JavaScript Date and Time

In JavaScript, date and time are represented by the `Date` object.
The `Date` object provides the date and time information and also provides various methods.
A JavaScript date defines the **EcmaScript epoch** that represents milliseconds since **1 January 1970 UTC**. This date and time is the same as the UNIX epoch (predominant base value for computer-recorded date and time values).

Creating Date Objects

There are four ways to create a date object.

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

---

new Date()

You can create a date object using the `new Date()` constructor. For example,

```
const timeNow = new Date();
console.log(timeNow); // shows current date and time
```
Run Code

Output

```
Mon Jul 06 2020 12:03:49 GMT+0545 (Nepal Time)
```

Here, `new Date()` creates a new date object with the current date and local time.

---

new Date(milliseconds)

The `Date` object contains a number that represents milliseconds since **1 January 1970 UTC**.

`new Date(milliseconds)` creates a new date object by adding the milliseconds to the zero time. For example,

```
const time1 = new Date(0);

// epoch time
console.log(time1); // Thu Jan 01 1970 05:30:00

// 100000000000 milliseconds after the epoch time
const time2 = new Date(100000000000)
console.log(time2); // Sat Mar 03 1973 15:16:40
```
Run Code

Note: 1000 milliseconds is equal to 1 second.

---

new Date(date string)

`new Date(date string)` creates a new date object from a date string.

In JavaScript, there are generally three date input formats.

ISO Date Formats

You can create a date object by passing ISO date formats. For example,

```
// ISO Date(International Standard)
const date = new Date("2020-07-01");

// the result date will be according to UTC
console.log(date); // Wed Jul 01 2020 05:45:00 GMT+0545
```

You can also pass only the year and month or only the year. For example,

```
const date = new Date("2020-07");
console.log(date); // Wed Jul 01 2020 05:45:00 GMT+0545

const date1 = new Date("2020");
console.log(date1); // Wed Jul 01 2020 05:45:00 GMT+0545
```

You can also pass specific time to ISO dates.

```
const date = new Date("2020-07-01T12:00:00Z");
console.log(date); // Wed Jul 01 2020 17:45:00 GMT+0545
```

Note: Date and time are separated with capital letter **T**. And UTC time is defined with capital **Z**.

Short and Long date format

The other two date formats are **short date format** and **long date format**.

```
// short date format "MM/DD/YYYY"
const date = new Date("03/25/2015");
console.log(date); // Wed Mar 25 2015 00:00:00 GMT+0545

// long date format "MMM DD YYYY"
const date1 = new Date("Jul 1 2020");
console.log(date1); // Wed Jul 01 2020 00:00:00 GMT+0545
```

```
// month and day can be in any order
const date2 = new Date("1 Jul 2020");
console.log(date2); // Wed Jul 01 2020 00:00:00 GMT+0545

// month can be full or abbreviated. Also month names are insensitive.
// comma are ignored
const date3 = new Date("July 1 2020");
console.log(date3); // Wed Jul 01 2020 00:00:00 GMT+0545

const date4 = new Date("JULY, 1, 2020");
console.log(date4); // Wed Jul 01 2020 00:00:00
```
Run Code

---

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

`new Date(year, month,...)` creates a new date object by passing specific date and time. For example,

```
const time1 = new Date(2020, 1, 20, 4, 12, 11, 0);
console.log(time1); // Thu Feb 20 2020 04:12:11
```

The passed argument has a specific order.

If four numbers are passed, it represents year, month, day and hours. For example,

```
const time1 = new Date(2020, 1, 20, 4);
console.log(time1); // Thu Feb 20 2020 04:00:00
```

Similarly, if two arguments are passed, it represents year and month. For example,

```
const time1 = new Date(2020, 1);
console.log(time1); // Sat Feb 01 2020 00:00:00
```

Note: If you pass only one argument, it is treated as milliseconds. Hence, you have to pass two arguments to use this date format.
In JavaScript, months are counted from **0 to 11**. January is **0** and December is **11**.

JavaScript Date Methods

There are various methods available in JavaScript Date object.

Method	Description
now()	Returns the numeric value corresponding to the current time (the num of milliseconds elapsed since January 1, 1970 00:00:00 UTC)
getFullYear()	Gets the year according to local time
getMonth()	Gets the month, from 0 to 11 according to local time
getDate()	Gets the day of the month (1–31) according to local time
getDay()	Gets the day of the week (0-6) according to local time
getHours()	Gets the hour from 0 to 23 according to local time
getMinutes	Gets the minute from 0 to 59 according to local time
getUTCDate()	Gets the day of the month (1–31) according to universal time
setFullYear()	Sets the full year according to local time
setMonth()	Sets the month according to local time
setDate()	Sets the day of the month according to local time
setUTCDate()	Sets the day of the month according to universal time

Example: Date Methods

```
const timeInMilliseconds = Date.now();
console.log(timeInMilliseconds); // 1593765214488

const time = new Date;

// get day of the month
const date = time.getDate();
console.log(date); // 30

// get day of the week
const year = time.getFullYear();
console.log(year); // 2020

const utcDate = time.getUTCDate();
console.log(utcDate); // 30

const event = new Date('Feb 19, 2020 23:15:30');
// set the date
event.setDate(15);
console.log(event.getDate()); // 15

// Only 28 days in February!
event.setDate(35);

console.log(event.getDate()); // 7
```
Run Code

Formatting a Date

Unlike other programming languages, JavaScript does not provide a built-in function for formatting a date.

However, you can extract individual bits and use it like this.

```
const currentDate = new Date();
const date = currentDate.getDate();
const month = currentDate.getMonth();
const year = currentDate.getFullYear();
```

```
// show in specific format
let monthDateYear = (month+1) + '/' + date + '/' + year;

console.log(monthDateYear); // 7/3/2020
```
Run Code

Note: The above program gives inconsistent length as date and month have single-digit and double-digit.

AutoCorrection in Date Object

When you assign out of range values in the `Date` object, it auto-corrects itself. For example,

```
const date = new Date(2008, 0, 33);
// Jan does not have 33 days

console.log(date);
```
Run Code

Output

```
Sat Feb 02 2008
```