TELUS Component Library

Countdown

Web Component | Figma UI KIT

```
 9 : 23 : 59 : 58

function DefaultCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return <Countdown targetTime={targetTime} />
}
```

Introduction

The purpose of the `Countdown` component is to provide a dynamic display of time remaining to a specific moment in the future.

Follow the appropriate instructions to add this component in to your app.

Guidance

- Use `Countdown` to display time remaining to a specific moment
- Default variant of the `Countdown` will inherit text style of parent container
- Use `large` or `feature` variants to render a standalone component for visual emphasis
- `Countdown` is only responsive for XS breakpoint (keep in mind, however, that you may still need to adjust its props to make sure it does not exceed the smallest expected width)

> ⚠ CAUTION
>
> Note that the component will throw an exception if the target time provided is invalid. If the time is in the past, the all-zero countdown will be displayed.

Accessibility

Currently, `Countdown` component is focusable for a11y purposes, it also allows passing standard accessibility attributes via props.

Platform considerations

This component is currently only supported on web.

Props

Name	Type	Platform	Default	Description
copy	union	web only	'en'	Copy language identifier (`'en'` or `'fr'`) or a dictionary instance (an object with the following keys: days, day, hours, hour, minutes, minute, seconds, second)
variant	variant	web only	{}	System variant prop, see variants for more details
tokens	tokens	web only		System tokens prop, see tokens for more details
targetTime	instanceOf	web only		An instance of JavaScript `Date` object or a string parseable via `Date.parse()`

Name	Type	Platform	Default	Description
				representing a point in the future to count down to.

Tokens

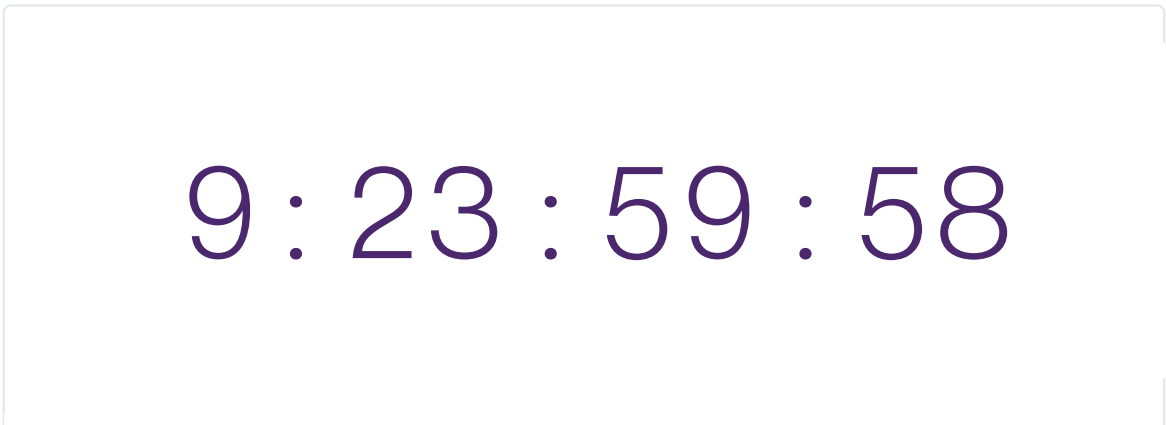
In exceptional circumstances, the following tokens can be passed to this component to override its default styles. **Do not do this unless absolutely necessary.** Read more about overriding styles.

▶ View Tokens

Variants

Feature

Use the `feature` variant in order to display a visually emphasized version of the `Countdown`:



