Dates and Times

Overview

The `date` and `datetime` DataTable column data types utilize the built-in JavaScript Date class.

Important: In JavaScript Date objects, months are indexed starting at zero and go up through eleven, with January being month 0 and December being month 11.

(https://google-developers.appspot.com/chart/interactive/docs/constructor) Dates and Times Using the Date Constructor

(https://google-developers.appspot.com/chart/interactive/docs/dateconstructor)Dates Using the Date Constructor

To create a new Date object, you call the `Date()` constructor with the `new` keyword, with arguments to specify components of the date. These arguments take the form of several numbers corresponding to the different properties of your date.

```
new Date(Year, Month, Day, Hours, Minutes, Seconds, Milliseconds)
```

When using the Date constructor with the `date` data type, you only need to specify the Year, Month, and Day.

The Date constructor can also take the form: `new Date(Milliseconds)`, where Milliseconds is the distance in milliseconds of the desired date from January 1, 1970 00:00:00 UTC. For dates and times prior to that date, a negative number of Milliseconds would be given.

Using the Date constructor is useful when manually constructing your DataTable using the `addColumn()`, `addRow()`, and `addRows()` methods, as well as the `arrayToDataTable()` method. However, if using JSON to specify data, the string representation (#datestring) needs to be used.

The JavaScript Date constructor can also accept a string representation of the date as an argument. This string can take several different forms. The most reliable forms conform to

either the RFC 2822 specification (//tools.ietf.org/html/rfc2822#page-14) or the ISO 8601 specification (//www.w3.org/TR/NOTE-datetime). The formats for these are:

- RFC 2822 — `'MMM DD, YYYY'` or `'DD MMM, YYYY'` (Example: `new Date('Jan 1, 2015')` or `new Date('1 Jan, 2015')`)
- ISO 8601 — `'YYYY-MM-DD'` (Example: `new Date('2015-01-01')`)

Warning: The string representation in the Date constructor may be parsed differently by different browsers and different versions of browsers, thus returning different dates for the same string. As such, it is **not recommended** to pass in strings to the Date constructor. Instead, it is encouraged to only use numbers for the Date constructor's arguments.

The timeline below shows the Super Bowl champion of each NFL season since the year 2000.

CODE IT YOURSELF ON JSFIDDLE

Below is the code for creating this timeline. Note the use of the `new Date()` constructors, and the numbers given for each date, using 0-based months as mentioned earlier.

```
google.charts.load('current', {'packages':['timeline']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
  var data = new google.visualization.DataTable();
  data.addColumn('string', 'Team');
  data.addColumn('date', 'Season Start Date');
  data.addColumn('date', 'Season End Date');

  data.addRows([
    ['Baltimore Ravens',     new Date(2000, 8, 5), new Date(2001, 1, 5)],
    ['New England Patriots', new Date(2001, 8, 5), new Date(2002, 1, 5)],
    ['Tampa Bay Buccaneers', new Date(2002, 8, 5), new Date(2003, 1, 5)],
    ['New England Patriots', new Date(2003, 8, 5), new Date(2004, 1, 5)],
    ['New England Patriots', new Date(2004, 8, 5), new Date(2005, 1, 5)],
    ['Pittsburgh Steelers',  new Date(2005, 8, 5), new Date(2006, 1, 5)],
    ['Indianapolis Colts',   new Date(2006, 8, 5), new Date(2007, 1, 5)],
    ['New York Giants',      new Date(2007, 8, 5), new Date(2008, 1, 5)],
    ['Pittsburgh Steelers',  new Date(2008, 8, 5), new Date(2009, 1, 5)],
    ['New Orleans Saints',   new Date(2009, 8, 5), new Date(2010, 1, 5)],
    ['Green Bay Packers',    new Date(2010, 8, 5), new Date(2011, 1, 5)],
    ['New York Giants',      new Date(2011, 8, 5), new Date(2012, 1, 5)],
    ['Baltimore Ravens',     new Date(2012, 8, 5), new Date(2013, 1, 5)],
    ['Seattle Seahawks',     new Date(2013, 8, 5), new Date(2014, 1, 5)],
  ]);

  var options = {
    height: 450,
    timeline: {
      groupByRowLabel: true
    }
  };

  var chart = new google.visualization.Timeline(document.getElementById('

  chart.draw(data, options);
}
```

