momentjs

About the Tutorial

MomentJS is a JavaScript library which helps is parsing, validating, manipulating and displaying date and time in JavaScript in a very easy way. MomentJS can be used directly inside a browser and also with Node.js. Working with dates and time using JavaScript can be quite challenging, specifically if you have lots of manipulation to be done with dates. MomentJS comes with lots of features that eases your work with date and time.

Audience

This tutorial is designed for software programmers who want to learn the basics of MomentJS and its programming concepts in simple and easy way. This tutorial will give you enough understanding on various functionalities of MomentJS with suitable examples.

Prerequisites

This tutorial is designed keeping in mind that its readers have a basic understanding of HTML, CSS, and JavaScript. If you are new to these concepts, we recommend you to go through relevant tutorials and gain an understanding on them, before you proceed with this tutorial.

Copyright &Disclaimer

Table of Contents

1. MomentJS – Overview

MomentJS is a JavaScript library which helps is parsing, validating, manipulating and displaying date/time in JavaScript in a very easy way. This chapter will provide an overview of MomentJS and discusses its features in detail.

Moment JS allows displaying of date as per localization and in human readable format. You can use MomentJS inside a browser using the script method. It is also available with Node.js and can be installed using npm.

In MomentJS, you can find many easy to use methods to add, subtract, validate date, get the maximum, minimum date etc. It is an open source project and you can easily contribute to the library and add features in the form of plugins and make it available on GitHub and in Node.js.

Features

Let us understand in detail all the important features available with MomentJS:

Parsing

Parsing allows you to parse the date in the format required. Parsing of date is available in string, object and array. It allows you to clone the moment using **moment.clone**. There are methods available which gives the date output in UTC format.

Date Validation

Date Validation is very easy with MomentJS. You can use the method **isValid()** and check whether the date is valid or not. MomentJS also provides many parsing flags which can be used to check for date validation.

Manipulation

There are various methods to manipulate date and time on the moment object. add, subtract, startoftime, endoftime, local, utc, utcoffset etc., are the methods available which gives details required on date/time in MomentJS.

Get/Set

Get/Set allows to read and set the units in the date. It allows to change as well as read hour , minute, seconds, millisecond, date of month, day of week, day of year, week of year, month, year, quarter, week year, weeks in year, get/set, maximum , minimum etc. Get /Set is a very helpful feature available in MomentJS.

Display

Display provides formats to display date in different ways. There are methods available which tells the time from a given moment, from the current moment, difference between two moments etc. It allows to display date in JSON format, Array, Object, String etc.

Date Queries

Date Queries has easy to use methods which tells if the date is greater or lesser than the input, in between the dates given, is a leap year, is a moment, is a date etc. It is very useful with date validation.

Durations

Durations is one of the important features in MomentJS. It basically handles length of the time for given units. The **humanize** method available displays date in a human readable format.

Internationalization

Internationalization is yet another important features in MomentJS. You can display date and time based on locale. The locale can be applied to a specific moment if required. You will get a minified file from the MomentJS home site which has all the locales. In case you are dealing with a specific locale, you can also add just that locale file and work with it. The names of months, weeks and days are displayed in the locale specified.

Customization

MomentJS allows customization to the locale created. You can customize month names, month abbreviation, weekday names, weekday abbreviation, long date format, and calendar format for a defined locale as per your requirements.

Utilities

Utilities comes with two methods: **normalize units** and **invalid.** They are used with the moment and helps us change or customize the output as we need. It also allows to set our own custom validation on the moment object.

Plugins

Plugins are additional features of MomentJS. There are many plugins added to calendars, date format, parsing, date ranges, precise range etc. You can add your own plugins and make them available with Node.js and GitHub.

2. MomentJS –Environment Setup

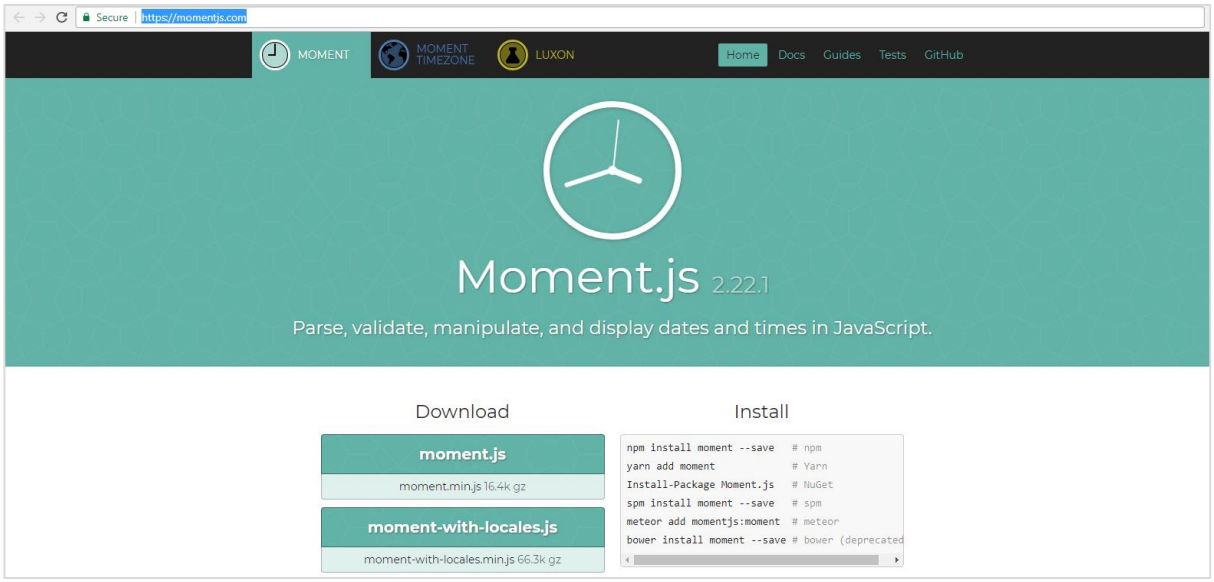
In this chapter, you will learn in detail about setting up the working environment of MomentJS on your local computer. Before you begin with working on MomentJS, you need to have the access to the library. You can access its files in any of the following methods:

Method 1: Using MomentJS File in Browser

In this method, we are going to need MomentJS file from its official website and will use it directly in the browser.

Step 1

As a first step, go to the official website of MomentJS (https://MomentJS.com/). You will find the home page as shown here:



Observe that there is a download option available which gives you the latest MomentJS file available. Note that the file is available with and without minification.

Step 2

Now, include **moment.js** inside the **script** tag and start working with MomentJS**.** For this, you can use the code given below:

```
<script type="text/JavaScript"
src="https://MomentJS.com/downloads/moment.js"></script>
```

Given here is a working example and its output for a better understanding:

Example

```
<html>
<head>
    <title>MomentJS - Working Example</title>
    <script type="text/JavaScript"
src="https://MomentJS.com/downloads/moment.js"></script>
    <style>
        div {
            border: solid 1px #ccc;
            padding:10px;
            font-family: "Segoe UI",Arial,sans-serif;
            width: 50%;
        }
    </style>
</head>
<body>
    <div style="font-size:25px" id="todaysdate"></div>
<script type="text/JavaScript">
    var a = moment().toString();
    document.getElementById("todaysdate").innerHTML = a;
</script>
</body>
</html>
```

Output

Tue Apr 17 2018 16:05:53 GMT+0530

The **moment-locale** file to work with different locales is also available as shown in the screenshot above. Now, add the file to the script tag as shown below and work with different locales of your choice. For this, you can use the code given below:

```
<script type="text/JavaScript" src="https://MomentJS.com/downloads/moment-with-
locales.js"></script>
```

Given here is a working example for moment-locale and its output for a better understanding:

```
<html>
    <head>
        <script type="text/JavaScript"
src="https://MomentJS.com/downloads/moment-with-locales.js"></script>
    </head>
    <body>
        <h1>Moment Locale</h1>
        <div id="datedisplay" style="font-size:30px;"></div>
        <script type="text/JavaScript">
            var a = moment.locale("fr");
            var c = moment().format("LLLL");
            document.getElementById("datedisplay").innerHTML = c;
        </script>
</body></html>
```

Output

Method 2: Using Node.js

If you are opting for this method, make sure you have **Node.js** and **npm** installed on your system. You can use the following command to install MomentJS:

```
npm install moment
```

You can observe the following output once MomentJS is successfully installed:

```
C:\moment>npm install moment
C:\moment
`-- moment@2.22.1

npm WARN enoent ENOENT: no such file or directory, open 'C:\moment\package.json'

npm WARN moment No description
npm WARN moment No repository field.
npm WARN moment No README data
npm WARN moment No license field.

C:\moment>
```

Now, to test if MomentJS works fine with Node.js, create the file **test.js** and add the following code to it:

```
var moment = require('moment');

var a = moment().toString();

console.log(a);
```

Now, in the command prompt, run the command node test.js as shown in the screenshot given below:

```
C:\moment\node_modules\moment\src>node test.js
Tue Apr 17 2018 16:30:42 GMT+0530

C:\moment\node_modules\moment\src>
```

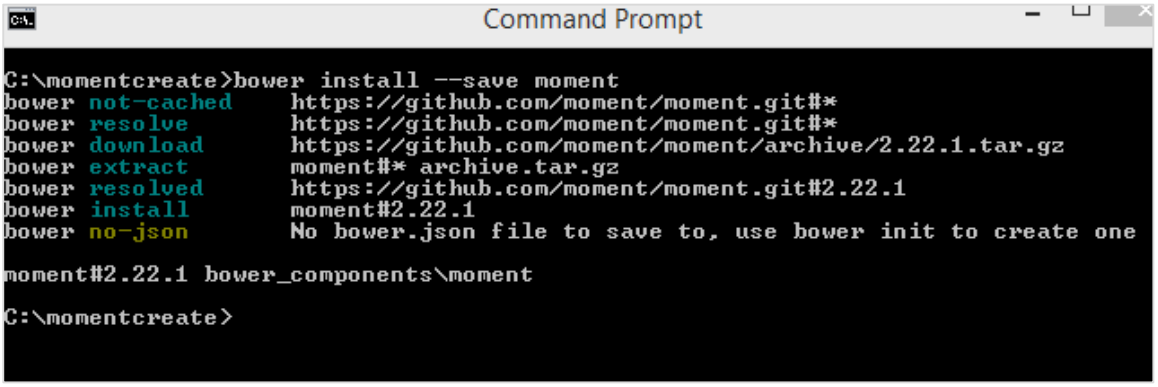
Note that this command displays the output for **moment().toString()**.

Method 3: Using Bower

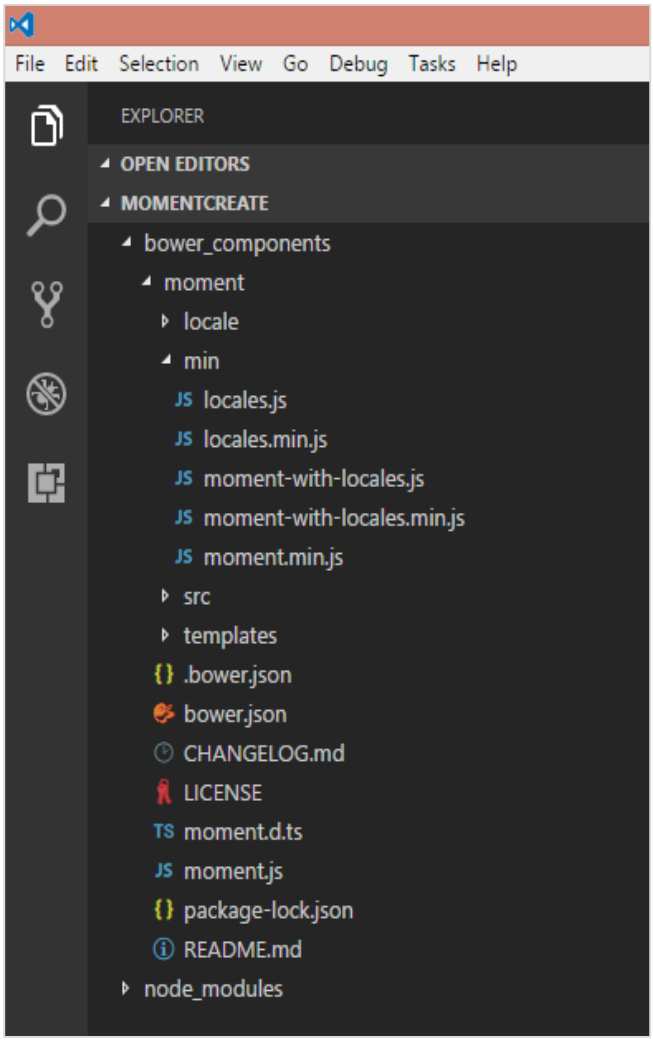
Bower is another method to get the required files for MomentJS. You can use the following command to install MomentJS using Bower:

```
bower install --save moment
```

The screenshot given below shows the installation of MomentJS using Bower:



These are the files loaded from Bower for MomentJS to install. The installed moment and locale files are shown in the image given below:

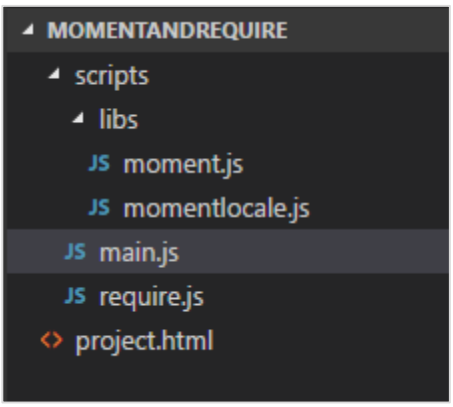


3. MomentJS – Introduction

In this chapter, we will discuss how to work with **MomentJS using RequireJS** and **MomentJS and TypeScript.**

MomentJS and RequireJS

To understand the working of MomentJS using RequireJS, let us analyze a working example with MomentJS and RequireJS. The folder structure of the corresponding app is shown in the following image:



You can obtain the **require.js** file fetched from RequireJS official site: http://requirejs.org/docs/download.html. Observe the following code for a better understanding:

Example project.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>RequireJS and MomentJS</title>
        <!-- data-main attribute tells require.js to load
             scripts/main.js after require.js loads. -->
        <script data-main="scripts/main" src="scripts/require.js"></script>
    </head>
    <body>
        <h1>RequireJS and MomentJS</h1>
        <div id="datedisplay" style="font-size:25px;"></div>
    </body>
</html>
```

main.js

```
require.config({
    paths:{
        'momentlocale':'libs/momentlocale',
    },
});
 require(['momentlocale'], function (moment) {
    moment.locale('fr');
    var a = moment().format("LLLL");
    document.getElementById("datedisplay").innerHTML = a;
});
```

Note that **Moment.js** and **momentlocale.js** are in the folder **libs**.

The following is the output for **project.html** that you will observe in the browser:

MomentJS and TypeScript

The code used for building MomentJS and Typescript project are as given below:

package.json

```
{
  "name": "momenttypescript",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "dependencies": {
    "browserify": "^16.2.0",
    "gulp": "^3.9.1",
    "gulp-connect": "^5.5.0",
    "gulp-typescript": "^4.0.2",
    "moment": "^2.22.1",
    "tsify": "^4.0.0",
```

```
    "typescript": "^2.8.3",
    "vinyl-source-stream": "^2.0.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Note that the dependencies available in **package,json** needs to be installed using **npm install**.

main.ts

```
import * as moment from 'moment';
let now = moment().format('LLLL');
document.getElementById("datedisplay").innerHTML = now;
```

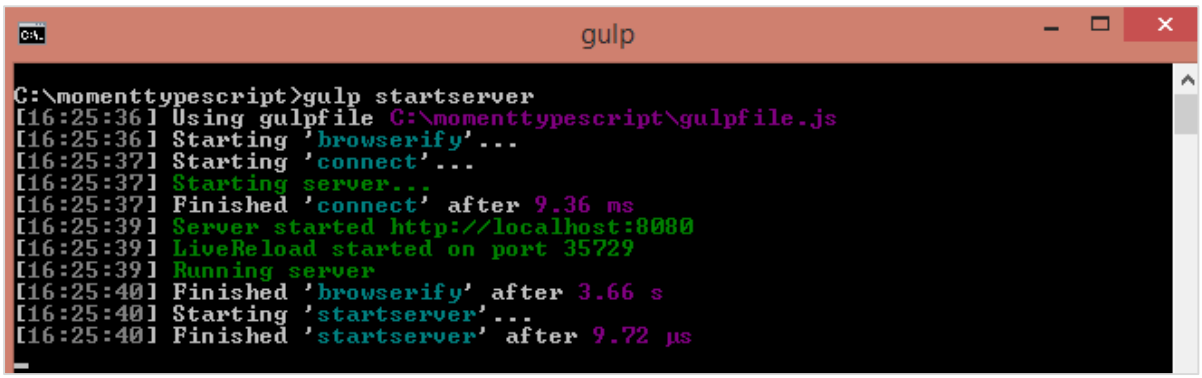
You need to use **Gulp** to build the file from typescript to JavaScript, that is from **main.ts** to **main.js**. The following code shows the **gulpfile.js** which is used to build the file. Note that we have used **gulp-connect** package which opens a local server to display the output.

gulpfile.js

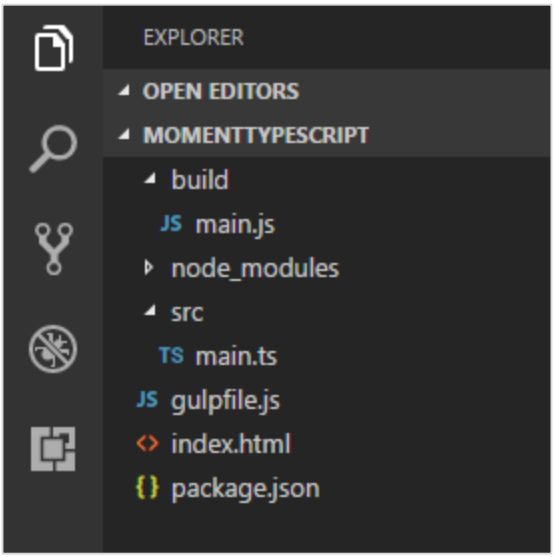
```
var gulp = require("gulp");
var connect = require("gulp-connect");
var browserify = require("browserify");
var tsify = require("tsify");
var source = require("vinyl-source-stream");
gulp.task("build", function (cb) {
  runSequence("browserify", "minify", cb);
});
gulp.task("startserver", ["browserify", "connect"]);
gulp.task("browserify", function () {
var b = browserify({
    insertGlobals: true,
    debug: false
})    .add("src/main.ts")    .plugin(tsify, { typescript: require("typescript")
});
```

```
return b
    .bundle()
    .pipe(source("main.js"))
    .pipe(gulp.dest("build/"));
});
gulp.task("connect", function () {
  connect.server({
    root: ".",
    //  port: '80',
    livereload: true
  });
});
```

This is the output that you observe when you run the code given above:



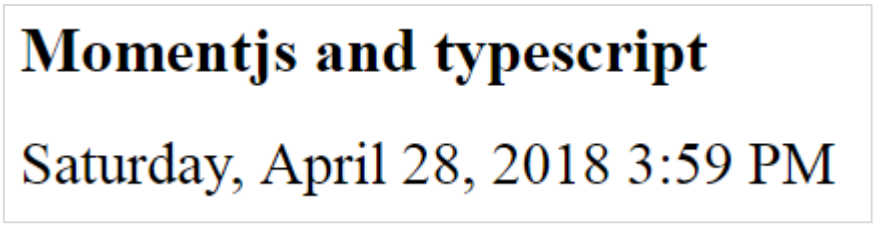
You can see the folder structure as shown in the following format:



The code for index.html is shown below:

```
<html>
    <head></head>
    <body>
        <h1>MomentJS and typescript</h1>
        <div id="datedisplay" style="font-size:30px;"></div>
        <script src="build/main.js"></script>
    </body>
</html>
```

Now, if you open http://localhost:8080/ , you can see the output as shown below:

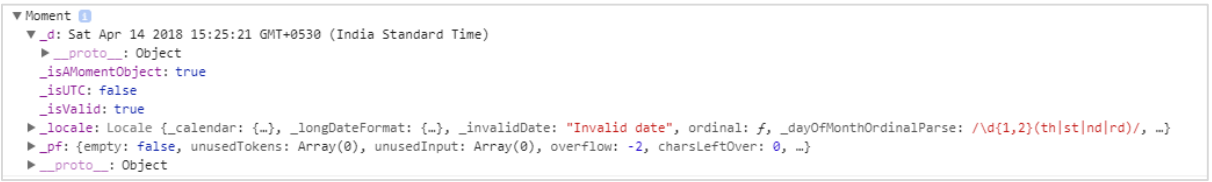


4. MomentJS – Parsing Date and Time

MomentJS has many easy to use methods which helps in parsing date and time. It can parse dates in the form of object, string, array, JavaScript native date object etc. This chapter discusses them in detail.

