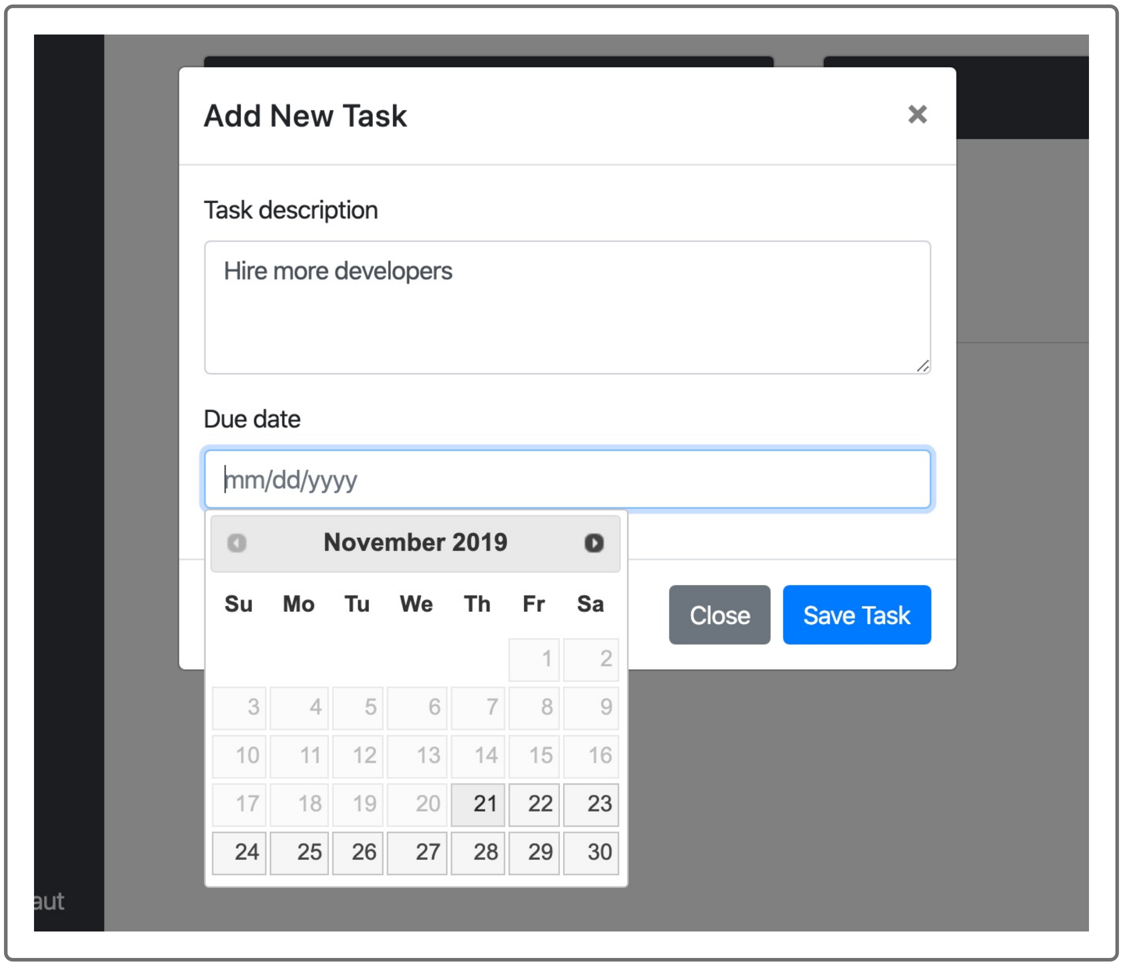
5.4.4 Add jQuery UI's Datepicker

The client envisions a date picker on the site looking something like this image:



We need to not only implement a date picker in the Add New Task form but also dynamically create one every time a user edits a due date. We could implement this feature in a few ways. For one, we could use the native HTML5 date picker—as simple as adding the `type="date"` attribute to the date field's `<input>` element. Although this solution would be ideal, not every browser supports that attribute just yet. We shouldn't optimize the app for only a portion of its user base.