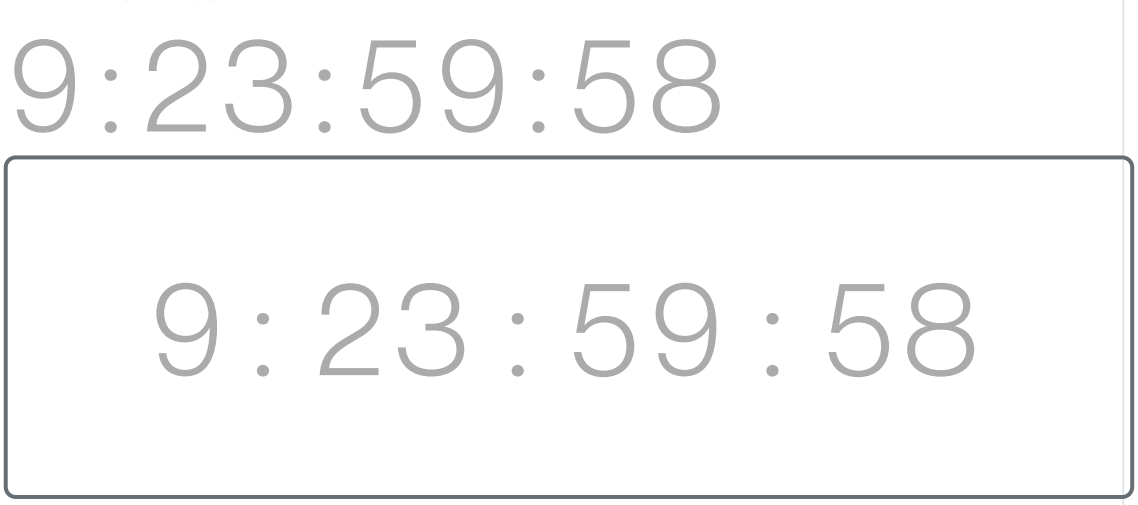

```
function FeatureCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return <Countdown targetTime={targetTime} variant={{ feature:
true }} />
}
```

Inverse

Use the `inverse` variant to display the `Countdown` on darker backgrounds:

9 : 23 : 59 : 58

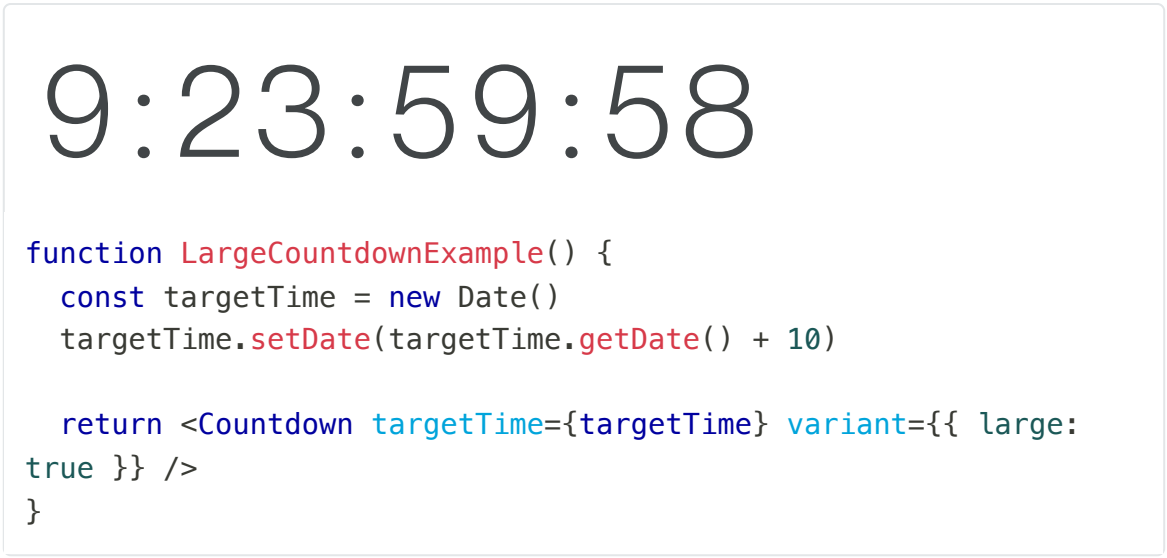



```
function InverseCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return (
    <Box variant={{ background: 'darkest' }} space={1}>
      <Countdown targetTime={targetTime} variant={{ inverse:
true }} />
      <Countdown targetTime={targetTime} variant={{ inverse:
true, large: true }} />
      <Countdown targetTime={targetTime} variant={{ inverse:
true, feature: true }} />
    </Box>
  )
}
```