Parsing Date

MomentJS gives wrapper object as output when **moment()** is called. You can observe the following when you console the output in the browser.



MomentJS provides various methods to parse the Date as listed below:

Method	Syntax
Now	moment()
String	moment(string)
Object	moment(object)
Date	moment(Date)
Array	moment(Array[])
Unix Timestamp	moment(number)
Moment Clone	moment(Moment)
UTC	moment.utc()
parseZone	moment.parseZone()
Creation Data	moment().creationData();
Defaults	var m = moment({hour: 3, minute: 40, seconds: 10});

In this section, let us discuss each method in detail:

Now

This will display the current date.

Example

```
var now = moment();
```

To get the current date, we have to use the following code:

```
now._d
```

Output

```
Current date/time :Sat Apr 14 2018 15:56:09 GMT+0530 (India Standard Time)
```

String

This will take string as date for parsing with moment.

Syntax

```
moment(string)
```

Observe the following examples and their outputs when different date strings are given to moment.

Example 1

```
var day = moment("2017-04-15");
```

To display the date, we have used **day._d** to get the date details from the moment.

Output

```
Current date/time :Sat Apr 15 2017 00:00:00 GMT+0530 (India Standard Time)
```

Example 2

```
var day = moment("2017-W10-5");
```

or

```
var day = moment("2017W105");
```

Observe that in the string **"2017-W10-5"** given to the moment, **W** represents the week. You can observe the following output, where W10 falls for March month.

Output

```
Current date/time :Fri Mar 10 2017 00:00:00 GMT+0530 (India Standard Time)
```

Example 3

```
var day = moment("2017-080");
```

or

```
var day = moment("2017080");
```

Here, the string **2017-080** is the 80th day which falls on March 21st as shown below.

Output

```
Current date/time :Tue Mar 21 2017 00:00:00 GMT+0530 (India Standard Time)
```

Example 4

```
var day = moment("2017-05-08T09");
```

or

```
var day = moment("20170508T09");
```

Here, the string **2017-05-08T09** is given to **moment** where the number after **T** represents the hour to be shown.

Output

```
Current date/time :Mon May 08 2017 09:00:00 GMT+0530 (India Standard Time)
```

Example 5

```
var day = moment("2017-06-08 06:30:26");
```

or

```
var day = moment("20170608T063026");
```

We can also pass the hour, minutes, or seconds to the moment and the output as follows:

Output

```
Current date/time :Thu Jun 08 2017 06:30:26 GMT+0530 (India Standard Time)
```

Moment with String Format

The table given below shows the format details for year, month, and day.

Format	Example	Details
YYYY	2018	Displays 4 digit year
YY	18	Displays 2 digit year
Q	1-4	Displays the Quarter
M or MM	1-12	Month number
MMM or MMMM	Jan-Dec or January - December	Name of the month
D or DD	1-31	Day of month
Do	1st-31st	Day of month with ordinal
DDD or DDDD	1-365	Day of year
X	1598773566.565	Unix Timestamp
x	1598773566565	Unix Timestamp in milliseconds

The table given below shows the format details for week, weekyear and week days:

Input	Example	Details
gggg	2018	Locale 4 digit week year
gg	18	Locale 2 digit week year
w or ww	1-53	Week of the year
e	0-6	Day of week

ddd or dddd	Mon-Sun or Monday-Sunday	Name of the day in the week
GGGG	2018	ISO 4 digit year
GG	18	ISO 2 digit year
W or WW	1-53	ISO week of the year
E	1-7	ISO day of the week

The table given below shows the format details for hour, minute, seconds , milliseconds:

Format	Example	Details
H or HH	0-23	24 hrs time
h or hh	1-12	12 hrs time
k or kk	1-24	24 hrs time starting from 1
a A	am pm	Post or ante meridian
m or mm	0-59	minutes
s or ss	0-59	seconds
S or SS or SSS	0-999	Fractional seconds
Z or ZZ	+12:00	Offset from UTC as +-HH:mm, +-HHmm, or Z

You can check if date is valid as per the string formats using the command as shown:

```
var day = moment('2018.05.25', 'YYYY-MM-DD').isValid();
```

As you can observe in the output shown below, this will return **true** since the date is in proper format: YYYY, MM, and DD.

```
Output :true
```

If the same date is changed as given below, the output will be **false,** as shown below:

```
var day = moment('05.25', 'YYYY-MM-DD').isValid();
```

```
Output :false
```

You can also format the date as per your requirement as shown in the following examples:

Example 1

```
var day = moment('2018/05/25').format("YYYY-MM-DD");
```

Output

```
Output :2018-05-25
```

Example 2

```
var day = moment('20170608T063026').format("YYYY-MM-DD HH:mm:ss");
```

Output

```
Output :2017-06-08 06:30:26
```

Example 3

```
var day = moment("634", "Hmm").format("HH:mm");
```

Output

```
Output :06:34
```

Example

It is possible to parse multiple formats, where the formats are passed in array form as shown below:

```
var day = moment("12-25-1995",["MM-DD-YYYY", "YYYY-MM-DD"]).isValid();
```

Output

```
Output :true
```

Since the given date matches with one of the formats, the output given is true.

Object

Moment can take the date, time units in object form as shown below:

Syntax

```
moment({unit: value, ...});
```

Example

```
var day = moment({ year :2017, month :10, day :3, hour :15, minute :10, second
:3, millisecond :123});
```

Output

```
Output :Fri Nov 03 2017 15:10:03 GMT+0530 (India Standard Time)
```

Date

We can use new **Date()** JavaScript object inside a moment.

Syntax

```
moment(Date);
```

Example

```
var day = moment(new Date(2018,10,08));
```

Output

```
Output :Thu Nov 08 2018 00:00:00 GMT+0530 (India Standard Time)
```

Array

A moment can be created with input as an array and the values passed in the array to be same as how we would pass to new Date() JavaScript object.

Syntax

```
moment(Number[]);
```

Example

```
var day = moment([2018,10,08, 10, 30,40]);
```

Output

```
Output :Thu Nov 08 2018 10:30:40 GMT+0530 (India Standard Time)
```

Unix Timestamp

Using this, you can pass the integer value in milliseconds and seconds to moment.

Syntax

```
moment(Number);
```

Example

```
var day = moment(1315681876406);
```

Output

```
Output :Sun Sep 11 2011 00:41:16 GMT+0530 (India Standard Time)
```

To use seconds inside moment will have to use it as moment.unix(timestamp)

Syntax

```
moment.unix(Number)
```

Example

```
var day = moment.unix(1968781876);
```

Output

```
Output :Sat May 22 2032 01:21:16 GMT+0530 (India Standard Time)
```

Moment Clone

Moment Clone will create a copy of the moment created.

Syntax

```
moment(Moment);
```

There are 2 ways to clone a moment. Observe the following examples for a better understanding:

Example1

```
var date1 = moment([2017,06,10]);
var date2 = moment(date1);
```

Output

```
Output :Mon Jul 10 2017 00:00:00 GMT+0530 (India Standard Time)
```

```
Output from cloning :Mon Jul 10 2017 00:00:00 GMT+0530 (India Standard Time)
```

Example2

```
var date1 = moment([2017,06,10]);
var date2 = date1.clone();
```

Output

```
Output :Mon Jul 10 2017 00:00:00 GMT+0530 (India Standard Time)
```

```
Output from cloning :Mon Jul 10 2017 00:00:00 GMT+0530 (India Standard Time)
```

UTC

For **moment.utc**, the date will be parsed and displayed in local time. By default, date will be shown in local time. In case you need to parse it in UTC, you can make use of this method.

Syntax

```
moment.utc();
```

moment.utc can take params as number, string, moment, date etc. This method can be used when the date displayed has to be in UTC format.

Example

```
var date1 = moment.utc();
```

Output

```
Output :Sun Apr 15 2018 09:41:07 GMT+0530 (India Standard Time)
```

Any moment created with **moment.utc()** will be in UTC mode, and any moment created with **moment()** will be not.

The following example shows the hours fetched from local and utc moment:

Example

```
var a = moment().hours();
var b = moment.utc().hours();
```

Output

```
Output from local moment:9
```

```
Output from utc moment :4
```

Observe an additional example showing the display of date in utc and local mode:

Example

```
var a = moment().format();
var b = moment.utc().format();
```

Output

```
Output from local moment:2018-04-15T09:54:00+05:30
```

```
Output from utc moment :2018-04-15T04:24:00Z
```

parseZone

This method parses the string provided and keeps the resulting moment in a fixed timezone.

Syntax

```
moment.parseZone()
moment.parseZone(String)
```

Example

```
var a = moment().parseZone();
```

Output

```
Output from local moment:1523766759169
```

moment.parseZone is similar to using moment.utcOffset with a string.

Example

```
var a = moment("2013-01-01T00:00:00-13:00").utcOffset("2013-01-01T00:00:00-13:00");
var b = moment.parseZone("2013-01-01T00:00:00-13:00");
```

Output

```
Output from local moment:1357045200000
```

```
Output from utc moment :1357045200000
```

Creation Data

This method allows to access the data, that is the input provided once the moment is created. It returns the object with the inputs.

Syntax

```
moment().creationData();
```

Observe the following example to understand the working of creationData in MomentJS:

Example

```
var m = moment("2013-01-02", "YYYY-MM-DD").creationData();
 m.input;
 m.format;
```

Output

```
Input: 2018-01-02
```

```
Format: YYYY-MM-DD
```

Defaults

When we create moment, it gives the output as the current day, month, year, hour, minutes and seconds. We can specify the units to be default one while creating moment and the same will be displayed keeping rest same as per the current date/time.

Example

```
var m = moment({hour: 3, minute: 40, seconds: 10});
```

Note that in the above example we have defaulted the hour, minute and seconds to 03:40:10 and the same is displayed keeping rest the current date .

Output

```
Default Output : Sun Apr 15 2018 03:40:10 GMT+0530 (India Standard Time)
```

5. MomentJS - Date Validation

MomentJS handles date validation in an easy way. You need not write lots of code to validate date. **isValid()** is the method available on moment which tells if the date is valid or not. MomentJS also provides many parsing flags which can be used to check for date validation.

Parsing Flags

MomentJS provides the following parsing flags in cases where the date given is considered as invalid:

overflow: This will occur when the month given is 13th, day is 367th in an year or 32nd in a month, 29th for Feb on a non-leap year etc. Overflow contains the index of the invalid unit to match towards **invalidAt**. Note that **-1** means no overflow.

invalidMonth: It shows an invalid month name. It will give the invalid month string or null.

Empty: When an input is given which is not a date. It gives a Boolean.

nullInput: A null input, like moment(null);It returns a Boolean.

invalidFormat: When the format given is empty such as **moment('2018-04-25', [])**. It gives Boolean back.

userInvalidated: A date created explicitly as invalid, such as **moment.invalid()**. It returns Boolean.

meridiem: Indicates the meridiem (AM/PM) parsed, if any. It returns string.

parsedDateParts: It returns an array of date parts parsed such as **parsedDateParts[0]** as year, **parsedDateParts[1]** as month and **parsedDateParts[2]** as day. If no parts are present, but meridiem has value, date is invalid. It returns an array.

Consider the following example to understand date validation:

```
var a = moment("2018-18-10T10:20:25");
a.isValid();
a.invalidAt();
```

Output

```
check for isvalid:false
```

```
check for invalidAt :1
```

The invalidAt gives the output as 1 , which points to the month as the month value is greater than 12 and it overflows. If there is an overflow, invalidAt will give the output as shown in the table given here:

0	years
1	months
2	days
3	hours
4	minutes
5	seconds
6	milliseconds

If there are multiple overflows in the date given, it will be an output for the first overflowed index.

6. MomentJS - Getter/Setter

MomentJS has many methods to get/set the date inputs. Get will allow us to read the required input unit and set will allow to modify the input unit. This chapter discusses in detail the get/set methods to be used on the moment.

The following table shows the get/set methods available:

Method	Syntax

Millisecond	moment().millisecond(Number);
	moment().millisecond();
	moment().milliseconds(Number);
	moment().milliseconds();

Second	moment().second(Number);
	moment().second();
	moment().seconds(Number);
	moment().seconds();

Minute	moment().minute(Number);
	moment().minute();
	moment().minutes(Number);
	moment().minutes();

Hour	moment().hour(Number);
	moment().hour();
	moment().hours(Number);
	moment().hours();

Date of Month	moment().date(Number);
	moment().date();
	moment().dates(Number);
	moment().dates();

Day of week	moment().day(Number|String); moment().day();

	moment().days(Number|String); moment().days();
Day of year	moment().dayOfYear(Number); moment().dayOfYear();

Week of year	moment().week(Number);
	moment().week();
	moment().weeks(Number);
	moment().weeks();

Week of year (ISO)	moment().isoWeek(Number);
	moment().isoWeek();
	moment().isoWeeks(Number);
	moment().isoWeeks();

Month	moment().month(Number|String); moment().month();

Quarter	moment().quarter();
	moment().quarter(Number);
	moment().quarters();
	moment().quarters(Number);

Year	moment().year(Number); moment().year();
Week year	moment().weekYear(Number); moment().weekYear();
Weeks in year	moment().weeksInYear();

Get	moment().get('year');
	moment().get('month');
	moment().get('date');
	moment().get('hour');

	moment().get('minute');
	moment().get('second');
	moment().get('millisecond');

Set	moment().set(String, Int); moment().set(Object(String, Int));
Maximum	moment.max(Moment[,Moment...]); moment.max(Moment[]);
Minimum	moment.min(Moment[,Moment...]); moment.min(Moment[]);

Let us discuss each of these methods in detail:

Millisecond

This method will get / set the milliseconds. To set millisecond it takes value from 0-999, if the range is beyond, then it will add to the seconds.

Syntax

```
moment().millisecond(Number);
moment().millisecond();
moment().milliseconds(Number);
moment().milliseconds();
```

Example 1

```
var m = moment().millisecond();
```

Output

Milliseconds: 697

Example 2

```
var m = moment().millisecond(555);
```

Output

```
Output : 1523780327555
```

Second

This method will get/set the seconds. It takes input from 0-59; if greater than the range provided, then it will add to the minutes.

Syntax

```
moment().second(Number);
moment().second();
moment().seconds(Number);
moment().seconds();
```

Example

```
var m = moment().second(); //gives current second
var d = moment().second(10); //sets the second to 10 as shown below
```

Output

```
Get seconds : 15
```

```
Set seconds : Sun Apr 15 2018 13:54:10 GMT+0530 (India Standard Time)
```

Minute

This method will get/set the minutes. It takes input from 0-59, if greater than the range provided, then it will add to the hour.

Syntax

```
moment().minute(Number);
moment().minute();
moment().minutes(Number);
moment().minutes();
```

Example

```
var m = moment().minute(); // gets the current minute
 var d = moment().minute(5); // sets the minute as shown below
```

Output

```
Get minutes : 56
```

```
Set minutes : Sun Apr 15 2018 13:05:25 GMT+0530 (India Standard Time)
```

Hour

This method will get/set the hour. It takes input from 0-23, if greater than the range provided, it will add to the day.

Syntax

```
moment().hour(Number);
moment().hour();
moment().hours(Number);
moment().hours();
```

Example

```
var m = moment().hour(); // gets the current hour
 var d = moment().hour(8); // sets the hour as shown below
```

Output

```
Get hour : 13
```

```
Set hour : Sun Apr 15 2018 08:59:29 GMT+0530 (India Standard Time)
```

Date of Month

This method will get/set the day of the month. It takes input from 1-31, if greater than the range provided it will add to the next month.

Syntax

```
moment().date(Number);
moment().date();
moment().dates(Number);
moment().dates();
```

Example

```
var m = moment().date(); // gets the current day of the month
 var d = moment().date(2); // sets the day of month as shown below
 var k = moment().date(40); //sets the day of month which is greater than the
range so the output shows the next month as shown in the output
```

Output

```
Date of month: 19
```

```
Date of month: Mon Apr 02 2018 09:49:05 GMT+0530 (India Standard Time)
```

```
Date of month: Thu May 10 2018 09:49:05 GMT+0530 (India Standard Time)
```

Day of Week

This method will get/set the day of the week. It takes input from 0-6, where 0 is for Sunday and 6 as Saturday. If the value is greater than the range, it will fall in the next week. You can set the day of week using number or string.

