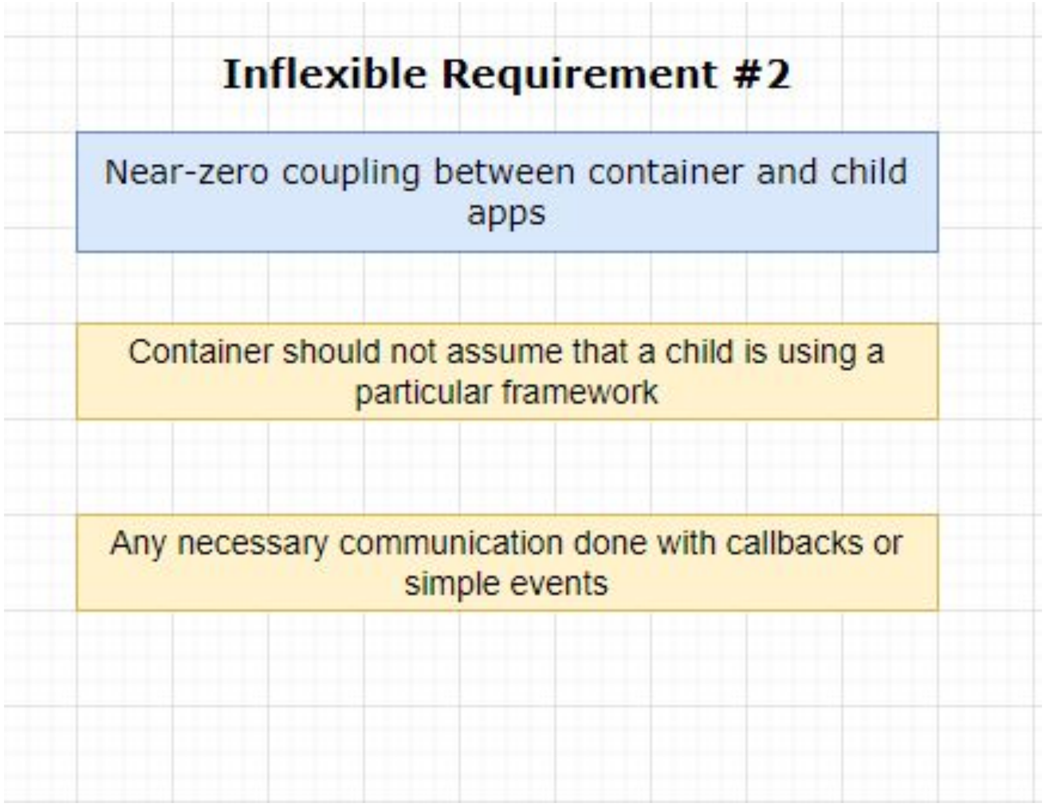
Zero coupling between child projects

No importing of functions/objects/classes/etc

No shared state

Shared libraries only through Module Federation

Architectural Requirements



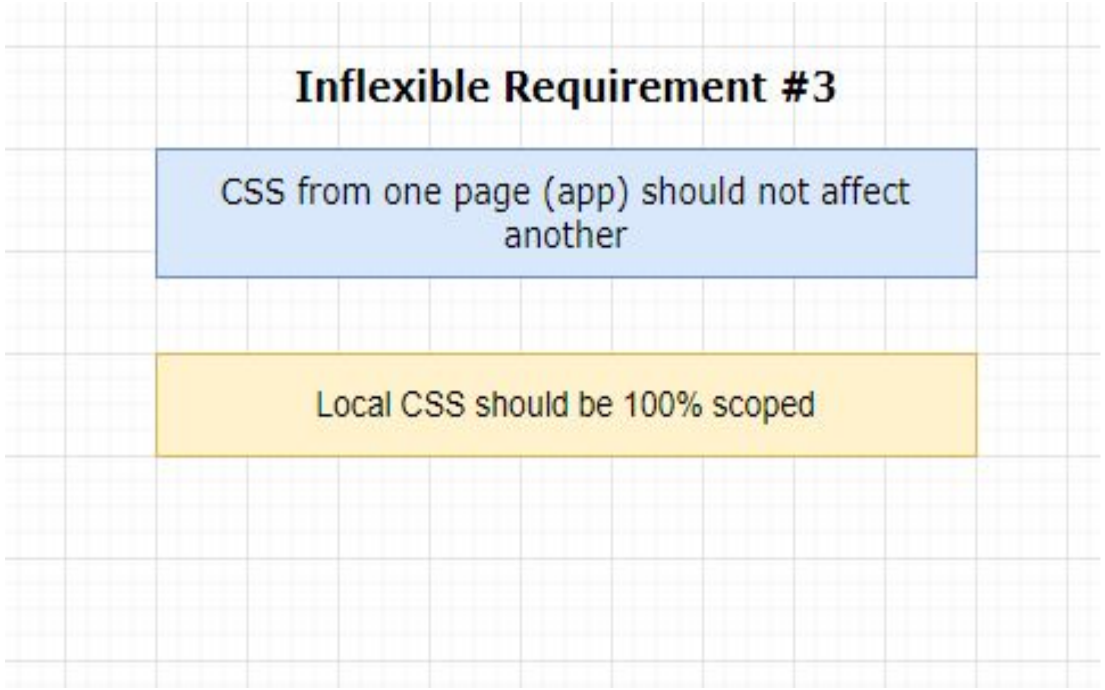
Inflexible Requirement #2

Near-zero coupling between container and child apps

Container should not assume that a child is using a particular framework

Any necessary communication done with callbacks or simple events

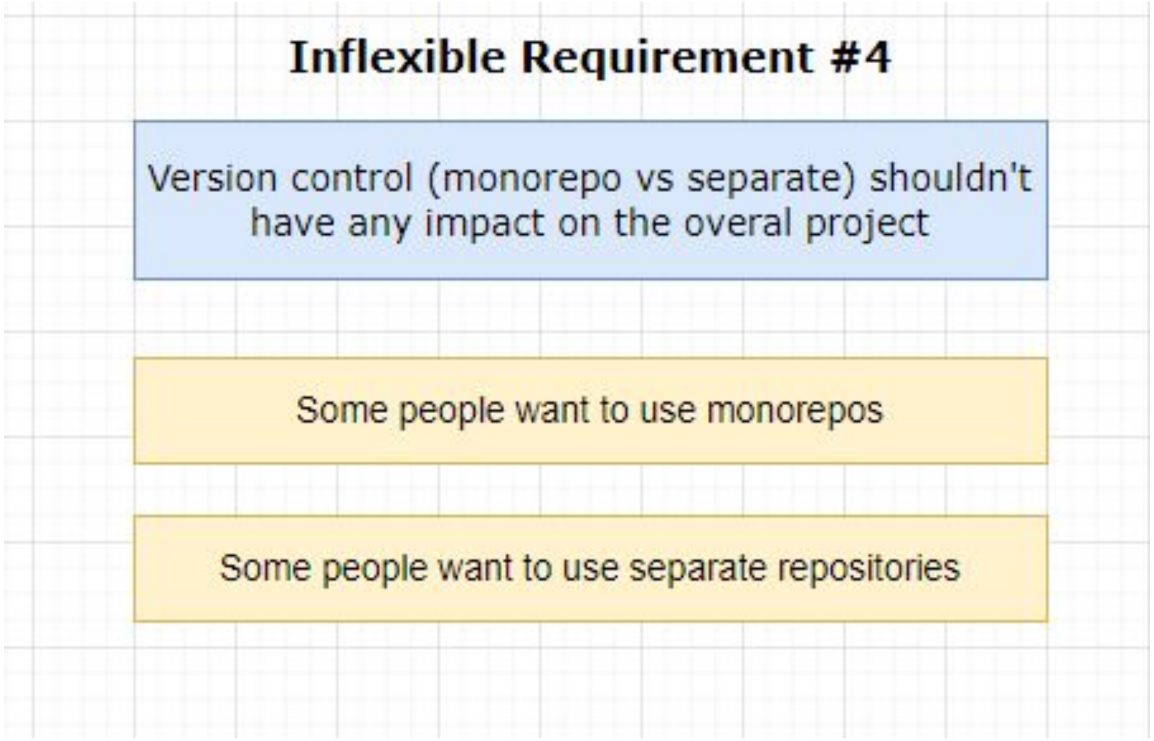
Architectural Requirements


Inflexible Requirement #3

CSS from one page (app) should not affect another

Local CSS should be 100% scoped

Architectural Requirements

