JavaScript getTime()

The internal clock in JavaScript counts from midnight January 1, 1970.

The getTime() function returns the number of milliseconds since then:

1550704419883

The getFullYear() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getFullYear();
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getFullYear()</h2>

<p>The getFullYear() method returns the full year of a date:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getFullYear();
</script>

</body>
</html>
```

JavaScript getFullYear()

The getFullYear() method returns the full year of a date:

2019

The getMonth() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getMonth();
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getMonth()</h2>

<p>The getMonth() method returns the month of a date as a number from 0 to 11.
</p>
<p>To get the correct month, you must add 1:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getMonth() + 1;
</script>

</body>
</html>
```

JavaScript getMonth()

The getMonth() method returns the month of a date as a number from 0 to 11.

To get the correct month, you must add 1:

2

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

Example

```
var d = new Date();
var months = ["January", "February", "March", "April", "May", "June", "July", "August",
"September", "October", "November", "December"];
document.getElementById("demo").innerHTML = months[d.getMonth()];
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getMonth()</h2>

<p>The getMonth() method returns the month as a number:</p>

<p>You can use an array to display the name of the month:</p>

<p id="demo"></p>

<script>
var d = new Date();
var months =
["January","February","March","April","May","June","July","August","September",
"October","November","December"];
document.getElementById("demo").innerHTML = months[d.getMonth()];
</script>

</body>
</html>
```

JavaScript getMonth()

The getMonth() method returns the month as a number:

You can use an array to display the name of the month:

February

The getDate() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getDate();
```

The getHours() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getHours();
```

The getMinutes() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getMinutes();
```

The getSeconds() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getSeconds();
```

The getMilliseconds() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getMilliseconds();
```

The getMilliseconds() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getMilliseconds();
```

The getDay() Method

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.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

```
var d = new Date();
var days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
document.getElementById("demo").innerHTML = days[d.getDay()];
```

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

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

```
<script>
var d = new Date();
d.setFullYear(2020);
document.getElementById("demo").innerHTML = d;
</script>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript setFullYear()</h2>

<p>The setFullYear() method sets the year of a date object:</p>

<p id="demo"></p>

<script>
var d = new Date();
d.setFullYear(2020);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript setFullYear()

The setFullYear() method sets the year of a date object:

Fri Feb 21 2020 04:55:19 GMT+0530 (India Standard Time)

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

Example

```
<script>
var d = new Date();
d.setFullYear(2020, 11, 3);
document.getElementById("demo").innerHTML = d;
</script>
```

The setMonth() Method

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

Example

```
<script>
var d = new Date();
d.setMonth(11);
document.getElementById("demo").innerHTML = d;
</script>
```

```
<html>
<body>

<h2>JavaScript setMonth()</h2>

<p>The setMonth() method sets the mont of a date object.</p>
<p>Note that months count from 0. December is month 11:</p>

<p id="demo"></p>

<script>
var d = new Date();
d.setMonth(11);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript setMonth()

The setMonth() method sets the mont of a date object.

Note that months count from 0. December is month 11:

Sat Dec 21 2019 04:57:01 GMT+0530 (India Standard Time)

The setDate() Method

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

Example

```
<script>
var d = new Date();
d.setDate(20);
document.getElementById("demo").innerHTML = d;
</script>
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript setDate()</h2>

<p>The setDate() method sets the day of a date object:</p>

<p id="demo"></p>

<script>
var d = new Date();
d.setDate(15);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript setDate()

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

Fri Feb 15 2019 04:58:58 GMT+0530 (India Standard Time)

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

Example

```
<script>
var d = new Date();
d.setDate(d.getDate() + 50);
document.getElementById("demo").innerHTML = d;
</script>
```

The setHours() Method

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

Example

```
<script>
var d = new Date();
d.setHours(22);
document.getElementById("demo").innerHTML = d;
</script>
```

The setMinutes() Method

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

Example