(https://google-developers.appspot.com/chart/interactive/docs/datetimeconstructor) Datetimes

Using the Date Constructor

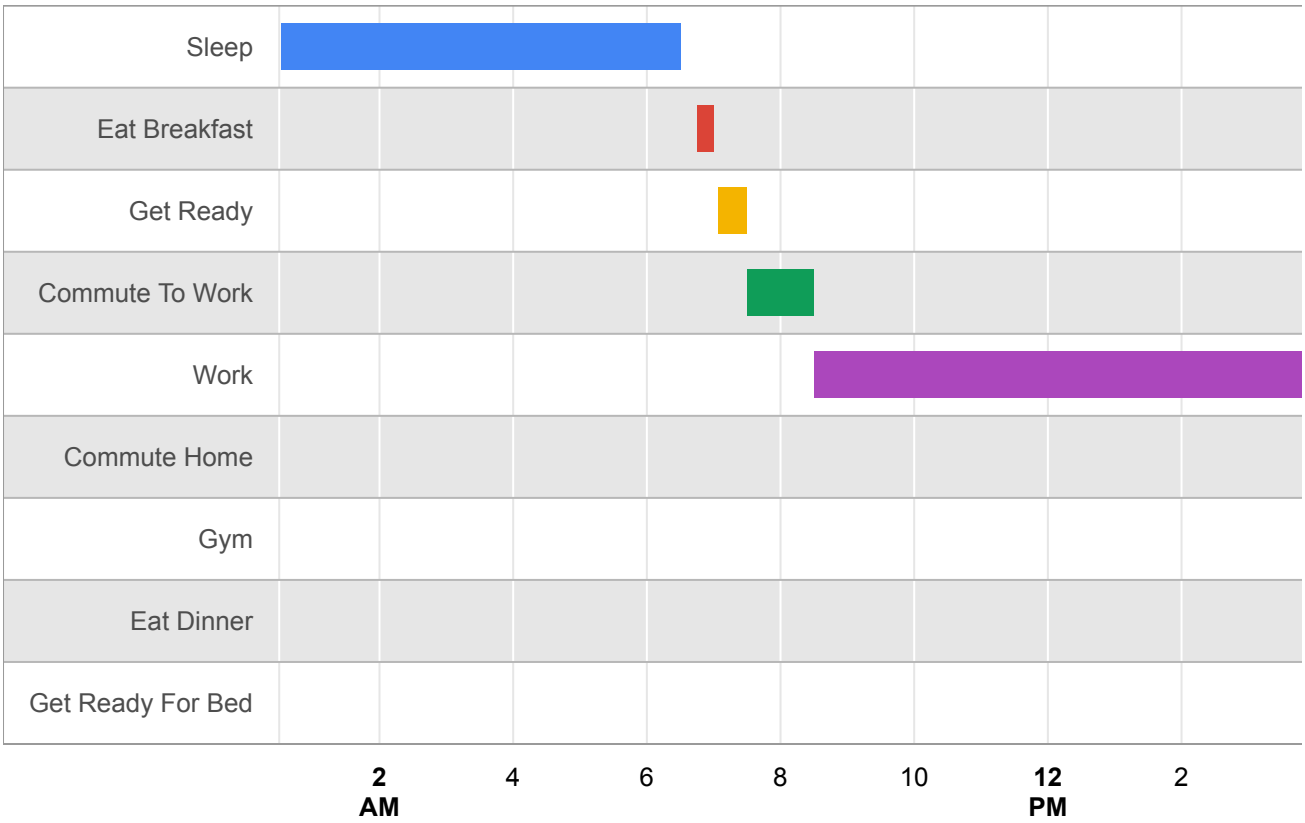
The DataTable **`datetime`** column data type uses the same Date constructor as the **`date`** data type, but now uses all of the arguments to fill out the time.

