**JavaScript Tutorial: Date & Time In JavaScript | Web Development Tutorials #59**

In this tutorial, we are going to learn about the **date object.**We will learn how to set, initialize, and get the date through JavaScript. Make a new file as *tut59.html* and add the boilerplate to get the HTML template. Then give the title as **JavaScript Dates** under the <title> tag.

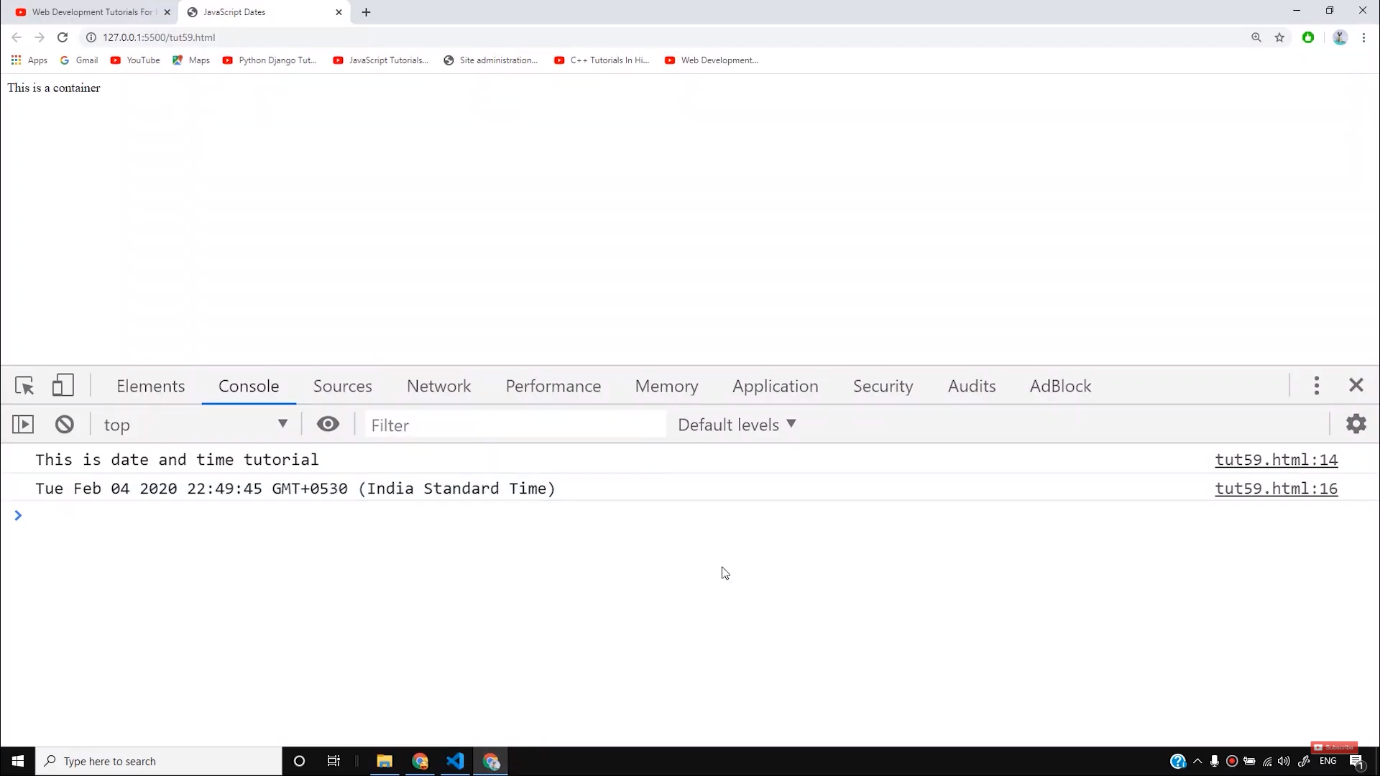
To add a date, we can write as-

let now = new Date();

console.log(now);

Copy

This is a constructor and *date* here is an object. The output from the above code that we get is as follows-



