ska-react-webapp-skeleton Documentation

Release 0.2.2

SKAO, (Trevor A Swain)

Mar 26, 2024

CONTENTS:

This repository is intended to act as a skeleton for any SKA developer looking to make a React based web application.

It includes tools for linting, code formatting, and testing which are easily integrated into various IDEs. It also includes modular federation, exposing the ReactSkeleton component, which can be imported into other applications.

CHAPTER

ONE

OVERVIEW

This project is intended to act as a skeleton for any SKA developer looking to make a React based web application.

It includes tools for linting, code formatting, and testing which are easily integrated into various IDEs. It also includes modular federation, exposing the ReactSkeleton component, which can be imported into other applications.

1.1 IDE Integration

1.2 VS Code

Install the [ESLint](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-ESLint) and [Prettier](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode) extensions and reload the IDE.

Errors should now show in the editor. *shift* + *alt* + *F* will format a file, or you can turn on the format on save setting.

JetBrains (WebStorm, etc.)

ESLint is integrated into the Ultimate versions of all JetBrains IDEs

Prettier can be installed through a [plugin](https://plugins.jetbrains.com/plugin/10456-prettier). Follow the steps [here](https://www.jetbrains.com/help/idea/prettier.html) to configure it.

CHAPTER

TWO

REQUIREMENTS

This skeleton requires **Node** and **YARN** to install and run. To install these follow the instructions for your operating system at

[https://nodejs.org/en/download/{]}(https://nodejs.org/en/download/).

Alternatively the official Node docker image can be used. Instructions can be found on the

[official Node docker image site](https://github.com/nodejs/docker-node/blob/master/README.md#how-to-use-this-image).

CHAPTER

THREE

INSTALLATION

All the following notes assume you are at the command prompt for your chosen environment.

1. Confirm Node and YARN are installed and configured correctly, both the following commands should return the relevant version number.

   > node –version > yarn –version

2. Clone the project from GitHub
3. Allow yarn to be able to include required SKAO libraries

> npm config set @ska-telescope:registry https://artefact.skao.int/repository/npm-internal/

4. Install all the necessary project dependencies by running

> yarn init

5. INstall required SKAO libraries

It is expected that required SKAO libraries would have been included at this point, however if this is found not to be the case, the following command will include them.

> yarn skao:update

Steps to convert to your own application.

Here are the steps required to migrate this application for use within the Portal. These are required so that we can ensure a unique reference into the Portal. For this example we will use the name NewApp as the name of the new application

1. Clone the application into the appropriate folder
2. Follow the installation instructions in the previous paragraph
3. ReactSkeleton.tsx : Change *ReactSkeleton* to *NewApp* in 2 locations
4. ReactSkeleton.tsx : Change filename to `NewApp.tsx
5. ReactSkeleton.cy.tsx : Change *ReactSkeleton* to *NewApp* in 4 locations
6. ReactSkeleton.cy.tsx : Change filename to `NewApp.cy.tsx
7. Change Folder name from ReactSkeleton to NewApp
8. App.tsx : Change *ReactSkeleton* to *NewApp* in 4 locations
9. webPack.config.js : Change path for *ReactSkeleton* so it points to *NewAppNewApp*.

Compilation and running this application will allow it to be shown within the ReactSkeleton menu item within the developer section of the SKA-Portal. Whilst initial development is being done and until the application is allocated a permanent location, it is suggested that no other WebPack changes are done.

Steps to create a new release

Note, this does not currently work in the Windows Shell. Use either Linux, Mac, or Windows WSL.

Also note that the chart names should be updated when you use a different repo name.

The following steps and commands is to create a new release for the portal.

1. Create a new branch from `main` branch.
2. Run one of `make bump-major-release`, `make bump-minor-release`, or `make bump-patch-release`
3. Update the `charts/ska-react-webapp-skeleton/values.yaml` file, the *image.version* should be updated.
4. Make sure the following files have the new version: * `charts/ska-react-webapp-skeleton/Chart.yaml` * `package.json` * `.release`
5. Run `make git-create-tag`
6. Run `make git-push-tag`
7. You will then be able to merge that branch back in, and the new release should be created.

CHAPTER

FOUR

DURING DEVELOPMENT

It is note that the introduction of new libraries may throw an error. This is usually because WebPack requires the library to be included as part of the ModuleFederationPlugin entry within the webpack.config.js It is suggested that the new library be added into the area on the configuration annotated mixture.