Syntax

```
moment().day(Number|String);
moment().day();
moment().days(Number|String);
moment().days();
```

Example

```
var m = moment().day(); // gives 4 for thursday
var d = moment().day(0); //shows sunday
var a = moment().day('Monday'); //set the day of week to monday
var k = moment().day(10); //since it greater than 0-6 it sets to the next week
and outputs Wed.
var o = moment().day(-5); // since the value is -ve it will set for last week
```

Output

```
Day of week: 4
```

```
Day of week: Sun Apr 15 2018 09:54:05 GMT+0530 (India Standard Time)
```

```
Day of week: Mon Apr 16 2018 09:54:05 GMT+0530 (India Standard Time)
```

```
Day of week: Wed Apr 25 2018 09:54:05 GMT+0530 (India Standard Time)
```

```
Day of week: Tue Apr 10 2018 09:54:05 GMT+0530 (India Standard Time)
```

Day of week (Locale)

This gets or sets the day of the week according to the locale.

Syntax

```
moment().weekday(Number);
moment().weekday();
```

As per the locale, if Sunday is set as the first day of week, you will have to set **moment.weekday(0)** to Sunday. If Monday is the first day of week you will see **moment.weekday(0)** to set as Monday.

The working of it remains same as day of week where if greater than range it will set to next week , if -ve value it will go for last week.

Example

```
var m = moment().weekday();
 var d = moment().weekday(4);
 var a = moment().weekday('Monday');
 var k = moment().weekday(10);
 var o = moment().weekday(-5);
```

Output

```
Day of week(locale): 4
```

```
Day of week(locale): Thu Apr 19 2018 09:56:52 GMT+0530 (India Standard Time)
```

```
Day of week(locale): Thu Apr 19 2018 09:56:52 GMT+0530 (India Standard Time)
```

```
Day of week(locale): Wed Apr 25 2018 09:56:52 GMT+0530 (India Standard Time)
```

```
Day of week(locale): Tue Apr 10 2018 09:56:52 GMT+0530 (India Standard Time)
```

ISO Day of week

This method will set / get the day of week as per ISO where 1 is Monday and 7 is Sunday. So the range is 1-7 and anything greater than the range will fall in the next week and less than the range will fall in the last week.

Syntax

```
moment().isoWeekday(Number);
moment().isoWeekday();
```

Example

```
var m = moment().isoWeekday();
var d = moment().isoWeekday(4);
var a = moment().isoWeekday('Monday');
var k = moment().isoWeekday(10);
var o = moment().isoWeekday(-5);
```

```
ISO Day of week: 4
```

```
ISO Day of week: Thu Apr 19 2018 09:58:13 GMT+0530 (India Standard Time)
```

```
ISO Day of week: Mon Apr 16 2018 09:58:13 GMT+0530 (India Standard Time)
```

```
ISO Day of week: Wed Apr 25 2018 09:58:13 GMT+0530 (India Standard Time)
```

```
ISO Day of week: Tue Apr 10 2018 09:58:13 GMT+0530 (India Standard Time)
```

Day of Year

This method will take care of getting and setting the day of year. The range given is 0-366. If the input to set is greater than the range, it will show up the next year.

Syntax

```
moment().dayOfYear(Number);
moment().dayOfYear();
```

Example

```
var m = moment().dayOfYear();
var d = moment().dayOfYear(320);
var a = moment().dayOfYear(400);
```

Output

```
Day of year: 109
```

```
Day of year: Fri Nov 16 2018 09:59:28 GMT+0530 (India Standard Time)
```

```
Day of year: Mon Feb 04 2019 09:59:28 GMT+0530 (India Standard Time)
```

Week of Year

This method will get/set the week of the year. The output will depend on the locale being used, as first day of week differs from country to country.

Syntax

```
moment().week(Number);
moment().week();
moment().weeks(Number);
moment().weeks();
```

Example

```
var m = moment().week();
var d = moment().week(50);
var a = moment().week(90);
```

Output

```
Week of year: 16
```

```
Week of year: Thu Dec 13 2018 10:00:39 GMT+0530 (India Standard Time)
```

```
Week of year: Thu Sep 19 2019 10:00:39 GMT+0530 (India Standard Time)
```

Week of Year (ISO)

This method gets/sets the week of the year as per the ISO.

Syntax:

```
moment().isoWeek(Number);
moment().isoWeek();
moment().isoWeeks(Number);
moment().isoWeeks();
```

Example

```
var m = moment().isoWeek();
var d = moment().isoWeek(50);
var a = moment().isoWeek(90);
```

Output

```
Week of year ISO: 16
```

```
Week of year ISO: Thu Dec 13 2018 10:01:20 GMT+0530 (India Standard Time)
```

```
Week of year ISO: Thu Sep 19 2019 10:01:20 GMT+0530 (India Standard Time)
```

Month

This method gets/sets the month. The range is from 0-11, if greater it will show the next year, if negative it will show the last year.

Syntax

```
moment().month(Number|String);
moment().month();
```

It will take number or string to set the month.

Example

```
var m = moment().month();
var d = moment().month(8);
var a = moment().month("November");
var k = moment().month(20);
var o = moment().month(-5);
```

Output

```
Month: 3
```

```
Month: Wed Sep 19 2018 10:03:00 GMT+0530 (India Standard Time)
```

```
Month: Mon Nov 19 2018 10:03:00 GMT+0530 (India Standard Time)
```

```
Month: Thu Sep 19 2019 10:03:00 GMT+0530 (India Standard Time)
```

```
Month: Sat Aug 19 2017 10:03:00 GMT+0530 (India Standard Time)
```

Quarter

This method will get and set the quarter. The range is 1-4. Inputs greater than the range will fall in the next year and less will fall in the previous years.

Syntax

```
moment().quarter();
moment().quarter(Number);
moment().quarters();
moment().quarters(Number);
```

Example

```
var m = moment().quarter();
var d = moment().quarter(3);
var a = moment().quarter(6);
var k = moment().quarter(-1);
```

Output

```
Quarter: 2
```

```
Quarter: Thu Jul 19 2018 10:04:03 GMT+0530 (India Standard Time)
```

```
Quarter: Fri Apr 19 2019 10:04:03 GMT+0530 (India Standard Time)
```

```
Quarter: Wed Jul 19 2017 10:04:03 GMT+0530 (India Standard Time)
```

Year

This method will get/set the year. The range for year is -270,000 to 270,000.

Syntax

```
moment().year(Number);
moment().year();
```

Example

```
var m = moment().year();
var d = moment().year(2020);
```

Output

```
Year: 2018
```

```
Year: Sun Apr 19 2020 10:04:51 GMT+0530 (India Standard Time)
```

Week Year

This method will get/set the week of the year. Its setting will depend on the locale. It will differ from based on country.

Syntax

```
moment().weekYear(Number);
moment().weekYear();
```

Example

```
var m = moment().weekYear();
var d = moment().weekYear(2020);
```

Output

```
Week of Year: 2018
```

```
Week of Year: Thu Apr 16 2020 10:05:31 GMT+0530 (India Standard Time)
```

Week Year (ISO)

This method will get/set the week year in the ISO form.

Syntax

```
moment().isoWeekYear(Number);
moment().isoWeekYear();
```

Example

```
var m = moment().isoWeekYear();
var d = moment().isoWeekYear(2020);
```

Output

```
ISO Week of Year: 2018
```

```
ISO Week of Year: Thu Apr 16 2020 10:06:14 GMT+0530 (India Standard Time)
```

Weeks in Year

This method will get the number of weeks according to locale in the current moment's year.

Syntax

```
moment().weeksInYear();
```

Example

```
var m = moment().weeksInYear();
```

Output

```
Weeks in Year: 52
```

Weeks in Year (ISO)

This method will get the number of weeks according to the ISO in the current moment's year.

Syntax

```
moment().isoWeeksInYear();
```

Example

```
var m = moment().isoWeeksInYear();
```

Output

```
Weeks in Year ISO: 52
```

Get

This method gets the year, month, date, hour, minute, second, and millisecond.

Syntax

```
moment().get('year');
moment().get('month');
moment().get('date');
moment().get('hour');
moment().get('minute');
moment().get('second');
moment().get('millisecond');
```

Example

```
var m = moment().get('year');
var d = moment().get('month');
var a = moment().get('date');
var k = moment().get('hour');
var o = moment().get('minute');
var s = moment().get('second');
```

```
var ms = moment().get('millisecond');
```

Output

Get Year: 2018

Get Month: 3

Get Date: 19

Get Hour: 10

Get Minute: 10

Get Second: 22

Get Millisecond: 162

For the **get** method, the units taken are shown in the table below. Note that the units can be case insensitive or plural. Observe the table shown below:

year or years or y
month or months or M
date or dates or D
hour or hours or H
Minute or minutes or m
second or seconds or s
millisecond or milliseconds or ms

Set

This method will set the unit with the value provided. The unit is the first param and the value as the second. It can also take the unit, value pair in object form.

Syntax

```
moment().set(String, Int);
moment().set(Object(String, Int));
```

Example

```
var m = moment().set('year', 2020);
var d = moment().set('month', 5);
var a = moment().set('date', 28);
var k = moment().set({hour:10, minute:25, second:50});
```

Output

```
Set Year: Sun Apr 19 2020 10:12:04 GMT+0530 (India Standard Time)

Set Month: Tue Jun 19 2018 10:12:04 GMT+0530 (India Standard Time)

Set Date: Sat Apr 28 2018 10:12:04 GMT+0530 (India Standard Time)

Set hour:minute:second: Thu Apr 19 2018 10:25:50 GMT+0530 (India Standard Time)
```

Maximum

This method gives the maximum from the moment instance.

Syntax

```
moment.max(Moment[,Moment...]);
moment.max(Moment[]);
```

Example

```
var m = moment([2020,10,11]);
var k = moment();
var c = moment.max(m,k);
```

Output

```
Maximum: Wed Nov 11 2020 00:00:00 GMT+0530 (India Standard Time)
```

Minimum

This method gives the minimum from the moment instance.

Syntax

```
moment.min(Moment[,Moment...]);
moment.min(Moment[]);
```

Example

```
var m = moment([2020,10,11]);
var k = moment();
var c = moment.min(m,k);
```

Output

```
Minimum: Thu Apr 19 2018 10:14:13 GMT+0530 (India Standard Time)
```

7. MomentJS - Manipulate Date and Time

MomentJS provides various methods to manipulate date and time on the moment object. This chapter deal with all such methods in detail.

Methods to Manipulate Date and Time

The following table shows the methods available in MomentJS which helps in manipulating the date and time as required:

Add	moment.add()
Subtract	moment.subtract()
Start of Time	moment.startof()
End of Time	moment.endof()
Local	moment.local()
UTC	moment.utc()
UTC offset	moment.utcOffset()

In this section, let us discuss each of these methods in detail:

Add

This method allows you add days, time, years, hours, seconds etc., on the moment object.

Syntax

```
moment().add(Number, String);
moment().add(Object);
moment().add(Duration);
```

Using the add method we can add number, hours, days etc.

The following table shows the lists of keys/shorthand which you can use with add method.

Key	Shorthand
years	y
quarters	Q
months	M
weeks	w
days	d
hours	h
minutes	m
seconds	s
milliseconds	ms

You can make use of key or shorthand unit inside the add method as follows:

Example

```
moment.add(5, 'days'); or moment.add(5, 'd');
```

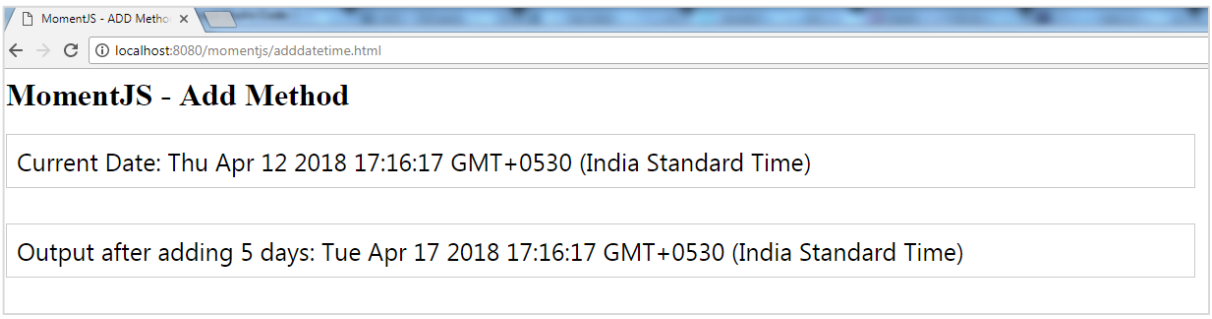
Let see a working example which shows to add days to the current date using add method.

Example

```
<html>
<head>
    <title>MomentJS - ADD Method</title>
    <scrip type="text/JavaScript"
src="https://MomentJS.com/downloads/moment.js"></script>
    <style>
        div {            border: solid 1px #ccc;
            padding:10px;
```

```
            font-family: "Segoe UI",Arial,sans-serif;
            width: 75%;
        }
    </style>
</head>
<body>
    <h1>MomentJS - Add Method</h1>
    <div style="font-size:25px" id="currentdate"></div>
     <br/>
     <br/>
     <div style="font-size:25px" id="changeddate"></div>
     <script type="text/JavaScript">
            var day = moment(); //outputs current date.
            document.getElementById("currentdate").innerHTML = "Current Date: "
+ day._d;
            var changeddate = moment().add(5, 'days'); // adding 5 days to
current date.
            document.getElementById("changeddate").innerHTML = "Output after
adding 5 days: " + changeddate._d;
     </script>
</body>
</html>
```

Output



Note that the above code displays the current date and the date after adding 5 days to it.

You can also use the **key** with add method as follows:

```
var changeddate = moment().add(5, 'days'); // adding 5 days to current date.
```

Example

Let see an example which adds 5 hours to the given date:

```
var changeddate = moment([2017, 2, 31]).add(5, 'hours');
```

Output

```
Output : Sat Mar 31 2012 05:00:00 GMT+0530 (India Standard Time)
```

If there are multiple additions to be done to the date, you can do it using add method chaining or using object literal.

Add Method using Chaining

Consider you want to add days and months to the current date. It can be done using method chaining as shown in below example:

Example

```
var changeddate = moment().add(5, 'days').add(2, 'months');
```

Output

```
Current Date: Sat Apr 14 2018 14:51:51 GMT+0530 (India Standard Time)

Output after adding 5 days and 2 month: Tue Jun 19 2018 14:51:51 GMT+0530 (India
Standard Time)
```

To add days and months to the current date, we can use method chaining as follows:

```
var changeddate = moment().add(5, 'days').add(2, 'months');
```

We can also use key as shown in the code given below:

```
var changeddate = moment().add(5, 'd').add(2, 'M');
```

Add Method using Object

Using this technique, you can use object literal for adding multiple keys to the current date.

Example

```
var changeddate = moment().add({ days: 5, months: 2 });
```

Output

```
Current Date: Sat Apr 14 2018 14:51:51 GMT+0530 (India Standard Time)
```

```
Output after adding 5 days and 2 month: Tue Jun 19 2018 14:51:51 GMT+0530 (India Standard Time)
```

The object method is used as follows:

```
var changeddate = moment().add({ days: 5, months: 2 });
```

You can also use keys in the object form as follows:

```
var changeddate = moment().add({ d: 5, M: 2 });
```

In case, we need to add days or months to a given date, the same can be done as shown below:

Example

```
var changeddate = moment([2014, 10, 10]).add({ d: 5, M: 2 });
```

Output

```
Output after adding 5 days and 2 month: Thu Jan 15 2015 00:00:00 GMT+0530 (India Standard Time)
```

We have added 5 days and 2 months to the date 10/10/2014 which gives the output as 15/01/2015.

Adding Duration to Add Method

We can also use duration method to add days, months, years, time etc. to a given date.

Observe the following example that shows how to add 5 weeks to a given date using duration:

Example

```
var duration = moment.duration({ 'weeks': 5 });
var changeddate = moment([2012, 0, 31]).add(duration);
```

Note that we have added 5 weeks to 31/01/2012 and thus the output is as follows:

Output

```
Output : Tue Mar 06 2012 00:00:00 GMT+0530 (India Standard Time)
```

Special Cases for Months and Years

In case we are trying to add months to the date whose days are greater than the months added, it will take the last day of the month which is added to the date.

Example

```
var changeddate = moment([2017, 0, 31]).add(1, 'months');
```

In the above example, we are adding one month to 31/01/2017, now since February has 28 days it takes the last day of February and displays the date as shown below:

```
Output : Tue Feb 28 2017 00:00:00 GMT+0530 (India Standard Time)
```

Subtract

Just like the add method, **subtract** allows to subtract days, months, hours, minutes, seconds etc., from a given date.

