TELUS Component Library

TextArea

Multi-Platform Component | [Figma UI KIT](#)

Label ❓

A usually much longer hint below

```
<TextArea
  label="Label"
  hint="A usually much longer hint below"
  hintPosition="below"
  tooltip="Tooltip content"
/>
```

Introduction

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

Use the `TextArea` component to collect long-form information such as product feedback or support queries. The `TextArea` height will automatically increase when new lines are added - this behavior can be limited within the theme.

[Follow the appropriate instructions](#) to add this component in to your app.

Guidance

Controlled component

If it is required that the state of the `TextArea` be controlled by the application or other external methods, `value` and `onChange` props must be passed to the `TextArea`.

Controlled

Will show an error if longer than 6 characters and success otherwise

Expand editable code

If the `TextArea` value should not be changed by user input, a `readOnly` prop must be provided.

Read only

won't change

```
<TextArea label="Read only" readOnly value="won't change" />
```

Uncontrolled component

If it is not necessary to control the `TextArea` state, you can create one without a value prop. In this case its value can be accessed within the

`onChange` callback.
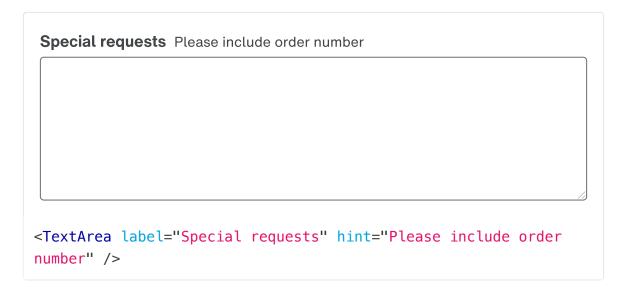
Getting feedback for entered values

Use the `feedback` prop to give the user feedback regarding their input (keep it as brief as possible). You can affirm that the user's input was correct, or highlight errors that must be corrected by setting the `validation` prop.

Order instructions

Leave at the main door ✅

```
<TextArea label="Order instructions" initialValue="Leave at the
main door" validation="success" />
```

Order instructions

❗

This field must be filled.

```
<TextArea label="Order instructions" validation="error"
feedback="This field must be filled." />
```

This component does not perform input validations, as that is an application level concern. You will need to track the value of your form fields and perform any required data validations either client side or server side, depending on the context.

Supplying a hint

Use a `hint` to succinctly clarify attributes of the expected input data, such as the expected format, or an indicator that the field is optional. It is a more usable and accessible option than the HTML `placeholder` attribute.

Special requests Please include order number

```
<TextArea label="Special requests" hint="Please include order
number" />
```

- `hint` can be used beside the label (inline) or below
  - Label and inline hint text should not exceed 40 characters total, otherwise, use hint text below the label
- The `hint` should **not** be longer than 85 characters
- The `hint` should **not** wrap more than 3 lines on all viewport sizes
- Developers can use `hintPosition` to re-position the `hint` below the `label` if the text supplied to the inline `hint` wraps

Using a tooltip

When a more in-depth explanation is needed to fill out a `TextArea` properly, additional instructions may be provided to the `tooltip` prop.

Describe your inquiry Please include customer ID ❓

```
<TextArea
  label="Describe your inquiry"
  hint="Please include customer ID"
  tooltip="Inquiries will be addressed by our staff ASAP. We
will reach out if the query cannot be fulfilled. Please check
your confirmation email for your customer ID."
/>
```

Assigning an ID

The `TextArea` component provides the capability to specify a custom identifier through the use of the `nativeID` property.

Label

```
<TextArea
  label="Label"
  nativeID="My custom id goes here"
  onClear={() => console.log('Back to square one')}
/>
```

> ⊘ **INFO**
>
> In order to support more advanced usages all additional props will be passed down to the React Native's `TextArea` component. See React Native documentation for supported props and React Native Web documentation for their implementation on the web.

Accessibility

`TextArea` is accessible to screen readers on both native platforms and Web. It will be announced with the `label` prop if it is provided, and optionally

described with the `hint` , `tooltip` and `feedback` .

Platform considerations

The component is available on both native platforms and web.

> **(!) INFO**
>
> Due to React Native's implementation of `TextInput` it's not possible to access the current value of the input by passing a ref. You can only access it within the `onChange` callback.

Props

Name	Type	Platform	Default	Description
value	string	standard		If the input's state is to be controlled by a parent component, use this prop together with the `onChange` to pass down and update the lifted state.
initialValue	string	standard		Use this to set the initial value of an uncontrolled input. Updating `initialValue` will **not** update the actual value.
inactive	bool	standard		Disables all user interactions with the input.
readOnly	bool	standard		Makes it impossible to change the input's

Name	Type	Platform	Default	Description
				value.
onChange	func	standard		Use to react upon input's value changes. Required when the `value` prop is set. Will receive the input's value as an argument.
tokens	tokens	standard		System tokens prop, see tokens for more details
variant	variant	standard		System variant prop, see variants for more details

Tokens

In exceptional circumstances, the following tokens can be passed to this component to override its default styles. **Do not do this unless absolutely necessary.** Read more about overriding styles.

▶ View Tokens

Variants

TextArea has no visual variants, but its appearance may change based on the validation and inactive props.

Last name

Potter

✓

```
<TextArea label="Last name" validation="success"
initialValue="Potter" />
```

Last name

!

```
<TextArea label="Last name" validation="error" />
```

Inactive

can't touch this

```
<TextArea label="Inactive" inactive initialValue="can't touch
this" />
```

Feedback

- Spotted a problem with this component? Raise an <u>issue on GitHub</u>
- See any <u>existing issues</u> for this component
- Contact the team on slack in <u>#ds-support</u>