Adding Tasks to Columns

Hope you've been following everything up till now. Next, we'll add the functionality of creating new tasks!

Exercise:

Create a `+` button inside the column header. When this `+` is pressed, open a modal window with the same form fields as the ones at the bottom of the page. When submitting the form, the new card should appear inside the column where it was created. Use the rmodal library!

As the form at the bottom of the page is now useless, remove it.

Source code:

Use the PomodoroTracker4 folder as a starting point. The result is in PomodoroTracker5.

Solution:

This is a user experience exercise involving a third party library. Let's install rmodal using npm. Navigate to the folder of the application, then execute the following command:

```
npm i rmodal --save
```

This command installs Bootstrap using the Node Package Manager, and stores it inside the `node_modules` folder. We can find the rmodal JavaScript and CSS file in the `dist` folder of `rmodal`. Let's add them to the head of our `index.html` file:

```
<link rel="stylesheet"
      href="node_modules/rmodal/dist/rmodal.css" type="text/css" />
<script type="text/javascript"
        src="node_modules/rmodal/dist/rmodal.js"></script>
```

Let's prepare the form in the HTML file by wrapping the form with three nested `div` elements. The `div` element classes and the `form-horizontal` class can be found in the documentation.

As we are planning to open the modal window from the column we want to add the card to, the column chooser dropdown has to be removed from the form. We will, therefore, change the column chooser dropdown list to a hidden input field.

```
<div id="form-modal" class="modal">
    <div class="modal-dialog animated">
        <div class="modal-content">
            <form class="js-add-task  form-horizontal"
                  action="javascript:void(0)">
                <input type="text"
                       name="task-name"
                       class="js-task-name"
                       placeholder="Task Name" />
                <select name="pomodoro-count"
                        class="js-pomodoro-count">
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                </select>
                <input type="hidden"
                       class="js-column-chooser"
                       name="column-chooser" />
                <input type="submit" class="js-add-task-submit" />
                <button class="js-add-task-cancel">Cancel</button>
            </form>
        </div>
    </div>
</div>
```

Notice the `js-add-task-submit` and `js-add-task-cancel` classes. The first submits the form, adds the task, and closes the modal window. The second simply closes the modal window without any changes.

We also need to add some manual styling. Let's remove the old form styles:

```
form {
    padding: 3rem;
    background-color: #ddd;
}
```

To override the default styles of the modal window, we have to target the following class:

```
.modal .modal-dialog {
    position: fixed !important;
    width: 400px !important;
    top: 50px;
    left: calc( 50% - 250px);
    padding: 50px;
    background-color: #eee;
    border: 3px #444 solid;
}

.modal .modal-dialog input {
    margin: 5px;
}
```

The fixed position makes sure that it appears on top of our screen in the coordinates specified by `top` and `left`, regardless of where we scroll horizontally.

Now it's time to handle the form from the JavaScript file.

```
const modalAddTaskCancel =
    document.querySelector( '.js-add-task-cancel' );

const addCardModal = new RModal(
    document.getElementById('form-modal'), {}
);

const columnTemplate = ( { header } ) => `
    <div class="task-column">
        <div class="task-column__header">
            ${ header }
            <span class="task-controls__icon  js-task-create"
                  data-column-index="${header}">\u{2795}</span>
        </div>
        <div class="task-column__body  js-${ header }-column-body"
             data-name="${ header }">
        </div>
    </div>
`;

const openCreateTaskModal = columnLabel => {
    const columnIndex = columns.indexOf( columnLabel );
    document.querySelector( '.js-column-chooser' ).value = columnIndex;
    addCardModal.open();
}

modalAddTaskCancel.addEventListener( 'click', e => {
    e.preventDefault();
    addCardModal.close();
} );
```

First, we get a handle for the cancel button so that we can add a click listener to it later.

The `addCardModal` constant contains the `RModal` instance created using our third-party library.

To open the modal window, we have to place a `js-task-create` icon to the header template. Whenever we click the `+` button on the header, the modal appears on a screen. The modal window will be opened by the `openCreateTaskModal` function, which will be called from the click event handler callback.

Finally, the modal window will be closed from by the click event handler attached to `modalAddTaskCancel`. Notice that we called the `preventDefault` method of the click event. This is because we want to avoid submitting the form.

The `handleTaskButtonClick` event handler monitors click events on the whole pomodoro board. Therefore, we can just call `openCreateTaskModal` from there:

```
const handleTaskButtonClick = function( event ) {
    const classList = event.target.className;
    const taskId = event.target.dataset.id;
    const columnIndex = event.target.dataset.columnIndex;

    /js-task-done/.test( classList ) ?
        finishTask( taskId, columnIndex ) :
    /js-increase-pomodoro/.test( classList ) ?
        increasePomodoroDone( taskId, columnIndex ) :
    /js-delete-task/.test( classList ) ?
        deleteTask( taskId, columnIndex ) :
    /js-task-create/.test( classList ) ?
        openCreateTaskModal( columnIndex ) :
    null;

    saveAndRenderState();
}
```

Finally, in the `addTask` submit event callback, we have to close the modal window:

```
const addTask = function( event ) {
    event.preventDefault();
    addCardModal.close();
    // ...
}
```

As `.task.selected` is a stronger selector than `.task`, the background color will be overridden automatically to the value inside `.task.selected`.