(https://google-developers.appspot.com/chart/interactive/docs/datestringargument) Alternatively, a string representation of `datetime` can also be passed into the Date constructor. A string representation of `datetime` consists of adding the hours, minutes, and seconds, in addition to the 4-digit timezone offset (e.g. Pacific Standard Time (PST) is -0800). For the RFC 2822 spec, the time and timezone are added with spaces between the date and the time, and the time and the timezone. In the ISO 8601 spec, there are no spaces, instead the date is followed by an uppercase "T" to denote a time component. There is also no space between the time and the timezone offset. The full `datetime` date string for December 6, 2014 at 10:30am PST would be:

- RFC 2822 — `Dec 6, 2014 10:30:00 -0800`.
- ISO 8601 — `2014-12-06T10:30:00-0800`.

Warning: Again, the string representation may be parsed differently by different browsers/versions. Notably, when dealing with time and timezones, there are differences in whether the datetime is returned with a UTC (GMT) timezone, or is offset and returned in local time. This is another reason why the use of datetime strings is **not recommended**.

The below timeline breaks down an average day, using the datetime data type.



CODE IT YOURSELF ON JSFIDDLE

```
google.charts.load('current', {'packages':['timeline']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
  var data = google.visualization.arrayToDataTable([
    ['Activity', 'Start Time', 'End Time'],
    ['Sleep',
     new Date(2014, 10, 15, 0, 30),
     new Date(2014, 10, 15, 6, 30)],
    ['Eat Breakfast',
     new Date(2014, 10, 15, 6, 45),
     new Date(2014, 10, 15, 7)],
    ['Get Ready',
     new Date(2014, 10, 15, 7, 4),
     new Date(2014, 10, 15, 7, 30)],
    ['Commute To Work',
     new Date(2014, 10, 15, 7, 30),
     new Date(2014, 10, 15, 8, 30)],
    ['Work',
     new Date(2014, 10, 15, 8, 30),
     new Date(2014, 10, 15, 17)],
    ['Commute Home',
     new Date(2014, 10,  15, 17),
     new Date(2014, 10,  15, 18)],
    ['Gym',
     new Date(2014, 10, 15, 18),
     new Date(2014, 10,  15, 18, 45)],
    ['Eat Dinner',
     new Date(2014, 10,  15, 19),
     new Date(2014, 10,  15, 20)],
    ['Get Ready For Bed',
     new Date(2014, 10,  15, 21),
     new Date(2014, 10,  15, 22)]
  ]);

  var options = {
    height: 450,
  };

  var chart = new google.visualization.Timeline(document.getElementById('

  chart.draw(data, options);
}
```

(https://google-developers.appspot.com/chart/interactive/docs/timezones)Dates, Times, and Timezones

Using the Date constructor, either for `date` or `datetime`, will return the desired date or datetime in the timezone set by the user's browser. Setting your Date object to a specific timezone can be achieved in a few ways. First, Google Charts provides a Date Formatter (https://google-developers.appspot.com/chart/interactive/docs/reference#dateformatter) in which you can specify a `timeZone`. This will provide a formatted value for each of your `date` and `datetime` values in your DataTable. You can also pass in a string (#datestringargument) as your argument to the `new Date()` constructor, or you can wrap your arguments in the `Date.UTC()` method, such as:

```
new Date(Date.UTC(Year, Month, Day, Hours, Minutes, Seconds, Milliseconds))
```

This will set a Date object to the specified date and time in the UTC (GMT) timezone. From there you can calculate the desired offset for timezone and set your dates and times as desired.

Dates and Times Using the Date String Representation

When serializing data using the JavaScript DataTable object literal notation (https://google-developers.appspot.com/chart/interactive/docs/reference#dataparam) to build your DataTable, the `new Date()` constructor cannot be used. Instead, Google Charts provides a Date string representation that allows your `date` or `datetime` to be serialized and parsed properly when creating a DataTable. This Date string format simply drops the `new` keyword and wraps the remaining expression in quotation marks:

```
"Date(Year, Month, Day, Hours, Minutes, Seconds, Milliseconds)"
```

Important: When using this Date String Represenation, as when using the `new Date()` constructor, months are indexed starting at zero (January is month 0, December is month 11).

