CSE 3002

INTERNET & WEB PROGRAMMING

Theory DA

G2 | SJT601

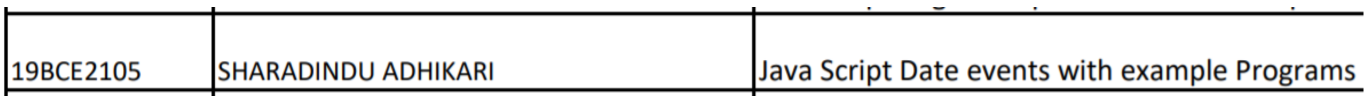
FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105

Question



19BCE2105	SHARADINDU ADHIKARI	Java Script Date events with example Programs

Q. Write a short note on JavaScript Date events with example programs.

Solution

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 are 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()`

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

```
const timeNow = new Date();
console.log(timeNow); // shows current date and 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);
const time2 = new Date(100000000000);
console.log(time2);
```

<u>`new Date(date string)`</u>

`new Date(date string)` creates a new date object from a date string.
In JavaScript, there are generally three date input formats.

<u>ISO Date formats:</u>

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

```
const date = new Date("2021-07");
console.log(date); // Wed Jul 01 2020 05:45:00 GMT+0530

const date1 = new Date("2021");
console.log(date1); // Wed Jul 01 2021 05:45:00 GMT+0530
```

<u>Short and Long date formats:</u>

The other two date formats are short date format and long date format. For example,

```
// short date format "MM/DD/YYYY"
const date = new Date("03/25/2021");
console.log(date); // Wed Mar 25 2021 00:00:00 GMT+0530
// long date format "MMM DD YYYY"
const date1 = new Date("Jul 1 2021");
console.log(date1); // Wed Jul 01 2021 00:00:00 GMT+0530
// month and day can be in any order
const date2 = new Date("1 Jul 2021");
console.log(date2); // Wed Jul 01 2021 00:00:00 GMT+0530
// month can be full or abbreviated. Also, month names are insensitive.
const date3 = new Date("July 1 2021");
console.log(date3); // Wed Jul 01 2021 00:00:00 GMT+0530
const date4 = new Date("JULY, 1, 2021");
console.log(date4); // Wed Jul 01 2021 00:00:00
```

`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(2021, 1, 20, 4, 12, 11, 0);
console.log(time1); // Thu Feb 20 2021 04:12:11
```

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 number 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

For example,

```
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); // 2021
const utcDate = time.getUTCDate();
console.log(utcDate); // 30
const event = new Date('Feb 19, 2021 23:15:30');
event.setDate(15);
console.log(event.getDate()); // 15
// Only 28 days in February!
event.setDate(35);
console.log(event.getDate()); // 7
```

Formatting a Date:

Unlike other programming languages, JavaScript does not provide a built-in function for formatting a date. However, we 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();
let monthDateYear = (month+1) + '/' + date + '/' + year;
console.log(monthDateYear);
```

Comparing Dates:

As with everything else related to date, comparing dates has its own gotchas.

First, we need to create date objects. Fortunately, <, >, <=, and >= all work. So, comparing July 19, 2014 and July 18, 2014 is as easy as:

```
const date1 = new Date("July 19, 2014");
const date2 = new Date("July 28, 2014");

if(date1 > date2) {
    console.log("First date is more recent");
} else {
    console.log("Second date is more recent");
}
```

Checking for equality is trickier, since two date objects representing the same date are still two different date objects and will not be equal. Comparing date strings is a bad idea because, for example, “July 20, 2014” and “20 July 2014” represent the same date but have different string representations. The snippet below illustrates the first point:

```
const date1 = new Date("June 10, 2003");
const date2 = new Date(date1);

const equalOrNot = date1 == date2 ? "equal" : "not equal";
console.log(equalOrNot);
```

The output will never be equal.

This particular case can be fixed by comparing the integer equivalents of the dates (their time stamps) as follows:

```
date1.getTime() == date2.getTime();
```

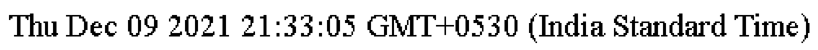
Testing:

1. Create a date object in JavaScript.

Code:

```
<script>
   var d = new Date();
   document.write(d);
</script>
```

Output:



```
Thu Dec 09 2021 21:33:05 GMT+0530 (India Standard Time)
```

2. Passing parameters to the date object.

Code:

```
<script>
    var d = new Date(2018,0,31,14,35,20,50);
    document.write(d);
</script>
```

Output:

```
Wed Jan 31 2018 14:35:20 GMT+0530 (India Standard Time)
```

3. Construct a date object by passing a date string.

Code:

```
<script>
   var d = new Date("31 January 2018");
    document.write(d);
</script>
```

Output:

```
Wed Jan 31 2018 00:00:00 GMT+0530 (India Standard Time)
```

4. Get year, month, and day from a date object.

Code:

```
<script>
   var d = new Date();
   // Extracting date part
   document.write(d.getDate() + "<br>"); // Display the day of the month
   document.write(d.getDay() + "<br>"); // Display the number of days into the
week (0-6)
    document.write(d.getMonth() + "<br>"); // Display the number of months
into the year (0-11)
    document.write(d.getFullYear()); // Display the full year (four digits)
</script>
```

Output:

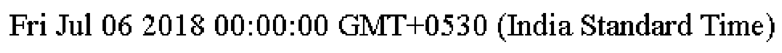
```
9
4
11
2021
```

5. Specify the date value outside of the range.

Code:

```
<script>
    var d = new Date(2018, 5, 24); // June 24, 2018
    d.setDate(36); // Sets day to 36, new date will be July 6, 2018
    document.write(d);
</script>
```

Output:



```
Fri Jul 06 2018 00:00:00 GMT+0530 (India Standard Time)
```