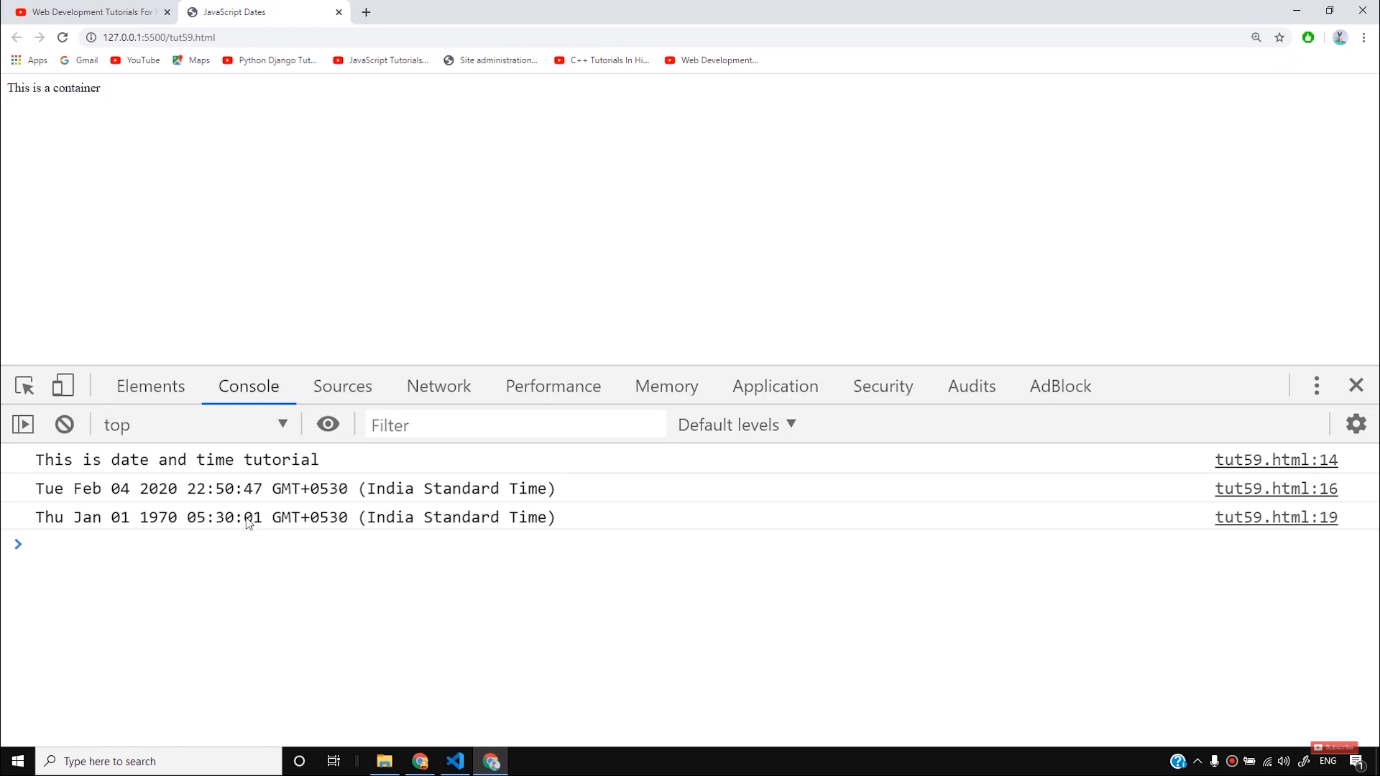
But if we write like this-

let dt = new Date(1000);

console.log(dt);

Copy

The output we get is as follows-



The value inside the date object gives the time after that particular milliseconds from where the reference is taken.