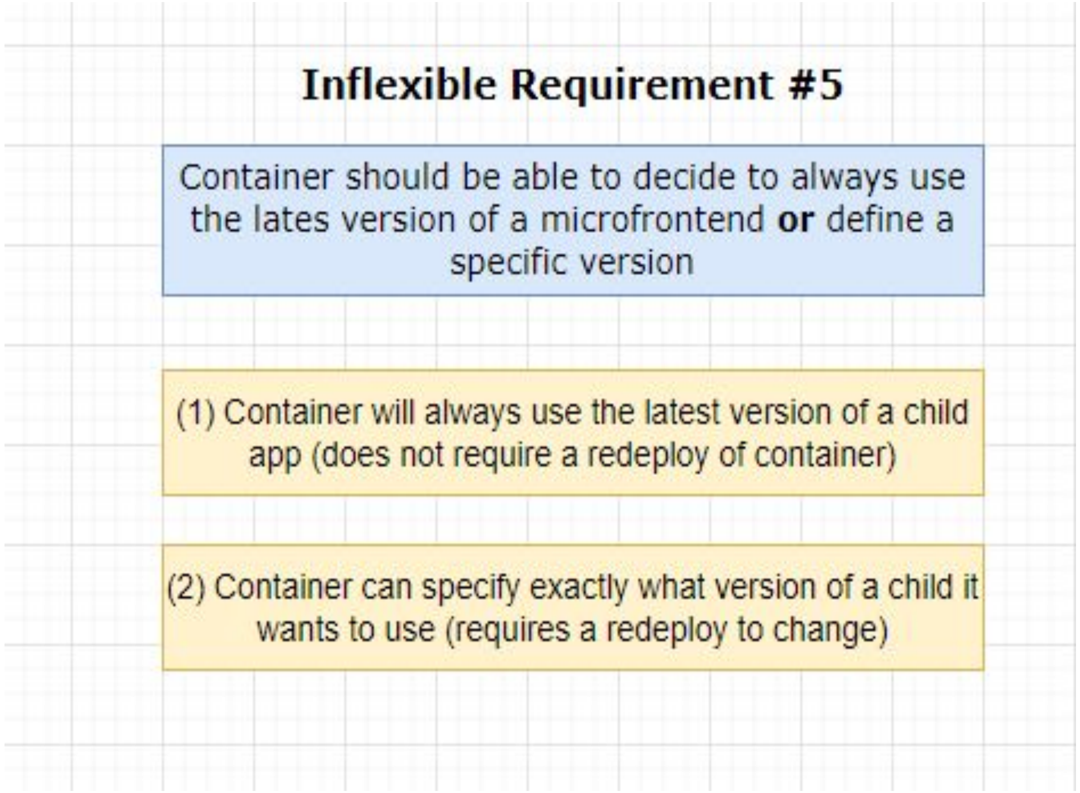

Inflexible Requirement #4

Version control (monorepo vs separate) shouldn't have any impact on the overal project

Some people want to use monorepos

Some people want to use separate repositories

Architectural Requirements

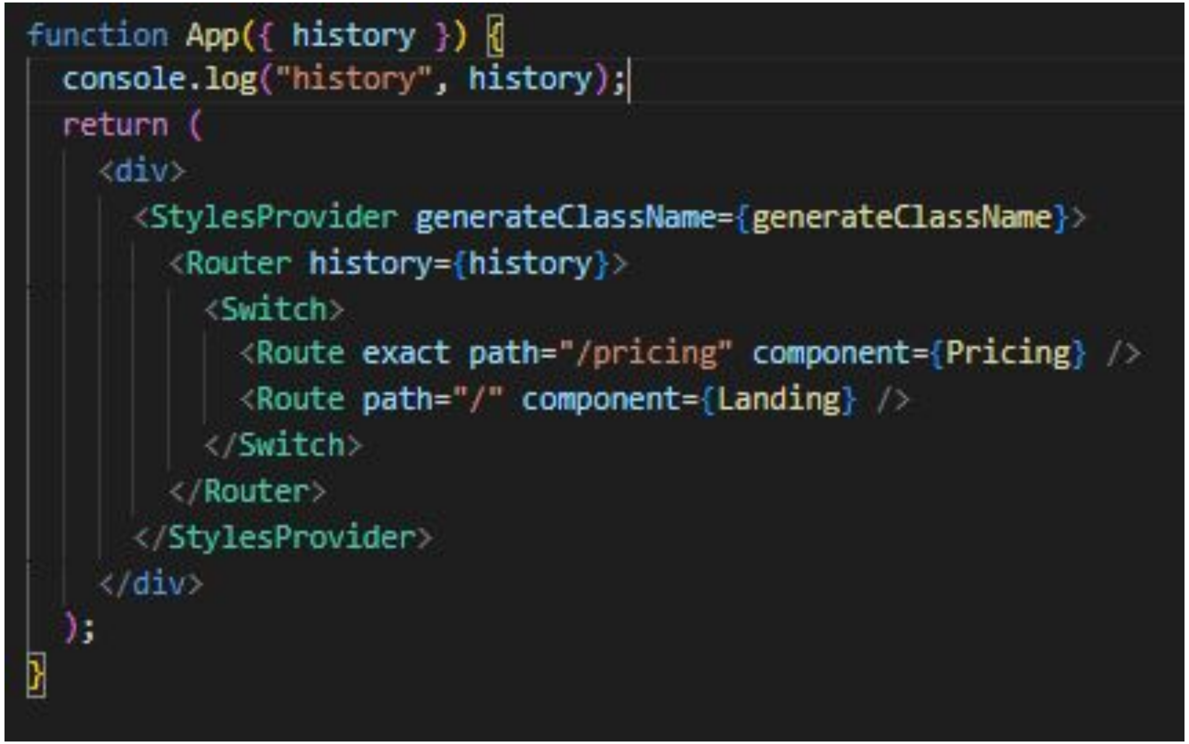

Inflexible Requirement #5

Container should be able to decide to always use the lates version of a microfrontend **or** define a specific version

(1) Container will always use the latest version of a child app (does not require a redeploy of container)

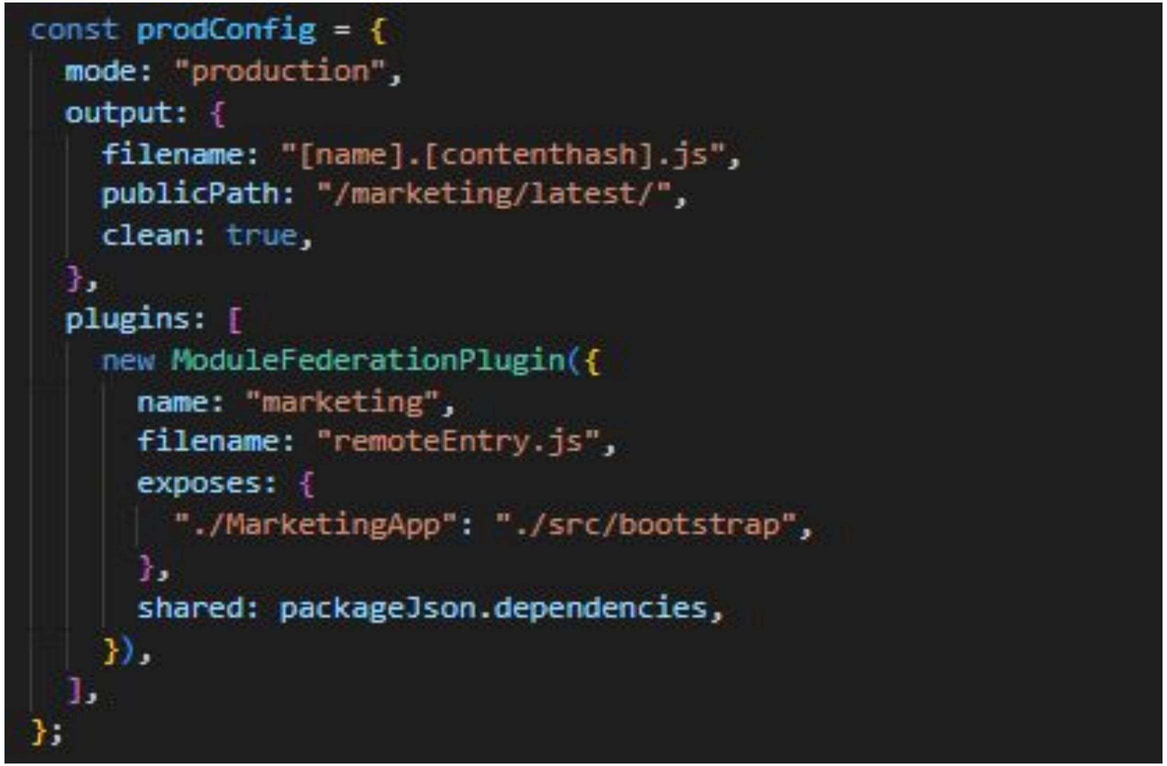
(2) Container can specify exactly what version of a child it wants to use (requires a redeploy to change)

