**Converting React app to AEM SPA editable**

1. In the terminal, navigate to react-app directory and execute “npm install --save-dev webpack webpack-cli” to install Webpack
2. Create a file named webpack.config.js and paste the following

  const path = require("path");

module.exports = {

entry: "./src",

output: {

  filename: "App.js",

  path: path.resolve(\_\_dirname, "dist")

},

module: {

  rules: [

    {

      test: /\.js$/,

      exclude: /(node\_modules)/,

      use: {

        loader: "babel-loader",

        options: {

          presets: ["@babel/preset-react"]

        }

      },

      enforce: "post"

    },

    {

      test: /\.css$/,

      loader: "style-loader!css-loader"

    }

  ]

}

};

1. Add .babelrc file in project root and paste the following configuration

{

"presets": [

  [

    "@babel/preset-react",

    {

      "pragma": "dom", // default pragma is React.createElement

      "pragmaFrag": "DomFrag", // default is React.Fragment

      "throwIfNamespace": false // defaults to true

    }

  ]

]

}

1. Open package.json and add the following under scripts

"develop": "webpack --mode development"

And add the following DevDependencies section

"devDependencies": {

  "@babel/preset-react": "^7.0.0",

 "babel-loader": "^8.0.5",

   "webpack": "^4.28.3",

  "webpack-cli": "^3.3.0"

}

1. Run “npm install” and “npm run develop” to build project.

Note: If you encounter any issues, try deleting packaga-lock.json and node\_modules and try again

1. You should see a folder named dist and App.js under it (if you don’t have any styles)
2. Take the index.html from public folder of your react-app and place it in your web server’s doc root folder. Update the file to include <script src="App.js" type="text/JavaScript"></script> before body tag close
3. You should see your React application work fine in browser without dev server

**Big Note:** Please be wary that the web pack versioned used, babel/other babel related modules version used and the webpack.config.js configurations and .babelrc configuration are inter related. If anything changes like right versions with wrong config or right config with different versions might lead to build errors and worst successful build with blank white page

It’s time to create an AEM Archetype project in Eclipse or Terminal so that we will have an AEM project to onboard the react-app. For this demo I am using archetype 16 as I had some issues with 18 and 17 is very latest too.

I am using the command to create the archetype project

mvn org.apache.maven.plugins:maven-archetype-plugin:2.4:generate -DarchetypeGroupId=com.adobe.granite.archetypes -DarchetypeArtifactId=aem-project-archetype -DarchetypeVersion=16 -DarchetypeCatalog=https://repo.adobe.com/nexus/content/groups/public/

Enter Group ID, apps folder name and other properties when prompted. Following is what I provided

groupId: com.techmystyle.spa

groupId: com.techmystyle.spa

version: 1.0-SNAPSHOT

package: com.techmystyle.spa

appsFolderName: react-demo

artifactId: com.techmystyle.spa.react-demo

artifactName: react-demo

componentGroupName: react-demo

confFolderName: react-demo

contentFolderName: react-demo

cssId: react-demo

packageGroup: react-demo

siteName: react-demo

change the directory to com.techmystyle.spa.react-demo and build the project

mvn clean install -PautoInstallPackage -Padobe-public -Daem.port=6402

Note: -Daem.port is optional if you are running AEM on default port 4502

Open page <http://localhost:6402/editor.html/content/react-demo/en.html> to ensure everything looks good

Once you are able to build and verify everything works from AEM side, copy the whole react-app folder into newly created maven project. Before you add pom.xml and make this new one a sub module, install and integrate AEM client lib generator

1. In terminal cd into react-app folder and execute “npm install --save-dev aem-clientlib-generator”
2. Create a file named clientlib.config.js and paste the following

module.exports = {

// default working directory (can be changed per 'cwd' in every asset option)

context: \_\_dirname,

// path to the clientlib root folder (output)

clientLibRoot:

  "./../ui.apps/src/main/content/jcr\_root/apps/react-demo/clientlibs",

libs: {

  name: "clientlib-react-demo",

  allowProxy: true,

  categories: ["clientlib-react-demo"],

  jsProcessor: ["min:gcc"],

  serializationFormat: "xml",

  assets: {

    js: ["dist/\*\*/\*.js"],

    css: ["dist/\*\*/\*.css"]

  }

}

};