We could use JavaScript and build this functionality ourselves, but we'd need to establish all the rules and logic. Alternatively, we've already installed jQuery UI for the drag and drop functionality, so there's no harm in using it for another feature.

We can see a demonstration of this date picker in the **demos section of jQuery UI (https://jqueryui.com/datepicker/)**, but that doesn't tell us exactly how to use it. We'll need to refer to the **Datepicker API documentation (https://api.jqueryui.com/datepicker/)** to understand how to implement it and what options we can set, just like the sortable UI functionality we added in the last lesson. Let's take a minute to peruse the documentation and review the available configuration settings.

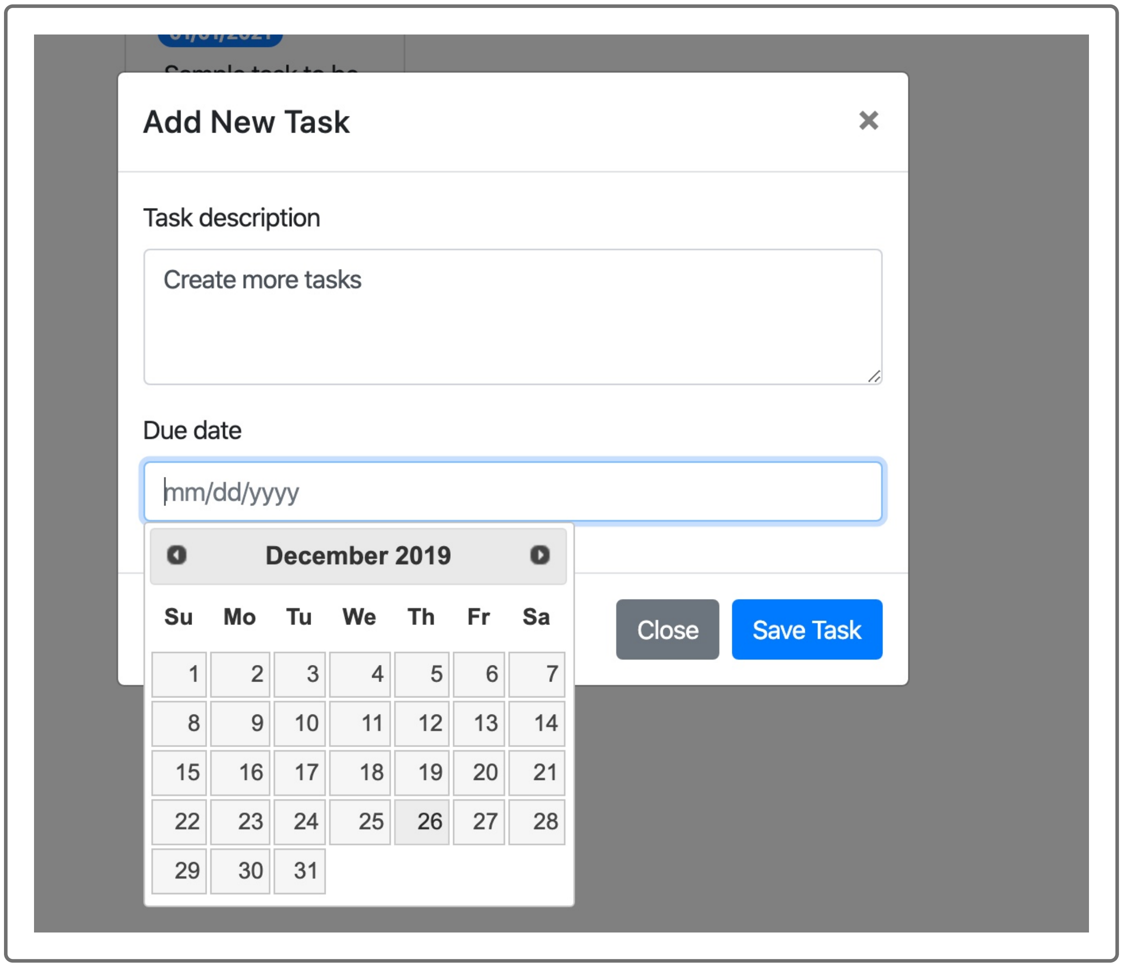
We'll add the date picker to the modal's date field first. To do that, we need to select the form `<input>` element with an `id` of `modalDueDate` using jQuery, then we'll chain the special `.datepicker()` method to it. Let's add the following code to the `script.js` file:

IMPORTANT

> If you have been using hard-coded `<li>` items in your HTML to expedite testing the app up to now, you must delete those items before continuing. The JavaScript that follows will not work as described if there are any hard-coded `<li>` items in the task lists in your HTML.

```
$("#modalDueDate").datepicker();
```

Save `script.js`, refresh the browser, and try to enter a date in the modal's form. It should look something like this image:



Once we select a date, the tool automatically fills in the value of the `<input>` element for a due date, even in the same format we want! So we have less configuration to worry about compared to the `.sortable()` method implemented earlier. But the client does want one particular setting in place: they want to prevent users from accidentally selecting due dates that have passed. We'll add an option called `minDate` to the `.datepicker()` method.

DEEP DIVE

DEEP DIVE

To learn more, see the [jQuery UI documentation on minDate (https://api.jqueryui.com/datepicker/#option-minDate)](https://api.jqueryui.com/datepicker/#option-minDate).

Update the `.datepicker()` method we just created to look like this code:

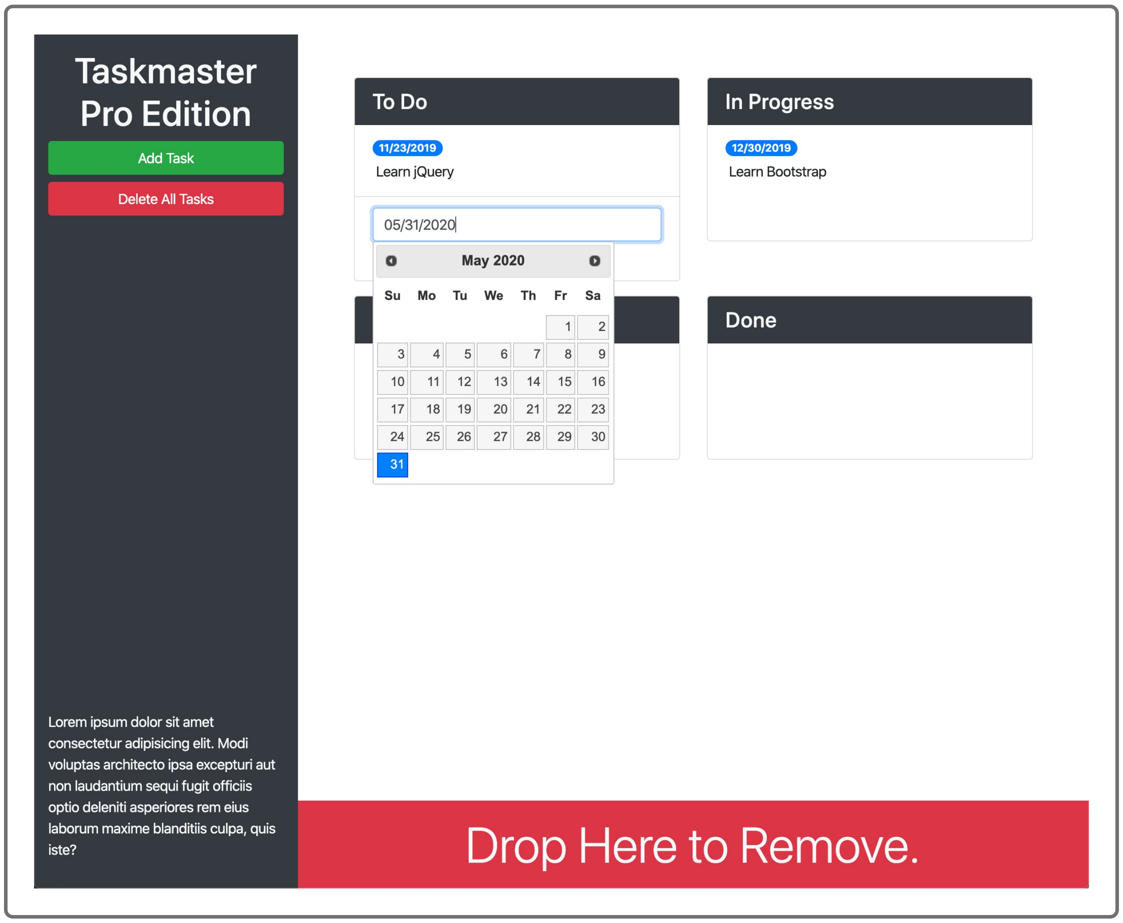
```
$("#modalDueDate").datepicker({
  minDate: 1
});
```