Marketing - App



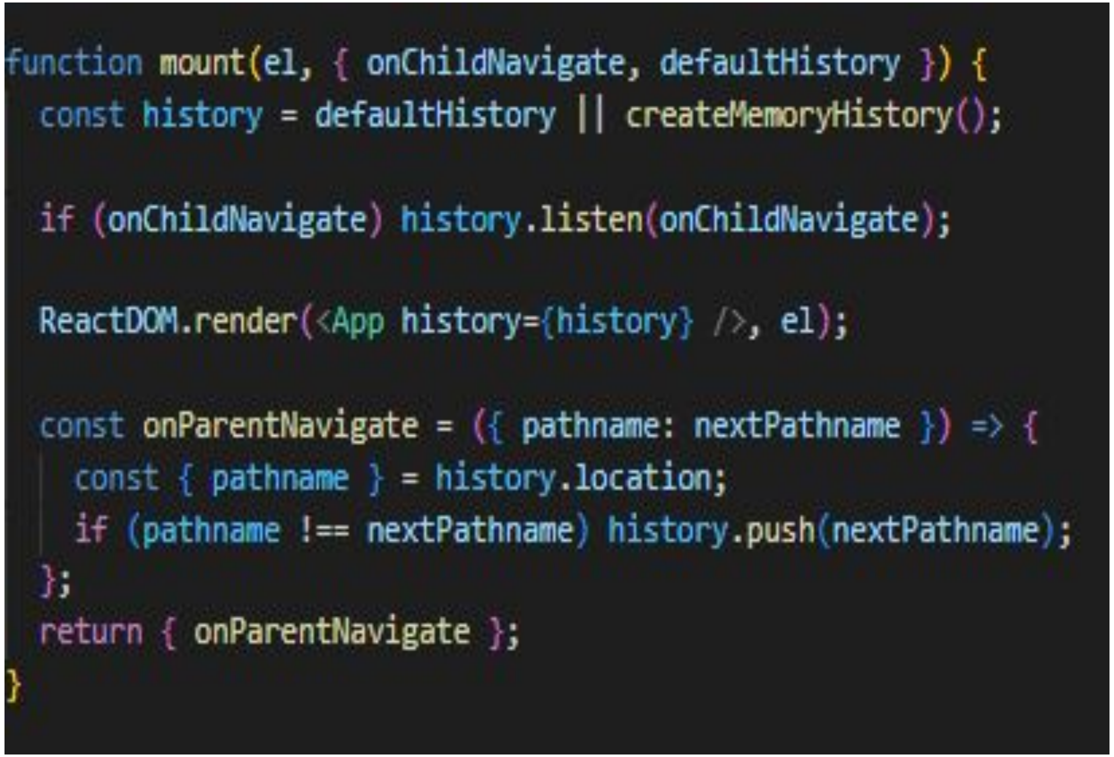
```
function App({ history }) {
  console.log("history", history);
  return (
    <div>
      <StylesProvider generateClassName={generateClassName}>
        <Router history={history}>
          <Switch>
            <Route exact path="/pricing" component={Pricing} />
            <Route path="/" component={Landing} />
          </Switch>
        </Router>
      </StylesProvider>
    </div>
  );
}
```

Marketing - webpack.prod.js



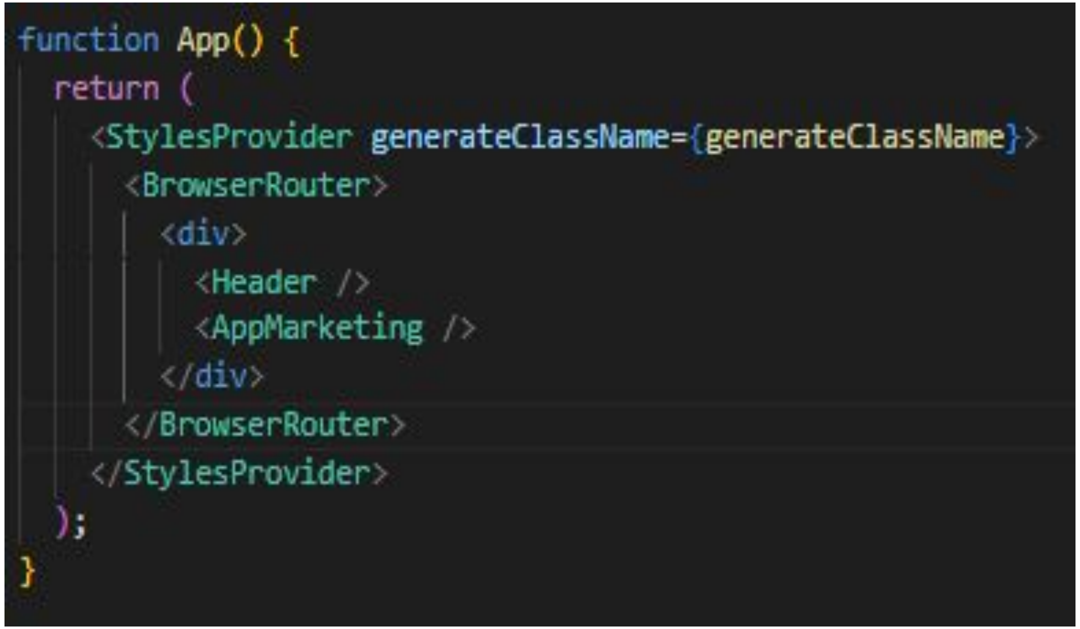
```
const prodConfig = {
  mode: "production",
  output: {
    filename: "[name].[contenthash].js",
    publicPath: "/marketing/latest/",
    clean: true,
  },
  plugins: [
    new ModuleFederationPlugin({
      name: "marketing",
      filename: "remoteEntry.js",
      exposes: {
        "./MarketingApp": "./src/bootstrap",
      },
      shared: packageJson.dependencies,
    }),
  ],
};
```