Below is the same Super Bowl timeline from before, but now using the JavaScript object literal notation and the Date string format.

CODE IT YOURSELF ON JSFIDDLE

```
google.charts.load('current', {'packages':['timeline']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
  var data = new google.visualization.DataTable({

    cols: [
      {id: 'team', label: 'Team', type: 'string'},
      {id: 'start', label: 'Season Start Date', type: 'date'},
      {id: 'end', label: 'Season End Date', type: 'date'}
    ],

    rows: [
      {c: [{v: 'Baltimore Ravens'},     {v: 'Date(2000, 8, 5)'}, {v: 'Dat
      {c: [{v: 'New England Patriots'}, {v: 'Date(2001, 8, 5)'}, {v: 'Dat
      {c: [{v: 'Tampa Bay Buccaneers'}, {v: 'Date(2002, 8, 5)'}, {v: 'Dat
      {c: [{v: 'New England Patriots'}, {v: 'Date(2003, 8, 5)'}, {v: 'Dat
      {c: [{v: 'New England Patriots'}, {v: 'Date(2004, 8, 5)'}, {v: 'Dat
      {c: [{v: 'Pittsburgh Steelers'},  {v: 'Date(2005, 8, 5)'}, {v: 'Dat
      {c: [{v: 'Indianapolis Colts'},   {v: 'Date(2006, 8, 5)'}, {v: 'Dat
      {c: [{v: 'New York Giants'},      {v: 'Date(2007, 8, 5)'}, {v: 'Dat
      {c: [{v: 'Pittsburgh Steelers'},  {v: 'Date(2008, 8, 5)'}, {v: 'Dat
```

```
        {c: [{v: 'New Orleans Saints'},   {v: 'Date(2009, 8, 5)'}, {v: 'Dat
        {c: [{v: 'Green Bay Packers'},    {v: 'Date(2010, 8, 5)'}, {v: 'Dat
        {c: [{v: 'New York Giants'},      {v: 'Date(2011, 8, 5)'}, {v: 'Dat
        {c: [{v: 'Baltimore Ravens'},     {v: 'Date(2012, 8, 5)'}, {v: 'Dat
        {c: [{v: 'Seattle Seahawks'},     {v: 'Date(2013, 8, 5)'}, {v: 'Dat
      ]
    });

    var options = {
      height: 450,
      timeline: {
        groupByRowLabel: true
      }
    };

    var chart = new google.visualization.Timeline(document.getElementById('

    chart.draw(data, options);
  }
```

