Event Handling: Click Events With Event.target

Imagine a user is viewing a Web page showing 300 different products. The user carefully studies the page, makes a selection, and clicks on one of the 300 products. Could we find out through code which element was clicked on? Yes!

Previously we learned how to handle a click event using an element's ID. However, what if we don't know the ID of the clicked element before it's clicked? There is a simple property we can use to discover on which element the click event occurred: `event.target`.

According to the MDN doc on event.target, "the `target`property of the`Event`interface is a reference to the object that dispatched the event. It is different from event.currentTargetwhen the event handler is called during the bubbling or capturing phase of the event." Essentially:

- `event.target`refers to the element on which the event occurred (e.g. a clicked element).
- `event.currentTarget`refers to the element to which the event handler has been attached, which could be the parent element of the `event.target`element. (*Note: We'll talk about this in more detail in the reading on The Bubbling Principle.*)

It is common practice for developers to use `event.target`to reference the element on which the event occurs in an event handling function. Let's practice using this handy property to get the ID of a clicked element.

Use event.target to console.log the ID of a clicked div

Let's say we had an HTML page with 10 `divs`, each with a unique ID, like below. We want to click on any one of these divs and print the clicked div's ID to the console.

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="example.css" />
    <script type="text/javascript" src="example.js"></script>
  </head>
  <body>
    <div id="div-1" class="box">1</div>
    <div id="div-2" class="box">2</div>
    <div id="div-3" class="box">3</div>
    <div id="div-4" class="box">4</div>
    <div id="div-5" class="box">5</div>
    <div id="div-6" class="box">6</div>
    <div id="div-7" class="box">7</div>
    <div id="div-8" class="box">8</div>
    <div id="div-9" class="box">9</div>
    <div id="div-10" class="box">10</div>
  </body>
</html>
```

In our linked **example.css**file, we'll add style to the `.box`class to make our `divs`easier to click on:

CSS

```
.box {
  border: 2px solid gray;
  height: 50px;
  width: 50px;
  margin: 5px;
}
```

Now, we'll write Javascript to print the clicked div's ID to the console. Again, we want to wait for the necessary DOM elements to load before running our script using `DOMContentLoaded`. Then, we'll listen for a click event and `console.log`the clicked element's ID.

Javascript

```
// example.js

// Wait for the DOM to load
window.addEventListener("DOMContentLoaded", event => {
  // Add a click event listener on the document's body
  document.body.addEventListener("click", event => {
    // console.log the event target's ID
    console.log(event.target.id);
  });
});
```

If you open up your HTML in a browser, you should see the 10 `divs`. Click on any one of them. Open up the browser console by right-clicking, selecting*Inspect*, and opening the *Console*tab. The ID of the div you clicked should be printed to the console. Click on the other divs randomly, and make sure their IDs print to the console as well.

What we learned:

- The definition of `event.target`
- How `event.target`differs from `event.currentTarget`
- How to console.log the ID of a clicked element using `event.target`