# 5.4.5 Add Moment.js to the Application

Adding due dates to tasks allows users to prioritize and plan, but users still have to manually review each task's due date. The client wants to highlight tasks two days out from the due date, thus enabling users to quickly identify looming deadlines.

We'll need to determine the amount of time between the current date and the due date. If that time is within two days, we'll attach a Bootstrap background color class to that task item so that the user can see it more easily.

---

## The JavaScript Date Object

JavaScript has built-in capabilities for working with date and time. For example, run the following code in the Chrome DevTools console:

```
var rightNow = new Date(); // don't worry about this "new" keyword yet

console.log(rightNow); // the result would look like a string...
console.log(typeof rightNow); // but its data type is an "object"
```

Running this code should return information about the current date and time in the first `console.log()` statement. Then in the second, when we check the data type, it'll return `"object"`.

A lot of things about the built-in JavaScript date functionality can be confusing. Why are we using the word `new`, and what's with the word `Date` being capitalized? Thankfully, we don't need to concern ourselves with understanding this part of JavaScript just yet. We only need to know that date and time functionality exists in the language and that some libraries make working with it a lot easier. **Moment.js.** is one of those libraries.

### NERD NOTE

Computer operating systems based on Unix have trouble telling time in terms of a calendar. Instead, time is measured in the number of seconds that have elapsed since 00:00:00 UTC on January 1, 1970. Libraries like Moment.js and even the native JavaScript `Date` object can convert that time into human-readable values, but under the hood it'll always involve some form of this "Unix time."

For more information, read about **Unix time on Wikipedia (https://en.wikipedia.org/wiki/Unix_time)** .

---

## Just a Moment

Just as jQuery makes DOM functionality available in JavaScript easier for developers to use, Moment.js is a JavaScript library that makes the built-in functionality for date and time easier to use. Let's take a look at the **documentation for Moment.js (https://momentjs.com/docs/)** and see what this library can do.

Moment.js can help us parse dates just like with JavaScript, but then we can manipulate that data more easily than with regular JavaScript. For instance, adding two days to the current time is as easy as running this code:

```
var twoDaysFromNow = moment().add(2, "days");
```

To compare, here's how to do the same thing using regular JavaScript:

```
// get current date
var currentDate = new Date();

// set how many days from now we want
var daysFromNow = 2;

// get date two days from now
var twoDaysFromNow = new Date(currentDate.setDate(currentDate.getDate(
```

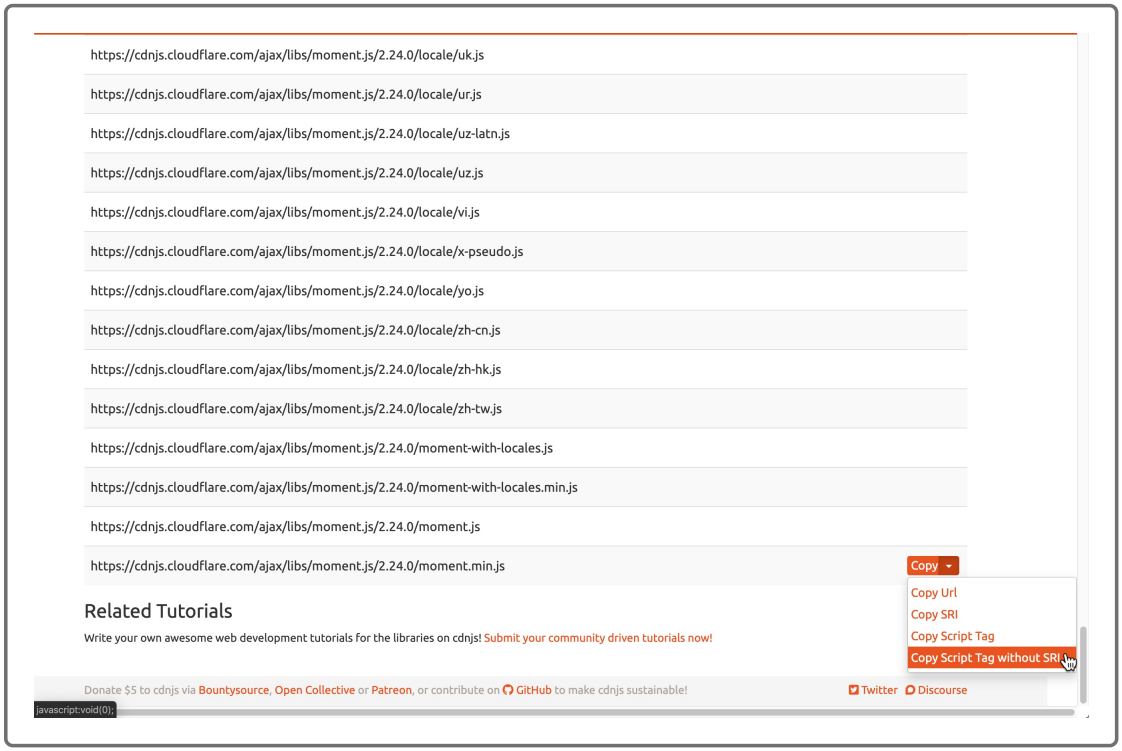
This code will get us to the same goal as the Moment.js method and may be preferable in certain situations, especially ones where we can't use external libraries like Moment.js. This is often the case in companies that prioritize security.