Syntax

```
moment().subtract(Number, String);
moment().subtract(Duration);
moment().subtract(Object);
```

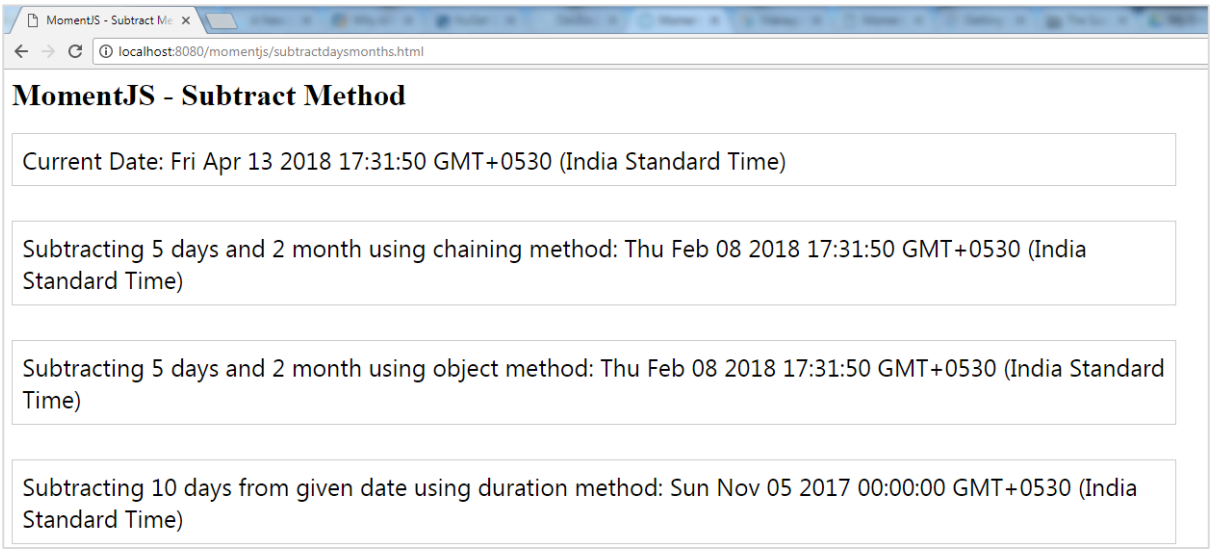
Observe the following example that shows how to use the subtract method:

Example

```
<html>
<head>
    <title>MomentJS - Subtract Method</title>
    <script type="text/JavaScript"
src="https://MomentJS.com/downloads/moment.js"></script>
    <style>
        div {
            border: solid 1px #ccc;
            padding:10px;
            font-family: "Segoe UI",Arial,sans-serif;
            width: 75%;
        }
    </style>
</head>
<body>
     <h1>MomentJS - Subtract Method</h1>
     <div style="font-size:25px" id="currentdate"></div>
     <br/>
     <br/>
     <div style="font-size:25px" id="changeddate"></div>
     <br/>
     <br/>
     <div style="font-size:25px" id="changeddate1"></div>
     <br/>
     <br/>
     <div style="font-size:25px" id="changeddate2"></div>
     <script type="text/JavaScript">
            var day = moment();
            document.getElementById("currentdate").innerHTML = "Current Date: "
+ day._d;
            var changeddate = moment().subtract(5, 'days').subtract(2,
'months');
            document.getElementById("changeddate").innerHTML = "Subtracting 5
days and 2 month using chaining method: " + changeddate._d;

            var changeddate1 = moment().subtract({ days: 5, months: 2 });
```

```
            document.getElementById("changeddate1").innerHTML = "Subtracting 5
days and 2 month using object method: " + changeddate1._d;

            var duration = moment.duration({ 'days': 10 });

            var changeddate2 = moment([2017, 10, 15]).subtract(duration);

            document.getElementById("changeddate2").innerHTML = "Subtracting 10
days from given date  using duration method: " + changeddate2._d;

      </script>

</body>

</html>
```

Output



To subtract days, months from the date we have done following:

```
//chaining subtract method
var changeddate = moment().subtract(5, 'days').subtract(2, 'months');


// subtract object method
var changeddate1 = moment().subtract({ days: 5, months: 2 });


//using duration in subract method
 var duration = moment.duration({ 'days': 10 });
 var changeddate2 = moment([2017, 10, 15]).subtract(duration);
```

The output for the same in shown in the above example.

Start of Time

Start of Time method will set the display to the start of the unit given.

Syntax

```
moment().startOf(String);
```

Example using year as the input

```
var day = moment().startOf('year');
```

This will display the first day of the year and will set the time, that is hours, minutes, seconds to 00:00:00 as shown below:

```
start of time using year: Mon Jan 01 2018 00:00:00 GMT+0530 (India Standard Time)
```

Example using month as the input

```
var day = moment().startOf('month');
```

This will display the first day of the month and will set the time, that is hours, minutes and seconds to 00:00:00 as shown below:

```
start of time using month: Sun Apr 01 2018 00:00:00 GMT+0530 (India Standard Time)
```

Example using quarter as the input

```
var day = moment().startOf('quarter');
```

This will give the details of the start of the current quarter. Time will be set to 12 am.

Output

```
start of time using quarter: Sun Apr 01 2018 00:00:00 GMT+0530 (India Standard Time)
```

If we check the quarter for the month of February using the command shown here, the output will be from 1st of Jan as shown below:

```
var day = moment([2018,01,10]).startOf('quarter');
```

Output

```
start of time using quarter: Mon Jan 01 2018 00:00:00 GMT+0530 (India Standard Time)
```

Example using week as the input

```
var day = moment().startOf('week');
```

Output

```
start of time using week: Sun Apr 08 2018 00:00:00 GMT+0530 (India Standard Time)
```

End of Time

End of Time method will set the display to the end of the unit given.

Syntax

```
moment().endtOf(String);
```

Example using year as the input

```
var day = moment().endOf('year');
```

This will display the last day of the year and will set the time to 23:59:59 as shown below:

```
end of time using year: Mon Dec 31 2018 23:59:59 GMT+0530 (India Standard Time)
```

Example using month as the input

```
var day = moment().endOf('month');
```

This will display the last day of the month and will set the time ie hours, minutes and seconds to 23:59:59 as shown below:

```
end of time using month: Mon Apr 30 2018 23:59:59 GMT+0530 (India Standard Time)
```

Local

With the local method, it gives time to display a moment instead of the original moment's time.

Syntax

```
moment().local();
```

Example

```
var day = moment().local();
```

Output

```
Date displayed : Sat Apr 14 2018 11:57:16 GMT+0530 (India Standard Time)
current hour:11 current minute:57
```

UTC

This method sets a flag on the original moment to use UTC to display a moment instead of the original moment's time.

Syntax

```
moment().utc();
```

Example

```
var day = moment().utc();
```

Output

```
Sat Apr 14 2018 12:12:10 GMT+0530 (India Standard Time)
```

UTC offset

This method gives the display in minutes.

Syntax

```
moment().utcOffset();
moment().utcOffset(Number|String);
```

Observe the following examples for a better understanding:

Example 1

```
var day = moment().utcOffset();
```

Output

```
UTC offset :330
```

Example 2

We can use pass value to utcOffset method as follows:

```
var day = moment().utcOffset(120);
```

In the above, we are adding 120 minutes to the current moment and its output is displayed as below showing the current date/time and after adding offset.

Output

```
Current date/time :Sat Apr 14 2018 12:53:14 GMT+0530 (India Standard Time)

After adding UTC offset of 120 :Sat Apr 14 2018 14:53:14 GMT+0530 (India Standard Time)
```

Example 3

When the value given to the utcOffset is less than 16 and greater than -16, it will consider it as hours and the hours in the date are changed as shown below:

```
var day = moment().utcOffset(8);
```

The output is as follows:

```
Current date/time :Sat Apr 14 2018 14:36:51 GMT+0530 (India Standard Time)

After adding UTC offset of 8 :Sat Apr 14 2018 22:36:51 GMT+0530 (India Standard Time)
```

Observe that the current date shows hours as 14 and after adding 8 in utcOffset it changes the hours to 22.

Example 4

We can also pass offset value as string as shown below:

```
var day = moment().utcOffset("+05:50");
```

To the offset, we are adding 5 hrs and 50 mins as string and its is output as shown below:

```
Current date/time :Sat Apr 14 2018 14:40:42 GMT+0530 (India Standard Time)

After adding UTC offset as string :Sat Apr 14 2018 20:30:42 GMT+0530 (India Standard
Time)
```

8. MomentJS – Formatting Date and Time

MomentJS provides formats to display date in different ways. There are methods available which tells the time from a given moment, from the current moment, difference between two moments etc. It can display date in JSON format, Array, Object, String etc.

Methods to Format Date and Time

The following table shows a list of methods available which helps in the displaying/formatting of the date as required.

Methods	Syntax
Format	moment().format(); moment().format(String);
Time from now	moment().fromNow(); moment().fromNow(Boolean);
Time from X	moment().from(Moment|String|Number|Date|Array);
Time to now	moment().toNow(); moment().toNow(Boolean);
Time to X	moment().to(Moment|String|Number|Date|Array); moment().to(Moment|String|Number|Date|Array, Boolean);

Calendar Time	moment().calendar();
	moment().calendar(referenceTime);
	moment().calendar(referenceTime, formats);

Difference	moment().diff(Moment|String|Number|Date|Array);
	moment().diff(Moment|String|Number|Date|Array, String);
	moment().diff(Moment|String|Number|Date|Array, String, Boolean);

Unix Timestamp(milliseconds)	moment().valueOf();

	+moment();
Unix Timestamp(seconds)	moment().unix();
Days in Month	moment().daysInMonth();
As JavaScript Date	moment().toDate();
As Array	moment().toArray();
As JSON	moment().toJSON();
As ISO 8601 String	moment().toISOString(); moment().toISOString(keepOffset);
As Object	moment().toObject();
As String	moment().toString();
Inspect	moment().inspect();

Format

This method will display the date/time details. It displays output based on the input. For example, **moment().format("MMMMD")** will display April for MMMM, that is the current month and current date for D. So the output is **April16**. With format, it tries to convert the units given to the corresponding display of date/time.

Syntax

```
moment().format();
moment().format(String);
```

Observe the following examples to gain a better understanding on displaying date using the **format** method.

Example

```
var changeddate = moment().format();
```

Output

```
Output : 2018-04-16T11:07:35+05:30
```

Note that when you use only the format method, it displays current date and time as shown above.

The following table shows a list of tokens to be taken as input string for format method:

Unit	Token	Output
Month	M	1-12
	Mo	1st, 2nd -12th
	MM	01-12
	MMM	Jan-Dec
	MMMM	January-December
Quarter	Q	1-4
	Qo	1st-4th
Day of Month	D	1-31
	Do	1st-31st
	DD	01-31
Day of Year	DDD	1-365
	DDDo	1st-365th
	DDDD	001-365
Day of Week	d	0-6
	do	0th-6th
	dd	Su,Mo,Tu,We,Th,Fr,Sa
	ddd	Sun-Sat

	dddd	Sunday-Saturday
Day of Week (locale)	e	0-6
Day of Week(ISO)	E	1-7
Week of Year	w	1-53
	wo	1st-53rd
	ww	01-53
Week of Year(ISO)	W	1-53
	Wo	1st-53rd
	WW	01-53
Year	YY	70,71---29,30
	YYYY	1970-2030
	Y	1970-9999
Week Year	gg	70,71 - 29,30
	gggg	1970,1971-2030
Week Year (ISO)	GG	70,71 - 29,30
	GGGG	1970,1971-2030
AM/PM	A	AM, PM
	a	am,pm
Hour	H	0-23
	HH	00-23
	h	1-12
	hh	01-12

	k	1-24
	kk	01-24
Minute	m	0-59
	mm	00-59
Second	s	0-59
	ss	00-59
Fractional Second	S	0-9
	SS	00-99
	SSS	000-999
	SSSS….	0000..-9999...
Time Zone	Z	-07:00 -06:00 ... +06:00 +07:00
	ZZ	-0700 -0600 ... +0600 +0700
Unix Timestamp	X	1360013296
Unix Millisecond Timestamp	x	1360013296123

The following table shows a list of tokens to be used on moment based on locale:

Unit	**Token**	**Output**
Time	LT	2:58 PM
Time with seconds	LTS	2:58:25 PM
Month numeral, day of month, year	L	16/04/2018
	I	16/4/2018
Month name, day of month, year	LL	April 16, 2018

	II	Apr 16, 2018
Month name, day of month, year, time	LLL	April 16, 2018 2:58 PM
	III	Apr 16, 2018 2:58 PM
Month name, day of month, day of week, year, time	LLLL	Monday, April 16, 2018,2:58 PM
	IIII	Mon, Apr 16, 2018, 2:58 PM

Observe the following examples to gain better understanding on token passed to format:

Example 1

```
var changeddate = moment().format("Do dddd MMMM gggg");
```

Output

```
Output : 16th Monday April 2018
```

Example 2

```
var changeddate = moment().format("MMMM Qo DD YYYY");
```

Output

```
Output : April 2nd 16 2018
```

Example 3

You can also add characters to the format method. For this purpose, put them in square brackets as shown below:

```
var changeddate = moment().format("[Today's Date is ] D MMM YYYY");
```

Output

```
Output : Today's Date is 16 Apr 2018
```

Example 4

```
var changeddate = moment().format("[Current Time is ] LTS");
```

Output

```
Output : Current Time is 3:21:14 PM
```

Example 5

```
var changeddate = moment().format("[As per locale the date is ] LLLL");
```

Output

```
Output : As per locale the date is Monday, April 16, 2018 3:23 PM
```

Time from now

This method will tell the length of time from now. Suppose for example if you pass a date to this method it will display the difference if it is in years, months, hours, minutes, or seconds.

Syntax

```
moment().fromNow();
moment().fromNow(Boolean);
```

Example

```
var changeddate = moment([2018, 0, 1]).fromNow();
```

Output

```
Output : 4 months ago
```

Observe that the date given to the moment is 01/01/2018, the difference till date is 4 months, so the output is given as 4 months ago.

Observe the following example to check the output when no inputs are given to the moment:

Example

```
var changeddate = moment().fromNow();
```

Output

```
Output : a few seconds ago
```

Note that in the above shown output, it has the **ago** keyword appended at the end. In case you do not need it, pass **true** to fromNow(**true**) method as shown below:

Example

```
var changeddate = moment([2015, 10, 01]).fromNow();
var changeddate1 = moment([2015, 10, 01]).fromNow(true);
```

Output

```
Without param to fromNow : 2 years ago
```

```
With true param to fromNow : 2 years
```

Time from X

This method will tell the length of time from another moment.

Syntax

```
moment().from(Moment|String|Number|Date|Array);
moment().from(Moment|String|Number|Date|Array, Boolean);
```

Example

```
var a = moment([1998, 10, 01]);
var b = moment([2015, 10, 01]);
var c = a.from(b);
var d = a.from(b, true);
```

OR

Example

```
var a = moment([1998, 10, 01]).from([2015, 10, 01]);
var b = moment([1998, 10, 01]).from([2015, 10, 01], true);
```

OR

Example

```
var a = moment([1998, 10, 01]);
var b = a.from([2015, 10, 01], true);
```

Output

```
Time from : 17 years ago
```

```
With true param to from : 17 years
```

You can also use JavaScript date as shown in the example given below:

Example

```
var a = moment([1995, 10, 01]).from(new Date());
```

Output

```
Time from : 22 years ago
```

If the date given to **from** is less than the checked date, it gives the output with **in** instead of **ago.** Look at the following example for a better understanding:

Example

```
var a = moment(new Date()).from([1995, 10, 01]);
```

Output

```
Time from : in 22 years
```

Time to now

This method tells the length of time from the date given to now.

Syntax

```
moment().toNow();
moment().toNow(Boolean);
```

Example

```
var a = moment([2015, 3, 15]).toNow();
```

Output

```
Time from : in 3 years
```

If you want to remove the prefix, use true with toNow() as shown below:

Example

```
var a = moment([1995, 3, 15]).toNow(true);
```

Output

```
Time from : 23 years
```

Time to X

This method will help to display moment as per in relation with another moment instead of current one.

Syntax

```
moment().to(Moment|String|Number|Date|Array);
moment().to(Moment|String|Number|Date|Array, Boolean);
```

Example

```
var a = moment([1995, 3, 15]);
var b = moment([1998, 10, 01]);
var c = a.to(b);
```

OR

Example

```
var a = moment([1995, 3, 15]).to(moment([1998, 10, 01]));
```

Output

```
Time to X : in 4 years
```

Example2

```
var a = moment([1995, 3, 15]).to(new Date());
```

Output

```
Time to X : in 23 years
```

Example 3

```
var a = moment([2018, 10, 11]).to(moment[2015, 10, 11], true);
```

Output

```
Time to X : 7 months
```

If you want to remove the prefix **in** or **ago** in the display, use **true**.

Calendar Time

This method displays the time difference to given reference time and the Calendar time.

Syntax

```
moment().calendar();
moment().calendar(referenceTime);
moment().calendar(referenceTime, formats);
```

Example 1

```
var changeddate = moment().calendar();
```

Output

Calendar Time: Today at 10:37 AM

Example 2

We have discussed chaining method in earlier chapter, so we can use the add/subtract methods with moment to get the calendar value as shown below:

```
var changeddate = moment().add(24, 'h').calendar();
```

Output

Calendar Time: Tomorrow at 11:13 AM

Example 3

```
var changeddate = moment().subtract(24, 'h').calendar();
```

Output

Calendar Time: Yesterday at 11:15 AM

Difference

This method gives the difference in milliseconds.

Syntax

```
moment().diff(Moment|String|Number|Date|Array);
moment().diff(Moment|String|Number|Date|Array, String);
moment().diff(Moment|String|Number|Date|Array, String, Boolean);
```

This method allows to get the difference in measurements, that is in years, months etc.,

The supported measurements are years, months, weeks, days, hours, minutes, and seconds.

Observe the following examples for a better understanding:

Example 1

```
var a = moment([2000, 2, 15]);
var b = moment([2007, 8, 16]);
var c = a.diff(b);
```

Or

Example

```
var a = moment([2000, 2, 15]).diff(moment([2007, 8, 16]));
```

Output

```
Difference Time: -236822400000
```

Example 2

Here is another example with measurements passed:

```
var a = moment([2010, 2, 15]).diff(moment([2007, 8, 16]), "years");
```

Output

```
Difference Time: 2
```

Example 3

In case you want the difference with the floating point number, pass true as the third parameter to difference as shown in the code here:

```
var a = moment([2010, 2, 15]).diff(moment([2007, 8, 16]), "years", true);
```

Output

```
Difference Time: 2.4972222222222222
```

Unix Timestamp (milliseconds)

This method gives the number of milliseconds since the Unix epoch.

Syntax

```
moment().valueOf();
+moment();
```

Example 1

```
var a = moment().valueOf();
```

Output

```
Unix Timestamp: 1523946683735
```

Example 2

```
var a = +moment();
```

Output

```
Unix Timestamp: 1523946821328
```

Unix Timestamp (seconds)

This method gives the no of seconds since the Unix epoch.

Syntax:

```
moment().unix();
```

Example

```
var a = moment().unix();
```

Output

```
Unix Timestamp: 1523946977
```

This value is floored to the nearest second, and does not include a milliseconds component.

Days in Month

This method will give the numbers of days in the current month.

Syntax

```
moment().daysInMonth();
```

Example

```
var a = moment().daysInMonth();
```

Output

```
Days in a month: 30
```

Example
Let us check the days in a month for a given date using this method. Use the following code for this purpose:

```
var a = moment([2016, 1]).daysInMonth();
```

Output

```
Days in a month: 29
```

For February in 2016, it displays days as 29.

As JavaScript Date

This method gives you the output same as the native JavaScript new Date().

Syntax

```
moment().toDate();
```

Example

```
var a = new Date();
var b = moment().toDate();
```

Output

```
From javascript new Date(): Tue Apr 17 2018 12:21:36 GMT+0530 (India Standard Time)
```

```
From momentjs toDate() : Tue Apr 17 2018 12:21:36 GMT+0530 (India Standard Time)
```

As Array

This method will return the params in array form and the values in it will be same as we get for new Date().

