Perfecting the Position

As you can see, the Tooltip that currently appears needs to be repositioned. In this lesson, we will try to reposition it to get a neater look!

WE'LL COVER THE FOLLOWING

Moving the tooltip to Javascript

Let's move the tooltip generation into JavaScript and have it replace the dot that we were testing with.

I also gave the text some properties so that it has some space for further confirmation that the tooltip is positioned correctly. Since absolute positioning's top and left attributes tell the tooltip to align its top-left corner at the specified coordinates, this appears to be working as expected.

the specified coordinates, this appears to be working as expected.

Avoiding multiple tooltips

We haven't written any code to tell the tooltip to disappear and reappear when different text is highlighted, so it's expected that multiple selections make the tooltip show up multiple times. This might be annoying for testing, so rather than save this for later, we'll fix it now.

We'll refactor to move the element creation at the root level, so it's not created every time the event happens. The only things that change are its existence on the DOM and the top and left attributes.

Output

JavaScript

HTML

CSS (SCSS)

We also had to move the `svgs` variable above, since JavaScript is a language that requires variables to be already defined by the time it's referenced.

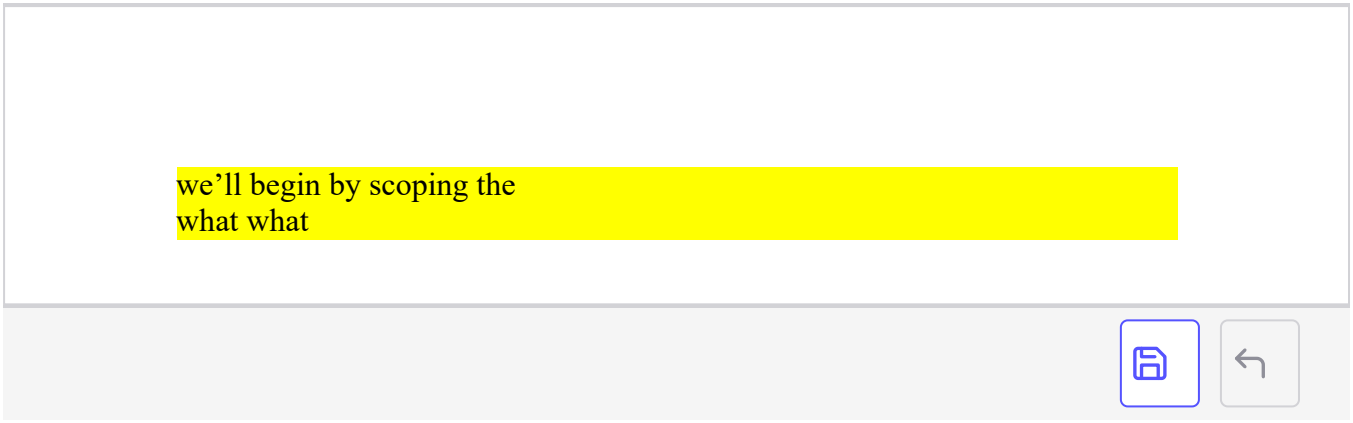
Next, we can position the tooltip such that the center of the bottom edge is where the top-left corner currently is.

Output

JavaScript

HTML

CSS (SCSS)



One thing to note is that the `offsetWidth` and `offsetHeight` aren't available until the element is added onto the DOM. Our options were to either define in JavaScript the `10rem` that the CSS sets the width to and take half of that, or calculate the width after it goes onto the DOM.

Q

Would setting the width a hardcoded value of `10rem` for the element in Javascript be a better option?

COMPLETED 0%

1 of 1

If we hadn't used JavaScript to get the width programmatically but instead used a constant like `10rem`, the result would have been the same, but the

drawback is that it couples the logic to the styling unnecessarily. The only

other answer that's possible is D, but setting any `style` attribute in JavaScript will be rendered just the same as if it had been defined in CSS.

Now let's add the little tail thing so we can align it to its final state vertically.

Adding the arrow end

To make a triangle, a quick search tells me, I can use empty width and heights and create the triangle with borders. Works for me! `

This looks fantastic! Let's smooth out the rough edges in the next lesson.