Selecting Cards

Add the functionality of selecting a particular task in your app.

Hint: a 'selected' property.

Exercise:

Make your cards selectable by clicking on them. Whenever you select a card, you have to deselect all other cards.

Source code:

Use the PomodoroTracker5 folder as a starting point. The end result is in PomodoroTracker6.

Solution:

The main purpose of this exercise is to check whether the candidate has a tendency of overcomplicating things. The solution of this exercise is fairly simple.

A `selected` class will take care of the styling of the selected task card:

```
.task.selected {
    background-color: #cce;
}
```

We can model the selected state by adding a `selected` property to the tasks. The absence of the property means that the card is not selected. When the property has a truthy value, the card is selected. Given only one card can be selected at a time, we select a card by deselecting all other cards, and then changing the selected state of the new card to `true`:

```
const deselectAllTasks = () => {
```

```
    for ( let column of board ) {
        for ( let task of column.tasks ) {
            delete task.selected;

        }
    }
}

const selectTask = (taskId, columnIndex) => {
    deselectAllTasks();
    board[ columnIndex ].tasks[ taskId ].selected = true;
}
```

Notice that we use the `board` global state, and we use `for...of` loops to drill down to the task level.

We are almost done. We only need to call the `selectTask` function in the click event handler. You might remember that event handling was implemented in the `handleTaskButtonClick` function.

```
const handleTaskButtonClick = function( { target } ) {
    const classList = target.className;
    const taskId = target.dataset.id;
    const columnIndex = 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 ) :
    /js-task/.test( classList ) ?
        selectTask( taskId, columnIndex ) :
    null;

    saveAndRenderState();
}
```

We did two things in the code. First, we simplified the `event.target` reference to `target` by applying destructuring in the function argument of `handleTaskButtonClick`. The code `{ target }` unwraps the event object, reads its `target` property, and makes it available.

The second change is that we are looking for the `/js-task/` pattern to identify when we click on a card.

If you test the application, you can conclude that sometimes clicking succeeds, while other times, you cannot change the selected state of cards. This is

because you have to click on the card `div` with the `.js-task` reference class. There are other `span` elements inside the card starting with `task__`. Once you click on those elements, nothing happens. Notice that these `div` elements are direct children of the card. Therefore, we can conclude that if the `target` DOM node has a class name starting with `task__`, we have to select the parent node of the card:

```
const handleTaskButtonClick = function( { target } ) {
    if ( /task__/.test( target.className ) ) {
        target = target.parentNode;
    }
    const classList = target.className;
    const taskId = target.dataset.id;
    const columnIndex = 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 ) :
    /js-task/.test( classList ) ?
        selectTask( taskId, columnIndex ) :
    null;

    saveAndRenderState();
}
```

If you test the application, everything works smoothly. There is only one major mistake in the code, namely, we violated separation of concerns. The `task__` prefixed classes are supposed to be used for styling only, and we attached functionality to them. What happens if a different styling team decides on renaming these classes? Most likely, our code will stop working.

Therefore, we have to add a `js-` prefixed class to the nodes we want to match in the regular expression. The class `js-task-child` will do:

```
const cardTemplate = ( { task, id, columnIndex } ) => `
    <div class="task  js-task ${ task.selected ? 'selected' : '' }"
         data-id="${id}"
         data-column-index="${columnIndex}">
        <span class="task__name  js-task-child">
            ${task.taskName}
        </span>
        <span class="task__pomodori  js-task-child">
            ${task.pomodoroDone} / ${task.pomodoroCount} pomodori
        </span>
        <div class="task__controls  js-task-child">
```

```
        ${ task.finished ? 'Finished' : `
            <span class="task-controls__icon  js-task-done"
                  data-id="${id}"
                  data-column-index="${columnIndex}">\u{2714}</span>
            <span class="task-controls__icon  js-increase-pomodoro"
                  data-id="${id}"
                  data-column-index="${columnIndex}">\u{2795}</span>`
        }
            <span class="task-controls__icon  js-delete-task"
                  data-id="${id}"
                  data-column-index="${columnIndex}">\u{1f5d1}</span>
        </div>
    </div>
`;
```

We have to apply the corresponding change in the `handleTaskButtonClick` function as well:

```
const handleTaskButtonClick = function( { target } ) {
    if ( /js-task-child/.test( target.className ) ) {
        target = target.parentNode;
    }
    const classList = target.className;
    const taskId = target.dataset.id;
    const columnIndex = 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 ) :
    /js-task/.test( classList ) ?
        selectTask( taskId, columnIndex ) :
    null;

    saveAndRenderState();
}
```

Now we are done. Even though the task was simple, there were a lot of pitfalls that had to be avoided.