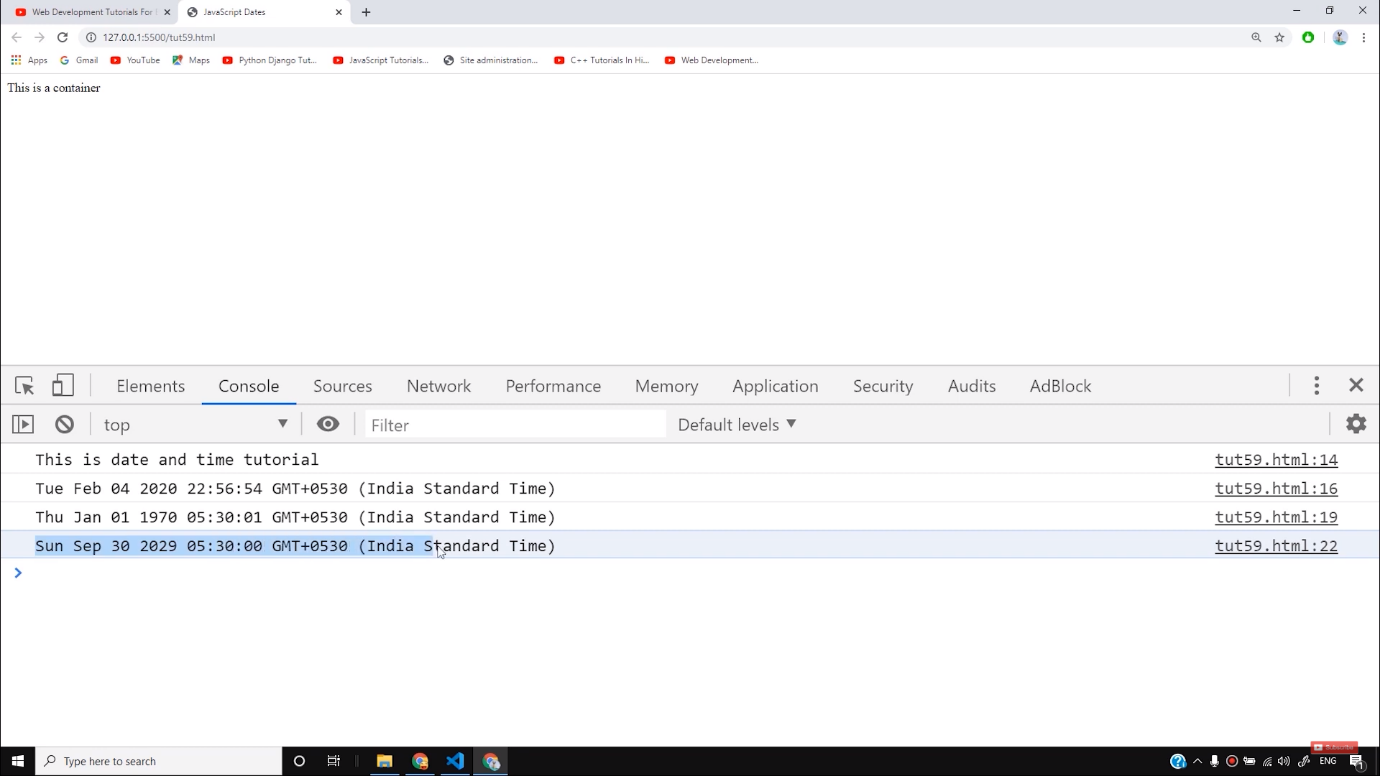
To initialize the date of any given time, we can write as follows-

let newDate = new Date("2029-09-30");

console.log(newDate)

Copy

After writing the above code, we get the output as-



However, there is another way to initialize the date. If we write as follows-

// let newDate = new Date(year, month, date, hours, minutes, seconds, milliseconds);

let newDate = new Date(3020, 4, 6, 9, 3, 2, 34);

console.log(newDate);