Marketing - bootstrap.js



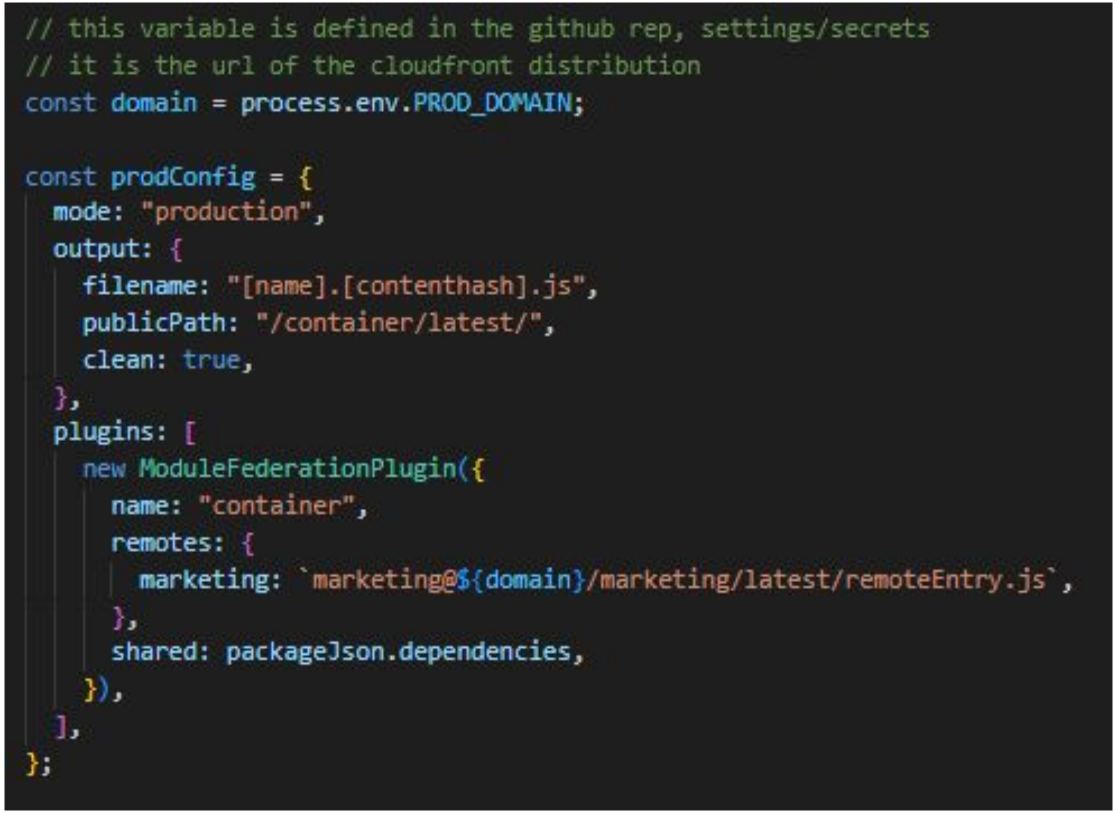
```javascript
function mount(el, { onChildNavigate, defaultHistory }) {
  const history = defaultHistory || createMemoryHistory();

  if (onChildNavigate) history.listen(onChildNavigate);

  ReactDOM.render(<App history={history} />, el);

  const onParentNavigate = ({ pathname: nextPathname }) => {
    const { pathname } = history.location;
    if (pathname !== nextPathname) history.push(nextPathname);
  };
  return { onParentNavigate };
}
```

Container - App



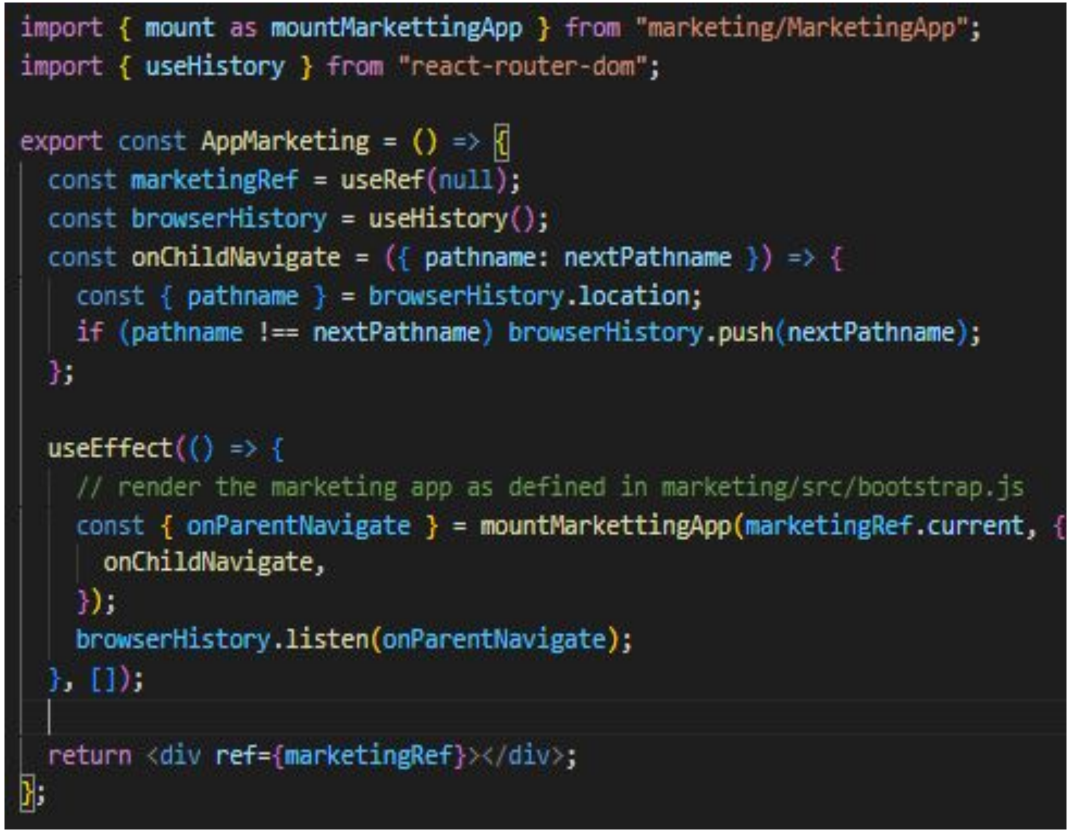
```
function App() {
  return (
    <StylesProvider generateClassName={generateClassName}>
      <BrowserRouter>
        <div>
          <Header />
          <AppMarketing />
        </div>
      </BrowserRouter>
    </StylesProvider>
  );
}
```

Container - webpack.prod.js



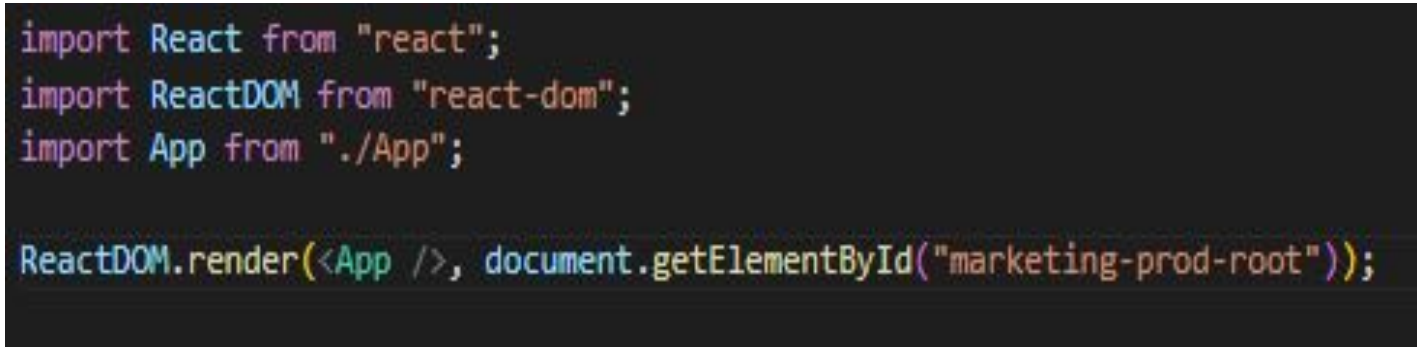
```
// this variable is defined in the github rep, settings/secrets
// it is the url of the cloudfront distribution
const domain = process.env.PROD_DOMAIN;

const prodConfig = {
  mode: "production",
  output: {
    filename: "[name].[contenthash].js",
    publicPath: "/container/latest/",
    clean: true,
  },
  plugins: [
    new ModuleFederationPlugin({
      name: "container",
      remotes: {
        marketing: `marketing@${domain}/marketing/latest/remoteEntry.js`,
      },
      shared: packageJson.dependencies,
    }),
  ],
};
```