Large

Set the `large` field on the `variant` prop to `true` in order to display a larger version of the `Countdown`:

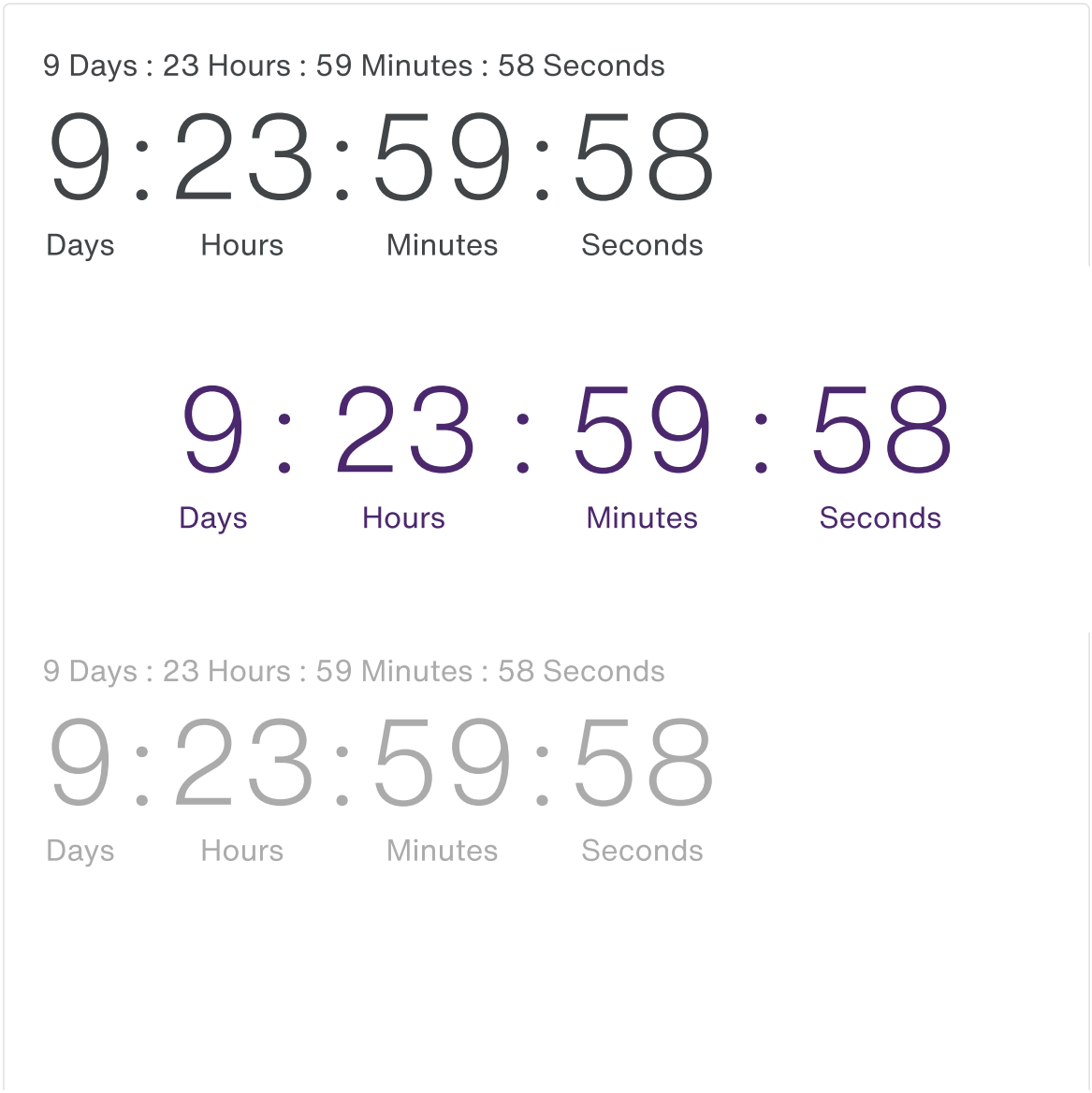


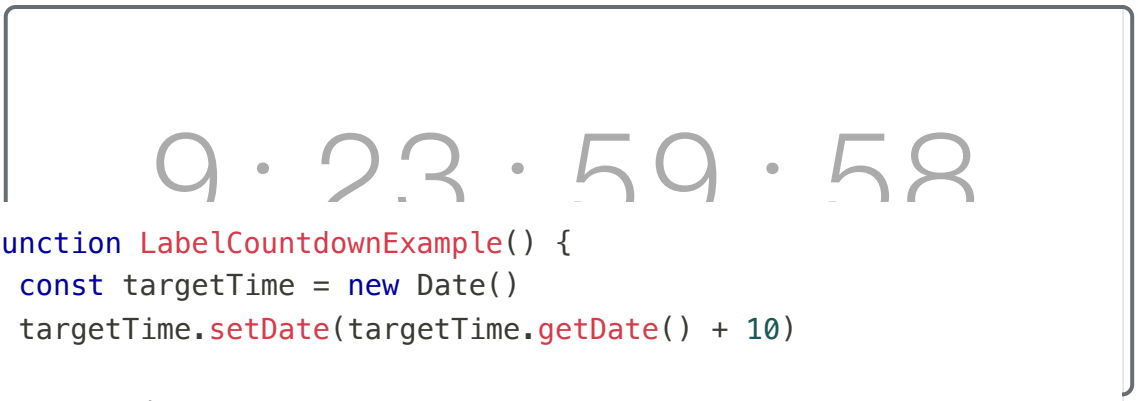

```
function LargeCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return <Countdown targetTime={targetTime} variant={{ large:
true }} />
}
```

Label

Use the `label` variant to annotate the numbers on the `Countdown`:



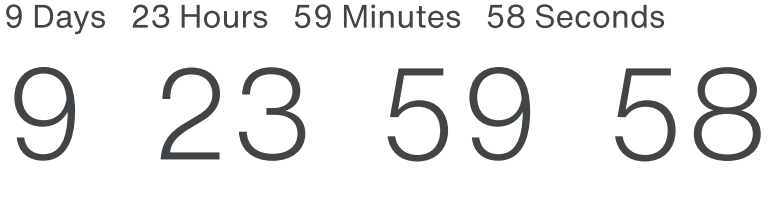


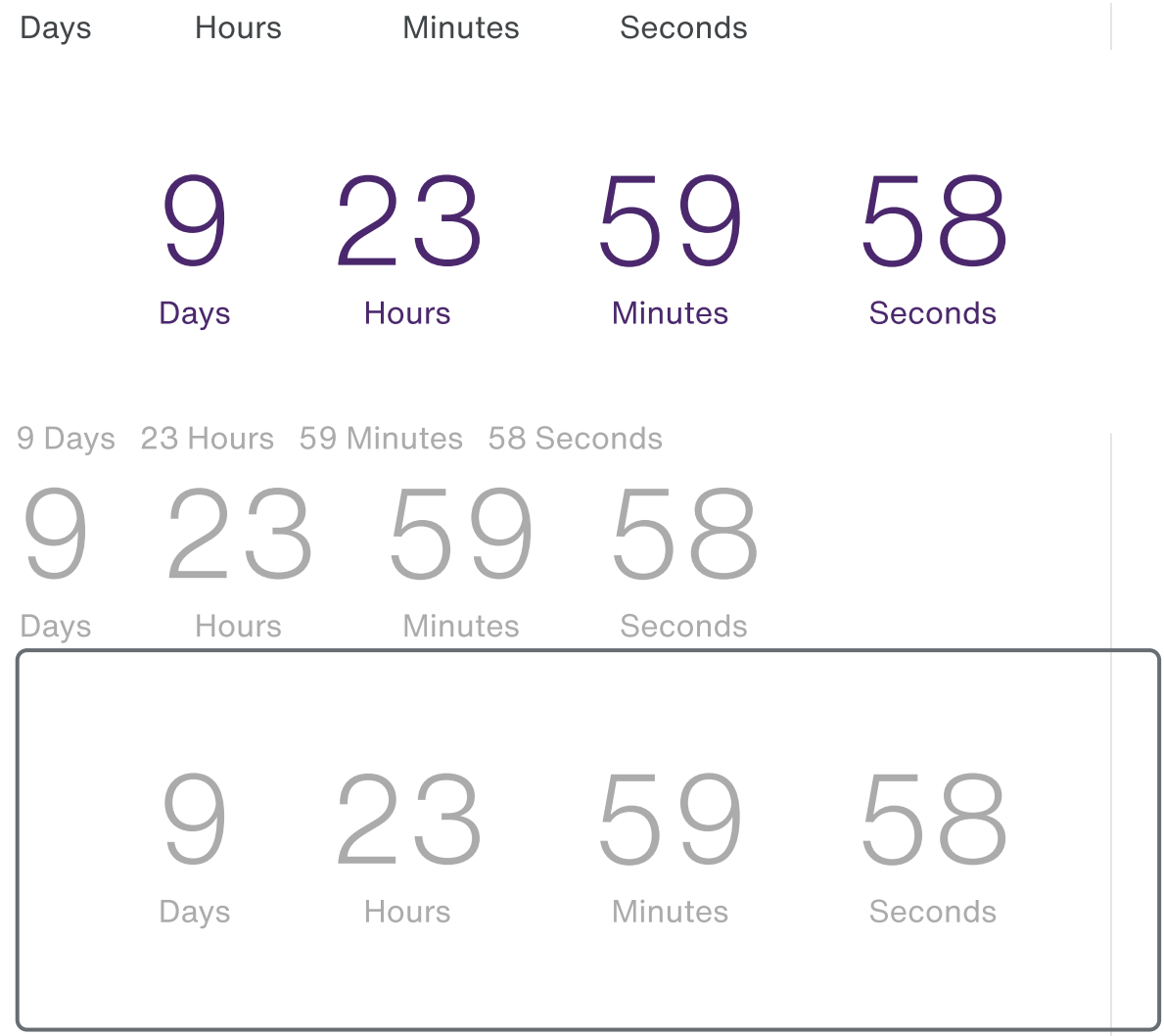

```
function LabelCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return (
    <StackView>
      <Box space={1}>
        <Countdown targetTime={targetTime} variant={{ label:
true }} />
        <Countdown targetTime={targetTime} variant={{ label:
true, large: true }} />
        <Countdown targetTime={targetTime} variant={{ feature:
true, label: true }} />
      </Box>
      <Box variant={{ background: 'darkest' }} space={1}>
        <Countdown targetTime={targetTime} variant={{ inverse:
true, label: true }} />
        <Countdown targetTime={targetTime} variant={{ inverse:
true, label: true, large: true }} />
        <Countdown
          targetTime={targetTime}
          variant={{ inverse: true, feature: true, label: true
}}
        />
      </Box>
    </StackView>
  )
}
```

No divider

Use the `noDivider` variant to omit the colon delimiters on the `Countdown`:






```
function NoDividerCountdownExample() {
  const targetTime = new Date()
  targetTime.setDate(targetTime.getDate() + 10)

  return (
    <StackView>
      <Box space={1}>
        <Countdown targetTime={targetTime} variant={{ label:
true, noDivider: true }} />
        <Countdown
          targetTime={targetTime}
          variant={{ label: true, large: true, noDivider: true
}}
        />
        <Countdown
          targetTime={targetTime}
          variant={{ feature: true, label: true, noDivider: true
}}
        />
```

```
      </Box>
      <Box variant={{ background: 'darkest' }} space={1}>
        <Countdown
          targetTime={targetTime}
          variant={{ inverse: true, label: true, noDivider: true
}}
        />
        <Countdown
          targetTime={targetTime}
          variant={{ inverse: true, label: true, large: true,
noDivider: true }}
        />
        <Countdown
          targetTime={targetTime}
          variant={{ inverse: true, feature: true, label: true,
noDivider: true }}
        />
      </Box>
    </StackView>
  )
}
```

> ⚠ CAUTION
>
> Note that the component throws an exception if you're trying to use the `noDivider` variant without the `label` one, and that is by design.

Feedback

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