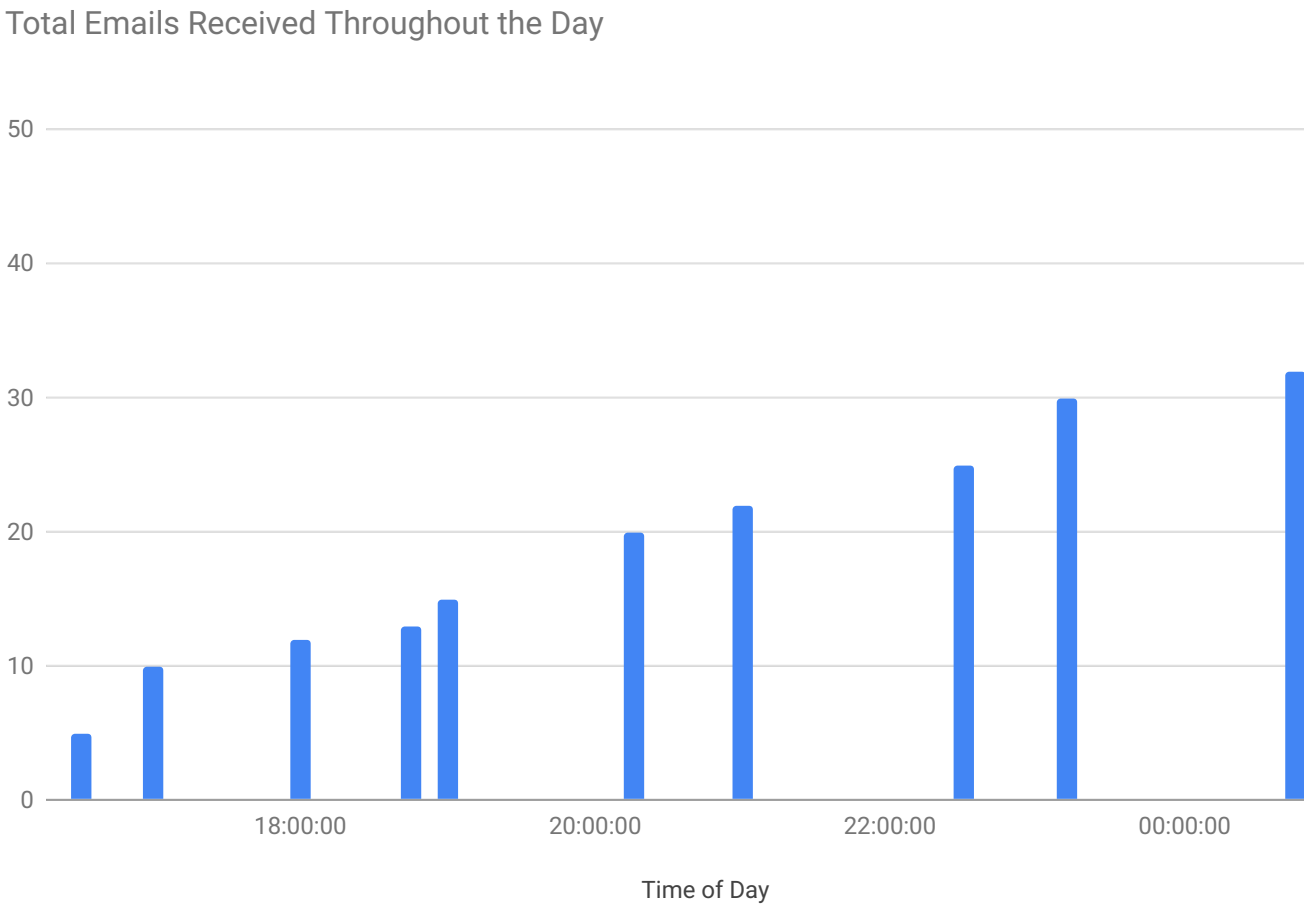
This format can also be used in the **`arrayToDataTable()`** method, provided that in the first array, where column labels are specified, you declare the necessary column as being of **`type: 'date'`** or **`type: 'datetime'`**.

```
var data = google.visualization.arrayToDataTable([
  ["Team", {type: 'date', label: 'Season Start Date'}, {type: 'date', label:
  ["Baltimore Ravens",     "Date(2000, 8, 5)", "Date(2001, 1, 5)"],
  ["New England Patriots", "Date(2001, 8, 5)", "Date(2002, 1, 5)"],
  ["Tampa Bay Buccaneers", "Date(2002, 8, 5)", "Date(2003, 1, 5)"],
  ["New England Patriots", "Date(2003, 8, 5)", "Date(2004, 1, 5)"],
  ["New England Patriots", "Date(2004, 8, 5)", "Date(2005, 1, 5)"],
  ["Pittsburgh Steelers",  "Date(2005, 8, 5)", "Date(2006, 1, 5)"],
  ["Indianapolis Colts",   "Date(2006, 8, 5)", "Date(2007, 1, 5)"],
  ["New York Giants",      "Date(2007, 8, 5)", "Date(2008, 1, 5)"],
  ["Pittsburgh Steelers",  "Date(2008, 8, 5)", "Date(2009, 1, 5)"],
  ["New Orleans Saints",   "Date(2009, 8, 5)", "Date(2010, 1, 5)"],
  ["Green Bay Packers",    "Date(2010, 8, 5)", "Date(2011, 1, 5)"],
  ["New York Giants",      "Date(2011, 8, 5)", "Date(2012, 1, 5)"],
  ["Baltimore Ravens",     "Date(2012, 8, 5)", "Date(2013, 1, 5)"],
  ["Seattle Seahawks",     "Date(2013, 8, 5)", "Date(2014, 1, 5)"]
]);
```