Syntax

```
moment().toArray();
```

Example

```
var a = new Date();
var b = moment().toArray();
```

Output

```
From javascript new Date(): Tue Apr 17 2018 12:25:16 GMT+0530 (India Standard Time)
```

```
From toArray() : 2018,3,17,12,25,16,984
```

As JSON

When this method is used on a moment object it will displayed as an ISO8601 string, adjusted to UTC.

Syntax

```
moment().toJSON();
```

Example

```
var a = moment().toJSON();
```

Output

```
From toJSON() : 2018-04-17T07:13:57.000Z
```

As ISO 8601 String

This method formats a string to the ISO8601 standard. It gives the timestamp in UTC form.

Syntax

```
moment().toISOString();
moment().toISOString(keepOffset);
```

Example

```
var a = moment().toISOString();
```

Output

```
ISO string: 2018-04-17T07:19:27.195Z
```

To avoid the display in UTC form use true in toISOString(true) as shown

Example

```
var a = moment().toISOString(true);
```

Output

```
ISO string: 2018-04-17T12:52:24.289+05:30
```

As Object

This method gives the output in object form containing year, month, day-of-month, hour, minute, seconds, milliseconds.

Syntax

```
moment().toObject();
```

Example

```
var a = moment().toObject();
var objstr = a.years + " " + a.months + " " + a.date + " " + a.hours + " " +
a.minutes + " " + a.seconds;
```

Output

```
As Object: 2018 3 17 12 56 46
```

As String

This will output the date as English string.

Syntax

```
moment().toString();
```

Example

```
var a = moment().toString();
```

Output

```
As String: Tue Apr 17 2018 12:59:13 GMT+0530
```

Inspect

This method gives a machine readable date string which when used again to evaluate the same moment. It is mostly used for debugging purpose.

Syntax

```
moment().inspect();
```

Example

```
var a = moment().inspect();
```

Output

```
Inspect: moment("2018-04-17T15:00:51.373")
```

Example

```
var a = moment(new Date("18/10")).inspect();
```

Output

```
Inspect output: moment.invalid(/* Invalid Date */)
```

Thus, when you have to debug the output of moment, you can inspect its displayed output.

9. MomentJS - Date Queries

MomentJS provides methods to query the date/time for leap year, date comparison, date validation etc. This chapter discusses them in detail.

Methods for Querying Date in MomentJS

The following table shows methods available in MomentJS and their syntax for querying date:

Method	Syntax
Is Before	moment().isBefore(Moment|String|Number|Date|Array); moment().isBefore(Moment|String|Number|Date|Array, String);
Is Same	moment().isSame(Moment|String|Number|Date|Array); moment().isSame(Moment|String|Number|Date|Array, String);
Is After	moment().isAfter(Moment|String|Number|Date|Array); moment().isAfter(Moment|String|Number|Date|Array, String);
Is Same or Before	moment().isSameOrBefore(Moment|String|Number|Date|Array); moment().isSameOrBefore(Moment|String|Number|Date|Array, String);
Is Same or After	moment().isSameOrAfter(Moment|String|Number|Date|Array); moment().isSameOrAfter(Moment|String|Number|Date|Array, String);
Is Between	moment().isBetween(moment-like, moment-like); moment().isBetween(moment-like, moment-like, String);
Is Daylight Saving Time	moment().isDST();
Is Leap Year	moment().isLeapYear();
Is a Moment	moment.isMoment(obj);
Is a Date	moment.isDate(obj);

Is Before

This method checks if the given moment is before another moment. It returns true or false.

Syntax

```
moment().isBefore(Moment|String|Number|Date|Array);
moment().isBefore(Moment|String|Number|Date|Array, String);
```

Example

```
var isbefore = moment().isBefore('2020-10-21');
```

Output

Note that the date used inside **isBefore** is greater than the current date, so we are getting true as the output.

Example

```
var isbefore = moment([2015, 10, 01]).isBefore([2000, 10, 21]);
```

Output

With isBefore, we can compare dates from one moment to another. We can also specify the units with isBefore. The units supported are year, month, week, day, hour, minute and second.

Example

Let us consider a working example with units passed to isBefore();

```
var isbefore = moment([2015, 10, 01]).isBefore([2010, 10, 21], 'year');
```

Output

Is Same

This method will check if the moment is same as another moment. It returns true or false.

Syntax

```
moment().isSame(Moment|String|Number|Date|Array);
moment().isSame(Moment|String|Number|Date|Array, String);
```

Example

```
var issame = moment([2015, 10, 01]).isSame([2015, 10, 01]);
```

Output

Note that as with isBefore, we can use unit with isSame() method. Following are the units supported: year, month, week, day, hour, minute and second.

Example

```
var issame = moment([2015, 10, 01]).isSame([2015, 05, 10], 'year');
```

Output

Since the year of both moments is matching, it gives **true** as the output.

Example

```
var issame = moment([2015, 10, 01]).isSame([2015, 05, 10], 'month');
```

Output

Note that in the above example the month is not matching, so it gives the output as false.

You will get false for month unit even if the year is not matching . Observe the following code for the same.

Example

```
var issame = moment([2015, 10, 01]).isSame([2014, 05, 10], 'month');
```

Output

Example

```
var issame = moment([2015, 10, 01]).isSame([2015, 05, 10], 'day');
```

Output

Is After

This method gives true or false if the given moment is after another moment.

Syntax

```
moment().isAfter(Moment|String|Number|Date|Array);
moment().isAfter(Moment|String|Number|Date|Array, String);
```

Example

```
  var isafter = moment().isAfter();
```

Output

If no param to **isAfter** it considers it as the current date. It gives the output as false since the current date is not after the moment compared.

Example

```
var isafter = moment([2015, 10, 01]).isAfter([2014, 05, 10]);
```

Output

If the given date to **isAfter** is less, it gives true as the date is after the another moment.

Example

```
var isafter = moment([2015, 10, 01]).isAfter([2018, 05, 10]);
```

Output

```
Is After: false
```

Here the date given to isAfter is greater than the date compared, so the output is false.

Consider the following examples to check **isAfter()** method with units.

Example

```
var isafter = moment([2015, 10, 01]).isAfter([2011, 05, 10], 'year');
```

Output

```
Is After: true
```

Is Same or Before

This method checks if a moment is same or before another moment.

Syntax

```
moment().isSameOrBefore(Moment|String|Number|Date|Array);
moment().isSameOrBefore(Moment|String|Number|Date|Array, String);
```

Example

```
var issameorbefore = moment('2017-10-10').isSameOrBefore('2017-11-21');
```

Output

```
Is Same or Before: true
```

Example

```
var issameorbefore = moment('2017-10-10').isSameOrBefore('2017-08-21');
```

Output

Observe that in the code shown above, the moment is neither the same nor before, so output is false.

We can use the units with isSameOrBefore() and the ones supported are year, month, week ,day, hour, minute and second.

Example

```
var issameorbefore = moment('2017-10-10').isSameOrBefore('2017-08-21', 'year');
```

Output

Example

```
var issameorbefore = moment('2017-10-10').isSameOrBefore('2017-10-21',
'month');
```

Output

Is Same or After

This method checks if the moment is same or after another moment. It returns true or false.

Syntax

```
moment().isSameOrAfter(Moment|String|Number|Date|Array);

moment().isSameOrAfter(Moment|String|Number|Date|Array, String);
```

Example

```
var issameorafter = moment('2017-10-10').isSameOrAfter('2017-10-09');
```

Output

We can use the units with isSameOrAfter() and the ones supported are year, month, week , day, hour, minute and second.

Example

```
var issameorafter = moment('2017-10-10').isSameOrAfter('2017-10-09', 'year');
```

Output

Example

```
var issameorafter = moment('2017-10-10').isSameOrAfter('2017-10-15', 'day');
```

Output

Is Between

This method will return true or false if the moment is between other two moments.

Syntax

```
moment().isBetween(moment-like, moment-like);
moment().isBetween(moment-like, moment-like, String);
```

Example

```
var isbetween = moment('2015-05-01').isBetween('2009-10-19', '2018-10-25');
```

Output

Example

```
var isbetween = moment('2015-05-01').isBetween('2009-10-19', '2018-10-25',
'year');
```

Output

You can use the units year, month, week, day, hour, minute and second along with the method as shown above.

Is Daylight Saving Time

This method returns true or false if the current moment falls in daylight saving time.

Syntax

```
moment().isDST();
```

Example

```
var isdylight = moment().isDST();
```

Output

If date is not given it will check for the current time.

Example

```
var isdylight = moment([2017, 2, 14]).isDST();
```

Output

```
Is Day light saving: false
```

Is Leap Year

This method will return true or false if the moment year is a leap year.

Syntax

```
moment().isLeapYear();
```

Example

```
var isleapyear = moment([2000]).isLeapYear();
```

Output

```
Is Leap Year: true
```

Is a Moment

This method will return true or false if the given variable to be checked is a moment.

Syntax

```
moment.isMoment(obj);
```

Example

```
var ismoment = moment.isMoment(moment());
```

Output

```
Is a Moment: true
```

Example

```
var ismoment = moment.isMoment(moment([2010, 05, 01]));
```

Output

```
Is a Moment: true
```

Is a Date

This method will return true or false if the given variable to be checked is a native JavaScript date, that is **new Date()**.

Syntax

```
moment.isDate(obj);
```

Example

```
var isdate = moment.isDate(new Date());
```

Output

```
Is a Date: true
```

10. MomentJS – Internationalization

Internationalization is one of the important features in MomentJS. You can display date and time based on localization, in other words, based on the country/region. The locale can be applied to specific moment if required.

This chapter discusses in detail about how to make apply locale globally, locally, work with locale using Node.js, in browser, get the units (months, weekdays etc.) in the required locale etc.

Global locale

We can assign locale globally and all the date /time details will be available in the locale assigned.

Syntax

```
moment.locale(String);
moment.locale(String[]);
moment.locale(String, Object);
```

Example

```
moment.locale("hi");
var a = moment.duration(1, 'days').humanize();
var b = moment.duration(1, 'week').humanize();
```

Output

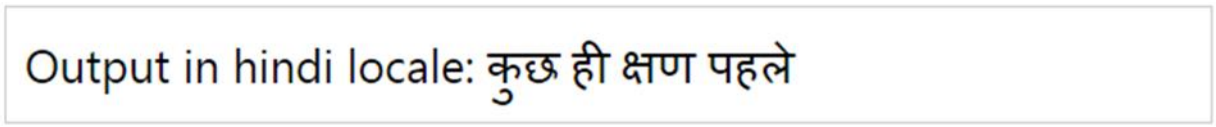
Output in hindi locale: एक दिन

Output in hindi locale: ७ दिन

Example

```
moment.locale("hi");
var a = moment().fromNow();
```

Output



Example

```
moment.locale("pt");
var a = moment().locale();
```

Output

Changing Locale Locally

We need locale to applied locally in case we need to handle many locales in an application.

Here is a way to work with locale locally.

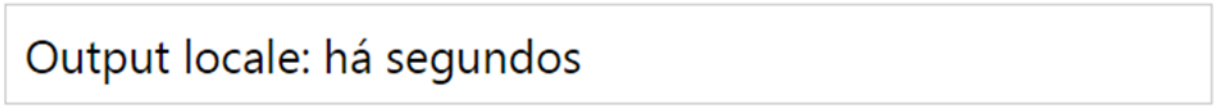
Syntax

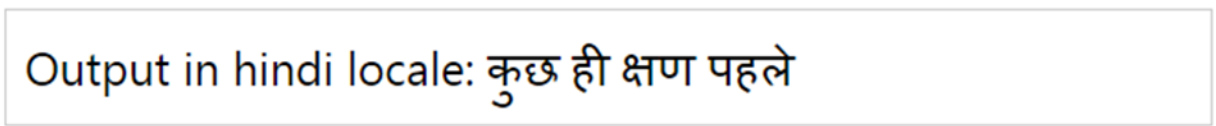
```
moment().locale(String|Boolean);
```

Example

```
moment.locale("pt");
var a = moment().fromNow();
moment.locale("hi");
var b = moment().fromNow();
```

Output





Example

```
moment.locale("pt");
var a = moment().format('LLLL');
moment.locale("hi");
var b = moment().format('LLLL');
```

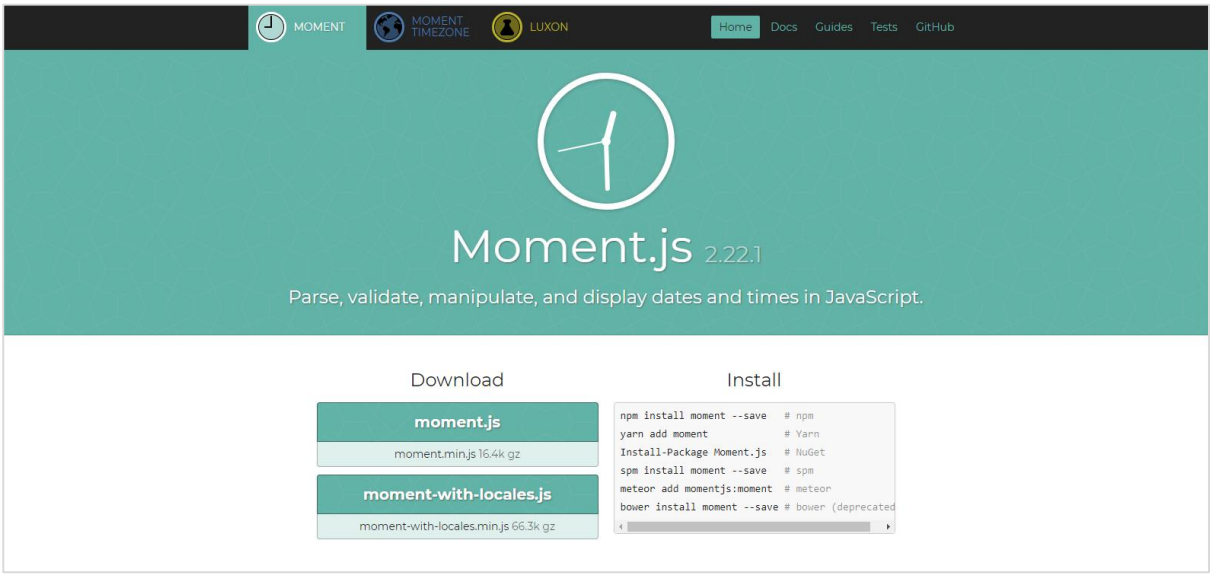
Output

Output locale: Sexta-feira, 20 de abril de 2018 17:50

Output in hindi locale: शुक्रवार, २० अप्रैल २०१८, शाम ५:५० बजे

Using Locale in Browser

We can start working with locale by including the locale file in script tag. You can get the minified files for all locales from the MomentJS home page as shown below:



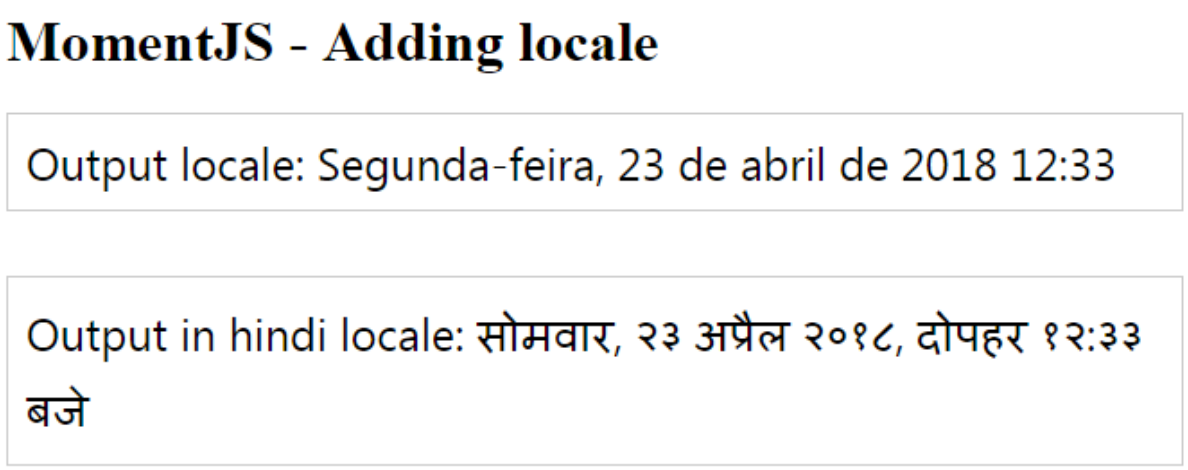
The same can be included in your application as shown here:

Example

```
<html>
<head>
```

```
    <title>MomentJS - Adding locale</title>
    <script type="text/JavaScript"
src="https://MomentJS.com/downloads/moment.js"></script>
    <script type="text/JavaScript" src="https://MomentJS.com/downloads/moment-
with-locales.js"></script>
    <style>
        div {
            border: solid 1px #ccc;
            padding:10px;
            font-family: "Segoe UI",Arial,sans-serif;
            width: 40%;
        }
    </style>
</head>
<body>
    <h1>MomentJS - Adding locale</h1>
    <div style="font-size:25px" id="currentdate"></div>
<br/>
<br/>
<div style="font-size:25px" id="changeddate"></div>
<script type="text/JavaScript">
    moment.locale("pt");
    var a = moment().format('LLLL');
    moment.locale("hi");
    var b = moment().format('LLLL');
    document.getElementById("currentdate").innerHTML = "Output locale: " + a;
    document.getElementById("changeddate").innerHTML = "Output in hindi locale:
" + b;
</script>
</body>
</html>
```

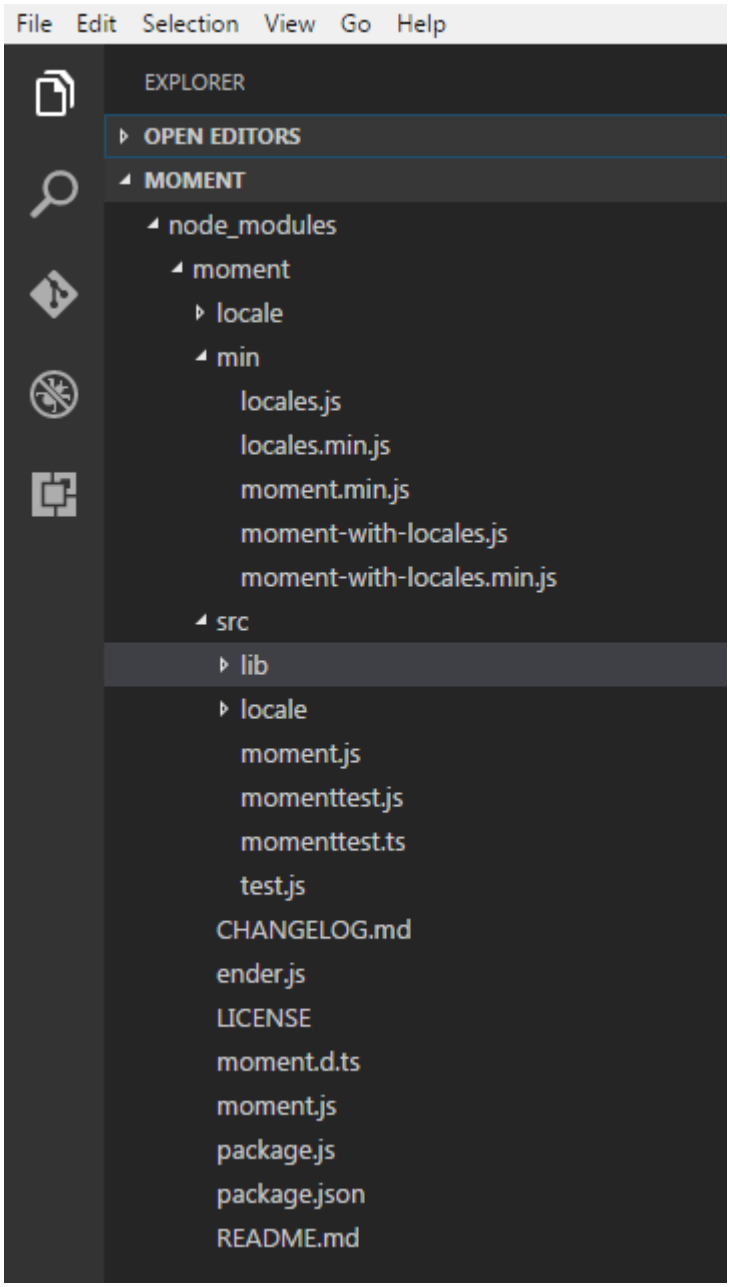
Output


MomentJS - Adding locale
Output locale: Segunda-feira, 23 de abril de 2018 12:33
Output in hindi locale: सोमवार, २३ अप्रैल २०१८, दोपहर १२:३३ बजे

Using Locale using Node.js

If you happen to use Node.js , you will have the locale files already in moment when you do npm install moment.