Container - App Marketing



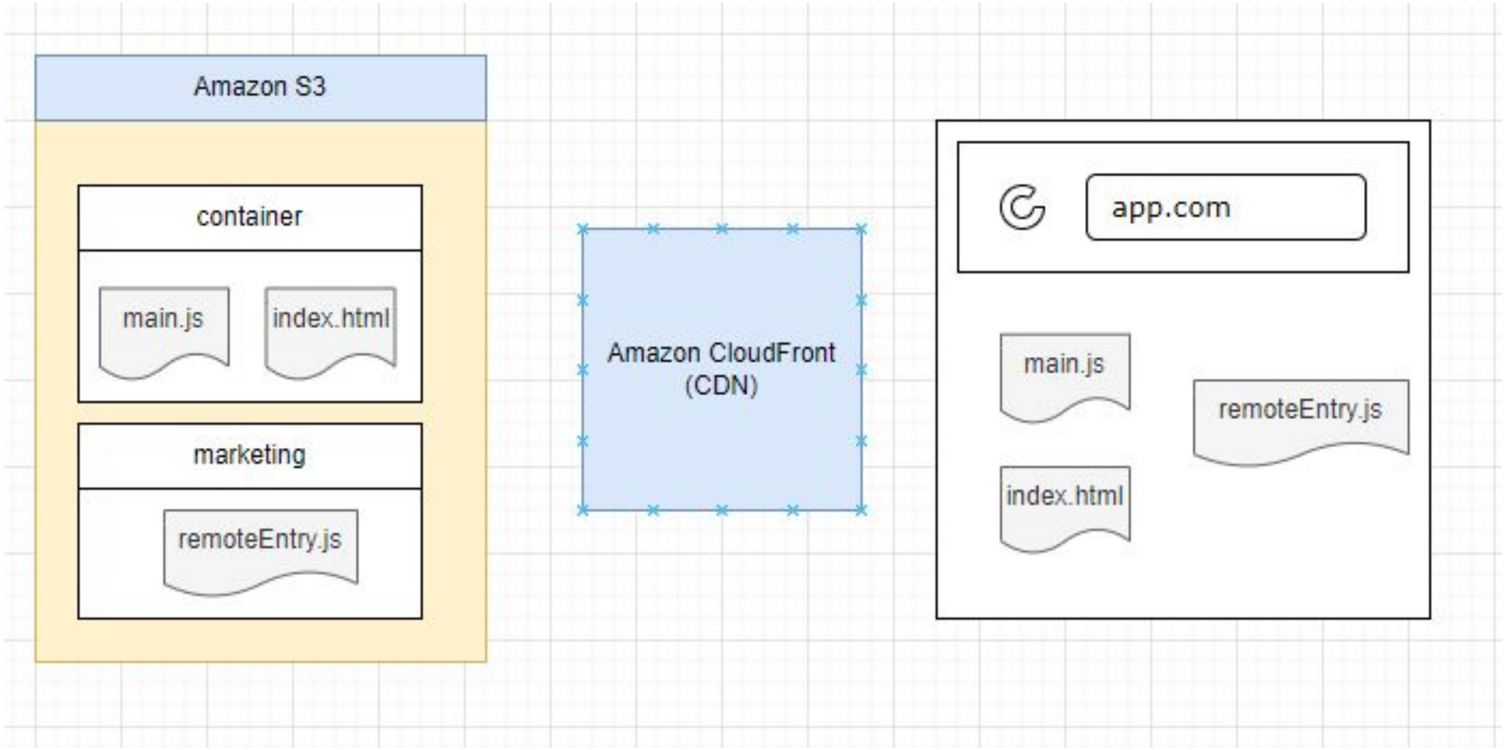
```
import { mount as mountMarkettingApp } from "marketing/MarketingApp";
import { useHistory } from "react-router-dom";

export const AppMarketing = () => {
  const marketingRef = useRef(null);
  const browserHistory = useHistory();
  const onChildNavigate = ({ pathname: nextPathname }) => {
    const { pathname } = browserHistory.location;
    if (pathname !== nextPathname) browserHistory.push(nextPathname);
  };

  useEffect(() => {
    // render the marketing app as defined in marketing/src/bootstrap.js
    const { onParentNavigate } = mountMarkettingApp(marketingRef.current, {
      onChildNavigate,
    });
    browserHistory.listen(onParentNavigate);
  }, []);

  return <div ref={marketingRef}></div>;
};
```

Container - bootstrap.js

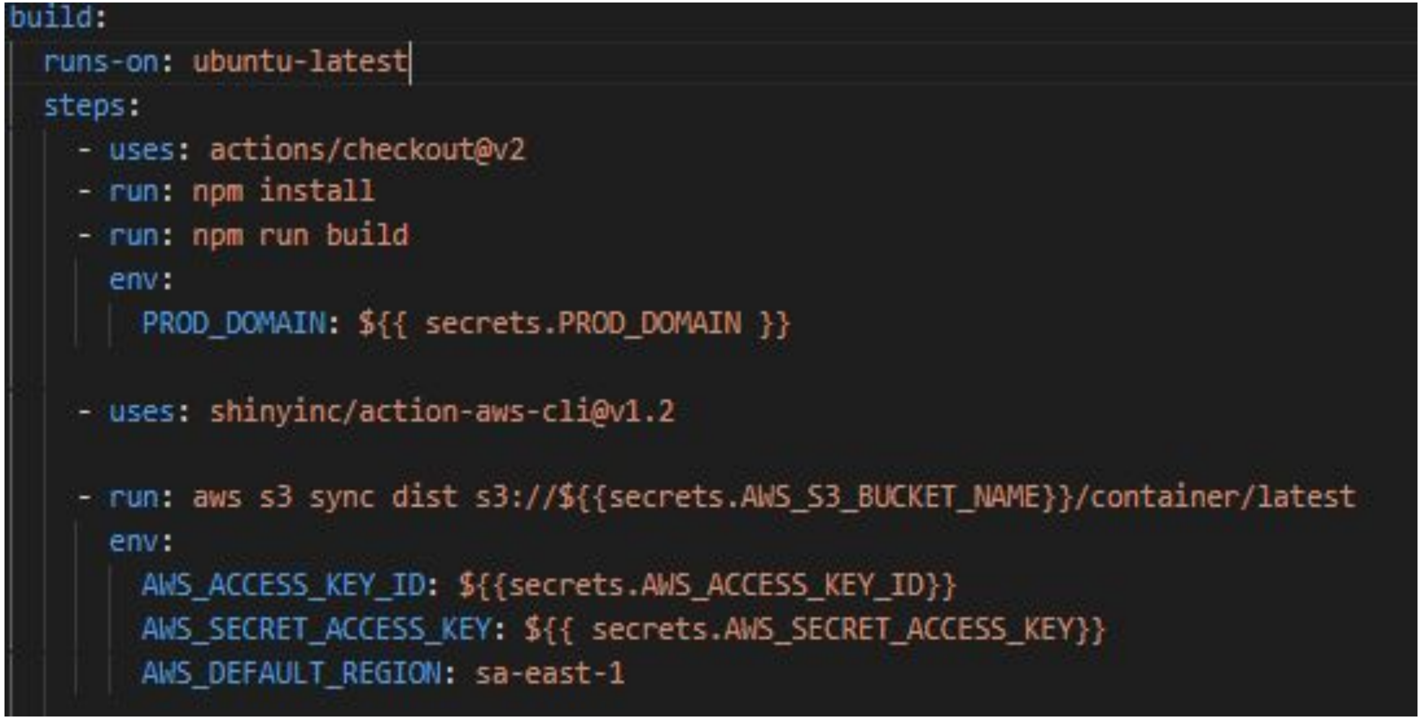


```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";

ReactDOM.render(<App />, document.getElementById("marketing-prod-root"));
```