In the preceding code block, we passed an object into the `.datepicker()` method and set key-value pairs of the options we'd like to use. We used a value of `1` for the `minDate` option to indicate how many days after the current date we want the limit to kick in. In other words, we've set the minimum date to be one day from the current date.

Now if we try to select a date in the modal's form, any days from the current date or prior are grayed out to inform the user what they can or can't select. We could implement a lot of other options; some deal with the overall UI of the date picker while others deal with the behind-the-scenes logic. We're happy with what we have, so let's move on and implement the date picker for editing due dates.

Add Date Picker for Editing Dates

We've figured out the date picker functionality, as we have it working for the form in the modal; now we just need to use that same `.datepicker()` method to edit the date of a task. The result should allow us to use a date picker for each task, like this image displays:


Taskmaster Pro Edition
Add Task
Delete All Tasks
To Do
11/23/2019
Learn jQuery
05/31/2020
May 2020
In Progress
12/30/2019
Learn Bootstrap
Done
Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi voluptas architecto ipsa excepturi aut non laudantium sequi fugit officiis optio deleniti asperiores rem eius laborum maxime blanditiis culpa, quis iste?
Drop Here to Remove.

This step will be a little more involved than adding the modal form's date picker, as the date `<input>` elements we use for editing dates are dynamically created and aren't present in the HTML on page load.

PAUSE

When and where do we need to add the `.datepicker()` method to a due date edit `<input>` element?

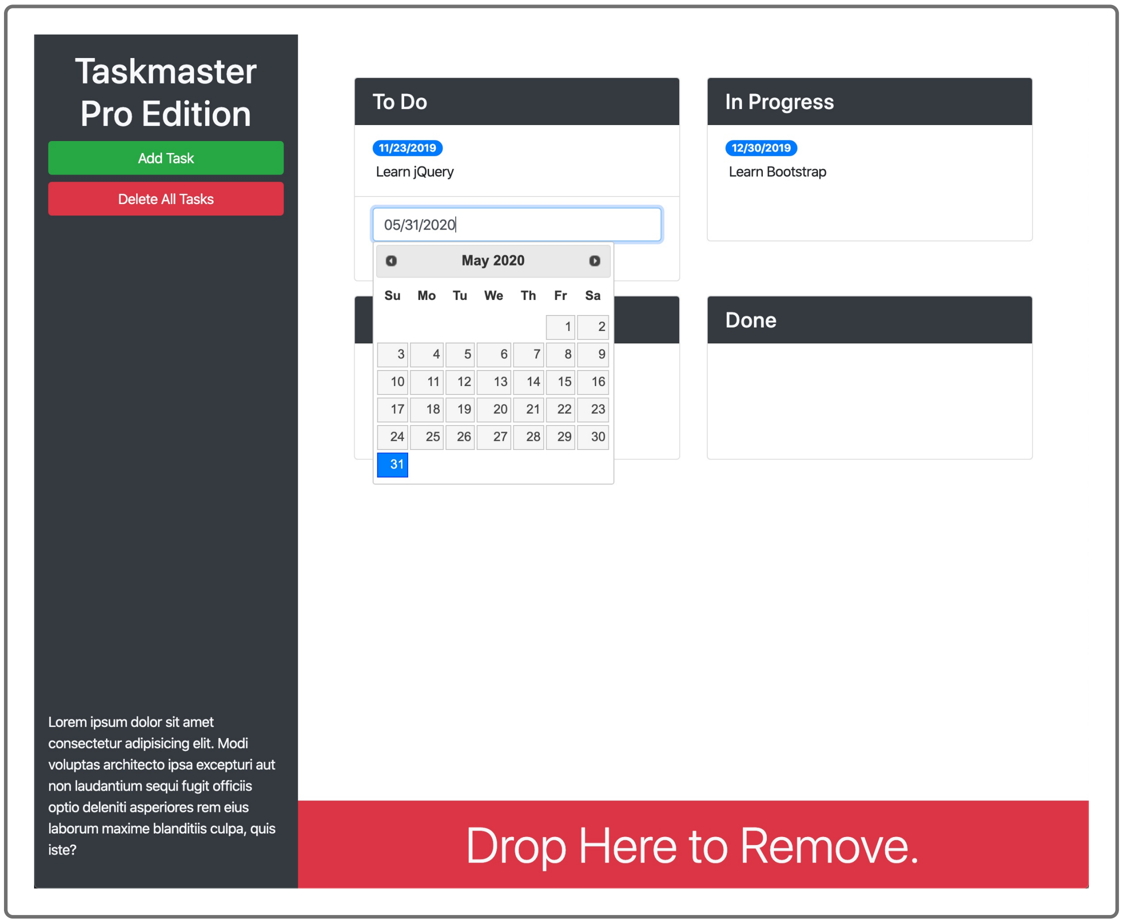
In the `click` event listener we added to any `<span>` element in `.list-group` to create a date `<input>` element.