3. Update package.json and append “ && clientlib” to dev build script

4. Now cd into com.techmystyle.spa.react-demo/react-app and execute npm run develop from terminal and you can see a new clientlib folder is added under mentioned clientlibRoot as per config

Now we are ready to create a page template and add this clientlib so that we can start seeing React application running in AEM

1. Copy react page component from aem-sample-we-retail-journal-we-retail-journal-1.2.0/ui.apps/jcr\_root/apps/we-retail-journal/react/components/structure/page into your project folder components/structure/react/ (create folder react)

This will serve as our page rendering component. All this component will do is having <div id=”root”></div> in body.html and have headlibs and footlibs pointing to our built files out of react project

**Note:** Make the following edits

1. Open body.html and change the id value to “root” from “page”
2. Open customfooterlibs.html and change the clientlib category to clientlib-react-demo
3. Open customheaderlibs.html and change the clientlib category to clientlib-react-demo and hardcode meta property cq:pagemodel\_root\_url value to /content/react-demo/en.model.json
4. Open .content.xml and update resourceSuperType to core/wcm/components/page/v2/page
5. Edit the property sling:resourceType on /content/react-demo/en/jcr:content and change it to react-demo/components/structure/react/page (This way we are avoiding creating a new template and creating another page for the demo purpose)

CD into com.techmystyle.spa.react-demo/ui.apps and run mvn clean install -PautoInstallPackage -Padobe-public -Daem.port=6402. This will install the new page component along with clientlibs. If we clear cache and refresh the page <http://localhost:6402/editor.html/content/react-demo/en.html> we should see AEM rendering react app just like what we saw from web server. We don’t have anything editable at this time

It is a good idea to make react-app a sub module of AEM Archetype project and add proper profile/config to run npm build as part of AEM build. Follow the instructions below to do so

1. Open parent pom.xml file and add <module>react-app</module> at the top of <modules> section
2. Add the following plugin in <pluginManagement><plugins> section

<!-- Maven Exec Plugin -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>1.6.0</version>

</plugin>

1. Add the following pom.xml under react-app folder



1. Run mvn clean install -PautoInstallPackage,buildDev -Padobe-public -Daem.port=6402 from project root. The profile “buildDev” runs npm run develop so that react app is built and copied to clientlibs before apps is packages

**Installing AEM JS SDK for React**

From terminal execute the following commands

npm install @adobe/cq-react-editable-components

npm install @adobe/cq-spa-component-mapping

npm install @adobe/cq-spa-page-model-manager

Install the following peer dependencies as well

npm install react-fast-compare

npm install typescript --save-dev

npm install ajv --save-dev

npm install clone --save-dev

Open BusinessOverview component and add the following

import { MapTo} from "@adobe/cq-react-editable-components";

const TextEditConfig = {

emptyLabel: "Text",

isEmpty: function(props) {

return !props || !props.text || props.text.trim().length < 1;

}

};

export default MapTo("react-demo/components/content/text")(

BusinessOverview,

TextEditConfig

);

And add cqPath property pointing to some node of type ‘text’. This is required as we are statically including the components

<BusinessOverview

cqPath={"/content/react-demo/en/jcr:content/root/responsivegrid/text"}

/>

Open index.js and make the following changes to integrate with Model Manager and initialize it

Note: Using MapTo without initializing ModelManager breaks the application.

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import { ModelManager, ModelClient } from "@adobe/cq-spa-page-model-manager";

import "bootstrap/dist/css/bootstrap.min.css";

function render() {

ReactDOM.render(<App />, document.getElementById("root"));

}

document.addEventListener("DOMContentLoaded", () => {

ModelManager.initialize().then(model => {

render();

});

});

Once you build after these changes, you should see an editable Text component decoration around our BusinessOverview react component(still with hardcoded value). When opened configure dialog, you should see the content from the **cqPath** location.

Now head over to BusinessOverview component and replace the hard coded jsx markup to this one

<div

data-rte-editelement

dangerouslySetInnerHTML={{ \_\_html: this.props.text }}

/>

(replace the whole return markup in render method with the above line)

Every time a component is edited, AEM automatically passed the component properties as properties to React component and triggers render. So we will see updates reflect without refreshing the page

To show the text on initial page load, we need to read it manually from ModelManager and pass as prop. To do so, update the App.js to include the following prop on BusinessOverview component

text={

ModelManager.modelStore.\_findItemData("root/responsivegrid/text").data.text

}

With this step you should see the text loading on page load and updating on component update