Copy

Through this constructor, we can initialize the date in the given format. There are some more functions which help us to know the year, month, date, and time directly. The functions are as follows-

let yr = newDate.getFullYear();

console.log("The year is ", yr);

let dat = newDate.getDate();

console.log("The date is ", dat);

let month = newDate.getMonth();

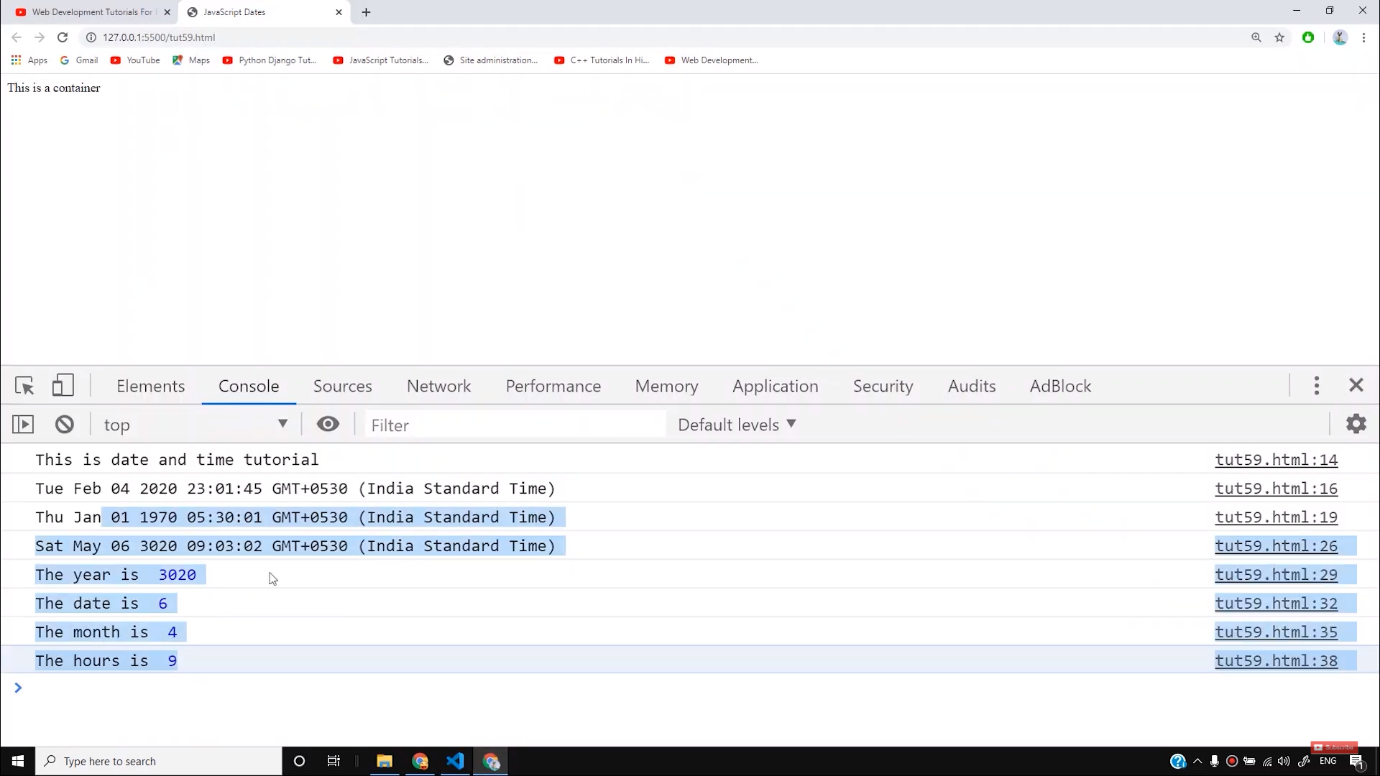
console.log("The month is ", month);