Deployment - AWS

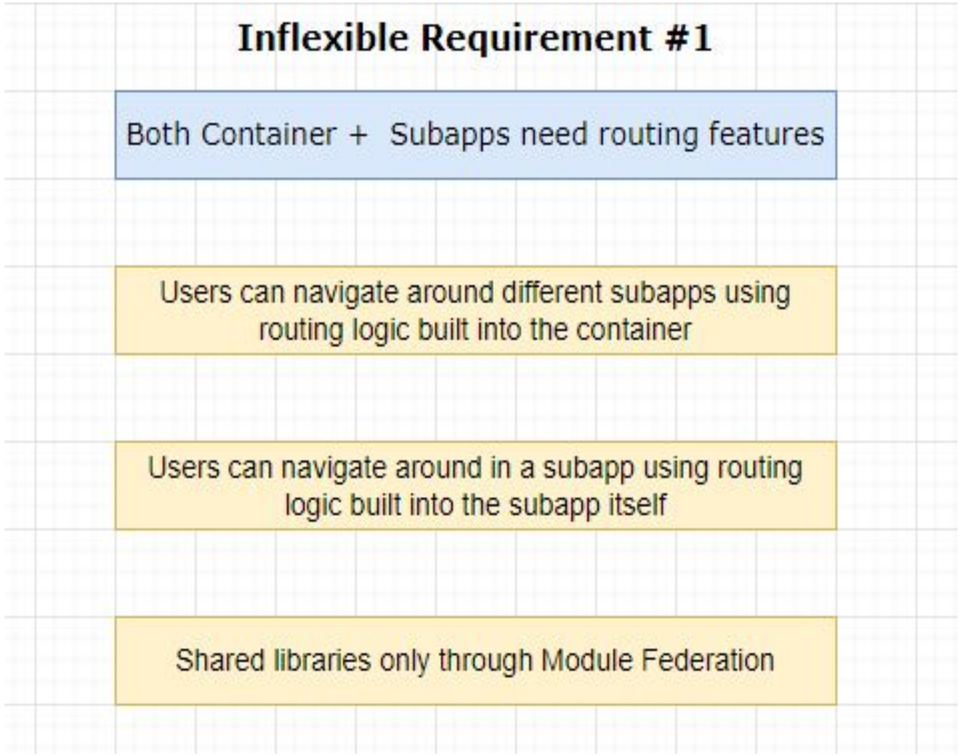


Container - CI/CD Pipeline



```
build:
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2
    - run: npm install
    - run: npm run build
      env:
        PROD_DOMAIN: ${{ secrets.PROD_DOMAIN }}

    - uses: shinyinc/action-aws-cli@v1.2

    - run: aws s3 sync dist s3://${{secrets.AWS_S3_BUCKET_NAME}}/container/latest
      env:
        AWS_ACCESS_KEY_ID: ${{secrets.AWS_ACCESS_KEY_ID}}
        AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY}}
        AWS_DEFAULT_REGION: sa-east-1
```

Routing - Requirements



Inflexible Requirement #1

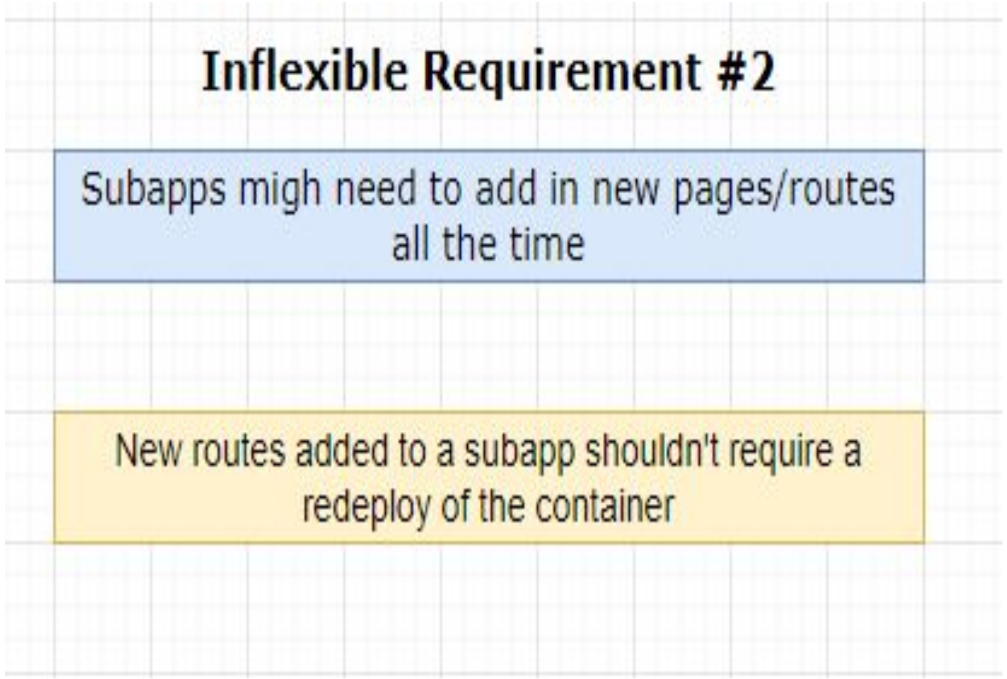
Both Container + Subapps need routing features

Users can navigate around different subapps using routing logic built into the container

Users can navigate around in a subapp using routing logic built into the subapp itself

Shared libraries only through Module Federation

Routing - Requirements

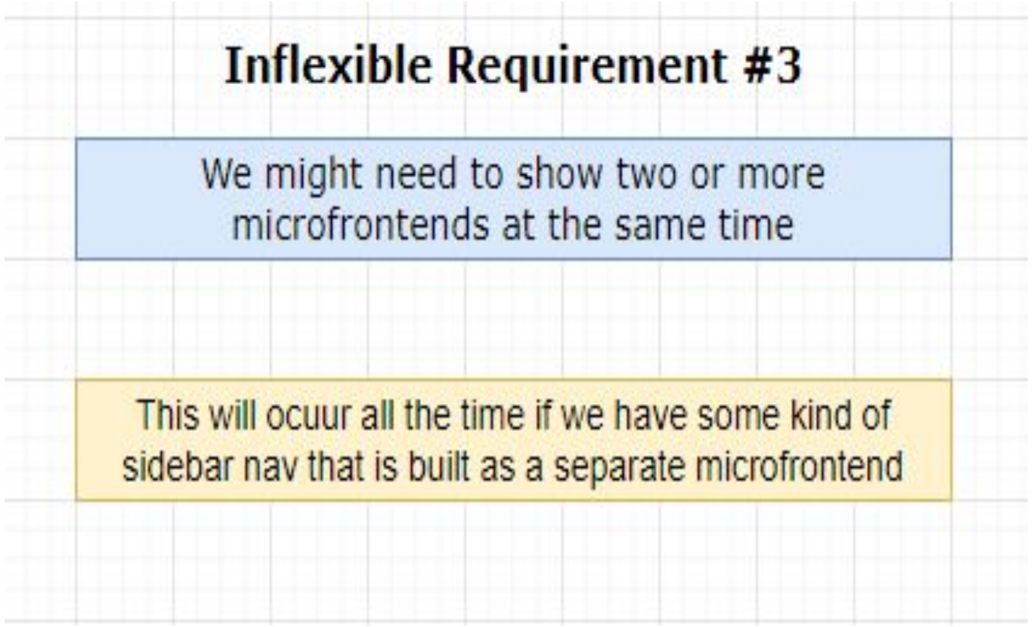


Inflexible Requirement #2

Subapps migh need to add in new pages/routes all the time

New routes added to a subapp shouldn't require a redeploy of the container

Routing Requirements

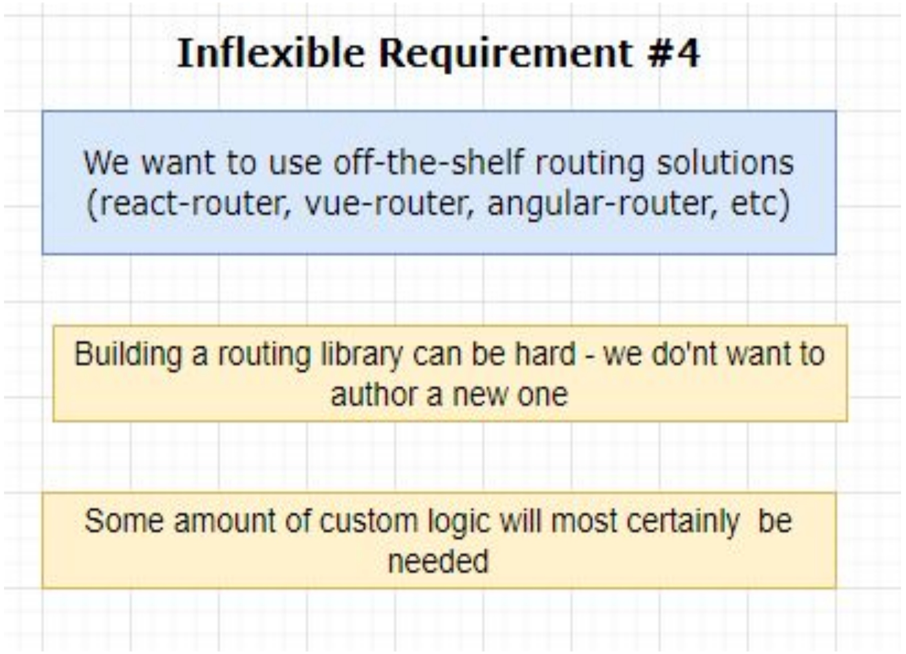


Inflexible Requirement #3

We might need to show two or more microfrontends at the same time

This will ocuur all the time if we have some kind of sidebar nav that is built as a separate microfrontend