Example

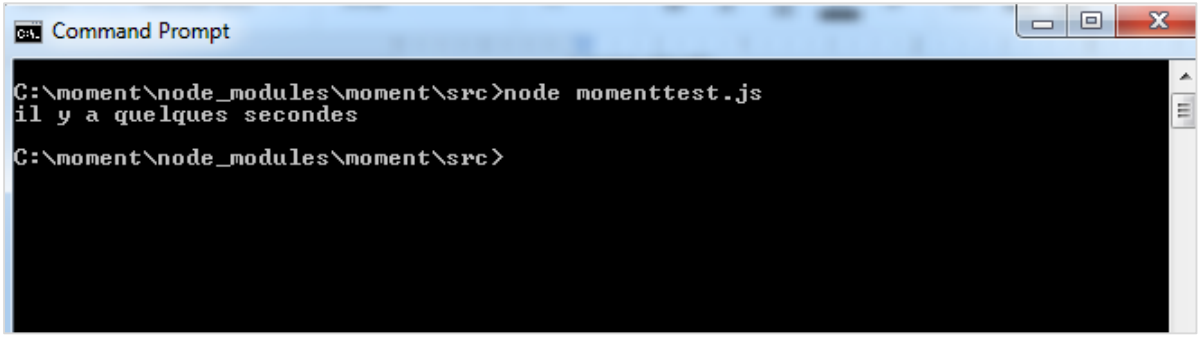


Observe the following example that is executed at Node.js end:

Example

```
"use strict";
var moment = require('moment');
moment.locale('fr');
var k = moment().fromNow();
console.log(k);
```

Output



Listing date/time details of current locale

You can set the locale and check the details like months, weekdays etc. In the examples given here, we have set the locale to French and below example shows the months, weekdays for French locale.

Locale Month

This method will get the months for the locale set.

Syntax

```
moment.months();
moment.monthsShort();
```

Example

```
moment.locale('fr');
var m = moment.months();
JSON.stringify(m);
```

Output

```
Months Details :
["janvier","février","mars","avril","mai","juin","juillet","août","septembre","octobre","novembre","décembre"]
```

Example

```
moment.locale('fr');
var m = moment.monthsShort();
JSON.stringify(m);
```

Output

```
Months Details :
["janv.","févr.","mars","avr.","mai","juin","juil.","août","sept.","oct.","nov.","déc."]
```

Weekdays

Syntax

```
moment.weekdays()
moment.weekdaysShort()
moment.weekdaysMin()
```

Example

```
moment.locale('fr');
var m = moment.weekdays();
JSON.stringify(m);
```

Output

```
Weekdays using locale:
["dimanche","lundi","mardi","mercredi","jeudi","vendredi","samedi"]
```

Example

```
moment.locale('fr');
var m = moment.weekdaysShort();
JSON.stringify(m);
```

Output

```
Weekdays using locale: ["dim.","lun.","mar.","mer.","jeu.","ven.","sam."]
```

Example

```
moment.locale('fr');
var m = moment.weekdaysMin();
JSON.stringify(m);
```

Output

```
Weekdays using locale: ["di","lu","ma","me","je","ve","sa"]
```

Example

```
moment.locale('fr');
var m = moment.weekdays(4);
```

The above examples displays the fourth day of the week.

Output

```
Weekdays using locale: jeudi
```

Example

```
moment.locale('fr');
 var m = moment.weekdays(true);
```

The locale is set to French. When true is passed to weekdays it will display the output as Saturday to Friday in French.

Output

```
Weekdays using locale:
lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
```

Example

```
moment.locale('fr');
var m = moment.weekdays(true, 1);
```

It will display Sunday as per French.

Output

```
Weekdays using locale: mardi
```

Checking current locale

We can check the current locale using moment.locale().

Syntax

```
moment.locale();
```

Example

```
moment.locale('fr');
 var k = moment.locale();
```

Output

```
Locale set to: fr
```

Example

```
moment.locale('fr');
 var k = moment.locale();
 moment.locale('ja');
 var s = moment.locale();
```

Output

```
Locale set to: fr
```

```
Locale set to: ja
```

moment.locales() gives the list of all the locales available for use or which are loaded. Observe the following line of code and its output:

```
moment.locales()
```

Output

```
Locale loaded are: en,af,ar-dz,ar-kw,ar-ly,ar-ma,ar-sa,ar-
tn,ar,az,be,bg,bm,bn,bo,br,bs,ca,cs,cv,cy,da,de-at,de-ch,de,dv,el,en-
au,en-ca,en-gb,en-ie,en-il,en-nz,eo,es-do,es-us,es,et,eu,fa,fi,fo,fr-ca,fr-
ch,fr,fy,gd,gl,gom-latn,gu,he,hi,hr,hu,hy-
am,id,is,it,ja,jv,ka,kk,km,kn,ko,ky,lb,lo,lt,lv,me,mi,mk,ml,mn,mr,ms-
my,ms,mt,my,nb,ne,nl-be,nl,nn,pa-in,pl,pt-br,pt,ro,ru,sd,se,si,sk,sl,sq,sr-
cyrl,sr,ss,sv,sw,ta,te,tet,tg,th,tl-ph,tlh,tr,tzl,tzm-latn,tzm,ug-cn,uk,ur,uz-
latn,uz,vi,x-pseudo,yo,zh-cn,zh-hk,zh-tw
```

In the above example, we are using the **locale.min.js** with all locales, so **moment.locales()** displays all the locales as shown above.

Accessing Locale Specific Functionality

Here will see the methods and properties available on currently loaded locale.

Syntax

```
moment.localeData()
```

You will observe the following output in the browser when we console moment.localeData():

Example

```
var localeData = moment.localeData();
 var m = localeData.months();
 var k = localeData.weekdays();
```

Output

```
Locale data
:January,February,March,April,May,June,July,August,September,October,November,December
```

```
Locale data : Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

The properties can be accessed using **moment.localeData('property')**. The following methods are available from **localeData** object:

```
var localeData = moment.localeData();
var m = localeData.months(moment()); // gets the month as per the moment such as April
var m1 = localeData.monthsShort(moment()); // gets the month name in shortform such as Apr
var k = localeData.weekdays(moment()); // returns the weekday for given moment.such as Monday
var k1 = localeData.weekdaysShort(moment()); // returns shortname for weekday for given moment.such as Mon
```

You can also use the following methods on **localeData** object:

```
localeData.weekdaysMin(aMoment);
localeData.weekdaysParse(minShortOrLongWeekdayString);
localeData.longDateFormat(dateFormat);
localeData.isPM(amPmString);
localeData.meridiem(hours, minutes, isLower);
localeData.calendar(key, aMoment);
localeData.relativeTime(number, withoutSuffix, key, isFuture);
localeData.pastFuture(diff, relTime);
localeData.ordinal(number);
localeData.preparse(str);
localeData.postformat(str);
localeData.week(aMoment);
localeData.invalidDate();
localeData.firstDayOfWeek();
localeData.firstDayOfYear();
```

11. MomentJS – Customization

MomentJS allows to add customization to the locale created. This chapter discusses them in detail.

The following list shows the customizations possible on localization:

- Month Names
- Month Abbreviation
- Weekday Names
- Weekday Abbreviation
- Minimal Weekday Abbreviation
- Long Date Formats
- Relative Time
- AM/PM
- AM/PM Parsing
- Calendar
- Calendar Format
- Ordinal
- Relative Time Thresholds
- Relative Time Rounding
- Changing Time Source

Syntax

```
moment.locale('en-my-settings', {
    // customizations.
});
```

When multiple locale are assigned, you can remove the assigned locale by passing **null** as second params to **moment.locale('fr', null);**

Example

```
var a = moment.locale('fr');
var b = moment.locale('en');
var c = moment.locale('ja', null);
var d = moment.locale('hi');
```

Output

You can define locale in the following ways:

Syntax

```
moment.defineLocale('fr-foo', {
  parentLocale: 'fr',
  /* */
});
```

To update locale, you can use the following code:

```
moment.updateLocale('fr', {
 /**/
});
```

To revert the update do:

```
moment.updateLocale('en', null);
```

Month Names

You can add month names to the locale customization.

Syntax

```
moment.updateLocale('en', {
    months : String[]
});
```

OR

```
moment.updateLocale('en', {
    months : Function
});
```

OR

```
moment.updateLocale('en', {
    months : {
        format : String[],
        standalone : String[]
    }
});
```

Example

```
var localeData = moment.updateLocale('fr', {
     months: [
            "Jan", "Feb", "Mar", "Apr", "May", "June", "July",
            "Aug", "Sept", "Oct", "Nov", "Dec"
     ]
});
var m = localeData.months();
```

Output

```
Locale output:Jan,Feb,Mar,Apr,May,June,July,Aug,Sept,Oct,Nov,Dec
```

Example

```
var localeData = moment.updateLocale('en', {
     nominative:
'January_February_March_April_May_June_July_August_September_October_November_D
ecember'.split('_'),
     subjective:
'January_February_March_April_May_June_July_August_September_October_November_D
ecember'.split('_'),
     months: function (momentToFormat, format) {
            if (/^MMMM/.test(format)) {
                  console.log(this._nominative);
                  return this._nominative[momentToFormat.month()];
            } else {
                  return this._subjective[momentToFormat.month()];
            }
     }
});
var m = localeData.months(moment(), "MMMM");
```

Output

```
Month Name:April
```

Example

```
var localeData = moment.updateLocale('en', {
     months : {
            format:
'January_February_March_April_May_June_July_August_September_October_November_D
ecember'.split('_'),
            standalone:
'January_February_March_April_May_June_July_August_September_October_November_D
ecember'.split('_'),
            isFormat:
/D[oD]?(\[[^\[\]]*\]|\s+)+MMMM?|MMMM?(\[[^\[\]]*\]|\s+)+D[oD]?/  // from 2.14.0
     }
});
var m = localeData.months();
```

Output

```
Months Details :
January,February,March,April,May,June,July,August,September,October,November,December
```

Month Abbreviations

This method helps in customizing the month abbreviations.

Syntax

```
moment.updateLocale('en', {
    monthsShort : String[]
});
moment.updateLocale('en', {
    monthsShort : Function
});
moment.updateLocale('en', {
    monthsShort : {
        format: String[],
        standalone : String[]
    }
});
```

Example

```
var localeData = moment.updateLocale('en', {
monthsShort : ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"]
});
var m = localeData.monthsShort();
```

Output

```
Months Details : Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
```

Example

```
var localeData = moment.updateLocale('en', {
     nominative:
'Jan_Feb_Mar_Apr_May_June_July_Aug_Sept_Oct_Nov_Dec'.split('_'),
     subjective:
'Jan_Feb_Mar_Apr_May_June_July_Aug_Sept_Oct_Nov_Dec'.split('_'),
     monthsShort: function (momentToFormat, format) {
            if (/^MMMM/.test(format)) {
                  console.log(this._nominative);
                  return this._nominative[momentToFormat.month()];
            } else {
                  return this._subjective[momentToFormat.month()];
            }
     }
});
var m = localeData.monthsShort(moment(), "MMMM");
```

Output

```
MonthsShort Name:Apr
```

Example

```
var localeData = moment.updateLocale('en', {
     monthsShort : {
            format:
'jan_feb_mar_apr_may_june_july_aug_sept_oct_nov_dec'.split('_'),
            standalone:
'jan_feb_mar_apr_may_june_july_aug_sept_oct_nov_dec'.split('_')
     }
});
var m = localeData.monthsShort();
```

Output

```
Months Details : jan,feb,mar,apr,may,june,july,aug,sept,oct,nov,dec
```

Weekdays Names

This method helps in customizing the weekdays names as per locale.

Syntax

```
moment.updateLocale('en', {
    weekdays : String[]
});
moment.updateLocale('en', {
    weekdays : Function
});
moment.updateLocale('en', {
    weekdays : {
        standalone : String[],
        format : String[],
        isFormat : RegExp
    }
});
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdays: [
            "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"
     ]
});
var m = localeData.weekdays();
```

Output

```
Weekdays:Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdaysarray: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"],
     weekdays: function (momentToFormat, format) {
           return this._weekdaysarray[momentToFormat.day()];
     }
});
var m = localeData.weekdays(moment(), "");
```

Output

```
Weekdays:Tuesday
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdays: {
           standalone:
"Sunday_Monday_Tuesday_Wednesday_Thursday_Friday_Saturday".split('_'),
           format:
"Sunday_Monday_Tuesday_Wednesday_Thursday_Friday_Saturday".split('_'),
           isFormat: /\[ ?[Вв] ?(?:прошлую|следующую|эту)? ?\] ?dd/
     }
});
var m = localeData.weekdays();
```

Output

```
Weekdays:Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

Weekday Abbreviations

This method helps in customizing the weekday abbreviations based on the locale set.

Syntax

```
moment.updateLocale('en', {
    weekdaysShort : String[]
});
moment.updateLocale('en', {
    weekdaysShort : Function
});
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdaysShort: ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
});
var m = localeData.weekdaysShort();
```

Output

```
Weekdays Abbreviation:Sun,Mon,Tue,Wed,Thu,Fri,Sat
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdaysShortarr: ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"],
     weekdaysShort: function (momentToFormat, format) {
            return this._weekdaysShortarr[momentToFormat.day()];
     }
});
var m = localeData.weekdaysShort(moment());
```

Output

```
Weekdays Abbreviation:Tue
```

Minimal Weekday Abbreviations

Syntax

```
moment.updateLocale('en', {
    weekdaysMin : String[]
});
moment.updateLocale('en', {
    weekdaysMin : Function
});
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdaysMin: ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"]
});
var m = localeData.weekdaysMin();
```

Output

```
Weekdays Abbreviation:Su,Mo,Tu,We,Th,Fr,Sa
```

Example

```
var localeData = moment.updateLocale('en', {
     weekdaysMinarr: ["Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"],
     weekdaysMin: function (momentToFormat, format) {
            return this._weekdaysMinarr[momentToFormat.day()];
     }
});
var m = localeData.weekdaysMin(moment());
```

Output

```
Weekdays Abbreviation:Tu
```

Long Date Formats

This method helps in customizing longdateformat for a locale.

Syntax

```
moment.updateLocale('en', {
   longDateFormat : Object
});
```

Here longDateFormat is an object containing key / value pair for each format.

Example

```
var localeData = moment.updateLocale('fr', {
     longDateFormat: {
            LT: "h:mm",
            LTS: "h:mm:ss A",
            L: "MM/DD/YYYY",
            l: "M/D/YYYY",
            LL: "MMMM Do YYYY",
            ll: "MMM D YYYY",
            LLL: "MMMM Do YYYY LT",
            lll: "MMM D YYYY LT",
            LLLL: "dddd, MMMM Do YYYY LT",
            llll: "ddd, MMM D YYYY LT"
     }
});
var m = moment().format('LT');
var x = moment().format('LLLL');
```

Output

```
LongDateFormat:11:39
```

```
LongDateFormat:mardi, avril 24 2018 11:39
```

Relative Time

This method helps in obtaining the relative time.

Syntax

```
moment.updateLocale('en', {
    relativeTime : Object
});
```

Example

```
var localeData = moment.updateLocale('en', {
     relativeTime: {
            future: "in %s",
            past: "%s ago",
            s: 'hello, a few seconds',
            ss: '%d seconds',
            m: "a minute",
            mm: "%d minutes",
            h: "an hour",
            hh: "%d hours",
            d: "a day",
            dd: "%d days",
            M: "a month",
            MM: "%d months",
            y: "a year",
            yy: "%d years"
     }
});
var m = moment().fromNow();
```

Output

Relative Time: hello, a few seconds ago

AM/PM

This method helps in customizing the meridiem as per locale.

Syntax

```
moment.updateLocale('en', {
    meridiem : Function
});
```

Example

```
var localeData = moment.updateLocale('fr', {
     meridiem: function (hours, minutes, isLower) {
            return hours < 12 ? 'PD' : 'MD';
     }
});
var m = moment('2016-01-01T05:00:00').format('hh a');
```

Output

```
Meridian: 05 PD
```

AM/PM Parsing

You can parse AM/PM using this method.

Syntax

```
moment.updateLocale('en', {
    meridiemParse : RegExp
    isPM : Function // will return true or false.
});
```

Example

```
var localeData = moment.updateLocale('fr', {
     meridiemParse: /PD|MD/,
     isPM: function (input) {
            return input.charAt(0) === 'M';
     }
```

```
});
var m = localeData.isPM("MD");
```

Output

```
Meridian: true
```

Calendar

This helps in customizing calendar object for a locale set.