let hours = newDate.getHours();

console.log("The hours is ", hours);

Copy

If we write the above functions, the output we get is as follows-



We can also set date and time manually with the functions like *setDate()*and *newMinutes*as follows-

newDate.setDate(8);

newDate.setMinutes(29);

console.log(newDate)

Copy

If we write the value inside the *setDate()*function more than 31, then the JavaScript automatically treats it as a new date and gives it a new value starting from 1. For example, if we write the value as 32, then it will automatically be converted to 1 and so on.

If we want to add the clock on our website, we can do this with the help of the above functions. But before that, we have to add the HTML which goes like this-

<div class="container">

Current time is <span id="time"></span>

</div>

Copy

Then we will define the above functions and create an inner HTML as follows-

newDate.setDate(8);

newDate.setMinutes(29);

console.log(newDate)

setInterval(updateTime, 1000);

function updateTime() {

time.innerHTML = new Date();

}

Copy

If we modify it using some CSS as follows-

.container {

font-size: 23px;

background-color: blanchedalmond;

border: 2px solid grey;

padding: 34px;

margin: 4px;

text-align: center;

}

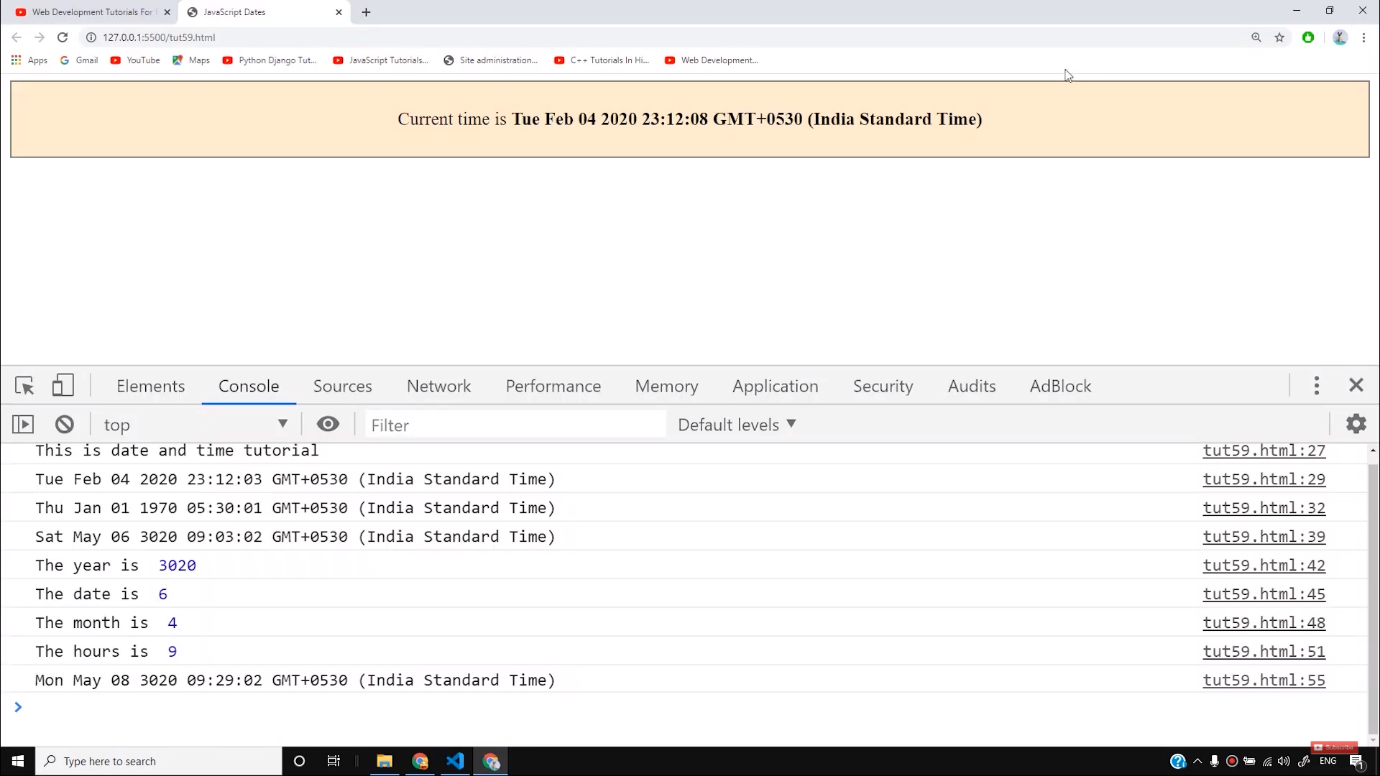
#time {

font-weight: bold;

}

Copy

Then the output will look as-



So I hope you must have understood how to use date and time functions on our website. Now you can try various ways to add date and time on your own website and make it look interesting.

**Code as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript Dates</title>

</head>

<style>

.container {

font-size: 23px;

background-color: blanchedalmond;

border: 2px solid grey;

padding: 34px;

margin: 4px;

text-align: center;

}

#time {

font-weight: bold;

}

</style>

<body>

<div class="container">

Current time is <span id="time"></span>

</div>

<script>

console.log("This is date and time tutorial");

let now = new Date();

console.log(now);

let dt = new Date(1000);

console.log(dt);

// let newDate = new Date("2029-09-30");

// console.log(newDate)

// let newDate = new Date(year, month, date, hours, minutes, seconds, milliseconds);

let newDate = new Date(3020, 4, 6, 9, 3, 2, 34);

console.log(newDate);

let yr = newDate.getFullYear();

console.log("The year is ", yr);

let dat = newDate.getDate();

console.log("The date is ", dat);

let month = newDate.getMonth();

console.log("The month is ", month);

let hours = newDate.getHours();

console.log("The hours is ", hours);

newDate.setDate(8);

newDate.setMinutes(29);

console.log(newDate)

setInterval(updateTime, 1000);

function updateTime() {

time.innerHTML = new Date();

}

</script>

</body>

</html>

# JavaScript Date Objects

[❮ Previous](https://www.w3schools.com/js/js_array_const.asp)[Next ❯](https://www.w3schools.com/js/js_date_formats.asp)

JavaScript **Date Object** lets us work with dates:

**Sat Jun 18 2022 22:00:02 GMT+0530 (India Standard Time)**

[Year: 2022](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getfullyear) [Month: 6](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getmonth) [Day: 18](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getdate) [Hours: 22](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_gethours) [Minutes: 0](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getminutes) [Seconds: 2](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getseconds)

### **Example**

const d = new Date();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_current)

## **JavaScript Date Output**

By default, JavaScript will use the browser's time zone and display a date as a full text string:

**Sat Jun 18 2022 22:00:02 GMT+0530 (India Standard Time)**

You will learn much more about how to display dates, later in this tutorial.

## **Creating Date Objects**

Date objects are created with the new Date() constructor.

There are **4 ways** to create a new date object:

new Date()  
new Date(year, month, day, hours, minutes, seconds, milliseconds)  
new Date(milliseconds)  
new Date(date string)

## **new Date()**

new Date() creates a new date object with the **current date and time**:

### **Example**

const d = new Date();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new)

Date objects are static. The computer time is ticking, but date objects are not.

## **new Date(year, month, ...)**

new Date(year, month, ...) creates a new date object with a **specified date and time**.