Routing - Requirements



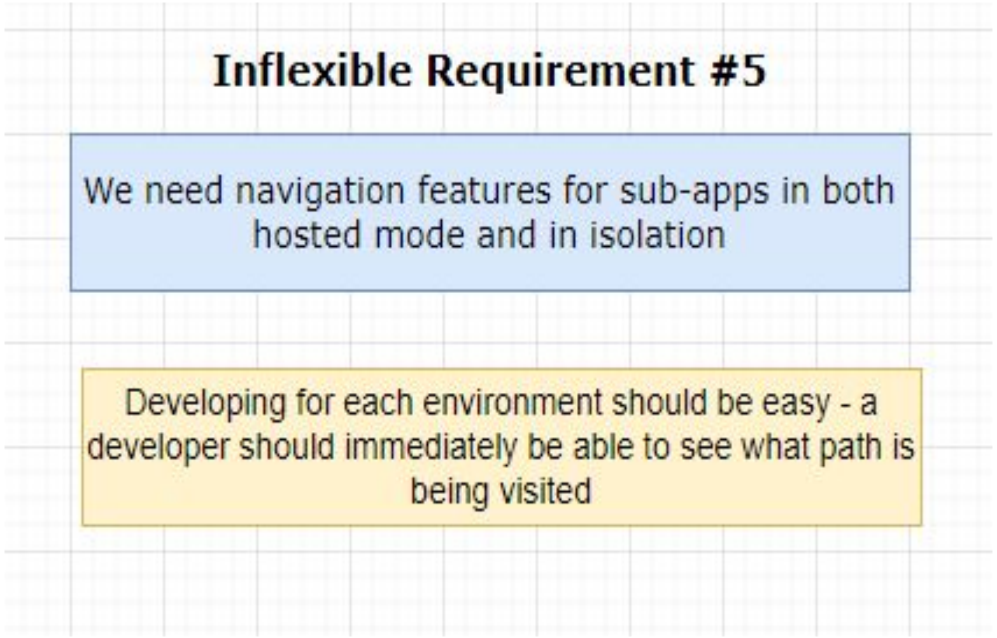
Inflexible Requirement #4

We want to use off-the-shelf routing solutions (react-router, vue-router, angular-router, etc)

Building a routing library can be hard - we do'nt want to author a new one

Some amount of custom logic will most certainly be needed

Routing - Requirements

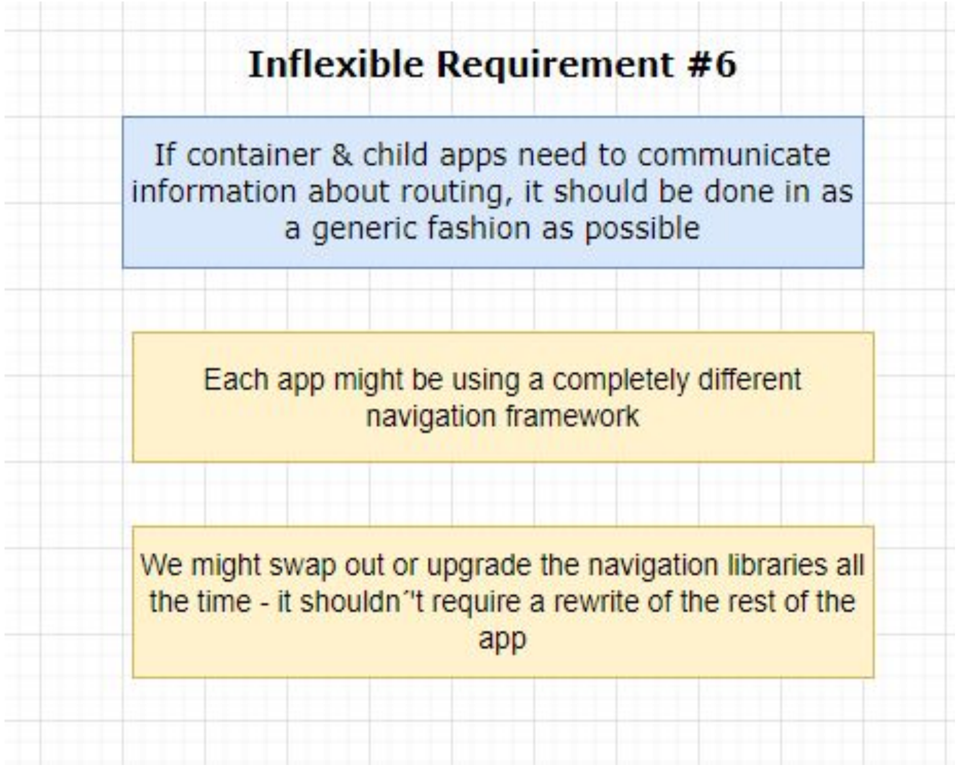


Inflexible Requirement #5

We need navigation features for sub-apps in both hosted mode and in isolation

Developing for each environment should be easy - a developer should immediately be able to see what path is being visited

Routing - Requirements



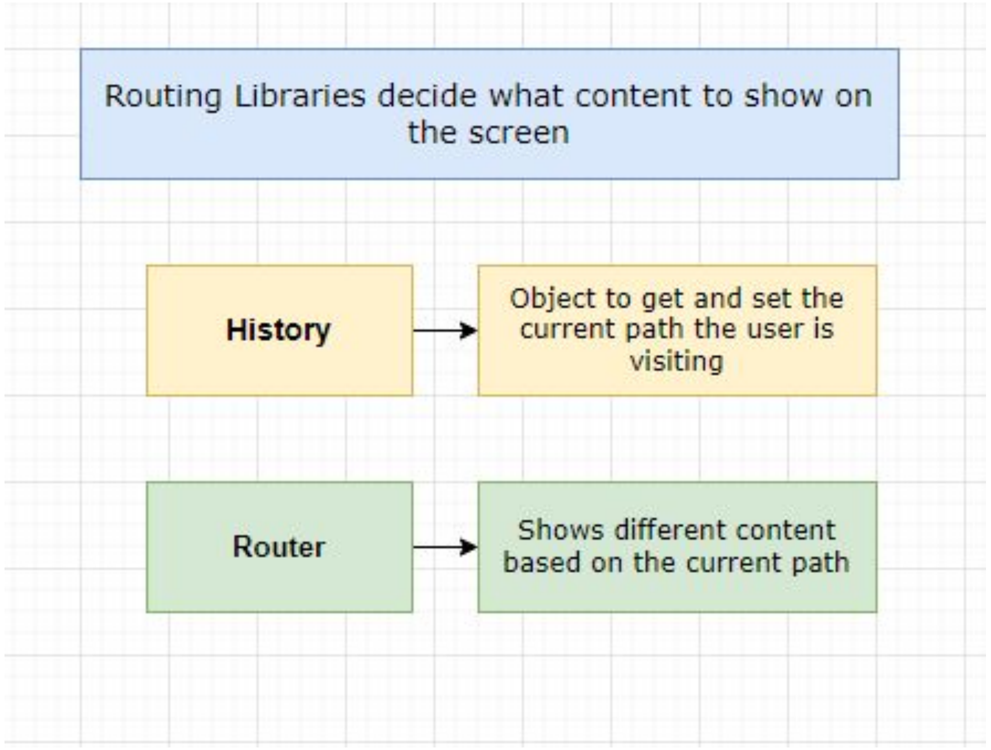
Inflexible Requirement #6

If container & child apps need to communicate information about routing, it should be done in as a generic fashion as possible