Hide Answer

We'll need to create the date picker functionality every time a user clicks on the due date for a task. Let's find and update the `click` event listener attached to the element with a class of `list-group`, so that it looks like this:

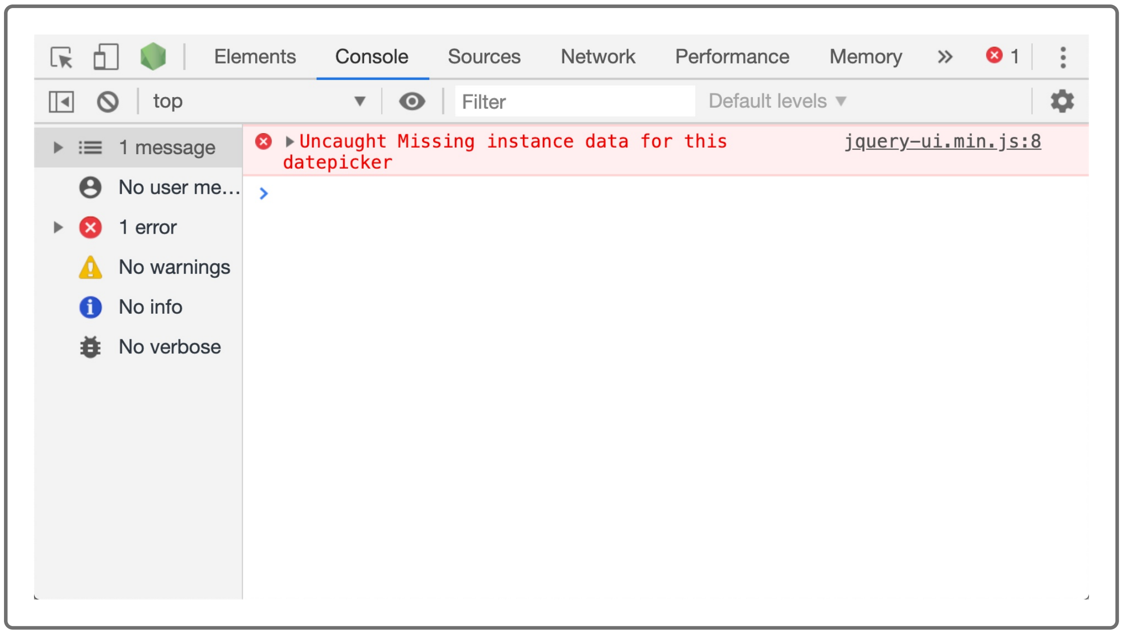
```
$(".list-group").on("click", "span", function() {
  // get current text
  var date = $(this).text().trim();

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

  $(this).replaceWith(dateInput);

  // enable jquery ui datepicker
  dateInput.datepicker({
    minDate: 1
  });

  // automatically bring up the calendar
  dateInput.trigger("focus");
});
```

