Develop a community component

Universal Design System encourages the community to contribute features, bug fixes and components. This guide covers everything you need to know to begin working in the UDS monorepo. Teams can choose to contribute components regardless of brand or platform they are speicific for. We welcome single-brand or multi-brand, web specific or multi-platform components. We envision that all components that are contributed by the community will begin their life in `@telus-uds/components-community`.

`@telus-uds/components-base` is the base multi-platform base component library that you could extend your components from. If your component is a web only react component you could extend from `@telus-uds/components-web` to benefit from the wider range of feature options. These components are:

- highly reusable for teams across the organization
- should have a relatively stable API
- actively maintained and supported by the UDS team

> INFO
>
> - If this is the first time you are contributing to the Universal Design System repository
>   - You will need Github access
>   - You may need to request write access for you and your team
> - If you are contributing to the UDS project, please follow the instructions to develop for UDS

Scaffold your component

We provide easy scripts to generate the basic scaffolding of a component. From the root directory of the monorepo, run the following script and you'll be prompted with a series of questions to best determine the brand and platform the component needs to work on.

> DANGER

Note that when creating new components you should use the UDS theming system to set rules for appearances and styles. **The Core themes (TELUS/Koodo/Public Mobile) will not receive any new style related to community components**. Please use the community theme in `packages/themes-community` for the respective brand to theme your component.

```
npm run create-component
```

The script will:

- Add a component folder with a component `.jsx` file and an index file to export it
- Add a test folder and file with a sample test
- Add a Storybook file with a default story of your component
- Add component routes to the appropriate storybook `.storybook/main.js` or/and `storybook/.storybook/main.js` file based on the platform your component is intendend for.
- Add a component documentation `.mdx` file
- Add component to the community section of the respective brand sidebar file in `docusaurus-plugin-component-docs-pages`.
- Update the main package index to export your component

You can check out the scaffolded component and start development by running `Storybook`. Make sure to select the platform and brand you'll be working on during the prompts.

INFO

`npm run create-component` script questions

1. Is it a community or a core component? Select **`Community`** to proceed with your component creation.
2. What platform are you developing for? You can create either a **`web`** (works only for web) or a **`multi-platform`** (can be used in react native) component.
3. What brand (theme) are you building the component for? Select a **`brand`** or **`multi-brand`**, pay attention that by selecting a *single*

brand you will only need to style the component for that brand, while selecting *multi-brand* you will need to define the styles for all brands.

4. Name the component. Follow the standards of the system by choosing a generic name that represents the component and use *Pascal Case* (e.g. *ExpandCollapse*).

```
npm run storybook
```

INFO

`npm run storybook` script questions

1. Which platform are you working on? `Storybook` will render all components compatible with the selected platform: **`Web`**, **`iOS`** or **`Android`**
2. Which brand are you working on? Defines if storybook will provide a theme switcher or scope the components to a single-brand.

To see the component on the documentation site you can run the below script and pick the right platform and theme.

```
npm run docs
```

INFO

Scaffolded component on `Storybook` web & mobile and documention site can be seen under the `Community section` to the bottom of the sidebar menu. For multi-brand components storybook will show an option to switch themes.

Build your component

It's now time to start coding! You can build your component as you would any `react` or `react-native` component.

Edit your component `.jsx` file as you want and add or edit your dependencies in the `package.json` file based on the theme palettes and base component libraries you wish to use.

Component composition

As much as possible your component should be composed of the existing UDS base and web components(TELUS branded).

Component standards

- Follow the component code checklist to ensure that your component meets all the criteria

Style your component

To style the component for the respective brands you can levarage the brand community themes and hooks exported by `@telus-uds/components-base`. In the UDS mono-repo there are three brand community packages for all core brands (in themes-community/) that you could use to achieve the desired default styles and configure appearances based on rules. Theming in UDS is as straightforward as creating a new entry for the component in the `theme.json` for the desired brands, and adding tokens, apprearances and rules. Below is a basic structure of a theme for a component in UDS.

```
"ComponentName": {
  "appearances": {},
  "rules": [],
  "tokens": {}
}
```

Learn more about themes in UDS. Drop a message in the #ds-support Slack channel or sign up for office hours if you need help theming a community component.

Test your component

- Write extensive test coverage using `@testing-library/react` or `@testing-library/react-native` for all behaviour and variants of your component
- Add stories that showcase the different variants of your component
- It is up for the contributing team to decide what tests should be added to the component. We only require that there are tests to at least one react version (17/18).

Document your component

- Use the scaffolded template as a starting point to add complete documentation of your component
- Edit your component `mdx` file in the `docs` folder. You may need to edit multiple `mdx` files in their respective platforms (multi-platform/web/ios/android)
- Temporarily for multi-platform components you'll need to update the component mdx file created in `docusaurus-plugin-component-docs-pages` at `docs`.

Edit this page