Working With Timeofday

The DataTable `timeofday` column data type takes an array of either 3 or 4 numbers, representing hours, minutes, seconds, and optionally milliseconds, respectively. Using `timeofday` is different than using `date` and `datetime` in that the values are not specific to a date, whereas `date` and `datetime` always specify a date.

For example, the time 8:30am would be: `[8, 30, 0, 0]`, with the 4th value being optional (`[8, 30, 0]` would output the same timeofday value).



CODE IT YOURSELF ON JSFIDDLE

```
google.charts.load('current', {'packages':['bar']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {

  var data = new google.visualization.DataTable();
  data.addColumn('timeofday', 'Time of Day');
  data.addColumn('number', 'Emails Received');

  data.addRows([
    [[8, 30, 45], 5],
    [[9, 0, 0], 10],
```

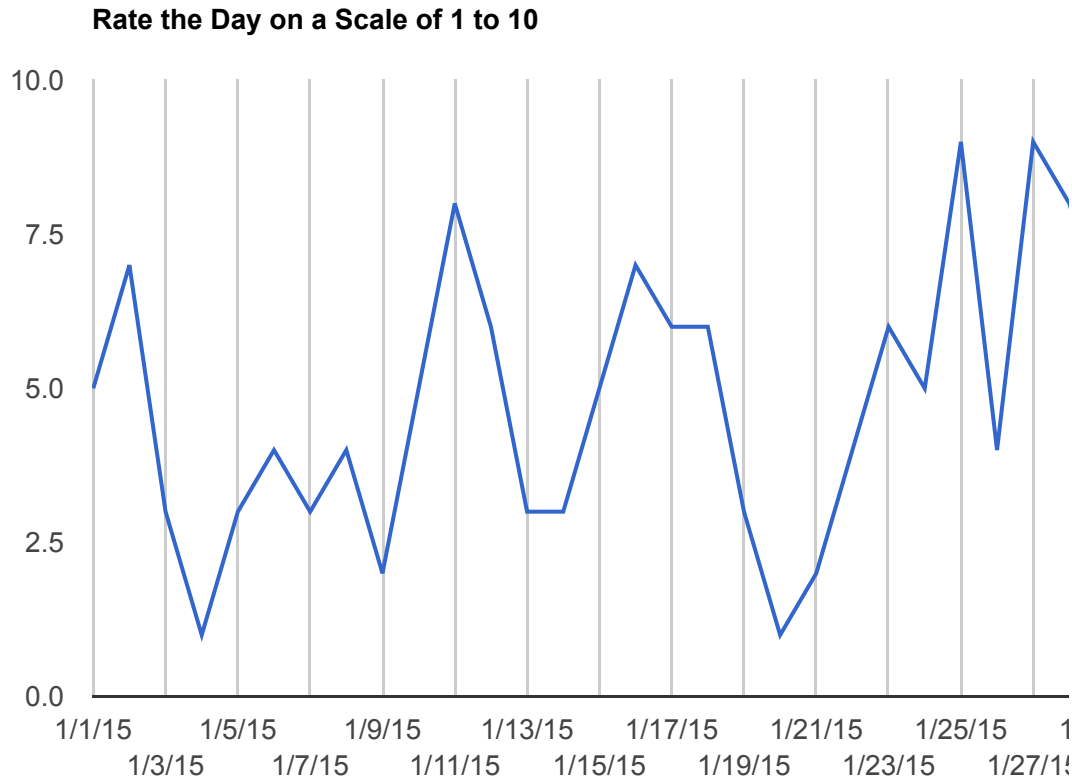
```
      [[10, 0, 0, 0], 12],
      [[10, 45, 0, 0], 13],
      [[11, 0, 0, 0], 15],
      [[12, 15, 45, 0], 20],
      [[13, 0, 0, 0], 22],
      [[14, 30, 0, 0], 25],
      [[15, 12, 0, 0], 30],
      [[16, 45, 0], 32],
      [[16, 59, 0], 42]
    ]);

    var options = google.charts.Bar.convertOptions({
      title: 'Total Emails Received Throughout the Day',
      height: 450
    });

    var chart = new google.charts.Bar(document.getElementById('chart_div'))

    chart.draw(data, options);
  }
```