```
<script>
var d = new Date();
d.setMinutes(30);
document.getElementById("demo").innerHTML = d;
</script>
```

Once we have a date, we can access all the components of the date with various built-in methods. The methods will return each part of the date relative to the local timezone. Each of these methods starts with `get`, and will return the relative number. Below is a detailed table of the `get` methods of the `Date` objec

Date/Time	Method	Range	Example
Year	`getFullYear()`	YYYY	1970
Month	`getMonth()`	0-11	0 = January
Day (of the month)	`getDate()`	1-31	1 = 1st of the month
Day (of the week)	`getDay()`	0-6	0 = Sunday
Hour	`getHours()`	0-23	0 = midnight
Minute	`getMinutes()`	0-59	
Second	`getSeconds()`	0-59	
Millisecond	`getMilliseconds()`	0-999	
Timestamp	`getTime()`	Milliseconds since Epoch time	

harryPotter.js

```
// Initialize a new birthday instance
const birthday = new Date(1980, 6, 31);
```

Now we can use all our methods to get each date component, from year to millisecond.

getDateComponents.js

```
birthday.getFullYear();      // 1980
birthday.getMonth();         // 6
birthday.getDate();          // 31
birthday.getDay();           // 4
birthday.getHours();         // 0
birthday.getMinutes();       // 0
birthday.getSeconds();       // 0
birthday.getMilliseconds();  // 0
birthday.getTime();          // 333849600000 (for GMT)
```

oct3.js

```
// Get today's date
const today = new Date();

// Compare today with October 3rd
if (today.getDate() === 3 && today.getMonth() === 9) {
  console.log("It's October 3rd.");
} else {
  console.log("It's not October 3rd.");
}
```

harryPotter.js

```
// Change year of birthday date
birthday.setFullYear(1997);

