Component quality checklist

This checklist should be consulted and completed before a component is accepted into the UDS repo. We also recommend that it be used as a reference for quality criteria for teams building their own components.

Design

Before building any component, please refer to the design checklist to ensure that as many considerations as possible have been covered at the design phase. Solving problems at the design phase will make component builds smoother and faster, and result in higher quality components.

Principles

1. **Composable**: Components are most effective when they can be combined to form more complex patterns
2. **Accessible**: Components should be designed and built for people of all abilities
3. **Responsive**: Mobile-first, works on any viewport size
4. **Quality**: Thorough testing is a first class concern

Implementation

- Functionally complete: we strive to keep base components stable
- Stable API: once accepted, breaking API changes must be avoided
- JSDoc component properties in their proptype definitions
- All design tokens should be passed via the `tokens` property
- Design tokens must be added to the theme schema
- Must not include business logic or proprietary information. Keep business logic, API calls, content, or other application-specific behaviour in the application. Focus on reusable functionality that can accept predictable data types.

Themes

- TELUS component token and variants complete
- Koodo component token and variants complete
- Public Mobile component token and variants complete

Accessibility

- Appropriate accessibility props are made available so engineers can implement the recommendations given in the TELUS Accessibility Guides and the TELUS Accessibility Matrix.

The following accessibility tasks should be considered:

Content structure

- Smart Headings recongize placement in page (if possible)
- Keyboard navigation (Default: Top-Down, then Left to Right)

Magnification

- Resized text
- Screen zoom
- Screen magnifier

Colour treatment e.g. contrast, colour blindness and smart invert

- Do not use colour alone to convey meaning
- Ensure there is enough contrast/luminosity between text and background
- Ensure there is enough contrast/luminosity between functional elements and background

Contrast Themese

- Ensure that content is visible in high-contrast themes
- Follow best practices for SVGs

Animation (Reduced motion)

- No autoplay (less than 3 seconds is ok)
- No looped animation
- Animation which conveys information must convey the same meaning without animations
- Respect Reduced motion settings (provide no animation or smooth animation alternative)

Test with assitive technologies

- Screen readers VoiceOver, NVDA or JAWS
- Switch Controls (alternate keyboards)

Testing

- Code passes repository linting and formatting checks
- Component UI tests using `React Testing Library` with comprehensive feature/branch coverage
- Strive for coverage complete, testing for as many cases as reasonably possible
- Add design tokens the to the test theme
- Component functionality, states, and responsive behavior manually verified across viewports on Web platform
- Component functionality, states, and responsive behavior manually verified across viewports on Android platform
- Component functionality, states, and responsive behavior manually verified across viewports on iOS platform

Where possible, components should be manually tested on physical devices (Web, Android, iOS) and using assistive technology as per the Accessibility responsibilities for testers

Documentation

Shared UDS documentation

- Component API
- Basic usage guidelines
- Limitations (platform-specific or otherwise)
- Security implications (if any)
- Usability and A11y guidance

Brand specific documentation

- Notify brand teams about new component

Edit this page