Rev. 08Aug2019

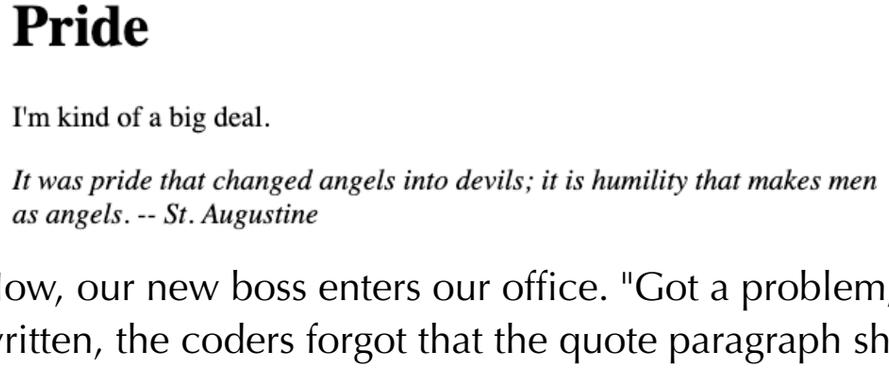
Exercise 8

As you learn JavaScript better, you'll see you can do a lot to manipulate the page. In fact, much deeper in the course, we'll be using JavaScript for the vast majority of things we want to do on a page. But that comes later; for now, let's just explore more.

BEYOND DOCUMENT.GETELEMENTBYID

In previous exercises, you've used both CSS and JavaScript to operate on an HTML page: CSS for styling and JavaScript for adding elements and for content changes.

Run **Exercise 8** in your repl; it should look something like this:



Now, our new boss enters our office. "Got a problem," he says. It turns out that when the HTML was written, the coders forgot that the quote paragraph should have had a `class` of `quote` *and* `inset`. Instead, it only got a `class` of `quote`. In other words, it should have been written like this:

```
<p class="quote inset"></p>
```

But we can fix this with JavaScript!

You've used `document.getElementById(id)` to grab hold of a particular element with a certain `id`. That works very well and you'll continue to use it. It doesn't work for identifying elements by `class` however. For that, let me introduce you to another JavaScript snippet: `document.querySelector`. This is a more generic way of identifying *anything* on the page.

While `getElementById` is a function that accepts an id as an argument, `querySelector` is a function that can accept any CSS-like identifier. Here is how we could capture the `<div>` with an `id` of `main`, using both functions:

```
document.getElementById("main")
document.querySelector("#main")
```

Notice than when we use `querySelector`, we provide the selector using the CSS-like syntax.

But how does this help us? We're still capturing elements by `id`, when we need to capture the the paragraph with a `class` of `quote` — but no `id`?

`querySelector` isn't restricted to taking an `id`; we can provide it with a CSS-like *class* description. `document.querySelector(".quote")` gets us a handle on the element we want. Now, we just need a little more JavaScript magic.

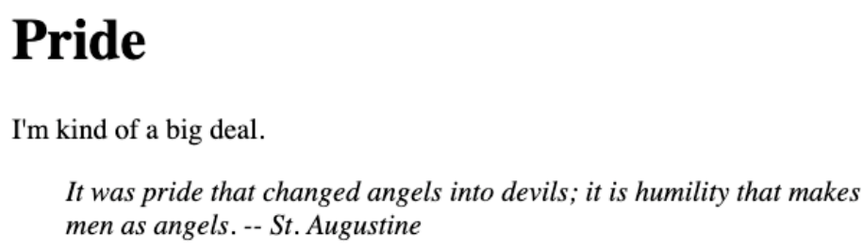
`classList.add(class-name)` is a function we can call on any element we have a handle on (whether we got that element by `getElementById` or `querySelector`). It does what you might guess it does: adds a `class` name to the element dynamically.

With that knowledge, we can fix the problem by adding this snippet of code to `index.js`:

```
let quoteElement = document.querySelector(".quote")
quoteElement.classList.add("inset")
```

- [ ] Try it!

Since the CSS rule for `.inset` already exists in `index.css`, our display now has the inset showing:



One final thing: if you wanted everything to be on one line, this code would do it:

```
document.querySelector(".quote").classList.add("inset")
```