go back home

# How To Make an Extremely Reusable Tooltip Component With React — And Nothing Else

Aug 2020 react javascript webdev css

*This article was originally published in the DEV Community.*

---

EDIT: Ok, after publishing I realized that "nothing else" is an exaggeration. There is quite a bit of CSS too. But no other JS dependency 🙊

---

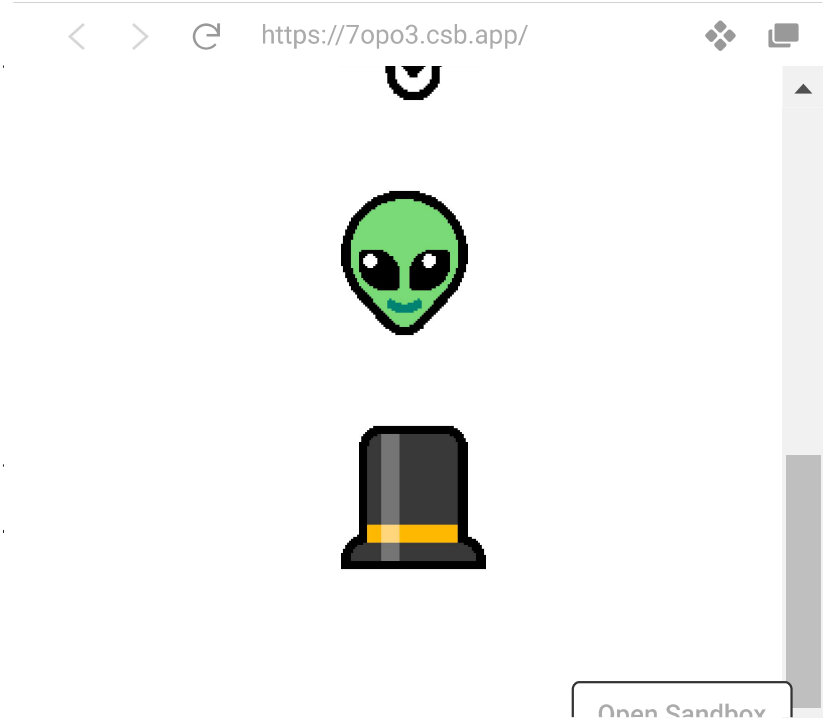
Tooltips are a fantastic way of adding context to a piece of UI.

Recently I had to use lots of them at work. And I'm a rather lazy person, so to make it easier I created a `<Tooltip />` component that I could use to add tooltips to pretty much anything.

It is a no-sweat component, with no dependency other than React itself.

Be aware that most of the patterns applied here are not exclusive to React so you may learn a few new things even React is not your cup of tea. 🙂

Here's a demo before we move into the details:




## How to make it #

What makes this component so good is that it leverages good old CSS sorcery with the flexibility of React `children` prop. We only need two files to make it work: `Tooltip.css` and `Tooltip.js`.

Let's talk about the CSS of it first.

## Tooltip.css #

There's a handful of techniques at play here:

1. Custom properties (CSS vars) that control color, spacing and arrow size
2. CSS border triangles and `before` pseudo-elements to make the arrows
3. Some smart absolute positioning and wrapping to put everything in the right place

Have a look at the `Tooltip.css` file.

Tooltip.css

```
82    top: 50%;
83    transform: translateX(0) trans
84  }
85  /* CSS border triangles */
86  .Tooltip-Tip.left::before {
87    left: auto;
88    right: calc(var(--tooltip-arro
89    top: 50%;
90    transform: translateX(0) trans
91    border-left-color: var(--toolt
92  }
93
```

You can see that half of it is styling to make the tooltip appear in different directions. A CSS preprocessor could make this code leaner but remember, we are keeping it simple.

The biggest takeaway of `Tooltip.css` is understanding that by wrapping a React component with `<Tooltip>` we are also wrapping it with an element styled by the `Tooltip-Wrapper` class.

That CSS class anchors the positioning of the tooltips with `position: relative`. That way we can use `position: absolute` in each tooltip with its `top`, `right`, `bottom`, and `left` values relative to the element we are wrapping.

Now that we understand that `Tooltip.css` handles how the tooltip looks and where it goes, let's talk about its `.js` counterpart.

## Tooltip.js #

`Tooltip.js` does four important things:

1. It takes everything inside a `<Tooltip>` component and moves it inside a `div` with `Tooltip-Wrapper` class by using `props.children`

2. It controls *what* content will be inside the tooltip bubble with `props.content`
3. It controls *where* the bubble will appear using the value passed to `props.direction` as a class.
4. It controls *when* it shows by listening to `onMouseEnter` and `onMouseLeave` events

Here, have a snoop at `Tooltip.js`:

Tooltip.js

```
1  import React, { useState } from '
2  import "./Tooltip.css";
3
4  const Tooltip = (props) => {
5    let timeout;
6    const [active, setActive] = use
7
8    const showTip = () => {
9      timeout = setTimeout(() => {
10       setActive(true);
11     }, props.delay || 400);
12   };
```

Can you see how it works together with `Tooltip.css`?

The biggest takeaway of this file is that it has the minimal necessary logic to make CSS shine. All the work it does is moving the props you passed to `<Tooltip>` into the right places.

So at the end of the day (or at the end of the reconciliation 😄), all that Tooltip.js does is transforming this:

```
<Tooltip content="Hello, I'm a tooltip" direction="right">
  <button>I'm a button</button>
</Tooltip>
```

Into this:

```
<div className="Tooltip-Wrapper" onMouseEnter={showTip} onMouseLeave
  <button>I'm a button</button>
  {active && <div className={`Tooltip-Tip right`}>Hello, I'm a toolt
</div>
```

## How to use it #

After learning how it works, the "how to use it" should be pretty simple to grasp.

All you need to do is import the `Tooltip` component and use it as a wrapper. Make it go above anything you want to show a tooltip on hover.

It takes three props:

1. `content`, which will be what's inside the tooltip

- Required, It can be anything JSX, text, images, other components, it's up to you

2. `direction`, which controls where the tooltip will show

- Optional, accepts `top`, `right`, `bottom`, and `left`. Defaults to `top`

3. `delay`, how much time, in milliseconds, it takes for the tooltip to show.

- Optional, defaults to 400ms

Add a simple wrap with a some of these props and *bam* now every hover on anything that is *inside* `<Tooltip>` will show a small balloon of content.

Here's how I did it in the demo:

App.js

```
1  import React from "react";
2  import Tooltip from "./Tooltip";
3  import "./styles.css";
4
5  export default function App() {
6    return (
7      <div className="App">
8        <h1>Hello, this is a React tooltip dem
9        <p> Try hovering the emojis below </p>
10
11       <div className="example-wrapper">
12         <Tooltip content="Yee-haw
```

Open Sandbox

Pretty cool, right?

What I love the most about modern web development is how components make stuff easier to implement after some initial setup.

Doing the same thing during jQuery times would take a lot of repetition, duplication, and much more elbow grease.

And as a final reflection, I'm sure that some things in front-end look crazy complex now but these kinds of techniques make me feel that we are moving in the right direction.

---

And that's it, thanks for reading. I hope this article is useful on your front-end journey!

As always, comments and feedback are super welcome, so what would you change or improve in this implementation?

Cover photo by Drew Beamer on Unsplash

---

## Hey, let's connect 👋 #

Follow me on Twitter and let me know you liked this article!

And if you *really* liked it, make sure to share it with your friends, that'll help me a lot 😄