birthday;
```

Output

```
Thu Jul 31 1997 00:00:00 GMT+0000 (UTC)
```

Modifying the Date with `set`

For all the `get` methods that we learned about above, there is a corresponding `set` method. Where `get` is used to retrieve a specific component from a date, `set` is used to modify components of a date. Below is a detailed chart of the `set` methods of the `Date` object.

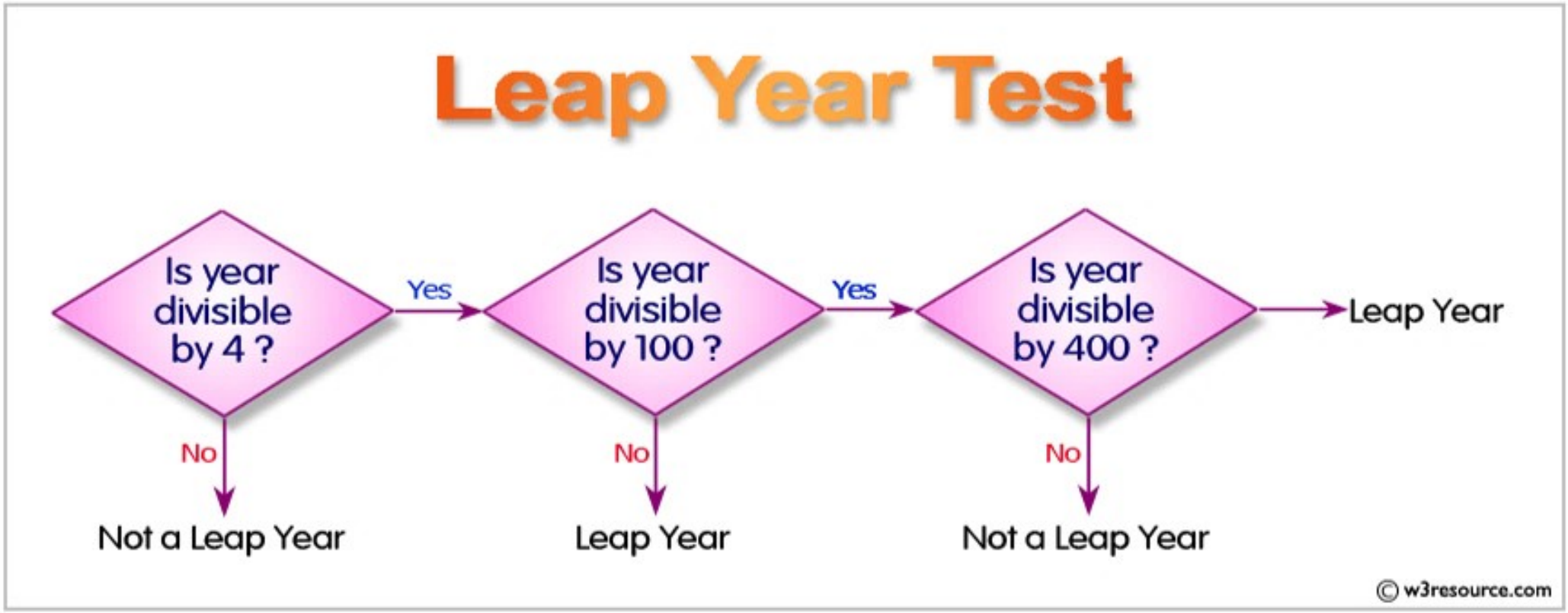
Date/Time	Method	Range	Example
Year	`setFullYear()`	YYYY	1970
Month	`setMonth()`	0-11	0 = January
Day (of the month)	`setDate()`	1-31	1 = 1st of the month
Day (of the week)	`setDay()`	0-6	0 = Sunday
Hour	`setHours()`	0-23	0 = midnight
Minute	`setMinutes()`	0-59	
Second	`setSeconds()`	0-59	
Millisecond	`setMilliseconds()`	0-999	
Timestamp	`setTime()`	Milliseconds since Epoch time	

Date Methods with UTC

The `get` methods discussed above retrieve the date components based on the user's local timezone settings. For increased control over the dates and times, you can use the `getUTC` methods, which are exactly the same as the `get` methods, except they calculate the time based on the UTC (Coordinated Universal Time) standard. Below is a table of the UTC methods for the JavaScript `Date` object.

Date/Time	Method	Range	Example
Year	`getUTCFullYear()`	YYYY	1970
Month	`getUTCMonth()`	0-11	0 = January
Day (of the month)	`getUTCDate()`	1-31	1 = 1st of the month
Day (of the week)	`getUTCDay()`	0-6	0 = Sunday
Hour	`getUTCHours()`	0-23	0 = midnight
Minute	`getUTCMinutes()`	0-59	
Second	`getUTCSeconds()`	0-59	
Millisecond	`getUTCMilliseconds()`	0-999	

Pictorial Presentation:

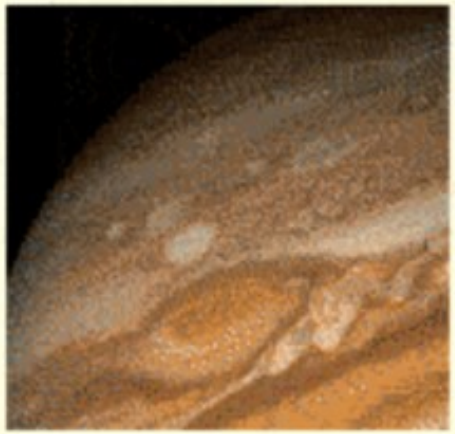


Selected Planets

Read essential details about the following planets:

- Venus
- Earth
- Jupiter

Jupiter



Jupiter is the fifth planet from the Sun and by far the largest. Jupiter is more than twice as massive as all the other planets combined (318 times Earth). Its orbit is 778,330,000 km (5.20 AU) from Sun; its diameter is 142,984 km (equatorial); and its mass is 1.900e27 kg.

Jupiter is the fourth brightest object in the sky (after the Sun, the Moon and Venus; at some times Mars is also brighter). It has been known since prehistoric times. Galileo's discovery, in 1610, of Jupiter's four large moons Io, Europa, Ganymede and Callisto (now known as the Galilean moons) was the first discovery of a center of motion not apparently centered on the Earth. It was a major point in favor of Copernicus's heliocentric theory of the motions of the planets; Galileo's outspoken support of the Copernican theory got him in trouble with the Inquisition.

Jupiter was first visited by Pioneer 10 in 1973 and later by Pioneer 11, Voyager 1, Voyager 2 and Ulysses. The spacecraft Galileo is currently in orbit around Jupiter and will be sending back data for at least the next two years

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

Thu Feb 21 2019 03:55:35 GMT+0530 (India Standard Time)

Year: 2019	Month: 2	Day: 21	Hours: 3	Minutes 55	Seconds: 35

Example

