5.1.7 Add Ability to Edit Task Dates

You've successfully updated Taskmaster to allow users to edit task descriptions, but editing the due date remains. Fortunately, the process will be very similar. Due dates are wrapped in `<span>` elements that are children of the same `.list-group`, meaning we can delegate the click the same way we did for `<p>` elements.

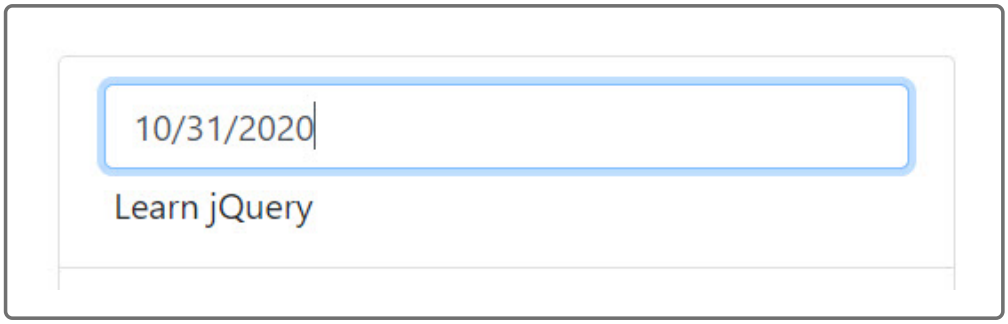
Add the following event listener and logic to `script.js`:

```
// due date was clicked
$(".list-group").on("click", "span", function() {
  // get current text
  var date = $(this)
    .text()
    .trim();

  // create new input element
  var dateInput = $("<input>")
    .attr("type", "text")
    .addClass("form-control")
    .val(date);

  // swap out elements
  $(this).replaceWith(dateInput);

  // automatically focus on new element
```

```
  dateInput.trigger("focus");
});
```

The main difference here is that we're creating an `<input>` element and using jQuery's `attr()` method to set it as `type="text"`. In jQuery, `attr()` can serve two purposes. With one argument, it gets an attribute (e.g., `attr("id")`). With two arguments, it sets an attribute (e.g., `attr("type", "text")`).

Test the app in the browser again, ensuring that due dates turn into text inputs when clicked, as the following screenshot demonstrates:



Next, we'll convert them back when the user clicks outside (i.e., when the element's blur event occurs).

Add the following event listener underneath the previous click listener:

```
// value of due date was changed
$(".list-group").on("blur", "input[type='text']", function() {
  // get current text
  var date = $(this)
    .val()
    .trim();

  // get the parent ul's id attribute
  var status = $(this)
    .closest(".list-group")
    .attr("id")
    .replace("list-", "");
```

```
  // get the task's position in the list of other li elements
  var index = $(this)
    .closest(".list-group-item")
    .index();

  // update task in array and re-save to localstorage
  tasks[status][index].date = date;
  saveTasks();

  // recreate span element with bootstrap classes
  var taskSpan = $("<span>")
    .addClass("badge badge-primary badge-pill")
    .text(date);

  // replace input with span element
  $(this).replaceWith(taskSpan);
});
```

Again, this due date logic looks similar to the task description logic. The important takeaway here is how we chain jQuery methods together to quickly accomplish certain tasks. Also, some jQuery methods are quite flexible, like `addClass()`, which can add one or multiple class names at a time.

jQuery's flexibility and syntax may confuse you at first, but many developers over the years have leaned on this time-tested library. Newer JavaScript frameworks and updates to JavaScript itself are slowly rendering jQuery unnecessary, but its previous popularity means a lot of legacy code still uses it.

There's a good chance you'll see jQuery again in the future, so we'll continue to practice it in the next few lessons. For now, test Taskmaster again to make sure the due date editing works. If anything's amiss, check the DevTools console for errors. With jQuery, it can be easy to omit or add an extra `)` or `}` character somewhere!