Formatting Axis, Gridline, and Tick Labels

When working with dates, datetime, and timeofday, it may be desired to format the axis labels, gridline labels, or tick labels a certain way. This can be achieved in a few ways.

First, you can use the **`hAxis.format`** or **`vAxis.format`** option. This option applies when the **`gridlines.count`** option is omitted, in which case the chart defaults to a count of 5, as well as when it is set to a number other than -1. This allows you to specify a format string, where you use placeholder letters for different parts of your date/datetime/timeofday. See the date formatter (https://google-developers.appspot.com/chart/interactive/docs/reference#dateformatter) reference, specifically the **`pattern`** section for more information on the placeholders and how they work.

CLICK TO CHANGE THE FORMAT



CODE IT YOURSELF ON JSFIDDLE

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {

  var data = new google.visualization.DataTable();
  data.addColumn('date', 'Time of Day');
  data.addColumn('number', 'Rating');

  data.addRows([
    [new Date(2015, 0, 1), 5],  [new Date(2015, 0, 2), 7],  [new Date(20
    [new Date(2015, 0, 4), 1],  [new Date(2015, 0, 5), 3],  [new Date(20
    [new Date(2015, 0, 7), 3],  [new Date(2015, 0, 8), 4],  [new Date(20
    [new Date(2015, 0, 10), 5], [new Date(2015, 0, 11), 8], [new Date(20
    [new Date(2015, 0, 13), 3], [new Date(2015, 0, 14), 3], [new Date(20
    [new Date(2015, 0, 16), 7], [new Date(2015, 0, 17), 6], [new Date(20
    [new Date(2015, 0, 19), 3], [new Date(2015, 0, 20), 1], [new Date(20
    [new Date(2015, 0, 22), 4], [new Date(2015, 0, 23), 6], [new Date(20
```

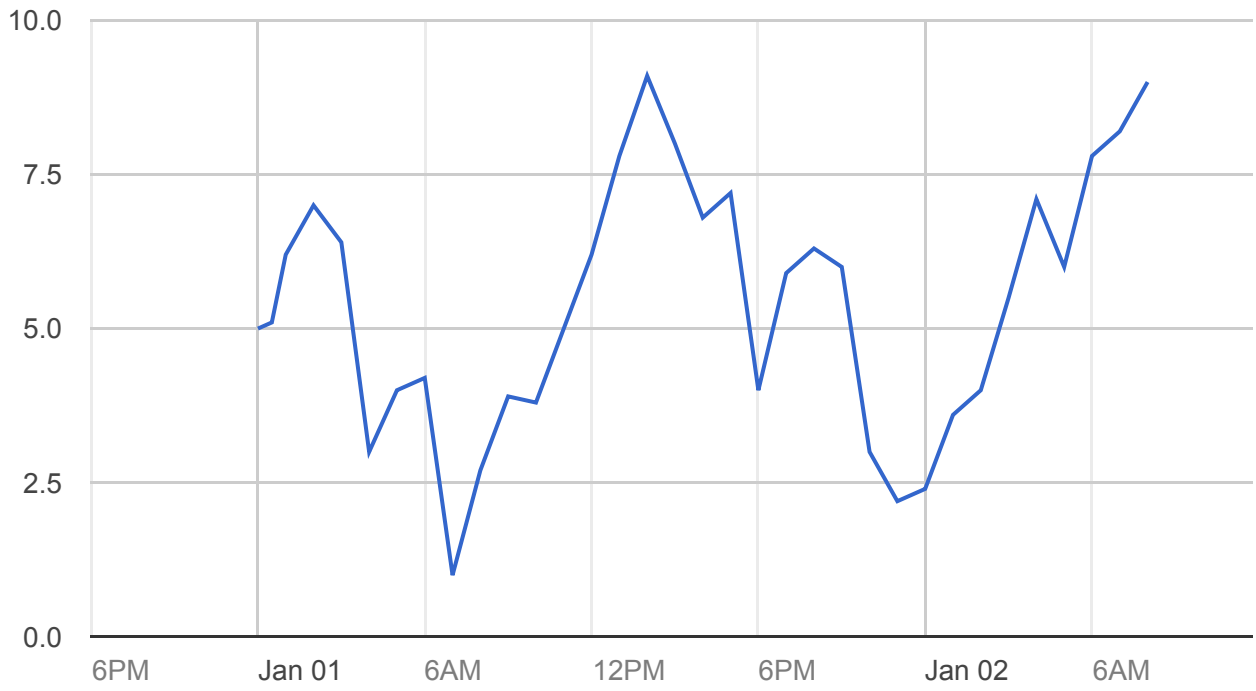
```
        [new Date(2015, 0, 25), 9], [new Date(2015, 0, 26), 4], [new Date(20
        [new Date(2015, 0, 28), 8], [new Date(2015, 0, 29), 6], [new Date(20
        [new Date(2015, 0, 31), 6], [new Date(2015, 1, 1), 7],  [new Date(20
      ]);