7 numbers specify year, month, day, hour, minute, second, and millisecond (in that order):

### **Example**

const d = new Date(2018, 11, 24, 10, 33, 30, 0);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_all)

**Note:** JavaScript counts months from **0** to **11**:

**January = 0**.

**December = 11**.

Specifying a month higher than 11, will not result in an error but add the overflow to the next year:

Specifying:

const d = new Date(2018, 15, 24, 10, 33, 30);

Is the same as:

const d = new Date(2019, 3, 24, 10, 33, 30);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_month_12)

Specifying a day higher than max, will not result in an error but add the overflow to the next month:

Specifying:

const d = new Date(2018, 5, 35, 10, 33, 30);

Is the same as:

const d = new Date(2018, 6, 5, 10, 33, 30);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_day_35)

## **Using 6, 4, 3, or 2 Numbers**

6 numbers specify year, month, day, hour, minute, second:

### **Example**

const d = new Date(2018, 11, 24, 10, 33, 30);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers6)

5 numbers specify year, month, day, hour, and minute:

### **Example**

const d = new Date(2018, 11, 24, 10, 33);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers5)

4 numbers specify year, month, day, and hour:

### **Example**

const d = new Date(2018, 11, 24, 10);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers4)

3 numbers specify year, month, and day:

### **Example**

const d = new Date(2018, 11, 24);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers3)

2 numbers specify year and month:

### **Example**

const d = new Date(2018, 11);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers2)

You cannot omit month. If you supply only one parameter it will be treated as milliseconds.

### **Example**

const d = new Date(2018);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_numbers1)

## **Previous Century**

One and two digit years will be interpreted as 19xx:

### **Example**

const d = new Date(99, 11, 24);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_two)

### **Example**

const d = new Date(9, 11, 24);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_one)

## **new Date(dateString)**

new Date(dateString) creates a new date object from a **date string**:

### **Example**

const d = new Date("October 13, 2014 11:13:00");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_string)

Date strings are described in the next chapter.

## **JavaScript Stores Dates as Milliseconds**

JavaScript stores dates as number of milliseconds since January 01, 1970, 00:00:00 UTC (Universal Time Coordinated).

Zero time is January 01, 1970 00:00:00 UTC.

Now the time is: **1655569802937** milliseconds past January 01, 1970

## **new Date(milliseconds)**

new Date(milliseconds) creates a new date object as**zero time plus milliseconds**:

### **Example**

const d = new Date(0);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_zero)

01 January 1970 **plus** 100 000 000 000 milliseconds is approximately 03 March 1973:

### **Example**

const d = new Date(100000000000);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_millisec)

January 01 1970 **minus** 100 000 000 000 milliseconds is approximately October 31 1966:

### **Example**

const d = new Date(-100000000000);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_millisec_minus)

### **Example**

const d = new Date(86400000);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_new_day)

One day (24 hours) is 86 400 000 milliseconds.

ADVERTISEMENT

## **Date Methods**

When a Date object is created, a number of **methods** allow you to operate on it.

Date methods allow you to get and set the year, month, day, hour, minute, second, and millisecond of date objects, using either local time or UTC (universal, or GMT) time.

Date methods and time zones are covered in the next chapters.

## **Displaying Dates**

JavaScript will (by default) output dates in full text string format:

### **Example**

Sat Jun 18 2022 22:00:02 GMT+0530 (India Standard Time)

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_current)

When you display a date object in HTML, it is automatically converted to a string, with the toString() method.

### **Example**

const d = new Date();  
d.toString();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_tostring)

The toUTCString() method converts a date to a UTC string (a date display standard).

### **Example**

const d = new Date();  
d.toUTCString();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_toutcstring)

The toDateString() method converts a date to a more readable format:

### **Example**

const d = new Date();  
d.toDateString();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_todatestring)

The toISOString() method converts a Date object to a string, using the ISO standard format:

### **Example**

const d = new Date();  
d.toISOString();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_toisostring)

# JavaScript Date Formats

[❮ Previous](https://www.w3schools.com/js/js_dates.asp)[Next ❯](https://www.w3schools.com/js/js_date_methods.asp)

## **JavaScript Date Input**

There are generally 3 types of JavaScript date input formats:

|  |  |
| --- | --- |
| **Type** | **Example** |
| ISO Date | "2015-03-25" (The International Standard) |
| Short Date | "03/25/2015" |
| Long Date | "Mar 25 2015" or "25 Mar 2015" |

The ISO format follows a strict standard in JavaScript.

The other formats are not so well defined and might be browser specific.

## **JavaScript Date Output**

Independent of input format, JavaScript will (by default) output dates in full text string format:

Sun Jun 19 2022 09:40:53 GMT+0530 (India Standard Time)

## **JavaScript ISO Dates**

ISO 8601 is the international standard for the representation of dates and times.

The ISO 8601 syntax (YYYY-MM-DD) is also the preferred JavaScript date format:

### **Example (Complete date)**

const d = new Date("2015-03-25");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_iso1)

The computed date will be relative to your time zone.  
Depending on your time zone, the result above will vary between March 24 and March 25.

