**JavaScript Date Format – How to Format a Date in JS**

JavaScript is one of the three fundamental web technologies you'll use when developing websites or web applications.

When creating these web pages, you will, at some point, likely need to use dates for some reason – such as displaying when something was uploaded, stored, and so on.

In this article, you will learn how to format dates in JavaScript and see how you can do this with a popular JavaScript date library known as moment.js.

**How to Get Dates in JavaScript**

In JavaScript, you use either the new Date() or Date() constructor to get your dates (either current date or a specific date).

The new Date() constructor returns a new Date object, while the Date() constructor returns a string representation of the current date and time.

Explain

let stringDate = Date();

console.log(stringDate); // "Tue Aug 23 2022 14:47:12 GMT-0700 (Pacific Daylight Time)"

let objectDate = new Date();

console.log(objectDate); // Tue Aug 23 2022 14:47:12 GMT-0700 (Pacific Daylight Time)

These full format dates comprise the day, month, year, minute, hour, and other information you don't need all the time.

Your primary concern is to see how you can format these date values to return dates in readable formats, which you can embed on your web page for everyone to understand.

**How to Format Dates in JavaScript**

Formatting dates depends on you and your needs. In some countries, the month comes before the day, then the year (06/22/2022). In others, the day comes before the month, then the year (22/06/2022), and lots more.

Regardless of the format, you want to break down the date object value and get the necessary information you want for your web page.

This is possible with the JavaScript date methods. There are many of these methods, but since you are only concerned with dates in this article, you will only need three of them:

* getFullYear() – You will use this method to get the year as a four-digit number (yyyy). For example, 2022.
* getMonth() – You will use this method to get the month as a number between 0-11, with each number representing the months from January to December. For example, 2 will be the index for March since it's zero-based indexing (meaning it starts from 0).
* getDate() – You will use this method to get the day as a number between 1-31 (this is not an index, but the exact day value, so it starts from 1 not 0).

**Note:** These methods can only be applied or will only work with the new Date() constructor, which returns a date object.

Explain

let objectDate = new Date();

let day = objectDate.getDate();

console.log(day); // 23

let month = objectDate.getMonth();

console.log(month + 1); // 8

let year = objectDate.getFullYear();

console.log(year); // 2022

You will notice that 1 is added to the month value above. This is because the month is 0 indexed. The value starts from 0. This means 7 will mean August instead of 8.

At this point, you have been able to extract all date values from the date object. You can now organize them in whatever format you desire:

Explain

let format1 = month + "/" + day + "/" + year;

console.log(format1); // 7/23/2022

let format2 = day + "/" + month + "/" + year;

console.log(format2); // 23/7/2022

let format3 = month + "-" + day + "-" + year;

console.log(format3); // 7-23-2022

let format4 = day + "-" + month + "-" + year;

console.log(format4); // 23-7-2022

In the above, you concatenated the values using the plus operator. You can also make use of template literals to concatenate:

Explain

let format1 = `${month}/${day}/${year}`;

console.log(format1); // 7/23/2022

let format2 = `${day}/${month}/${year}`;

console.log(format2); // 23/7/2022

let format3 = `${month}-${day}-${year}`;

console.log(format3); // 7-23-2022

let format4 = `${day}-${month}-${year}`;

console.log(format4); // 23-7-2022

Now, you have seen the possible ways you may want to format your date.

Another scenario can be if you want the month and day value to be preceded by 0 if it's a single numeric value (from 1-9). To do this, you would need to add a condition to handle this before formatting your dates:

Explain

if (day < 10) {

day = '0' + day;

}

if (month < 10) {

month = `0${month}`;

}

let format1 = `${month}/${day}/${year}`;

console.log(format1); // 07/23/2022

Interested in other ways you can format dates in JavaScript? Check out my article on "**How to Format Dates in JavaScript with One Line of Code**".

**How to Format Dates in JavaScript with Moment.js**

Moment.js is a JavaScript date and time library that you can use to quickly format your dates without handling the logic with so many lines of code.

To use this library, you have to install the package in your project using any of your preferred options in the documentation.

For this guide, you are only concerned with how you can format dates with Moment.js:

let date = moment().format();

console.log(date); // 2022-08-23T16:50:22-07:00

To format this date/time value, all you have to do is input your preferred format, and it will return the formatted date as seen below:

Explain

let dateFormat1 = moment().format('D-MM-YYYY');

console.log(dateFormat1); // 23-08-2022

let dateFormat2 = moment().format('D/MM/YYYY');

console.log(dateFormat2); // 23/08/2022

let dateFormat3 = moment().format('MM-D-YYYY');

console.log(dateFormat3); // 08-23-2022

let dateFormat4 = moment().format('MM/D/YYYY');

console.log(dateFormat4); // 08/23/2022

**Conclusion**

This article has taught you how to format dates in JavaScript, either from scratch or with the moment.js library.

Be mindful when using a library for small projects, because libraries increase the size of your project. This is because these libraries are designed to handle many more operations. But to use any minimal operation, you still must install the entire library.

It is always recommended that you perform simple operations like this from scratch. That is, unless you are forced to use the library, or the library has been installed, or you are working on a large-scale project that requires some complex formatting.