How do we unit test Dates with ngMock's TzDate Type?

How to unit test an Angular $filter

A Recap on JavaScript Dates

A Recap on JavaScript Dates

- Time Measured in Milliseconds

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970
  - new Date(0) => to Thu, 01 Jan 1970 00:00:00 GMT

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970
  - new Date(0) => to Thu, 01 Jan 1970 00:00:00 GMT
- Construct new dates via string representations

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970
  - new Date(0) => to Thu, 01 Jan 1970 00:00:00 GMT
- Construct new dates via string representations
  - Simplified ISO 8601 Extended Format

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970
  - new Date(0) => to Thu, 01 Jan 1970 00:00:00 GMT
- Construct new dates via string representations
  - Simplified ISO 8601 Extended Format
  - YYYY-MM-DDTHH:mm:ss.sssZ

A Recap on JavaScript Dates

- Time Measured in Milliseconds
  - Since 01 January 1970
  - new Date(0) => to Thu, 01 Jan 1970 00:00:00 GMT
- Construct new dates via string representations
  - Simplified ISO 8601 Extended Format
  - YYYY-MM-DDTHH:mm:ss.sssZ
  - YYYY-MM-DDTHH:mm:ss+HH:mm

Dates and Time Zones

Dates and Time Zones

- New Dates have a base value in UTC

Dates and Time Zones

- New Dates have a base value in UTC
  - Coordinated Universal Time

Dates and Time Zones

- New Dates have a base value in UTC
  - Coordinated Universal Time
- A Date object instance has two states

Dates and Time Zones

- New Dates have a base value in UTC
  - Coordinated Universal Time
- A Date object instance has two states
  - UTC time

Dates and Time Zones

- New Dates have a base value in UTC
  - Coordinated Universal Time
- A Date object instance has two states
  - UTC time
  - Local time based on system

Local Date/Time vs UTC Date/Time can cause problems in tests!

```
var d = new Date(0);
console.log('Local Date:   ' + d.toLocaleDateString());
console.log('Local Hours:  ' + d.getHours());
console.log('UTC Date:     ' + d.toISOString());
console.log('UTC Hours:    ' + d.getUTCHours());
```

UTC vs Paris CET

```
Local Date:   1/1/1970
Local Hours: 1
UTC Date:     1970-01-01T00:00:00.000Z
UTC Hours:   0
```

```
var d = new Date(0);
console.log('Local Date:   ' + d.toLocaleDateString());
console.log('Local Hours: ' + d.getHours());
console.log('UTC Date:     ' + d.toISOString());
console.log('UTC Hours:    ' + d.getUTCHours());
```

UTC vs London GMT

```
Local Date:  1/1/1970
Local Hours: 0
UTC Date:    1970-01-01T00:00:00.000Z
UTC Hours:   0
```

```
var d = new Date('1970-01-01T00:00:00-01:00');
console.log('Local Date:   ' + d.toLocaleDateString());
console.log('Local Hours:  ' + d.getHours());
console.log('UTC Date:     ' + d.toISOString());
console.log('UTC Hours:    ' + d.getUTCHours());
```

Date with Negative Offset

```
Local Date:   1/1/1970
Local Hours: 2
UTC Date:     1970-01-01T01:00:00.000Z
UTC Hours:   1
```

```
var d = new Date('1970-01-01T00:00:00+01:00');
console.log('Local Date:   ' + d.toLocaleDateString());
console.log('Local Hours:  ' + d.getHours());
console.log('UTC Date:     ' + d.toISOString());
console.log('UTC Hours:    ' + d.getUTCHours());
```

Date with Positive Offset

```
Local Date:   1/1/1970
Local Hours: 0
UTC Date:     1969-12-31T23:00:00.000Z
UTC Hours:   23
```

Angular Controller Using Dates

```
angular.module('countdownApp', [])
  .controller('CountdownController',  function($scope) {
      if ($scope.nowTime.getFullYear() === $scope.nextYear) {
        $scope.message = 'Happy new Year!';
      } else {
        $scope.message = 'Keep on counting down...!';
      }
});
```

Angular Controller Test Using Date

```
it('should display happy new year message', function () {
  // yay, we're in the new year!
  $scope.nowTime = new Date('2015-01-01:00:00:00Z');
  $scope.nextYear = 2015;

  var countdownController = $controller('CountdownController', { $scope: $scope });
  expect($scope.message).toBe('Happy new Year!');
});
```

Angular Controller Test Using Date

```
it('should display almost new year message', function () {
  // we're one hour away from the new year
  $scope.nowTime = new Date('2014-12-31:23:00:00Z');
  $scope.nextYear = 2015;

  var countdownController = $controller('CountdownController', { $scope: $scope });
  expect($scope.message).toBe('Keep on counting down...!');
});
```

```
Executed 2 of 2 SUCCESS
```

Test Output

London

```
Name of the group should display almost new year message
FAILED
  Expected 'Happy new Year!' to be 'Keep on counting
down...!'.
Executed 2 of 2 (1 FAILED)
```

Test Output

Paris

TzDate

Its main purpose is to create Date-like instances with timezone fixed to the specified timezone offset, so that we can test code that depends on local timezone settings without dependency on the time zone settings of the machine where the code is running.

TzDate

We can setup a timezone that will not change according to the local machine

Angular Controller Test Using TzDate

```
it('should display happy new year message', function () {
  // yay, we're in the new year!
  $scope.nowTime = new angular.mock.TzDate(0, ’2015-01-01:00:00:00Z');
  $scope.nextYear = 2015;

  var countdownController = $controller('CountdownController', { $scope: $scope });
  expect($scope.message).toBe('Happy new Year!');
});
```

Angular Controller Test Using TzDate

```
it('should display almost new year message', function () {
  // we're one hour away from the new year
  $scope.nowTime = new angular.mock.TzDate(0, ’2014-12-31:23:00:00Z');
  $scope.nextYear = 2015;

  var countdownController = $controller('CountdownController', { $scope: $scope });
  expect($scope.message).toBe('Keep on counting down...!');
});
```

```
Executed 2 of 2 SUCCESS
```

Test Output

London

```
Executed 2 of 2 SUCCESS
```

Test Output

Paris

```
angular.mock.TzDate(offset, date);
```

```
new Date(offsetDate);
```

Local Time Settings + offset =>

```
angular.mock.TzDate(offset, date);
```

Local Time Settings + **offset** =>

```
new Date(offsetDate);
```

```
getHours()
```

```
TzDate(-1, '1970-01-01T00:00:00Z').getHours(); => 1
TzDate(0,  '1970-01-01T00:00:00Z').getHours(); => 0
TzDate(1,  '1970-01-01T00:00:00Z').getHours(); => 23
```

```
angular.mock.TzDate(offset, date);
```

Local Time Settings + offset =>

```
new Date(offsetDate);
```

```
getHours()
```

~~`getUTCDay()`~~

Missing Date Methods

```
 var unimplementedMethods = ['getUTCDay', 'getYear', 'setDate',
'setFullYear', 'setHours', 'setMilliseconds', 'setMinutes',
'setMonth', 'setSeconds', 'setTime', 'setUTCDate',
'setUTCFullYear', 'setUTCHours', 'setUTCMilliseconds',
'setUTCMinutes', 'setUTCMonth', 'setUTCSeconds', 'setYear',
'toDateString', 'toGMTString', 'toJSON', 'toLocaleFormat',
'toLocaleString', 'toLocaleTimeString', 'toSource', 'toString',
'toTimeString', 'toUTCString', ‘valueOf'];
```