```
var d = new Date();
```

JavaScript Date Output

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

Thu Feb 21 2019 03:55:35 GMT+0530 (India Standard Time)

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

```
var d = new Date();
```

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

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

Note: JavaScript counts months from 0 to 11.

January is 0. December is 11.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>6 numbers specify year, month, day, hour, minute and second:</p>

<p id="demo"></p>

<script>
var d = new Date(2018, 11, 24, 10, 33, 30);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

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

Mon Dec 24 2018 10:33:30 GMT+0530 (India Standard Time)

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

Example

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

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

Example

```
var d = new Date(2018, 11, 24, 10);
```

3 numbers specify year, month, and day:

Example

```
var d = new Date(2018, 11, 24);
```

2 numbers specify year and month:

Example

```
var d = new Date(2018, 11);
```

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

Example

```
var d = new Date(2018);
```

Previous Century

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

Example

```
var d = new Date(99, 11, 24);
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>Two digit years will be interpreted as 19xx:</p>

<p id="demo"></p>

<script>
var d = new Date(99, 11, 24);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

Two digit years will be interpreted as 19xx:

Fri Dec 24 1999 00:00:00 GMT+0530 (India Standard Time)

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>One digit years will be interpreted as 19xx:</p>

<p id="demo"></p>

<script>
var d = new Date(9, 11, 24);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

One digit years will be interpreted as 19xx:

Fri Dec 24 1909 00:00:00 GMT+0530 (India Standard Time)

new Date(*dateString*)

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

Example

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>A Date object can be created with a specified date and time:</p>

<p id="demo"></p>

<script>
var d = new Date("October 13, 2014 11:13:00");
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

A Date object can be created with a specified date and time:

Mon Oct 13 2014 11:13:00 GMT+0530 (India Standard Time)

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: **1550701535747** milliseconds past January 01, 1970

new Date(*milliseconds*)

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

Example

```
var d = new Date(0);
```

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>Using new Date(milliseconds), creates a new date object as January 1, 1970,
00:00:00 Universal Time (UTC) plus the milliseconds:</p>

<p id="demo"></p>

<script>
var d = new Date(0);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

Using new Date(milliseconds), creates a new date object as January 1, 1970, 00:00:00 Universal Time (UTC) plus the milliseconds:

Thu Jan 01 1970 05:30:00 GMT+0530 (India Standard Time)

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

Example

```
var d = new Date(-100000000000);
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>100000000000 milliseconds from Jan 1, 1970, is approximately Oct 31, 1966:
</p>

<p id="demo"></p>

<script>
var d = new Date(-100000000000);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

JavaScript new Date()

10000000000 milliseconds from Jan 1, 1970, is approximately Oct 31, 1966:

Mon Oct 31 1966 19:43:20 GMT+0530 (India Standard Time)

Example

```
var d = new Date(86400000);
```

Try it Yourself »

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript new Date()</h2>

<p>Using new Date(milliseconds), creates a new date object as January 1, 1970,
00:00:00 Universal Time (UTC) plus the milliseconds:</p>

<p id="demo"></p>

<script>
var d = new Date(86400000);
document.getElementById("demo").innerHTML = d;
</script>

<p>One day (24 hours) is 86,400,000 milliseconds.</p>

</body>
</html>
```

