Rev. 08Aug2019

Exercise 37

EVENT LISTENERS

In previous exercises, we used event handlers similar to this:

```
<button value="p100" onclick="processClick(p100)">P 100</button>
```

And our `processClick` function might look like this:

```
let processClick = (buttonValue) => {
  // function body
}
```

That works, but it introduces what is often a bad practice: mixing JavaScript directly with HTML. Here's one way to think about why that might not be a best practice. We've hired two developers. Tony is expert with HTML and CSS. He's even submitted some things for `zengarden.com`. Yay, Tony! And we hired you who, though no slouch with HTML and CSS, really shines with JavaScript. Yay, you!

We give Tony some mockups and ask him to go to town. At the same time, we explain to you exactly what we want the application to *do*. Tony's heads down making things make sense to the user while looking beautiful. But JavaScript? That's not really Tony's thing (unlike you).

So we don't want Tony to be writing things like `onclick` event handlers; we just want him to concentrate on the user interface. And we certainly don't want you having to open up Tony's code and add event handlers. We have a conundrum.

Actually, it's pretty easy to solve with JavaScript. First, let's get rid of the event handler in Tony's code:

```
<button value="p100">P 100</button>
```

Now, in our JavaScript file, we'll attach an *event listener* to that button programmatically:

```
// get reference to Tony's button
let button = document.querySelector('button')

// create a function to be executed when a specified event fires
let logger = (event) => {
  console.log(event)
}

// attach an event listener to Tony's button so that when it's clicked, our logger
   function will execute

button.addEventListener('click', logger)
```

What's the difference between an *event handler* (as we've been using up until now) and this new *event listener* formulation? The primary difference is that, with an event listener, our function (`logger` in this case) is automatically sent the occurring *event*.

And what is this *event*? It's a JavaScript object generated by the browser and, as I said, sent automatically to the function you associated with the event listener. The event object contains *lots* of information, but the primary thing we want is the *target* property, which we can access inside our `logger` function as `event.target`.

- Enough talking! In `index.html`, locate Tony's button.
- In `index.js`, get a reference to that button using `document.querySelector`.
- Write the `logger` function. For the function body `console.log event.target`.
- Attach a click event listener to the button. (Look at the example above to see that the `addEventListener` function takes two arguments: the type of event, and the function to call when that event fires.)
- Run your repl and examine the **console** tab.

Hmmm...that was kind of unimpressive — but we did get something:

```
HTMLButtonElement {}
>
```

What we want is not the HTML element on which the event occurred but the `value` property of the HTML element:

```
<button value="p100">P 100</button>
```

That's very easy to get. Instead of using `event.target`, we use `event.target.value`.

- ☐ Change your `logger` function so that it `console.logs event.target.value`, then try out your repl.