4.5.7 Optimize the Code (Optional)

Developers deal with technical debt all the time. Because we don't always see the debt growing as we code, we should review the codebase later for areas of possible optimization.

The Taskinator technical debt isn't so huge that it would be considered terrible practice, but we should build these habits when we have less code to worry about.

Let's compare `createTaskEl()` and the `for` loop in `loadTasks()`. These steps look so similar that maybe we could just run `createTaskEl()` for each task object in `tasks`.

Because we'll remove new, working code, start by ensuring it's saved. Use the add, commit, and push commands to get the code up to GitHub.

After saving the working codebase, let's start updating the code for `loadTasks()`. We'll remove the `for` loop, as we don't need that code anymore. Now `loadTasks()` should look like this:

```
var loadTasks = function() {
  tasks = localStorage.getItem("tasks");

  if (!tasks) {
```

```
    tasks = [];
    return false;
  }

  tasks = JSON.parse(tasks);
}
```

Next, we don't want to use the `tasks` variable to store the tasks coming back from `localStorage`; we'll send one task object at a time to `createTaskEl()`, and the `tasks.push()` at the bottom will do it for us. So let's change the code in `loadTasks()` to look like this:

```
var loadTasks = function() {
  var savedTasks = localStorage.getItem("tasks");

  if (!savedTasks) {
    return false;
  }

  savedTasks = JSON.parse(savedTasks);
}
```

We're using the same actions as before, but we don't have to set `tasks` to an empty array anymore since we're not interfacing with it at all here.

We'll be done with `loadTasks()` after one more line of code. Add this to the bottom of the function under the `JSON.parse()`:

```
// loop through savedTasks array
for (var i = 0; i < savedTasks.length; i++) {
  // pass each task object into the `createTaskEl()` function
  createTaskEl(savedTasks[i]);
}
```

Now save `script.js` and try testing this code in the browser. It should work like before, but now we're using `createTaskEl()` for creating both a

brand-new task from the form and an existing task from `loadTasks()`!

Since the actions in `createTaskEl()` are identical to those we just deleted in `loadTasks()`, we can simply loop over each task in `savedTasks` and send its data to `createTaskEl()` as an argument!

Sometimes the hardest part of alleviating technical debt is just spotting it. Once you spot it, you have to decide whether it's worth fixing. You may decide not to fix it, if lots of other functions tied to the code getting refactored will have to be refactored too. But that depends on how much code needs to change.

Congratulations! You did a great job. Little things like this may not affect users, but you've simplified the code to make maintenance easier down the road.