**Customising Cognito SignIn UI for aws-amplify-react with TypeScript**

Let’s assume you are having trouble with customising Cognito Sign In UI for aws-amplify-react with TypeScript and have already know what AWS Cognito, aws-amplify and aws-amplify-react.

Customising Cogito Signin UI is easy with JavaScript React. All you need to do is to extend the SignIn class in aws-amplify-react with a new showComponent method returning your customised UI. [Here](https://aws-amplify.github.io/docs/js/authentication) is the official documentation on Cognito Authentication for React. There is an excellent blog post that explains how to extend SignIn class [here](https://blog.kylegalbraith.com/2018/11/29/how-to-easily-customize-the-aws-amplify-authentication-ui/).

If you are into plain JavaScript without any framework, check out this post, [Serverless Authentication with AWS Cognito and JavaScript](https://www.mydatahack.com/serverless-authentication-with-aws-cognito-and-javascript/).

Using aws-amplify-react with TypeScript requires a little bit more effort because the library does not offer types and you have to provide your own for customisation. It is surprising that aws-amplifiy is written in TypeScript, yet aws-amplify-react is only written in JavaScript and has no support for TypeScript (although this is likely to be changed in the future considering the popularity and benefits of developing application with TypeScript).

When you first use aws-amplify-react with TypeScript, you need to declare module to get it work as below (the file can be called as aws-amplify-react.d.ts in the types folder).

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

declare module 'aws-amplify-react';

</pre>

To customise anything in that library, you need to provide more detailed type. Here is the example for customising SignIn page. Basically, any type error you get, you add that class, variable or function in this file.

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

declare module 'aws-amplify-react' {

export declare class SignIn extends React.Component {

\_validAuthStates: string[];

handleInputChange: React.EventHandler;

signIn(): void;

}

interface IAuthenticatorProps {

hideDefault: boolean;

amplifyConfig: any;

}

export declare class Authenticator extends React.Component<IAuthenticatorProps> {

}

}

</pre>

Now, you can have your custom sign page by extending SignIn class.

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

import \* as React from 'react';

import { SignIn } from "aws-amplify-react";

class CustomSignIn extends SignIn {

constructor(props: any) {

super(props);

this.\_validAuthStates = ["signIn", "signedOut", "signedUp"];

}

showComponent(theme: any) {

return (

<form onSubmit={this.signIn}>

<label htmlFor="username">Username</label>

<input

onChange={this.handleInputChange}

type="text"

id="username"

key="username"

name="username"

placeholder="Username"

/>

<p></p>

<label htmlFor="password">Password</label>

<input

onChange={this.handleInputChange}

type="text"

id="password"

key="password"

name="password"

placeholder="\*\*\*\*\*\*\*\*\*\*"

/>

<p></p>

<button type="submit">Sign In Check</button>

</form>

)

}

}

export default CustomSignIn;

</pre>

The authState is passed down to the child component. You can return your UI when authState is ‘signedIn’.

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

import \* as React from 'react';

import SignOut from './SignOut';

interface AppProps {

  authState?: any;

}

class App extends React.Component<AppProps> {

  constructor(props: AppProps, context: any) {

    super(props, context);

  }

  render() {

    if (this.props.authState == 'signedIn') {

      return (

        <React.Fragment>

          <h1>Hello World</h1>

          <SignOut />

        </React.Fragment>

      );

    } else {

      return null;

    }

  }

}

export default App;

</pre>

Now wrap this in Authenticator with the CustomSignIn component we created. The authState will be available in the App component.

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

import \* as React from "react";

import config from "./aws-exports";

import CustomSignIn from "./CustomSignIn";

import App from "./App";

import { Authenticator } from "aws-amplify-react";

class AppWithAuth extends React.Component {

render() {

return (

<React.Fragment>

<Authenticator hideDefault={true} amplifyConfig={config}>

<CustomSignIn />

<App />

</Authenticator>

</React.Fragment>

)

}

}

export default AppWithAuth;

</pre>

Then, use AppWithAuth component in the index.tsx.

<pre class="theme:vs2012-black font:verdana font-size:13px toolbar:2 lang:js decode:true ">

import \* as React from 'react';

import \* as ReactDOM from 'react-dom';

import '../public/css/bootstrap.css';

import '../public/css/style.css';

import AppWithAuth from './AppWithAuth';

import { Auth } from 'aws-amplify';

import awsmobile from './aws-exports';

Auth.configure(awsmobile);

ReactDOM.render(

<AppWithAuth />,

document.getElementById('root')

);

</pre>

That’s it!

By the way, if you are having an error below, all you need to do is set **NODE\_ENV=production**. It will stop doing the unnecessary package checking and error will disappear.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

bundle.js:1442 Uncaught Error: Cannot use e "\_\_Schema" from another module or realm.

Ensure that there is only one instance of "graphql" in the node\_modules

directory. If different versions of "graphql" are the dependencies of other

relied on modules, use "resolutions" to ensure only one version is installed.

https://yarnpkg.com/en/docs/selective-version-resolutions

Duplicate "graphql" modules cannot be used at the same time since different

versions may have different capabilities and behavior. The data from one

version used in the function from another could produce confusing and

spurious results.

[/cc]