Save `script.js` and refresh the page, then try to edit a task's due date. The result should look something like this image:



Taskmaster Pro Edition
Add Task
Delete All Tasks
To Do
11/23/2019
Learn jQuery
05/31/2020
May 2020
In Progress
12/30/2019
Learn Bootstrap
Done
Lorem ipsum dolor sit amet consectetur adipisicing elit. Modi voluptas architecto ipsa excepturi aut non laudantium sequi fugit officiis optio deleniti asperiores rem eius laborum maxime blanditiis culpa, quis iste?
Drop Here to Remove.

The date picker shows up just fine, but what happens when we try to select a new date? It doesn't seem to work! If we open the Chrome DevTools console tab, we'll see an error that looks something like this image:



```
Uncaught Missing instance data for this          jquery-ui.min.js:8
datepicker
```

The error here, `Missing instance data for this datepicker`, might seem tough to dissect at first. We're new to this library, and unfamiliar with its means of handling errors, but maybe other people have had this issue. A brief Google search might reveal that this error means the date picker closed without collecting the date information we selected. We need access to that data so that we can print it to the page!

To fix this we'll look at how we finalize a due date edit. Up till now, `blur` worked to let the browser know we were done editing a due date, because we wrote the date directly into the `<input>` element. Now, however, we use the date picker to populate that element. This sounds more like a `change` event than a `blur` event, so we just need to listen for a change instead.

Let's edit the blur event listener. All we need to do is change the opening line of the function so that it listens on "change" instead of on "blur", like this:

```
$(".list-group").on("change", "input[type='text']", function()
```

The code inside the function can remain as-is.

Notice how we only had to update the browser event from `blur` to `change`? Sometimes that's all it takes to fix an error! Let's save `script.js`, refresh the page, and try to update a task's due date again. This time it should work, but there's one more thing we should test.

What happens when we click off of the date picker because we've decided not to edit the due date? The `<input>` element remains and doesn't switch back to the actual date's `<span>` element, because we removed the `blur` event earlier!

Luckily, the jQuery UI Datepicker documentation has a solution for us. We can add a method to the `.datepicker()` settings to handle its unique `onClose` event. To read about usage, see the **jQuery UI Datepicker** documentation for the onClose event (https://api.jqueryui.com/datepicker/#option-onClose) .

Per the documentation, the `onClose` option for `.datepicker()` allows us to execute a function when the date picker closes. It may close when a user clicks anywhere on the page outside the date picker, so we need to capture that event and change the date picker back to its original `<span>` element.

Let's revisit the due date `click` event again and update the date picker's settings in the callback function's code:

```
// enable jquery ui datepicker
dateInput.datepicker({
  minDate: 1,
  onClose: function() {
    // when calendar is closed, force a "change" event on the `dateInp
    $(this).trigger("change");
  }
});
```

Now when we go to update a due date and decide to leave it as is, we can click off of the date picker and the `<span>` element will reappear with the original date. By adding the `onClose` method, we can instruct the `dateInput` element to trigger its own `change` event and execute the callback function tied to it.

Bugs like this occur often, especially when you're learning something new. Documentation typically explains how to use a functionality in isolation, which doesn't help when you're implementing it into an app with other functionality. The good thing is, there's typically a fix! Just exercise your slick googling skills and find how to use that fix.

HIDE PRO TIP

When an error or bug occurs in the code, chances are that other developers have faced that issue already and the answer exists somewhere on the internet.

When searching for that answer, copy and paste the error into Google and see what comes back. If the answer isn't satisfactory, expand by lightly describing the issue. Consider including the coding language, to filter out answers for other programming languages. But avoid mentioning the application's name or the file path that it occurs at; no other users will match those details, and a search will yield no results.