Migrating from Allium package (ds-allium)

Update an existing application using @telus-uds/ds-allium to use the new @telus-uds/components-web. The Allium package `ds-allium` containing the single-brand TELUS theme components are set for deprecation in preference for a more efficient approach that works for all users and offers the versatility to create both single-brand and multi-brand web applications.

System requirements

Before installing, be sure to check the system requirements.

Compatibility

Allium package components are refactored to enable multi-brand functionality through tokenization. It's our intention to limit significant breaking changes. However, there are minor backward compatible changes in variant offerings and property deprecations that could impact your application's appearance and behavior. Thorough testing is recommended.

Migration process

1. Installation
2. BaseProvider configuration
3. Update component imports

1. Installation

1. Remove the existing `@telus-uds/ds-allium` package from your project,

```
npm uninstall @telus-uds/ds-allium
```

2. Install the new `@telus-uds/components-web` package,

```
npm install @telus-uds/components-web
```

3. To migrate to `@telus-uds/components-web`, you need to manually install the desired brand theme. The "allium" theme is no longer automatically included.

4. Install the theme package corresponding to the desired brand (e.g., "allium").

   ```
   npm install @telus-uds/theme-allium
   ```

2. BaseProvider configuration

1. In the application's entry point file (e.g., index.js), remove `AlliumProvider` if it exists and wrap the application with `BaseProvider` from `@telus-uds/components-web`.

2. Import the theme and pass it to the BaseProvider.

   ```
   import React from 'react'
   import { render } from 'react-dom'
   import alliumTheme from '@telus-uds/theme-allium'
   import { BaseProvider } from '@telus-uds/components-web'
   import App from './App'

   render(
     <BaseProvider defaultTheme={alliumTheme}>
       <App />
     </BaseProvider>,
     document.getElementById('root')
   )
   ```

3. Update component imports

Update all component imports from the old package to the new package.

```
// Old import
import { Button } from '@telus-uds/ds-allium'

// New import
import { Button } from '@telus-uds/components-web'
```

Troubleshoot migration from Isomorphic Starter Kit

⚠ CAUTION

If your app is based off of Isomorphic Starter Kit, you may encounter the following errors when you run the app after updating component imports.

Module parse failed: Unexpected token

```
ERROR in ./node_modules/@telus-uds/components-base/lib-module/Typography/Typography.js 84:48
Module parse failed: Unexpected token (84:48)
You may need an appropriate loader to handle this file type, currently no loaders are configured to process this file. See https://webpack.js.org/concepts#loaders
|
|       const childStyles = (child === null || child === void 0 ? void 0 : (_child$props = child.props) === null || _child$props === void 0 ? void 0 : _child$props.style) || {};
>       const supFontSize = childStyles.fontSize ?? superScriptFontSize;
|       const sanitizedChild = /*#__PURE__*/React.cloneElement(child, {
|         style: { ...childStyles,
```

```
ERROR: "webpack:client" exited with 2.
```

To resolve this issue, you need to update the webpack configuration files: `webpack.dev.client.config.js` and `webpack.prod.client.config.js`. Add an additional path to parse with babel-loader.

Example:

```
module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        use: ['babel-loader'],
        // include paths for babel-loader. multiple paths need to be in array
        include: [
          path.join(__dirname, 'src'),
          path.join(__dirname, 'node_modules/@telus-uds')
        ],
      }
```

Multiple paths can be included when they are in an array. The `@telus-uds` path needs babel-loader to parse optional chaining and nullish coalescing.

App runs, but app layout is incorrect

Error encountered on DevTools console:

```
viewports.js:63 Uncaught TypeError: Cannot assign to read only
property 'exports' of object '#<Object>'
    at Module.<anonymous> (viewports.js:63:1)
```

To fix this issue, update the babel.config.js file to add an `overrides` property with a test condition and plugin.

Example:

```
  const config = {
    presets: [
      [
        '@babel/preset-env',
        {
          modules: false,
          useBuiltIns: 'usage'
        }
      ],
      '@babel/preset-react'
    ],
    // overrides property with test condition and plugin
    overrides: [
      {
        test: './node_modules/@telus-uds/**/*.js',
        plugins: ['@babel/plugin-transform-modules-commonjs']
      }
    ],
```

The `@babel/plugin-transform-modules-commonjs` plugin does not need further installation since it should already be part of `@babel/preset-env`.

Testing and validation

Thoroughly test your application to ensure that the migrated components function as expected.

Support and questions

If you have any questions or concerns about the migration, you can always reach out to the UDS team on Slack.

Edit this page