Syntax

```
moment.updateLocale('en', {
    calendar : Object
});
```

Example

```
var localeData = moment.locale('de', {
     calendar: {
            sameDay: "[heute um] LT",
            sameElse: "L",
            nextDay: '[morgen um] LT',
            nextWeek: 'dddd [um] LT',
            lastDay: '[gestern um] LT',
            lastWeek: '[letzten] dddd'
     },
});
var m = moment().subtract(1, 'days').calendar();
```

Output

```
Calendar: gestern um 12:50
```

Ordinal

The ordinal display for dates can be changed based on locale.

Syntax

```
moment.updateLocale('en', {
    ordinal : Function
});
```

Example

```
var localeData = moment.updateLocale('en', {
     ordinal: function (number, token) {
            var b = number % 10;
            var output = (~~(number % 100 / 10) === 1) ? 'th' :
                  (b === 1) ? 'st' :
                        (b === 2) ? 'nd' :
                              (b === 3) ? 'rd' : 'TH';
            return number + output;
     }
});
var m = moment(['2015-05-8'], 'DD-MM-YYYY').format('Do');
```

In above example the **th** is changed to uppercase **TH** and the same is displayed in the output as the date is 8.

Output

```
Ordinal: 20TH
```

Relative Time Thresholds

This is used with duration.humanize where the length of duration is displayed as a **few seconds ago, in a minute, an hour ago** etc**.** No of seconds are predefined and displayed as few seconds ago, and the same is applicable for minute and hour. You can change the seconds, minute, hour, days limit using relative time threshold method.

Syntax

```
moment.relativeTimeThreshold(unit);  // getter
moment.relativeTimeThreshold(unit, limit);  // setter
```

The table given here shows the units used along with display message and description:

Unit	Display message	Description
ss	a few seconds	least number of seconds to be considered seconds
s	seconds	least number of seconds to be considered as a minute
m	minutes	least number of minutes to be considered as a hour
h	hours	least number of hours to be considered as a day
d	days	Least number of days to be considered as a month
M	months	Least number of months to be considered as a year

Example

Observe the following code that displays the default time as few second, seconds, minutes, hours, days and months:

```
var m = moment.relativeTimeThreshold('ss');
var x = moment.relativeTimeThreshold('s');
var c = moment.relativeTimeThreshold('m');
var d = moment.relativeTimeThreshold('h');
var y = moment.relativeTimeThreshold('d');
var t = moment.relativeTimeThreshold('M');
```

Output

Note that minute threshold is changed from default 45 to 5 and the output for humanize for 6 minutes is displayed as **in an hour**

Example

```
moment.relativeTimeThreshold('m', 5);
var c = moment.duration(6, "minutes").humanize(true);
```

Output

Minute threshold is changed from default 45 to 15 and the output for humanize for 6 minutes is displayed as **in 6 minutes.**

Example

```
moment.relativeTimeThreshold('m', 15);
 var c = moment.duration(6, "minutes").humanize(true);
```

Output

12. MomentJS – Durations

MomentJS provides an important feature called durations which handles length of time for given units. In this chapter, you will learn this in detail.

Methods Available with Durations

The following table shows the methods available with duration for different units to be used with moment duration:

Method	Syntax

Creating	moment.duration(Number, String);
	moment.duration(Number);
	moment.duration(Object);
	moment.duration(String);

Clone	moment.duration().clone();
Humanize	moment.duration().humanize();
Milliseconds	moment.duration().milliseconds(); moment.duration().asMilliseconds();
Seconds	moment.duration().seconds(); moment.duration().asSeconds();
Minutes	moment.duration().minutes(); moment.duration().asMinutes();
Hours	moment.duration().hours(); moment.duration().asHours();
Days	moment.duration().days(); moment.duration().asDays();
Weeks	moment.duration().weeks(); moment.duration().asWeeks();

Months	moment.duration().months(); moment.duration().asMonths();
Years	moment.duration().years(); moment.duration().asYears();

Add Time	moment.duration().add(Number, String);
	moment.duration().add(Number);
	moment.duration().add(Duration);
	moment.duration().add(Object);

Subtract Time	moment.duration().subtract(Number, String);
	moment.duration().subtract(Number);
	moment.duration().subtract(Duration);
	moment.duration().subtract(Object);

Using Duration with Diff	var duration = moment.duration(x.diff(y))
As Unit of Time	moment.duration().as(String);

Get Unit of Time	duration.get('hours');
	duration.get('minutes');
	duration.get('seconds');
	duration.get('milliseconds');

As JSON	moment.duration().toJSON();
Is a Duration	moment.isDuration(obj);
As ISO 8601 String	moment.duration().toISOString();
Locale	moment.duration().locale(); moment.duration().locale(String);

In this section, let us learn these methods in detail.

Create Duration

This method is used to create the duration.

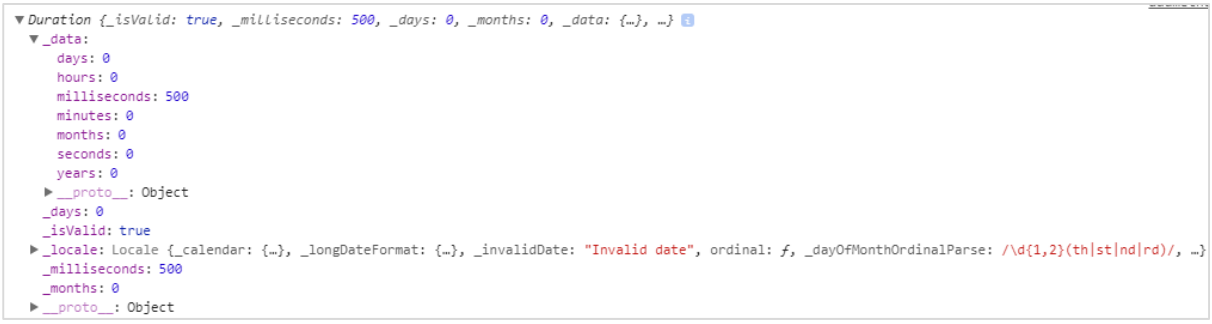
Syntax

```
moment.duration(Number, String);
moment.duration(Number);
moment.duration(Object);
moment.duration(String);
```

Example

```
var k = moment.duration(500);
JSON.stringify(k._data) // to get the object details from duration
```

The duration method gives the object with all details. The structure of duration that is visible in a console is shown here:



Output

```
Duration with millisecond:
{"milliseconds":500,"seconds":0,"minutes":0,"hours":0,"days":0,"months":0,"years":0}
```

Example

```
var k = moment.duration(1500);
```

Output

```
Duration with millisecond:
{"milliseconds":500,"seconds":1,"minutes":0,"hours":0,"days":0,"months":0,"years":0}
```

It also possible to create duration with units as params. Observe the following example for a better understanding:

Example

```
var k = moment.duration(5, 'seconds');
```

Output

```
Duration with millisecond:
{"milliseconds":0,"seconds":5,"minutes":0,"hours":0,"days":0,"months":0,"years":0}
```

Example

```
var k = moment.duration(12, 'months');
```

Output

Note that we have used 12 months in the duration, so it directly shows it in years as shown below:

```
Duration with millisecond:
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":0,"months":0,"years":1}
```

Example

```
var k = moment.duration(2, 'weeks');
```

The week's details are shown in for days. 2 weeks counts to 14 days as shown below:

Output

```
Duration with millisecond:
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":14,"months":0,"years":0}
```

The units which can be used with duration are years, months, weeks, days, hours, minutes, seconds and milliseconds. You can use the key/shorthand version discussed in earlier chapters for units with duration.

You can also update days, hour, minute, seconds as shown below:

Example

```
var k = moment.duration('6.23:50:40');
```

Output

```
Duration with millisecond:
{"milliseconds":0,"seconds":40,"minutes":50,"hours":23,"days":6,"months":0,"years":0}
```

Example with parsing method

```
var k = moment.duration('P5Y8M9DT4H5M25S');
```

Output

```
Duration with millisecond:
{"milliseconds":0,"seconds":25,"minutes":5,"hours":4,"days":9,"months":8,"years":5}
```

Clone

This method creates clone of the duration.

Syntax

```
moment.duration().clone();
```

Example

```
var a = moment.duration(2, 'days');
var b = a.clone();
```

Output

```
Duration with millisecond:
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":2,"months":0,"years":0}
```

Humanize

This method displays the date in human readable format.

Syntax

```
moment.duration().humanize();
```

Example

```
var a = moment.duration(24, "hours").humanize();
```

Output

```
Humanize: a day
```

In case you need suffix such as **ago** or **in** to the output, add true to humanize as follows:

Example

```
var a = moment.duration(24, "hours").humanize(true);
```

Output

```
Humanize: in a day
```

Example

```
var a = moment.duration(-1, "days").humanize(true);
```

Output

```
Humanize: a day ago
```

Milliseconds

This method will give no of milliseconds in a duration. It will show the value in the range 0-999. If the value is greater than 999, it will display as 0. To get the length of duration in milliseconds, use **asMilliseconds()** method.

Syntax

```
moment.duration().milliseconds();
moment.duration().asMilliseconds();
```

Example

```
var a = moment.duration(500).milliseconds();
```

Output

Example

```
var a = moment.duration(1000).milliseconds();
```

Output

Example

With **asMilliseconds**, you will be able to get the value in milliseconds as shown below:

```
var a = moment.duration(1000).asMilliseconds();
```

Output

Seconds

This method will give the number of seconds in the duration. The value returned will be between 0-59. If you want proper length of seconds in that duration make use of **moment.duration().asSeconds()**.

Syntax

```
moment.duration().seconds();
moment.duration().asSeconds();
```

Example

```
var a = moment.duration(20000).seconds();
```

Output

If you want to display the length of seconds in a duration make use of moment.duration().asSeconds() as follows:

Example

```
var a = moment.duration(600).asSeconds();
```

Output

Minutes

This method will give the no of minutes in the duration. The value returned will be between 0-59. If you want proper length of minutes in that duration make use of moment.duration().asMinutes().

Syntax

```
moment.duration().minutes();
moment.duration().asMinutes();
```

Example

```
var m = moment.duration(500, "minutes").minutes();
 var asmins = moment.duration(500, "minutes").asMinutes();
```

Output

Hours

This method will display the hours in the duration. The value returned will be between 0-23. In case you need the proper length of hours in that duration, make use of **moment.duration().asHours();**

Syntax

```
moment.duration().hours();
moment.duration().asHours();
```

Example

```
var h = moment.duration(500, "hours").hours();
var ashrs = moment.duration(500, "hours").asHours();
```

Output

```
hours: 20
```

```
as hours: 500
```

Days

This method will get the days in a duration. The value returned will be 0-30. **moment.duration().asDays()** gets the numbers of day in a given duration.

Syntax

```
moment.duration().days();
moment.duration().asDays();
```

Example

```
var d = moment.duration(500, "days").days();
var asdays = moment.duration(500, "days").asDays();
```

Output

Weeks

This method will the week details from a duration. To get the length of duration in weeks use **moment.duration().asWeeks()** method.

Syntax

```
moment.duration().weeks();
moment.duration().asWeeks();
```

Example

```
var m = moment.duration(500, "weeks").weeks();
  var asmins = moment.duration(500, "weeks").asWeeks();
```

Output

Months

This method will give no of months in a duration. It will return value from 0-11. To get the length of months in a given duration use **moment.duration().asMonths();**

Syntax

```
moment.duration().months();
moment.duration().asMonths();
```

Example

```
var m = moment.duration(100, "months").months();
 var asmins = moment.duration(500, "months").asMonths();
```

Output

Years

This method will give no of years. **moment.duration().asYears()** gets the length of the duration in years.

Syntax

```
moment.duration().years();
moment.duration().asYears();
```

Example

```
var m = moment.duration(500, "years").years();
var asmins = moment.duration(500, "years").asYears();
```

Output

Add Time

This method adds time to the current duration.

Syntax

```
moment.duration().add(Number, String);
moment.duration().add(Number);
moment.duration().add(Duration);
moment.duration().add(Object);
```

Example

```
var m = moment.duration().add(45, 'd');
 var output = JSON.stringify(m._data);
```

Output

```
Add Time to duration:
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":14,"months":1,"years":0}
```

Example

```
var m = moment.duration().add({ days: 7, months: 1 });
```

Output

```
Add Time to duration:
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":7,"months":1,"years":0}
```

Subtract Time

This method will subtract time from a given duration.

Syntax

```
moment.duration().subtract(Number, String);
moment.duration().subtract(Number);
moment.duration().subtract(Duration);
moment.duration().subtract(Object);
```

Example

```
var d = moment.duration(500, 'days');
 var st = moment.duration(500, 'days').subtract({ days: 7, months: 1 });
```

Output

```
Before Subtract Time :
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":13,"months":4,"years":1}
```

```
After Subtract Time :
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":6,"months":3,"years":1}
```

Using Duration with Diff

This method is used to get the duration difference between two moments.

Syntax

```
var duration = moment.duration(x.diff(y))
```

Example

```
var d = moment.duration(moment(["2017,07,04"]).diff(moment(["2015,08,20"])));
```

Output

```
Duration Diff :
{"milliseconds":0,"seconds":0,"minutes":0,"hours":0,"days":0,"months":0,"years":2}
```

As Unit of Time

This method is the alternative to the method used asHours, asMonths, asYears etc.

Syntax

```
moment.duration().as(String);
```

Example

```
var d = moment.duration(500, 'days').as('days');
```

Output

```
Unit of time : 500
```

Get Unit of Time

This is an alternative method for all **duration().hours()**, **duration().minutes()** etc.

Syntax

```
duration.get('hours');
duration.get('minutes');
duration.get('seconds');
duration.get('milliseconds');
```

Example

```
var d = moment.duration(500, 'hours').get('hours');
```

Output

```
Get Unit of time : 20
```

As JSON

We can get the duration object in json format. The output will be represented as an ISO8601 string.

Syntax

```
moment.duration().toJSON();
```

Example

```
var d = moment.duration(2, 'weeks').toJSON();
```

Output

```
Duration tojson : P14D
```

Is a Duration

This method is used to check if the variable is a moment duration object.

Syntax

```
moment.isDuration(obj);
```

Example

```
var d = moment.isDuration(moment.duration(2, 'weeks'));
var st = moment.isDuration(moment());
var s = moment.isDuration(new Date());
```

Output

```
Check for duration: true
```

```
Check for duration: false
```

```
Check for duration: false
```

An ISO 8601 String

This method will return the duration as an ISO 8601 string.

Syntax

```
moment.duration().toISOString();
```

Example

```
var d = moment.duration(2, 'weeks').toISOString();
```

Output

Note that the **P** shown in the output above stands for period.

The table given here shows the Format **PnYnMnDTnHnMnS** description:

Unit	Description
P	P stands for period. Placed at the start of the duration representation.
Y	Year
M	Month
D	Day
T	Designator that precedes the time components.
H	Hour
M	Minute
S	Second

Locale

This method helps to get/set the duration using locale. When used with humanize, you will see the difference in output for locale() method.

Syntax

```
moment.duration().locale();
moment.duration().locale(String);
```

Example

```
var hi = moment.duration(1, "day").locale("hi").humanize();
var en = moment.duration(1, "minutes").locale("en").humanize();
var ja = moment.duration(1, "seconds").locale("ja").humanize();
var it = moment.duration(1, "hours").locale("it").humanize();
var marathi = moment.duration(1, "day").locale("mr").humanize();
var konkani = moment.duration(1, "day").locale("gom-latn").humanize();
var kn = moment.duration(1, "day").locale("kn").humanize();
```

Output

Locale with humanize Hindi: एक दिन

Locale with humanize English: a minute

Locale with humanize Japanese: 数秒

Locale with humanize Italian: un'ora

Locale with humanize Marathi: एक दिवस

Locale with humanize Konkani: eka disan

Locale with humanize Kannada: ಒಂದು ದಿನ

13. MomentJS – Utilities

In MomentJS, you can change or customize the output as per the requirement using normalize units and invalid methods. You can also set your own custom validation on the moment object.

Observe the following table for more information:

Method	Syntax
Normalize Units	moment.normalizeUnits(String);
Invalid	moment.invalid(Object);

Normalize Units

This method allows to normalize the units used in your program. Note that the methods which makes use of units, such as moment().get('h') or moment().get('hours') are same.

In case you want to define the alias for units, you can do so using the Normalize method. Using the year as **y** or **YEa**r or **yEARs** will fallback to year using normalizeUnits.

Syntax

```
moment.normalizeUnits(String);
```

Example:

```
moment.normalizeUnits('y');
moment.normalizeUnits('Y');
moment.normalizeUnits('yEar');
moment.normalizeUnits('yeArs');
moment.normalizeUnits('YeARS');
```

Output

normalizeUnits: year

normalizeUnits: year

normalizeUnits: year

normalizeUnits: year

normalizeUnits: year

Example

```
var m = moment.normalizeUnits('M');
var a = moment.normalizeUnits('MONTHS');
var b = moment.normalizeUnits('Months');
var c = moment.normalizeUnits('MonTh');
```

Output

normalizeUnits: month

normalizeUnits: month

normalizeUnits: month

normalizeUnits: month

Invalid

This method allows you to define your own invalid object for a moment.

Syntax

```
moment.invalid(Object);
```

Example

In the following example, we are setting the invalidMonth and passing the same as object to invalid method. This helps to create the list of invalid checks of our choice to be validated with the moment.

```
var m = moment.invalid({ invalidMonth: 'reptember' });
var a = m.parsingFlags().invalidMonth;
```

Output

```
Invalid Month: reptember
```

14. MomentJS – Plugins

Plugins are extended features added on MomentJS. MomentJS is an open source project and many plugins are found in MomentJS which are contributed by its users and available using Node.js and GitHub.

This chapter discusses some of the calendars plugins and date formats plugins available in MomentJS.

