The JavaScript DOMContentLoaded Event

- Due No Due Date
- Points 1
- Submitting a website url

(https://github.com/learn-co-curriculum/phase-1-domcontentloaded) (https://github.com/learn-co-curriculum/phase-1-domcontentloaded/issues/new)

Learning Goals

- Understand why `DOMContentLoaded` is important
- Set up an event on `DOMContentLoaded`

Introduction

An important part of working with JavaScript is ensuring that your code runs at the right time. Every now and then, you may have to add some extra code to ensure your code doesn't run before the page is ready. Many factors go into determining the "right time," but there are two events that represent two particularly important milestones in terms of page load:

1. The `DOMContentLoaded` event fires when your page's DOM is fully parsed from the underlying html
2. The `load` event fires when a resource and all its dependent resources (including CSS and JavaScript) have finished loading

In this lesson, we'll be focusing on `DOMContentLoaded`.

Why is DOMContentLoaded Important?

The `DOMContentLoaded` event is the browser's built-in way to indicate when a page's html is loaded into the DOM. It isn't possible to manipulate HTML elements that haven't rendered yet, so trying to manipulate the DOM before the page fully loads can potentially lead to problems.

We need to make sure to wait until *after* the `DOMContentLoaded` event is triggered to safely execute our code. By creating an event listener, we can keep our code from immediately firing when `index.js` is loaded.

Set Up an Event Listener for DOMContentLoaded

As always, `addEventListener` takes a `String` with the name of the event and a *callback function*.

```
document.addEventListener("DOMContentLoaded", function() {
  console.log("The DOM has loaded");
});
```

If you put the above code in `index.js`, 'The DOM has loaded' will not be logged immediately. In fact, you can confirm this yourself by putting a second `console.log()` *outside* of the event listener callback:

```
document.addEventListener("DOMContentLoaded", function() {
  console.log("The DOM has loaded");
});

console.log(
  "This console.log() fires when index.js loads - before DOMContentLoaded is triggered"
);
```

Instructions

Code your solution in `index.js`. First, set up a `DOMContentLoaded` event listener to detect when the HTML page has loaded and the document is ready to be manipulated. Use the event's callback function to target the paragraph element with `id="text"` and replace the text with "This is really cool!"

Note: Using the `innerText` (https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/innerText) property to modify DOM element content will not work for this lab. Use `textContent` (https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent) or `innerHTML` (https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML) instead.

Test your event in the browser to confirm that it is working.

DOMContentLoaded Does Not Wait For Stylesheets and Images to Load

It is important to note that the `DOMContentLoaded` event fires once the initial HTML document finishes loading, but does not wait for CSS stylesheets or images to load. In situations where you need *everything* to completely load, use the `load` event instead.

While both will work, it is often the case that we only need the HTML content to fully load in order to execute our JavaScript. Since images can take some time to load, using the `load` event means visitors of a webpage may see your webpage in its original state for a couple of seconds before any JavaScript fires and updates the DOM.

For a comparison of the difference between `DOMContentLoaded` and `load`ed events, **check out this example (http://web.archive.org/web/20150405114023/http://ie.microsoft.com/testdrive/HTML5/DOMContentLoaded/Default.html)**.

Conclusion

JavaScript provides us the powerful ability to update webpage content without refreshing. We can, for instance, have a page with some basic HTML structure and use JavaScript to fill in the content, enabling the possibility of dynamic webpages.

This sort of action, however, will only work if the HTML content is loaded on the page before the JavaScript is executed. The `DOMContentLoaded` event ensures that our JavaScript code is being executed immediately after the HTML is finished loading.

Addendum

The `DOMContentLoaded` event is now a widely accepted standard. Modern web development, however, provides us with additional choices for setting up when we want our JavaScript to execute. For example, HTML5 now has a **`defer`** **(https://www.w3schools.com/tags/att_script_defer.asp)** attribute for use in `<script>` tags:

```
<script src="index.js" defer></script>
```

This functions in a similar way to `DOMContentLoaded`: the JavaScript code stored in `index.js` will be loaded up but won't execute until the HTML page completely loads.

Resources

- DOMContentLoaded (https://developer.mozilla.org/en-US/docs/Web/Events/DOMContentLoaded)
- Running Your Code at the Right Time (https://www.kirupa.com/html5/running_your_code_at_the_right_time.htm)