Time	~ 1 hour
Learning Goals	• Learn how to attach an event handler to a DOM element • Learn JavaScript functions for DOM element manipulation

Event Handlers

We have learnt about the DOM and what an event is previously. Recall that a function that executes response to an event is called an event handler. Let's now learn how to attach an event handler to a DOM element in vanilla javascript.

In jQuery, we can use `$(selector).on('event-name', someFunction())`. For this challenge, we will be using the JavaScript function `addEventListener()` to attach event handlers. Before you start coding, read this short tutorial and documentation so that yo have a better idea of what this function does.

Start challenge

Fork and clone the git repo for this challenge: https://github.com/NextAcademy/js-events. There will be two folders in the repo, namely:

1. `submit-button`
2. `inside-out-js`

Let's first start with the code in the `submit-button` folder.

Example 1: Submitting a Button

index.html:

Open the file `index.html` in your browser and play around with it. It will be a good idea to open up your developer's tool and the JavaScript console so that you can observe what happens in the console when you click on the button. Take note especially to what happens before and after you click the **Submit** button.

Once you are done exploring the website, go to `index.html` and read through the code. There are 3 things that we would like to draw your attention to:

1. Line 11: We have an `input` tag which takes in the word you keyed in. An ID `word` has been assigned to this tag.

2. Line 12: The ID `event-btn` has been assigned to the button. We will be attaching an event handler to this button so having an ID helps us with easier identification of the element that we would like to attach the handler to.

3. Line 13: Notice that we have an empty `p` tag here, i.e. `<p id="encrypted-word"></p>` with the ID `encrypted-word` to it. If you have your developer console open, pay attention to this element before and after you click the `Submit` button. You will notice that a new word will appear in this `p` tag after you have clicked the button. Hmm, what's going on here? Let's head to `index.js`to find out more!

index.js:

First read through the code in `index.js` carefully. Observe the lines where `console.log` and `debugger` is placed. If you have opened up your JavaScript console just now, you will be able to see `console.log` in action. Now trying uncommenting the `debugger`, save the file, refresh your browser and submit a word. What happens? How can you use the console to test code?

Once you are more comfortable with using the debugger, let's now pay attention to the following lines:

1. Line 4: `var str = document.getElementById("word").value;`

The function `document.getElementByID("word")` returns the element object in your HTML document referenced by the given ID. In this case, the element object will be the input field. However, if we want the text in the input field, we will need to include `.value` to return the text in the object. You can read more about the `value` property here.

2. Line 19: `getElementById("encrypted-word").innerHTML = "Your new word is: " + str;`

Here, the element that we are getting is the `p` tag with ID "encrypted-word". By calling `.innerHTML` on the element, we are saying that we want to change the content in the given `p` tag. When this function is executed, "str", which is our encrypted word, will replace the content in the `p` tag, which was empty at first. This is also why you only see the encrypted word appear after the button is clicked. `.innerHTML` is very useful whenever you want to change the contents of an element when an event is triggered!

3. Lines 23 - 24:

```
document.addEventListener('DOMContentLoaded', function(){
  document.getElementById('event-btn').addEventListener("click", getEncryptedWord);
});
```

This is an example of writing a function within a function, and it happens a lot in JavaScript, so it's good to start getting used to this syntax. Take note of how the curly and round brackets are placed. And the semicolons too!

Let's first focus on line 24. Here we get the button with ID `event-btn` and call `addEventListener` on the button. We then attach a click event to button such that the function `getEncryptedWord` is executed when the button is clicked. Note that we need not put the brackets after `getEncryptedWord` here.

Finally, we want our `addEventListener` event in line 24 to fire when the DOM is loaded, but before all page assets are loaded (CSS, images, etc.) for efficiency purposes. That's why we have the code in line 23. Find out more about DOMContentLoaded here. What happens if we did not include line 24? Give it a try!

Example 2: Adding and Manipulating DOM Elements

Now that you have a better idea of how to attach an event handler, let's move on to the second folder, i.e. `inside-out-js`. As usual, open `index.html` in your browser with the developer's tool open as well. Play around and observe what happens. Using the 5 emotions (Joy, Sadness, Anger, Disgust and Fear) from the animated movie "Inside Out", every time you click on the circle, an emotion will be returned to you. Notice also that the `Clear` button only appears after you clicked on the circle after the page is loaded

It's now your turn to explore the functions given in the JavaScript code. There will be some pointers in the code to guide you with what's going on. But before you even go into JavaScript, make sure you understand what is going on in both the HTML and CSS files. It would also be a good idea to do some wireframing before going into the code. Have fun! :)