Challenge: TaskLister Mini-Project

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

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

Learning Goals

- Build a functioning to-do list application
- Suppress a default action with `event.preventDefault()`

Introduction

In this lab, you'll be creating a simple to-do list application that uses JavaScript to manipulate the DOM.

Check out the **working demo** (https://learn-co-curriculum.github.io/js-task-lister-lite/) !

Lab: Build a Functioning To-Do List Application

Instead of relying on tests, this lab is *deliverable driven*. You will be responsible for ensuring that your solution works as intended by testing the behavior in the browser.

1. Fork and clone this repository
2. Open `index.html` in Chrome
3. Put your JavaScript knowledge to the test and work your way through the deliverables

Once you're done, be sure to commit and push your code up to GitHub, then submit the assignment using CodeGrade. Even though this lab does not have tests, it must still be submitted through CodeGrade in order to be marked as complete in Canvas.

Structuring Your Code

You've been provided with a basic HTML file, as well as an `index.js` file where you can implement your solution. Note that the `index.js` file is contained within a `src` folder — this is a common pattern that you will see in many labs moving forward. If you take a look at the `index.html` file, you'll see that the `script` tag that loads the code file includes the `src` directory in its path:

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

Deliverables

- As a user, I should be able to type a task into the input field.
- As a user, I should be able to click some form of a submit button.
- As a user, I expect to see the task string that I provided appear in the DOM after the submit button has been activated.

Note: While the example (https://learn-co-curriculum.github.io/js-task-lister-lite/) shows one possible working implementation of the TaskLister app, yours can (and is encouraged to!) look however you like!

HTML Forms

For this lab, we are going to be using the HTML `<form>` element (https://developer.mozilla.org/en-US/docs/Learn/Forms/Your_first_form) to capture the tasks the user enters. HTML forms can be quite complex and sophisticated but, at their most basic, consist of opening and closing `<form>` tags that enclose one or more `<input>` elements where users can enter information, and a way to submit the form. There are many types of input fields (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Input) to choose from; we use the `type` attribute to specify the one we want. For this lab, we are using two: a text field (`type="text"`) and a submit button (`type="submit"`).

If you look in the `index.html` file, you will see the following:

```
<form id="create-task-form" action="#" method="POST">
  <label for="new-task-description">Task description:</label>
  <input
    type="text"
    id="new-task-description"
```

```
      name="new-task-description"
      placeholder="description"
    />
    <input type="submit" value="Create New Task" />
  </form>
```

Now take a look at the page in your browser. The rendered form looks like this:

You can see each of the components that are in our form's HTML:

1. the label for our input field ("Task description:")
2. the input box, with the placeholder content "description", and
3. the button that's created by the `submit` input tag

Let's take a closer look at the opening `<form>` tag. You'll see it includes an (optional) `id` attribute and two other attributes:

```
<form id="create-task-form" action="#" method="POST"></form>
```

Because HTML forms were designed to be handled by backend programming languages such as PHP, the `action` attribute would normally contain a path to the backend code that processes the data captured from the user. Because we will be handling the form using JavaScript, we don't need to provide a path. By convention, we set that attribute to `"#"` .

The `method` attribute specifies the *type* of action we're executing when the form is submitted. The `method` attribute's value (in this case, "POST") is an *HTTP Verb*. (Although it is not required, you will often see HTTP verbs in all caps.) We will learn more about HTTP Verbs in the next section. For now, just know that the `POST` method is used when we want to capture the data submitted by our form and use it in some way.

By default, the HTML `<form>` element submits the form and redirects the browser to a new url when the `<submit>` button is clicked. This default behavior makes sense when form submission is being handled by a back-end programming language. However, this *is not* the experience we want to build in this lab. We instead want to handle the submission of the form using JavaScript and update the DOM without reloading the page. Therefore, we need to prevent that event from performing its default behavior.

Suppress a Default Action with `Event.preventDefault()`

The deliverables for this lab require you to use JavaScript to handle the clicking of the submit button. To do this, you'll need to listen for a `submit` event on the `<form>` element. In order to *prevent* the *default* behavior of the `submit` event, when our event listener "sees" the event, it needs to invoke the `preventDefault()` method on it.

Take a look at the **MDN Documentation on `Event.preventDefault()` (https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault)**. You'll see how JavaScript is used to prevent a form element (checkbox) from doing its *default* behavior (appearing checked upon click). You'll want to prevent `submit` from doing its default behavior in a similar fashion.

Stretch Deliverables

Once you've got the required deliverables working, you may want to try to implement one or more of the following:

- A delete function that will remove tasks from your list
- A priority value selected from a **dropdown (https://www.w3docs.com/learn-html/html-select-tag.html)** that is used to determine the color of the text in the list (e.g. red for high priority, yellow for medium, green for low)
  - As an additional challenge, implement a sorting functionality that displays the tasks in ascending or descending order based on priority
- An additional input field (e.g. user, duration, date due)
- Ability to edit tasks
- Something of your choice! The main objective is to add a feature that allows the user's input to affect the DOM