JavaScript new Date()

Using new Date(milliseconds), creates a new date object as January 1, 1970, 00:00:00 Universal Time (UTC) plus the milliseconds:

Fri Jan 02 1970 05:30:00 GMT+0530 (India Standard Time)

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

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

Example

```
d = new Date();
document.getElementById("demo").innerHTML = d;
```

Same as:

```
d = new Date();
document.getElementById("demo").innerHTML = d.toString();
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript toString()</h2>

<p>The toString() method converts a date to a string:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toString();
</script>

</body>
</html>
```

JavaScript toString()

The toString() method converts a date to a string:

Thu Feb 21 2019 04:17:33 GMT+0530 (India Standard Time)

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.toUTCString();
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Date()</h2>

<p>The toUTCString() method converts a date to a UTC string (a date display
standard):</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toUTCString();
</script>

</body>
</html>
```

JavaScript Date()

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

Wed, 20 Feb 2019 22:49:02 GMT

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

Example

```
var d = new Date();
document.getElementById("demo").innerHTML = d.toDateString();
```

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:

```
Wed Mar 25 2015 05:30:00 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)

```
var d = new Date("2015-03-25");
```

ISO Dates (Year and Month)

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

Example

```
var d = new Date("2015-03");
```

ISO Dates (Only Year)

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

Example

```
var d = new Date("2015");
```

ISO Dates (Date-Time)

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

Example

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

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

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript ISO Dates</h2>

<p>Modify the time relative to UTC by adding +HH:MM or subtraction -HH:MM to
the time.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
new Date("2015-03-25T12:00:00-06:00");
</script>

</body>
</html>
```

JavaScript ISO Dates

Modify the time relative to UTC by adding +HH:MM or subtraction -HH:MM to the time.

Wed Mar 25 2015 23:30:00 GMT+0530 (India Standard Time)

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

```
var d = new Date("03/25/2015");
```

WARNINGS !

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

```
var 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.

```
var 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.

```
var d = new Date("25-03-2015");
```

JavaScript Long Dates.

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

Example

```
var d = new Date("Mar 25 2015");
```

Try it Yourself »

Month and day can be in any order:

Example

```
var d = new Date("25 Mar 2015");
```

Try it Yourself »

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

Example

```
var d = new Date("January 25 2015");
```

Try it Yourself »

Example

```
var d = new Date("Jan 25 2015");
```

Try it Yourself »

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

```
var msec = Date.parse("March 21, 2012");
document.getElementById("demo").innerHTML = msec;
```

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Date.parse()</h2>

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

<p id="demo"></p>

<script>
var msec = Date.parse("March 21, 2012");
document.getElementById("demo").innerHTML = msec;
</script>

</body>
</html>
```

JavaScript Date.parse()

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

1332268200000

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

Example

```
var msec = Date.parse("March 21, 2012");
var d = new Date(msec);
document.getElementById("demo").innerHTML = d;
```

```
<!DOCTYPE html>
<html>
<body>

<p>Date.parse(string) returns milliseconds.</p>
<p>You can use the return value to convert the string to a date object:</p>

<p id="demo"></p>

<script>
var msec = Date.parse("March 21, 2012");
var d = new Date(msec);
document.getElementById("demo").innerHTML = d;
</script>

</body>
</html>
```

Date.parse(string) returns milliseconds.

You can use the return value to convert the string to a date object:

Wed Mar 21 2012 00:00:00 GMT+0530 (India Standard Time)

JavaScript Get Date Methods

‹ Previous

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

```
var d = new Date();
document.getElementById("demo").innerHTML = d.getTime();
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript getTime()</h2>

<p>The internal clock in JavaScript counts from midnight January 1, 1970.</p>
<p>The getTime() function returns the number of milliseconds since then:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getTime();
</script>

</body>
</html>
```