ADVERTISEMENT

## **ISO Dates (Year and Month)**

ISO dates can be written without specifying the day (YYYY-MM):

### **Example**

const d = new Date("2015-03");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_iso2)

Time zones will vary the result above between February 28 and March 01.

## **ISO Dates (Only Year)**

ISO dates can be written without month and day (YYYY):

### **Example**

const d = new Date("2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_iso3)

Time zones will vary the result above between December 31 2014 and January 01 2015.

## **ISO Dates (Date-Time)**

ISO dates can be written with added hours, minutes, and seconds (YYYY-MM-DDTHH:MM:SSZ):

### **Example**

const d = new Date("2015-03-25T12:00:00Z");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_iso4)

Date and time is separated with a capital T.

UTC time is defined with a capital letter Z.

If you want to modify the time relative to UTC, remove the Z and add +HH:MM or -HH:MM instead:

### **Example**

const d = new Date("2015-03-25T12:00:00-06:30");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_iso5)

UTC (Universal Time Coordinated) is the same as GMT (Greenwich Mean Time).

Omitting T or Z in a date-time string can give different results in different browsers.

## **Time Zones**

When setting a date, without specifying the time zone, JavaScript will use the browser's time zone.

When getting a date, without specifying the time zone, the result is converted to the browser's time zone.

In other words: If a date/time is created in GMT (Greenwich Mean Time), the date/time will be converted to CDT (Central US Daylight Time) if a user browses from central US.

## **JavaScript Short Dates.**

Short dates are written with an "MM/DD/YYYY" syntax like this:

### **Example**

const d = new Date("03/25/2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_7)

## **WARNINGS !**

In some browsers, months or days with no leading zeroes may produce an error:

const d = new Date("2015-3-25");

The behavior of "YYYY/MM/DD" is undefined.  
Some browsers will try to guess the format. Some will return NaN.

const d = new Date("2015/03/25");

The behavior of  "DD-MM-YYYY" is also undefined.  
Some browsers will try to guess the format. Some will return NaN.

const d = new Date("25-03-2015");

## **JavaScript Long Dates.**

Long dates are most often written with a "MMM DD YYYY" syntax like this:

### **Example**

const d = new Date("Mar 25 2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_1)

Month and day can be in any order:

### **Example**

const d = new Date("25 Mar 2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_2)

And, month can be written in full (January), or abbreviated (Jan):

### **Example**

const d = new Date("January 25 2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_4)

### **Example**

const d = new Date("Jan 25 2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_5)

Commas are ignored. Names are case insensitive:

### **Example**

const d = new Date("JANUARY, 25, 2015");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_string_6)

## **Date Input - Parsing Dates**

If you have a valid date string, you can use the Date.parse() method to convert it to milliseconds.

Date.parse() returns the number of milliseconds between the date and January 1, 1970:

### **Example**

let msec = Date.parse("March 21, 2012");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_parse)

You can then use the number of milliseconds to **convert it to a date** object:

### **Example**

let msec = Date.parse("March 21, 2012");  
const d = new Date(msec);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_convert)

# JavaScript Get Date Methods

[❮ Previous](https://www.w3schools.com/js/js_date_formats.asp)[Next ❯](https://www.w3schools.com/js/js_date_methods_set.asp)

These methods can be used for getting information from a date object:

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | Get the **year** as a four digit number (yyyy) |
| getMonth() | Get the **month** as a number (0-11) |
| getDate() | Get the **day** as a number (1-31) |
| getHours() | Get the **hour** (0-23) |
| getMinutes() | Get the **minute** (0-59) |
| getSeconds() | Get the **second** (0-59) |
| getMilliseconds() | Get the **millisecond** (0-999) |
| getTime() | Get the time (milliseconds since January 1, 1970) |
| getDay() | Get the weekday as a number (0-6) |
| Date.now() | Get the time. ECMAScript 5. |

## **The getTime() Method**

The getTime() method returns the number of milliseconds since January 1, 1970:

### **Example**

const d = new Date();  
d.getTime();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_gettime)

## **The getFullYear() Method**

The getFullYear() method returns the year of a date as a four digit number:

### **Example**

const d = new Date();  
d.getFullYear();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getfullyear)

ADVERTISEMENT

## **The getMonth() Method**

The getMonth() method returns the month of a date as a number (0-11):

### **Example**

const d = new Date();  
d.getMonth();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getmonth)

In JavaScript, the first month (January) is month number 0, so December returns month number 11.

