DOM Project: Create a profile page with Javascript

Now that you've learned about the DOM and how to access and manipulate it, put skills to use by building your own basic profile page! In this project, you will:

- Create an HTML file and link to a Javascript file
- Use Javascript to create and update page elements
- Add CSS classes with Javascript
- Create a clock using the Date object

Introduction

The best way to learn is to create something that is meaningful or relevant to you, so why not start by making a page all about yourself?

In this project, you'll create a simple profile page that displays details about you, such as who you are, what you like to do, and where you are located.

Put as many or as few details as you like. Don't worry, the government already knows where you live. It's your page, so feel free to give it your own flair!

Project Overview

You've learned about the DOM, and now it's time to put that knowledge into practice.

In this project, you'll create a simple profile page and fill it with details about yourself.

You could hard-code your content into your HTML file, but where's the fun in that? We'll practice using Javascript to access DOM elements and insert content into your page dynamically.

We'll also go over how to add CSS class attributes to elements dynamically, so you can add a bit of styling to your profile.

Phase 1: Setting up your HTML and Javascript files

Create an HTML file in a new project folder called `myProfile.html`. Set up your html file with a `head` and `body` section. Other than the appropriate HTML tags, leave the file empty of content, ids and classes.

In your HTML file, add a link to an external Javascript file in your project directory called `myProfile.js`. Test that your Javascript file is linked correctly by printing something you can read in the browser console. Example:

```
console.log("This is my profile page!")
```

Phase 2: Populating your profile

Again, don't hard-code any content in your HTML file. Instead, construct the page content using your Javascript file.

First, you should make sure all the DOM objects you need are loaded before you add new things to the page. Add a `DOMContentLoaded` event listener in your Javascript file.

```
window.addEventListener("DOMContentLoaded", event => {
  // Your Javascript goes here
});
```

Phase 2A: Creating and appending new elements

It's time to add some content to your profile page! Insert your name as an `h1` into the page using Javascript, and give it an `id`. *Hint: You may want to use the following:*

- `document.createElement()`
- `Element.setAttribute()`
- `document.createTextNode()`
- `Node.appendChild()`

After you've inserted the `h1`, open your HTML page in your browser and make sure the `h1` with your name appears on the page.

You've added your name to the page, but it still looks sparse. Fill out your page with some details about yourself by creating and appending new elements. Use what you've learned about manipulating the DOM to add to your Javascript file.

- Create a new unordered list element.
- Append at least **four list items** of details about yourself to your list.
- Append your list to the body of your page.

Below is an example list item:

```
const listItem1 = document.createElement("li");
const listItem1Content = document.createTextNode(
  "I like to drink iced lattes."
);
listItem1.appendChild(listItem1Content);
// Append listItem1 to your unordered list here
```

Phase 2B: Refactoring to make it programmatic

The code we wrote above works, but it is lengthy and leads to needless repetition. Imagine we want to display 20 hobbies. Following the pattern above means we would have to create an element, create a text node, and append a child node to the details list 20 times for each hobby. That's 80 extra lines of code!

Let's approach this differently and make the work easier for ourselves. Can we refactor it to make inserting the `li` elements more programmatic and easily repeatable? Yes, we can! First, let's create the `ul` and append it to the body of our page, as we did in the last section.

```
// Create the element with document.createElement()
// Set the attribute with Element.setAttribute()
// Append the element to the page with Node.appendChild()
```

Now, let's add the list items. We can shorten the code up by creating an array that stores all of the list items as string values, join all the string values into a single string, and insert that string into the DOM.

```
const detailsArr = [
  "<li>I like to drink iced lattes.</li>",
  "<li>I have two cats and eight kittens.</li>",
  "<li>My favorite place to get lunch is Chipotle.</li>",
  "<li>On the weekends, I play flag football.</li>"
];
const liString = detailsArr.join(" ");
```

```
const listElement = document.getElementById("details");
listElement.innerHTML = liString;
```

Notice that we used `innerHTML` here rather than `appendChild`. If we tried to insert the string using `appendChild`, what would happen? Why? Refer to the MDN documentation on Element.innerHTML and Node.appendChild for the answers.

You've cut down on the lines of code as well as made your code more readable! You can easily add new list items inside your array and they'll be automatically added to your `ul` element. Now that you've refactored your code, can you add new sections to your page?

Phase 3: Adding CSS classes and styles

You've added the details, but now they need some pizazz! Let's add some CSS classes to your elements that you can use to style the page. In your Javascript, add a class named `my-details` to the unordered list you added in the last section. You can use `Element.setAttribute` to set the class name to your `ul`. Here's an example:

```
const myDetails = document.createElement("ul");
myDetails.setAttribute("class", "my-details");
```

Now that you know how to set an element's class name, practice setting attributes by adding class names to the other elements you created.

- Add a class name of `name` to the `h1` containing your name.
- Add a class name of `detail` to each `li` element you created inside your list.
- Use `document.querySelectorAll` to access each `li`.
- Use `forEach()` to iterate over the `li` list and add a class using `Element.className`.

Create a CSS file and remember to link to it in your HTML file. In your CSS file, add some styles using the classes we added to the `ul` and the `li` elements: `details` and `detail`. Update your CSS to:

- Change the color of your `h1`
- Add a border around your `ul`
- Add padding to your list items

```
h1.my-name {
  color: green;
  padding: 40px 20px;
}

ul.my-details {
  border: 1px solid gray;
  padding: 40px;
}

li.detail {
  list-style-type: square;
  padding: 10px;
}
```

Feel free to add more CSS styles beyond the ones above to your page to personalize it!

Phase 4: Adding a clock with the Date object

By now, you should know how to add new elements to your page programmatically. Let's kick it up a notch by adding a clock that keeps the current time onto your profile page.

Objectives:

- Create a new element and add it to the body of your page
- Use the Javascript Date object to get the current time
- Insert the current time into a DOM element

You just created a live clock, and it's *time* for congratulations! (Har har.) Can you go the extra mile? Figure out how to insert your clock into a new list item under your personal details that says "I live in City, State, and it's currently `[CLOCK]` here.

Bonus: You're so extra!

Congratulations! You've created a basic profile page by manipulating the DOM and inserting elements dynamically with Javascript. But, why be *basic* when you can be a little *extra*? Make your profile extra shiny by adding more to your page.

Bonus A: Add more sections to your page

Using Javascript, create new elements and:

- Insert an image into your profile under your name. *Hint:* You could insert a new `img`, or you could add a `div` and set the background image using a CSS class.
- Insert more sections to your profile. Examples: "Likes" list and "Dislikes" list, "Favorite Restaurants", "My Activities", etc.

Bonus B: Use other Element methods

Check the MDN documentation for more Element methods you can use to manipulate the DOM. Try doing the following:

- Use `.outerHTML` to replace an element.
- Use the `classList` API to add/remove classes.
- Try using : `Element.closest`, `getElementsByClassName`, `getElementsByTagName` to select elements on your page.

Bonus C: Fire JS on different DOM events

You have used `DOMContentLoaded` to run Javascript on a DOM event. It is one of many different DOM events that developers can use to trigger functionality.

Check out MDN``s Event Reference documentation to see how many different DOM events there are. Trying using a few of them in your code. Some relevant methods to use would be under:

- Keyboard events
- Mouse events
- DOM mutation events

Bonus D: Make a countdown clock

Instead of keeping the current time on your profile page, make a countdown clock to your birthday.

Refer to the MDN documentation on the Date object for help with this task. Review how to calculate the elapsed time between two dates.