4.1 Adjustment once final positioning within the SKA-Portal has been determined

So that there is no clashes with other applications originating from a skeleton, the following steps should be taken. Once these are done the application will no longer be available via the ReactSKeleton menu item within the SKA-Portal

1. webpack.config.js : Change the devServer port number from 8090.
2. webpack.config.js : Change the final ReactSkeleton entry to NewApp.
3. Relate these new values back to the developer responsible for updating the SKA-Portal

Once these steps have been completed, the application should be accessible from it's new location in the SKA-Portal

4.2 SKA Components

These are supplied from the various components. An overview of the libraries have been provided here for reference. For specifics of the components/functions available, please refer to the appropriate repository

ska-javascript-components : library containing a few pure components and types

ska-gui-components : library containing components that have been built using Material UI.

ska-gui-local-storage : library containing redux local storage. This is used by the ska-gui-components components as applicable

It is noted that all the components from ska-javascript are provided in addition as part of the ska-gui-components library, so separate inclusion is not required if the ska-gui-components are included.

4.3 Services

Included are a number of services, which have also been implemented into the code, providing simple examples. It is expected that in the main that there will be no updates to these services directly

Below is a list of the current services, together with their purpose.

i18n : Allows for text to be displayed in the language of the browser, with English as the default

theme : Contains the initialization of the latest SKAO Theme

CHAPTER

FIVE

RUNNING AND BUILDING THE APPLICATION

Scripts for running, testing, and building the application are provided as part of the standard configuration. These are run using YARN and listed in the scripts section of the package.json file.

From the project directory, you can run any of the following:

- > *yarn dev*

  Runs the app in the development mode at [http://localhost:8090{}](http://localhost:8090). The app will recompile and restart if you make any edits to the source files. Any linting errors will also be shown in the console.

- > *yarn skao:update*

  yarn will update the repository with the latest SKAO libraries

- > *yarn start*

  Same as *yarn dev* but for some implementations it is prefixed with NODE_ENV=testing. This is used in the CI/CD Processes

- > *yarn cypress*

  Launches Cypress which has been set up to provide component testing. For further information on the use of Cypress, see https://docs.cypress.io/guides/component-testing/overview

- > *yarn test*

  Launches the test runner in the interactive watch mode. See the [testing](#testing) section for more information.

- > *yarn build*

  Builds the app for production to the *build* folder. The build is minified and any JSX is transpiled to JavaScript. Your app is ready to be deployed!

- > *yarn audit*

  Checks the dependencies to see if there are any vulnerabilities.

Running the application inside a container

There are two ways that this can be done as below:

1. Run using docker compose

```
  docker-compose up -d
```

2. build the docker file in the root directory and run the container exposing port 8090.

```
  docker build -t ska-react-webapp-skeleton . docker run -p 8090:8090 ska-react-webapp-skeleton
```

```

The project will then be accessible at the url http://localhost:8090/
```

CHAPTER

SIX

TESTING

6.1 Writing

We use Cypress as the test running framework. It will look for test files within a number of locations, however the standard that the SKAO will employ will be the use of *.cy.{tsx | jsx}* in the same folder as the component being tested.

``` components └─ App

App.cy.tsx
App.tsx

└─ **ReactSkeleton**

ReactSkeleton.cy.tsx
ReactSkeleton.tsx

```

Note that the ReactSkeleton component is exposed via WebPack 5 ModuleFederationPlugin, so this name should be changed to reflect the application being written (e.g. SignalDisplay, DataProductDashboard . . .)

See the developer guide for more information

6.2 Running

To run the interactive test runner, execute

> yarn test

This will also watch the source files and re-run when any changes are detected

To run the tests with coverage, execute

> yarn test:coverage

The coverage results are displayed in the console. They are also written to the *coverage* folder.

- *./build/coverage/index.html* - open in a web browser to view

All the tests should pass before merging the code

6.3 Code Analysis

[ESLint](https://ESLint.org/) and [Prettier](https://prettier.io/) are included as code analysis and formatting tools. These do not need installing as they're included in *node_modules* by running *yarn init*.

These tools can be run in the command line or integrated into your IDE (recommended).

JavaScript based SKA projects must comply with the [AirBnB JavaScript Style Guide](https://github.com/airbnb/javascript). These rules are included in this project and ESLint and Prettier are configured to use them.

Running

To run the analysis tools, execute

```
> yarn code-analysis
```

This will display any errors in the command line. If there are any errors, YARN will exit with a non-zero code, the *-s* argument suppresses this and cleans up the output.