Calendar Plugins

This section discusses two types of Calendar plugins: **ISO calendar** and **Taiwan calendar.**

ISO calendar

You can use the following command to install it with Node.js:

```
npm install moment-isocalendar
```

You can get the moment-isocalendar.js from GitHub : https://github.com/fusionbox/moment-isocalendar

Observe the following working example with isocalendar and MomentJS:

Example

```
var m = moment().isocalendar();
```

Output

```
ISO calendar details: 2018,17,3,704
```

Example

```
var m = moment.fromIsocalendar([2018, 51, 10, 670]).format('LLLL');
```

Output

```
ISO calendar details: Wednesday, December 26, 2018 11:10 AM
```

Taiwan Calendar

You can use the following command to install it with Node.js:

```
npm install moment-jalaali
```

You can get the moment-taiwan.js from GitHub : https://github.com/bradwoo8621/moment-taiwan

Observe the following working example with isocalendar and MomentJS:

Example

```
var m = moment('190/01/01', 'tYY/MM/DD');
 var c = m.twYear();
```

Output

Date formats Plugins

This section discusses the following types of Date format plugins:

- Java dateformat parser
- Short date formatter
- Parse date format
- Duration format
- Date Range
- Precise Range

Java DateFormat Parser

You can use the following command to install it with Node.js:

```
npm install moment-jdateformatparser
```

You can get the moment-jdateformatparser.js from GitHub : https://github.com/MadMG/moment-jdateformatparser

Observe the following working example for moment-jdateformatparser and MomentJS:

Example

```
var m = moment().formatWithJDF("dd.MM.yyyy");
```

Output

Short date formatter

The JavaScript file for shortdateformat can be fetched from GitHub : https://github.com/researchgate/moment-shortformat

Syntax

```
moment().short();
```

The display looks like as shown in the table here:

From moment	From moment().short()
0 to 59 seconds	0 to 59 s
1 to 59 minutes	1 to 59 m
1 to 23 hours	1h to 23h
1 to 6 days	1d to 6d
>= 7 days and same year	Display will be like such as feb 3, mar 6
>= 7 days and diff year	Display will be like such as feb 3, 2018, mar 6, 2018

You can take the script for momentshort from GitHub link given above.

Example

```
var a = moment().subtract(8, 'hours').short();
var b = moment().add(1, 'hour').short(true);
```

Output

If you want to remove the suffix **ago** or **in,** you can pass true to short(tru.

Parse date format

You can use the following command to install it with Node.js:

```
npm install moment-parseformat
```

Example

```
var a = moment.parseFormat('Friday 2018 27 april 10:28:10');
```

Output

Observe that the output shows that whatever parameters (date/ time) is given to the parseFormat, it gives the format of the date as shown above.

Duration Format

You can use the following command to install duration format on Node.js:

```
npm install moment-duration-format
```

The repository for duration format is available here : https://github.com/jsmreese/moment-duration-format

Let us see a working example with duration format:

Example

```
var a = moment.duration(969, "minutes").format("h:mm:ss");
```

Output

```
Duration : 16:09:00
```

This adds more details to the duration on moment created.

Date Range

You can use the following command to install date range on Node.js:

```
npm install moment-range
```

Example

```
window['moment-range'].extendMoment(moment);
var start = new Date(2012, 0, 15);
var end   = new Date(2012, 4, 23);
var range = moment.range(start, end);
console.log(range.start._d);
console.log(range.end._d);
```

Output

```
Moment Range start : Sun Jan 15 2012 00:00:00 GMT+0530 (India Standard Time)
```

```
Moment Range end : Wed May 23 2012 00:00:00 GMT+0530 (India Standard Time)
```

Precise Range

Precise range will display the exact date difference in date, time and in human readable format. You can use the following command to install precise range on Node.js:

```
npm install moment-precise-range-plugin
```

Example

```
var a = moment("1998-01-01 09:00:00").preciseDiff("2011-03-04 18:05:06");
```

Output

```
Precise Range Diff : 13 years 2 months 3 days 9 hours 5 minutes 6 seconds
```

15. MomentJS – Examples

Till now, we have learnt many concepts in MomentJS. This chapter gives you further examples for a better understanding.

Display Date Range Between Two Dates

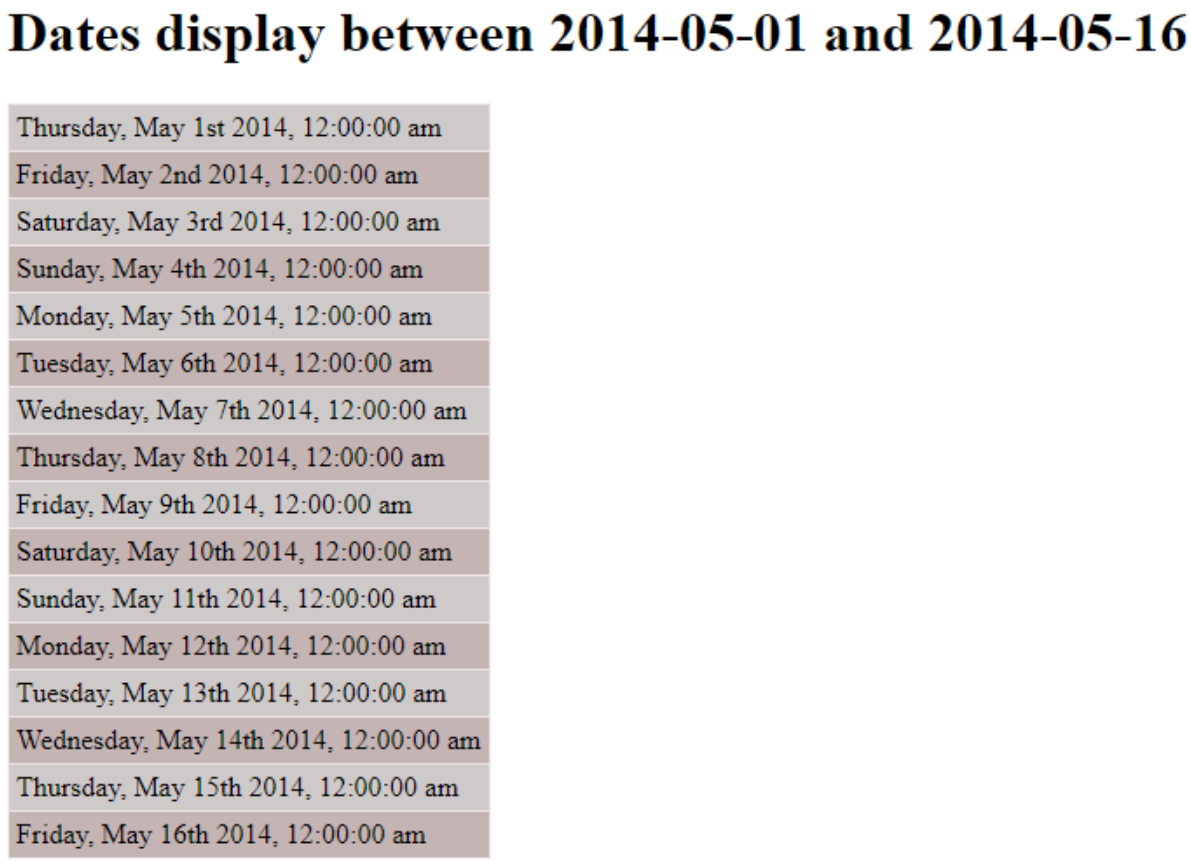
This is an example which displays the dates between two given dates.

```
<!DOCTYPE html>
<html>
<head>
<script type="text/JavaScript" src="MomentJS.js"></script>
<style>
table, td  {
  border: 1px solid #F1E8E8;
  border-collapse: collapse;
  padding: 4px;
}
table tr:nth-child(odd) {
  background-color: #CFCACA;
}
table tr:nth-child(even) {
  background-color: #C4B4B4;
}
</style>
</head>
<body>
<h1>Dates display between 2014-05-01 and 2014-05-16</h1>
<div id="container">
     <table id="datedetails"></table>
</div>
<script type="text/JavaScript">

     function getDaterange(start, end, arr) {
```

```
        if (!moment(start).isSameOrAfter(end)) {
            if (arr.length==0) arr.push(moment(start).format("dddd, MMMM Do
YYYY, h:mm:ss a"));
            var next = moment(start).add(1, 'd');
            arr.push(next.format("dddd, MMMM Do YYYY, h:mm:ss a"));
            getDaterange(next, end, arr);
        } else {
            return arr;
        }
        return arr;
    }
    var a = getDaterange("2014-05-01", "2014-05-16", []);
    var tr = "";
    for (var i = 0; i<a.length;i++ ) {
           tr += "<tr><td>"+a[i]+"</td></tr>";
    }
    document.getElementById("datedetails").innerHTML = tr;
</script>
</body>
</html>
```

We want to display all the dates between **2014-05-01** to **2014-05-16.** We have used date query **isSameOrAfter, date addition and date format** to achieve what we want.

Output

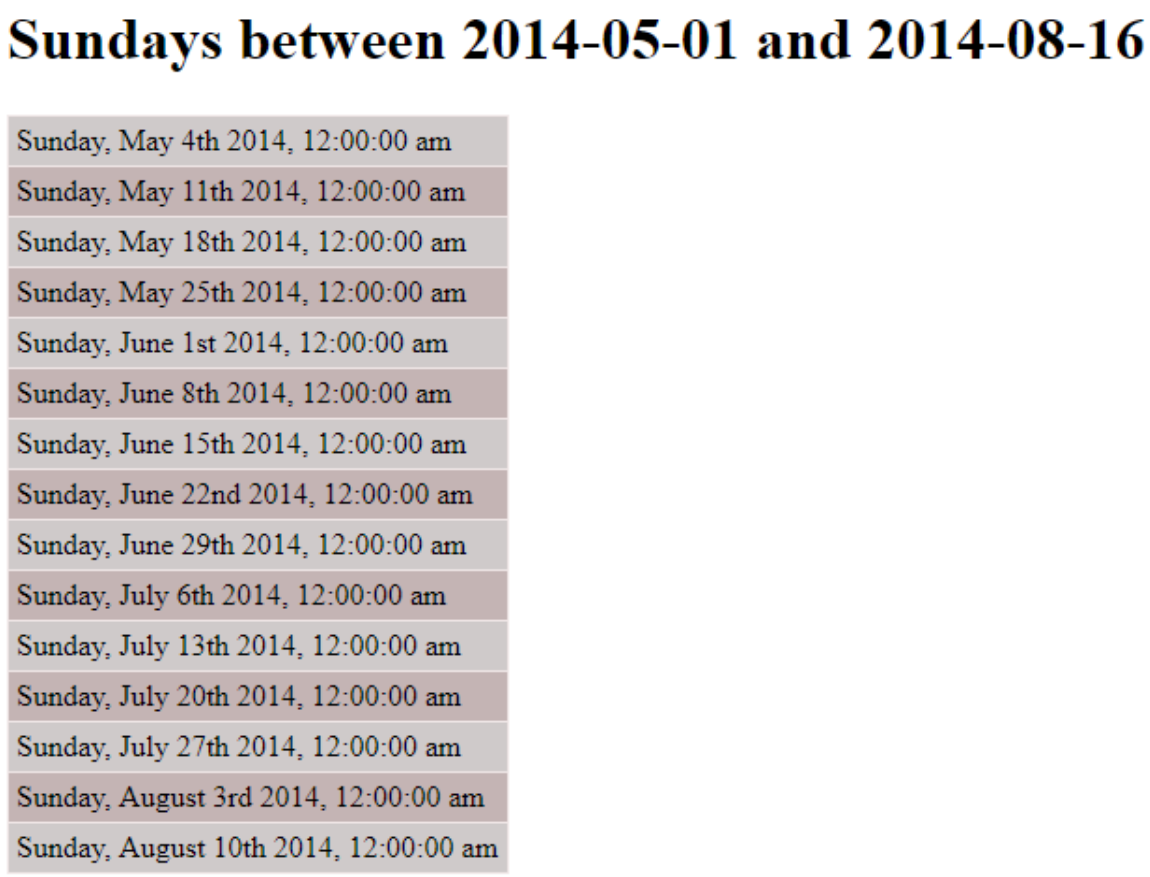


Display Sundays Between 2014-05-01 and 2014-08-16

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/JavaScript" src="MomentJS.js"></script>
   <style>
table, td  {
  border: 1px solid #F1E8E8;
  border-collapse: collapse;
  padding: 4px;
}
table tr:nth-child(odd) {
  background-color: #CFCACA;
}
table tr:nth-child(even) {
```

```
  background-color: #C4B4B4;
}
</style>
</head>
<body>
    <h1>Sundays between 2014-05-01 and 2014-08-16</h1>
    <div id="container">
        <table id="datedetails"></table>
    </div>
<script type="text/JavaScript">
function getDaterange(start, end, arr) {
  if (!moment(start).isSameOrAfter(end)) {
    if (arr.length==0) {
        if (moment(start).format("dddd") === "Sunday")  {
            arr.push(moment(start).format("dddd, MMMM Do YYYY, h:mm:ss a"));
        }
    }
    var next = moment(start).add(1, 'd');
    if (moment(next).format("dddd") === "Sunday")  {
        arr.push(next.format("dddd, MMMM Do YYYY, h:mm:ss a"));
    }
    getDaterange(next, end, arr);
  } else {
    return arr;
  }
  return arr;
}
var a = getDaterange("2014-05-01", "2014-08-16", []);
var tr = "";
for (var i = 0; i<a.length;i++ ) {
    tr += "<tr><td>"+a[i]+"</td></tr>";
}
document.getElementById("datedetails").innerHTML = tr;
</script>
</body>
</html>
```

Output



Display Date Details as per Locale

Here we are using moment.locale script which has all the locales.

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/JavaScript" src="MomentJS.js"></script>
  <script type="text/JavaScript" src="momentlocale.js" charset="UTF-
8"></script>
  <style type="text/css">
    div {
        margin-top: 16px!important;
        margin-bottom: 16px!important;
        width:100%;
    }
    table, td  {
```

```
  border: 1px solid #F1E8E8;
  border-collapse: collapse;
  padding: 4px;
}
table tr:nth-child(odd) {
  background-color: #CFCACA;
}
table tr:nth-child(even) {
  background-color: #C4B4B4;
}
  </style>
</head>
<body>
<div >
Select Locale : <select id="locale" onchange="updatelocale()"
style="width:200px;">
            <option value="en">English</option>
            <option value="fr">French</option>
            <option value="fr-ca">French Canada</option>
            <option value="cs">Czech</option>
            <option value="zh-cn">Chinese</option>
            <option value="nl">Dutch</option>
            <option value="ka">Georgian</option>
            <option value="he">Hebrew</option>
            <option value="hi">Hindi</option>
            <option value="id">Indonesian</option>
            <option value="it">Italian</option>
            <option value="jv">Japanese</option>
            <option value="ko">Korean</option>
     </select>
</div>
<br/>
<br/>
Display Date is different formats as per locale :<span
id="localeid"></span><br/>
<div>
     <table>
```

```
<tr>
	<th>Format</th>
	<th>Display</th>
</tr>
<tr>
	<td><i>dddd, MMMM Do YYYY, h:mm:ss a</i></td>
	<td>
		<div id="ldate"></div>
	</td>
</tr>
<tr>
	<td><i>LLLL</i></td>
	<td>
			<div id="ldate1"></div>
	</td>
</tr>
<tr>
	<td><i>moment().format()</i></td>
	<td>
			<div id="ldate2"></div>
	</td>
</tr>
<tr>
	<td><i>moment().calendar()</i></td>
	<td>
			<div id="ldate3"></div>
	</td>
</tr>
<tr>
	<td><i>Months</i></td>
	<td>
		<div id="ldate4"></div>
	</td>
</tr>
<tr>
	<td><i>Weekdays</i></td>
	<td>
```

```
                                <div id="ldate5"></div>
                    </td>
                </tr>
        </table>
</div>
<script type="text/JavaScript">
    var updatelocale = function() {
        var a = moment.locale(document.getElementById("locale").value);
        var k = moment().format("dddd, MMMM Do YYYY, h:mm:ss a");
        var k1 = moment().format("LLLL");
        var k2 = moment().format();
        var k3 = moment().calendar();
        var k4 = moment.months();
        var k5 = moment.weekdays();
        document.getElementById("localeid").innerHTML = "<b>"+a+"</b>";
        document.getElementById("ldate").innerHTML = k;
        document.getElementById("ldate1").innerHTML = k1;
        document.getElementById("ldate2").innerHTML = k2;
        document.getElementById("ldate3").innerHTML = k3;
        document.getElementById("ldate4").innerHTML = k4;
        document.getElementById("ldate5").innerHTML = k5;
    };
    updatelocale();
</script>
</body>
</html>
```

Output 1

Select Locale : English

Display Date is different formats as per locale :**en**

Format	Display
dddd, MMMM Do YYYY, h:mm:ss a	Friday, April 27th 2018, 9:51:51 pm
LLLL	Friday, April 27, 2018 9:51 PM
moment().format()	2018-04-27T21:51:51+05:30
moment().calendar()	Today at 9:51 PM
Months	January,February,March,April,May,June,July,August,September,October,November,December
Weekdays	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday

Output 2

Select Locale : Georgian

Display Date is different formats as per locale :**ka**

Format	Display
dddd, MMMM Do YYYY, h:mm:ss a	პარასკევი, აპრილი 27-ე 2018, 9:56:47 pm
LLLL	პარასკევი, 27 აპრილის 2018 9:56 PM
moment().format()	2018-04-27T21:56:47+05:30
moment().calendar()	დღეს 9:56 PM-ზე
Months	იანვარი,თებერვალი,მარტი,აპრილი,მაისი,ივნისი,ივლისი,აგვისტო,სექტემბერი,ოქტომბერი,ნოემბერი,დეკემბერი
Weekdays	კვირა,ორშაბათი,სამშაბათი,ოთხშაბათი,ხუთშაბათი,პარასკევი,შაბათი

Output 3

Select Locale : Hindi

Display Date is different formats as per locale :**hi**

Format	Display
dddd, MMMM Do YYYY, h:mm:ss a	शुक्रवार, अप्रैल २७ २०१८, ९:५७:१५ रात
LLLL	शुक्रवार, २७ अप्रैल २०१८, रात ९:५७ बजे
moment().format()	२०१८-०४-२७T२१:५७:१५+०५:३०
moment().calendar()	आज रात ९:५७ बजे
Months	जनवरी,फ़रवरी,मार्च,अप्रैल,मई,जून,जुलाई,अगस्त,सितम्बर,अक्टूबर,नवम्बर,दिसम्बर
Weekdays	रविवार,सोमवार,मंगलवार,बुधवार,गुरूवार,शुक्रवार,शनिवार

Output 4

Select Locale : Japanese

Display Date is different formats as per locale :**jv**

Format	Display
dddd, MMMM Do YYYY, h:mm:ss a	Jemuwah, April 27 2018, 9:57:39 ndalu
LLLL	Jemuwah, 27 April 2018 pukul 21.57
moment().format()	2018-04-27T21:57:39+05:30
moment().calendar()	Dinten puniko pukul 21.57
Months	Januari,Februari,Maret,April,Mei,Juni,Juli,Agustus,September,Oktober,Nopember,Desember
Weekdays	Minggu,Senen,Seloso,Rebu,Kemis,Jemuwah,Septu