      var options = {
        title: 'Rate the Day on a Scale of 1 to 10',
        width: 900,
        height: 500,
        hAxis: {
          format: 'M/d/yy',
          gridlines: {count: 15}
        },
        vAxis: {
          gridlines: {color: 'none'},
          minValue: 0
        }
      };

      var chart = new google.visualization.LineChart(document.getElementById

      chart.draw(data, options);

      var button = document.getElementById('change');

      button.onclick = function () {

        // If the format option matches, change it to the new option,
        // if not, reset it to the original format.
        options.hAxis.format === 'M/d/yy' ?
        options.hAxis.format = 'MMM dd, yyyy' :
        options.hAxis.format = 'M/d/yy';

        chart.draw(data, options);
      };
    }
```

You can also provide formatting rules for specific units of date and time values by including a `units` option under `gridlines` and `minorGridlines` for both axes. This option will only be used if the `gridlines.count` option is set to -1.

The `gridlines.units` option is an object, where you specify the format for the different aspects of date/datetime/timeofday for the computed gridline, and your chart will compute the gridlines based on the first format which fits the space for the gridline label. You can set the formats for years, months, days, hours, minutes, seconds, and milliseconds.

The format option accepts an array of string formats, and will use them in order until a format fits the label area. For this reason, it is recommended to list the formats in order from longest to shortest. The string formats use the same patterns as the date formatter (https://google-developers.appspot.com/chart/interactive/docs/reference#dateformatter) reference mentioned earlier.

CHANGE VIEW WINDOW



CODE IT YOURSELF ON JSFIDDLE

Note that in the above chart, when changing the view window, the format for the `hours` unit changes, given that hours switch from minor to major gridlines, and the format in the options changes with them. Also, note that minorGridlines are using the second, shorter formats, as the first formats do not fit the space in each instance.

```
        hAxis: {
          viewWindow: {
            min: new Date(2014, 11, 31, 18),
```

```
          max: new Date(2015, 0, 3, 1)
        },
        gridlines: {
          count: -1,
          units: {
            days: {format: ['MMM dd']},
            hours: {format: ['HH:mm', 'ha']},
          }
        },
        minorGridlines: {
          units: {
            hours: {format: ['hh:mm:ss a', 'ha']},
            minutes: {format: ['HH:mm a Z', ':mm']}
          }
        }
      }
```

(https://google-developers.appspot.com/chart/interactive/docs/moreinfomdn)

More Information on JavaScript Dates

If you would like to learn more about the JavaScript `Date()` object, the Mozilla Developer Network is a great resource. There you can learn all about JavaScript Date objects. (//developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date)

上次更新日期：二月 23, 2017