

Additional User Props

In this lesson, we'll learn how users can pass additional props to prop getters!

WE'LL COVER THE FOLLOWING



- Passing Additional Props to Prop Getters
- The Final Output
- Quick Quiz

Passing Additional Props to Prop Getters

With the prop getter, `getTogglerProps`, we can cater to other props the user may be interested in passing down to the toggle element.

For example, we currently have this usage by the user:

```
<button {...getTogglerProps()}>Click to view awesomeness...</button>
```



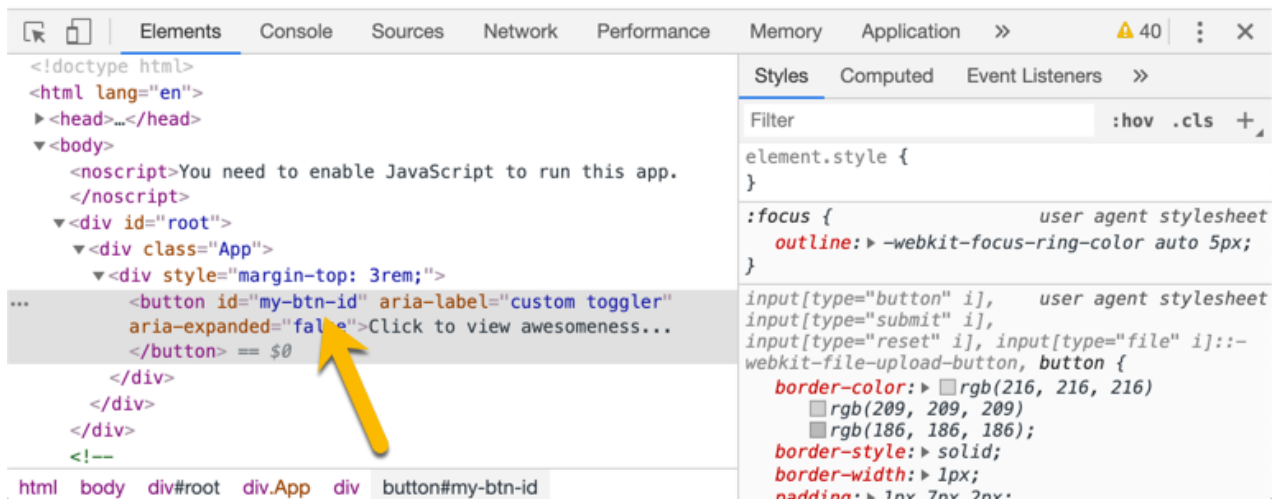
If the user needs some more props that we haven't catered for in the prop collection, they could do this:

```
<button id='my-btn-id' aria-label='custom toggler' {...getTogglerProps()}>  
  Click to view awesomeness...  
</button>
```



Passing `id` and `aria-label` works just fine.

Click to view awesomeness...



However, since these are still part of the “toggler props”, though not common enough to make them a part of the default props collection, we could allow for the following usage:

```
<button
  {...getTogglerProps({
    id: 'my-btn-id',
    'aria-label': 'custom toggler'
  })}
>
  Click to view awesomeness...
</button>
```

It's arguably more expressive. So, let's extend the `getTogglerProps` function to handle this use case.

```
...
const getTogglerProps = useCallback(
  ({ ...customProps }) => ({
    onClick: toggle,
    'aria-expanded': expanded,
    ...customProps
  }),
  [toggle, expanded]
)
```

Since the user will invoke `getTogglerProps` with an object of custom props, we could use the `rest` parameter to hold a reference to all the custom props.

```
({ ...customProps }) => ({
})
```

Then we could `spread` over all custom props into the returned props collection.

```
({ ...customProps }) => ({
  onClick: toggle,
  'aria-expanded': expanded,
  // look here
  ...customProps
})
```

With this extension to `getTogglerProps` we've handled the case for custom props not catered to in the default props collection we expose.

The Final Output

Here it is all put together.

```
.Expandable-panel {
  margin: 0;
  padding: 1em 1.5em;
  border: 1px solid hsl(216, 94%, 94%);
  min-height: 150px;
}
```

How interesting!

Quick Quiz

I started this article with a question: how can we handle custom props in a React component?

Let's take a quick quiz before moving forward

Q

What would happen if we passed custom props like so?

```
({ customProps }) => ({  
  ...
```

COMPLETED 0%

1 of 1



In the next section, we'll consider an even more interesting case.