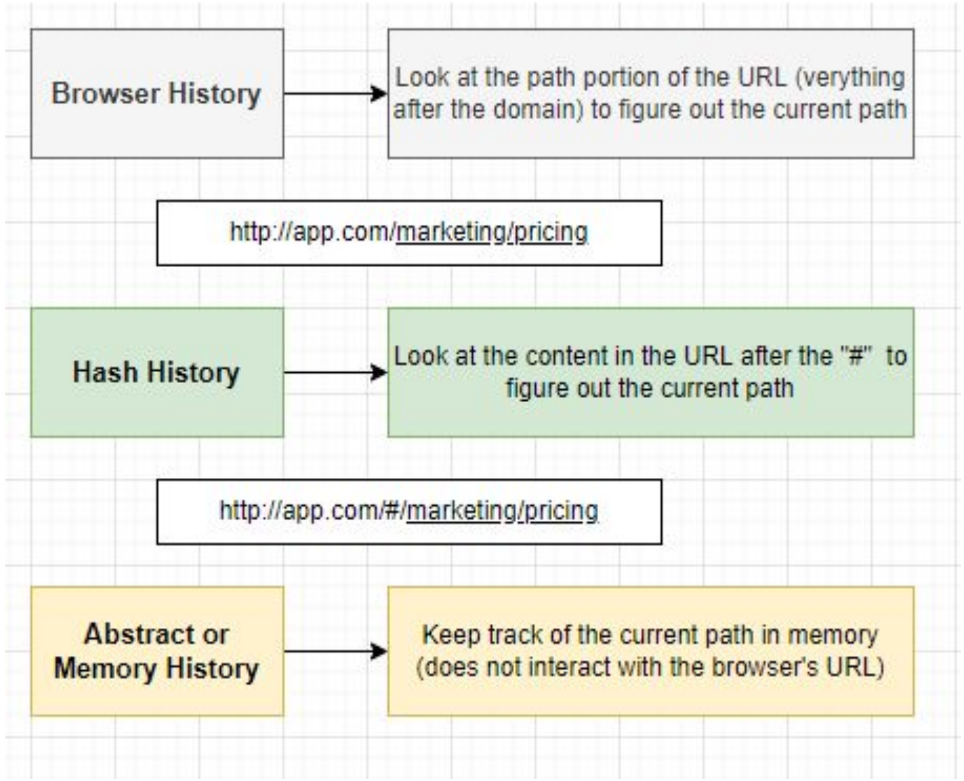
Each app might be using a completely different navigation framework

We might swap out or upgrade the navigation libraries all the time - it shouldn´'t require a rewrite of the rest of the app

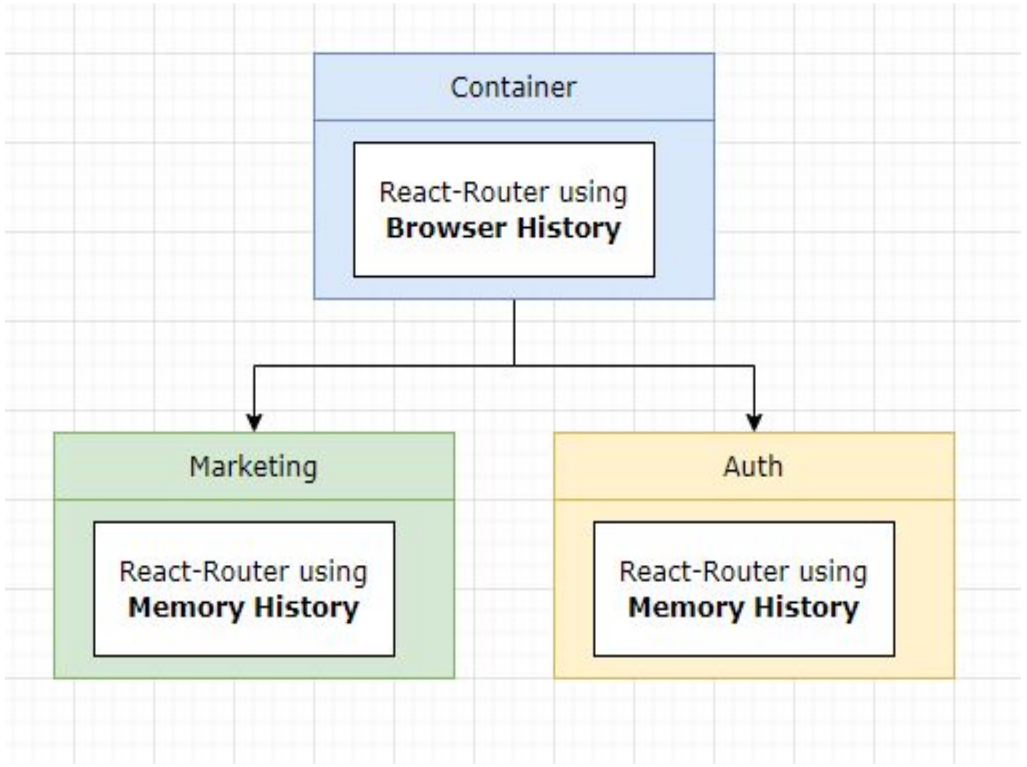
Routing Libraries - Essential Components


Routing Libraries decide what content to show on the screen
History
Object to get and set the current path the user is visiting
Router
Shows different content based on the current path

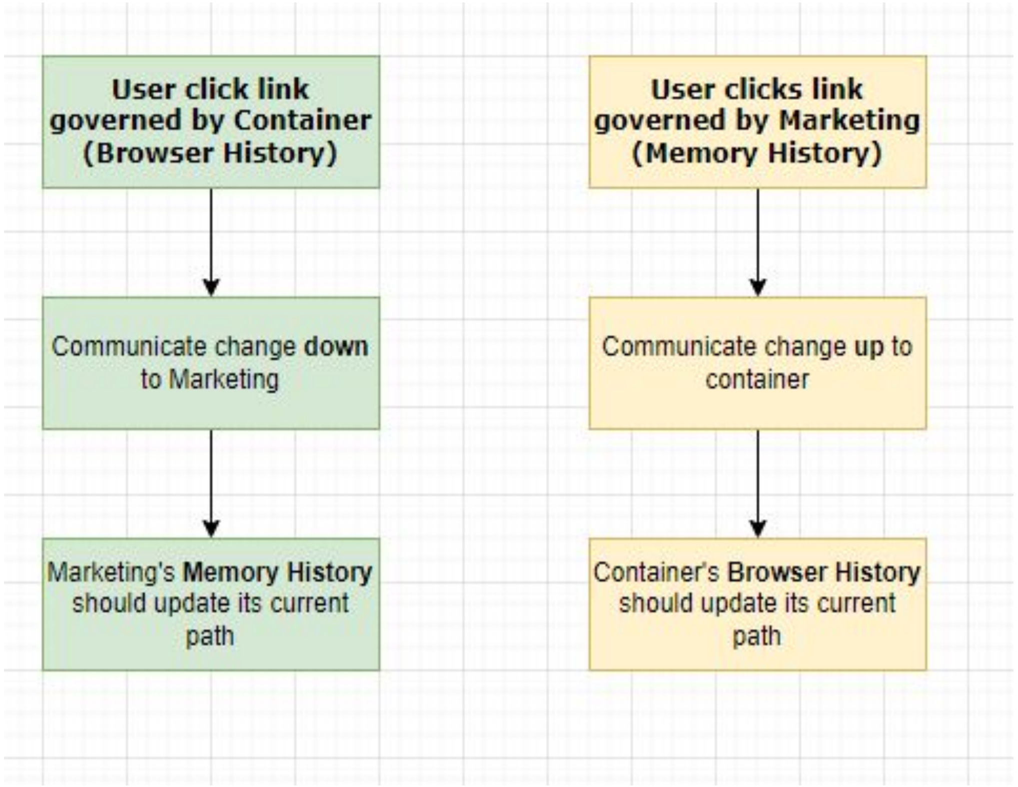
Routing - Types of Libraries


Browser History
Look at the path portion of the URL (verything after the domain) to figure out the current path
http://app.com/marketing/pricing
Hash History
Look at the content in the URL after the "#" to figure out the current path
http://app.com/#/marketing/pricing
Abstract or Memory History
Keep track of the current path in memory (does not interact with the browser's URL)

Routing Libraries - Container and Remotes


Container
React-Router using **Browser History**
Marketing
React-Router using **Memory History**
Auth
React-Router using **Memory History**

Routing Communication - Container and Remotes


User click link governed by Container (Browser History)
Communicate change **down** to Marketing
Marketing's **Memory History** should update its current path
User clicks link governed by Marketing (Memory History)
Communicate change **up** to container
Container's **Browser History** should update its current path