DEEP DIVE

**DEEP DIVE**

Even though we'll use Moment.js for most of our date and time needs, you can still learn more about how native dates work from the **MDN web docs for the JavaScript Date**

Most of the options on this list deal with how to handle date and time values in different parts of the world. We don't need to worry about any of these; let's just concern ourselves with the very last one, ending with `moment.min.js`. Move the cursor to it, and to the right there should be a button with a little down arrow. Click on that arrow to reveal a menu that looks like this image:




Select the option that says Copy Script Tag without SRI to get a copy of the `<script>` element for Moment.js. Let's paste it into the `index.html` file, right before the `<script src="./assets/js/script.js"></script>` element. If there's an issue with what was copied from the Moment.js CDN, use this code:

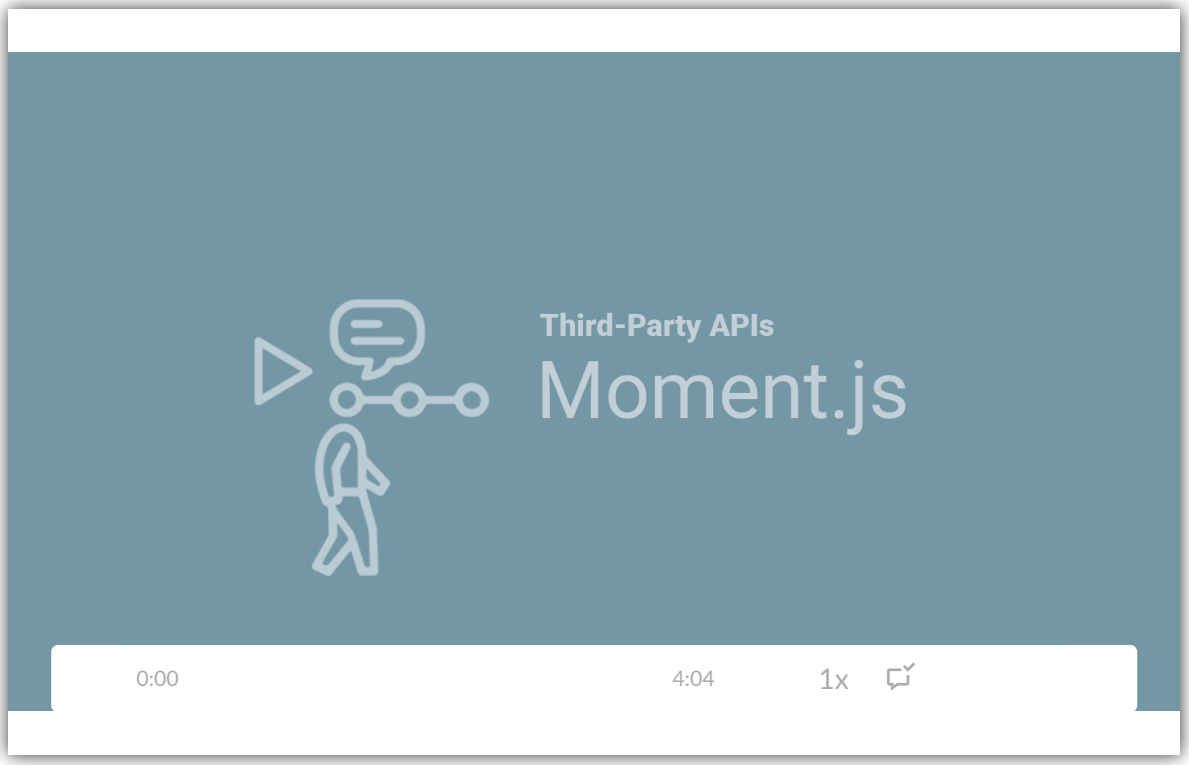
```
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.24.0/m
```

Let's test it. Save `script.js` and refresh the page. Then open the Chrome DevTools console and type the following:

**Object** (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date).

When using Moment.js, we just have to execute the `moment()` function with no arguments to get a Moment.js date object for the exact time of execution. From there, we can chain a number of methods to it and manipulate that date object's value.

