Event Handling: Input Focus and Blur

Form inputs are one of the most common HTML elements users interact with on a website. By now, you should be familiar with how to listen for a *click event*and run a script. In this reading, we'll learn about a couple of other events on an input field and how to use them:

- Element: focus event
- Element: blur event

Listening for focus and blur events

According to MDN, the focus eventfires when an element, such as an input field, receives focus (i.e. when a user has clicked on that element).

The opposite of the focus event is the blur event. The blur event fires when an element has lost focus (i.e. when the user clicks out of that element).

Let's see these two events in action. We'll set up an HTML page that includes`<input type="text" id="fancypants">`. Then, we'll write JavaScript that changes the background color of the `fancypants`textbox to `#E8F5E9`when the focus is on the textbox and turns it back to its normal color when focus is elsewhere.

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <script src="script.js">
  </head>
  <body>
   <input type="text" id="fancypants">
  </body>
</html>
```

Javascript

```
// script.js

window.addEventListener("DOMContentLoaded", event => {
  const input = document.getElementById("fancypants");

  input.addEventListener("focus", event => {
    event.target.style.backgroundColor = "#E8F5E9";
  });
  input.addEventListener("blur", event => {
    event.target.style.backgroundColor = "initial";
  });
});
```

In the code above, we changed the background color of the input on `focus` and changed it back to its initial value on `blur`. This small bit of functionality signals to users that they've clicked on or off of an input field, which is especially helpful and more user-friendly when there is a long form on the page. Now you can use `focus` and `blur` on your form inputs!

What we learned:

- How to listen for the focus event on inputs
- How to listen for the blur event on inputs