You can use an array of names, and getMonth() to return the month as a name:

### **Example**

const months = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];  
  
const d = new Date();  
let month = months[d.getMonth()];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_month)

## **The getDate() Method**

The getDate() method returns the day of a date as a number (1-31):

### **Example**

const d = new Date();  
d.getDate();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getdate)

## **The getHours() Method**

The getHours() method returns the hours of a date as a number (0-23):

### **Example**

const d = new Date();  
d.getHours();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getHours)

## **The getMinutes() Method**

The getMinutes() method returns the minutes of a date as a number (0-59):

### **Example**

const d = new Date();  
d.getMinutes();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getMinutes)

## **The getSeconds() Method**

The getSeconds() method returns the seconds of a date as a number (0-59):

### **Example**

const d = new Date();  
d.getSeconds();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getSeconds)

## **The getMilliseconds() Method**

The getMilliseconds() method returns the milliseconds of a date as a number (0-999):

### **Example**

const d = new Date();  
d.getMilliseconds();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getmilliseconds)

## **The getDay() Method**

The getDay() method returns the weekday of a date as a number (0-6):

### **Example**

const d = new Date();  
d.getDay();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_getday)

In JavaScript, the first day of the week (0) means "Sunday", even if some countries in the world consider the first day of the week to be "Monday"

You can use an array of names, and getDay() to return the weekday as a name:

### **Example**

const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];  
  
const d = new Date();  
let day = days[d.getDay()];

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_weekday)

## **UTC Date Methods**

UTC date methods are used for working with UTC dates (Universal Time Zone dates):

|  |  |
| --- | --- |
| **Method** | **Description** |
| getUTCDate() | Same as getDate(), but returns the UTC date |
| getUTCDay() | Same as getDay(), but returns the UTC day |
| getUTCFullYear() | Same as getFullYear(), but returns the UTC year |
| getUTCHours() | Same as getHours(), but returns the UTC hour |
| getUTCMilliseconds() | Same as getMilliseconds(), but returns the UTC milliseconds |
| getUTCMinutes() | Same as getMinutes(), but returns the UTC minutes |
| getUTCMonth() | Same as getMonth(), but returns the UTC month |
| getUTCSeconds() | Same as getSeconds(), but returns the UTC seconds |

# JavaScript Set Date Methods

[❮ Previous](https://www.w3schools.com/js/js_date_methods.asp)[Next ❯](https://www.w3schools.com/js/js_math.asp)

Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.

## **Set Date Methods**

Set Date methods are used for setting a part of a date:

|  |  |
| --- | --- |
| **Method** | **Description** |
| setDate() | Set the day as a number (1-31) |
| setFullYear() | Set the year (optionally month and day) |
| setHours() | Set the hour (0-23) |
| setMilliseconds() | Set the milliseconds (0-999) |
| setMinutes() | Set the minutes (0-59) |
| setMonth() | Set the month (0-11) |
| setSeconds() | Set the seconds (0-59) |
| setTime() | Set the time (milliseconds since January 1, 1970) |

## **The setFullYear() Method**

The setFullYear() method sets the year of a date object. In this example to 2020:

### **Example**

const d = new Date();  
d.setFullYear(2020);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setfullyear)

The setFullYear() method can **optionally** set month and day:

### **Example**

const d = new Date();  
d.setFullYear(2020, 11, 3);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setfullyear_options)

ADVERTISEMENT

## **The setMonth() Method**

The setMonth() method sets the month of a date object (0-11):

### **Example**

const d = new Date();  
d.setMonth(11);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setmonth)

## **The setDate() Method**

The setDate() method sets the day of a date object (1-31):

### **Example**

const d = new Date();  
d.setDate(15);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setdate)

The setDate() method can also be used to **add days** to a date:

### **Example**

const d = new Date();  
d.setDate(d.getDate() + 50);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_ahead)

If adding days shifts the month or year, the changes are handled automatically by the Date object.

## **The setHours() Method**

The setHours() method sets the hours of a date object (0-23):

### **Example**

const d = new Date();  
d.setHours(22);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_sethours)

## **The setMinutes() Method**

The setMinutes() method sets the minutes of a date object (0-59):

### **Example**

const d = new Date();  
d.setMinutes(30);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setminutes)

## **The setSeconds() Method**

The setSeconds() method sets the seconds of a date object (0-59):

### **Example**

const d = new Date();  
d.setSeconds(30);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_setseconds)

## **Compare Dates**

Dates can easily be compared.

The following example compares today's date with January 14, 2100:

### **Example**

let text = "";  
const today = new Date();  
const someday = new Date();  
someday.setFullYear(2100, 0, 14);  
  
if (someday > today) {  
  text = "Today is before January 14, 2100.";  
} else {  
  text = "Today is after January 14, 2100.";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_date_compare)