Check out the following video for other examples of some of the methods `moment()` provides:



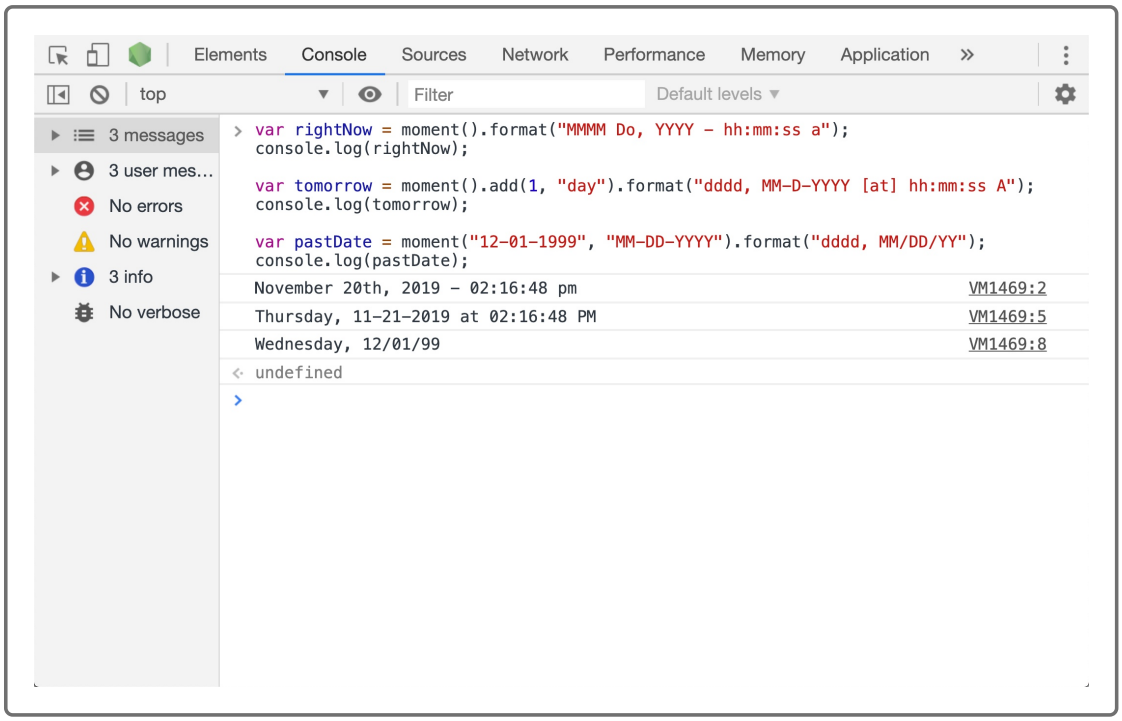

We'll need to install Moment.js into the application before we can work with any of its functionality, so let's go ahead and do that now. The documentation provides instructions under **"Where to use it"** (https://momentjs.com/docs/#/use-it/browser/), but it points us to a CDN to get the link. So let's go to **the Moment.js CDN** (https://cdnjs.com/libraries/moment.js).

```
var rightNow = moment().format("MMMM Do, YYYY - hh:mm:ss a");
console.log(rightNow);

var tomorrow = moment().add(1, "day").format("dddd, MM-D-YYYY [at] hh:
console.log(tomorrow);

var pastDate = moment("12-01-1999", "MM-DD-YYYY").format("dddd, MM/DD/
console.log(pastDate);
```

The result should look something like this image:




Not only did we just ensure Moment.js is loaded in the application, but we also ran through a few examples of how to use it.

First, we formatted the current date and time information to print in a certain way, using the `.format()` method. In the second example, we added a day to the current date and time, then formatted it in a different way.

In the third, we wanted information about a specific date, not just right now, so we passed that date as an argument into the `moment()` function

and then provided the format the first argument is being inserted as. This way, the `moment()` function knows that we're looking for "month/day/year" format and not "day/month/year" format.

DEEP DIVE

**DEEP DIVE**

---

All of this information can be found in the Moment.js documentation:

- **Parsing dates with Moment.js (https://momentjs.com/docs/#/parsing/)**
- **Formatting dates for displaying in Moment.js (https://momentjs.com/docs/#/displaying/)**
- **Manipulating dates in Moment.js (https://momentjs.com/docs/#/manipulating/)**

Now that we know it works in the app, let's add task auditing to Taskmaster Pro!