Calendar Chart

Overview

Note: JavaScript counts months starting at **zero**: January is 0, February is 1, and December is 11. If your calendar chart seems off by a month, this is why.

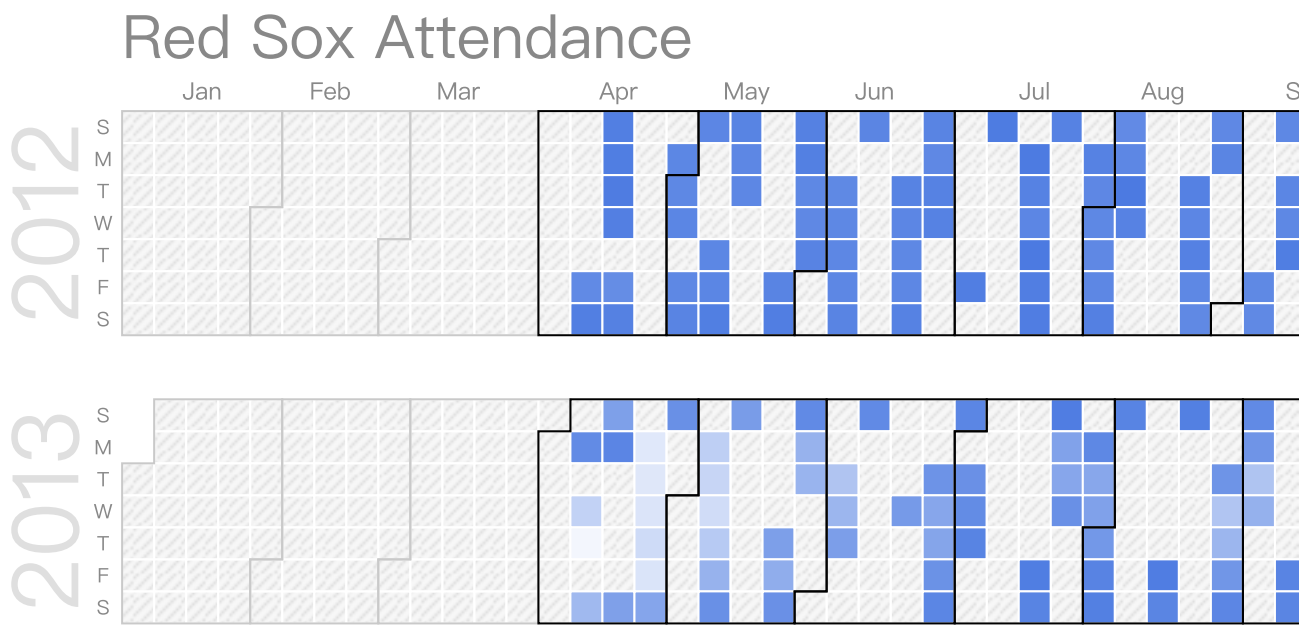
A *calendar chart* is a visualization used to show activity over the course of a long span of time, such as months or years. They're best used when you want to illustrate how some quantity varies depending on the day of the week, or how it trends over time.

The calendar chart may be undergoing substantial revisions in future Google Charts releases.

Calendar charts are rendered in the browser using SVG (http://www.w3.org/Graphics/SVG/) or VML (http://en.wikipedia.org/wiki/Vector_Markup_Language), whichever is appropriate for the user's browser. Like all Google charts, calendar charts display tooltips when the user hovers over the data. And credit where credit is due: our calendar chart was inspired by the D3 calendar visualization.

A Simple Example

Let's say we wanted to display how the attendance for a sports team varied throughout the season. With a calendar chart, we can use brightness to indicate the values and let people see trends at a glance:



You can mouse over the individual days to see the underlying data values.

To create a calendar chart, load the `calendar` package and then create two columns, one for the dates and one for the values. (An optional third column for customized styling is coming in a future Google Charts release.)

Then fill in your rows with date-value pairs, as shown below.

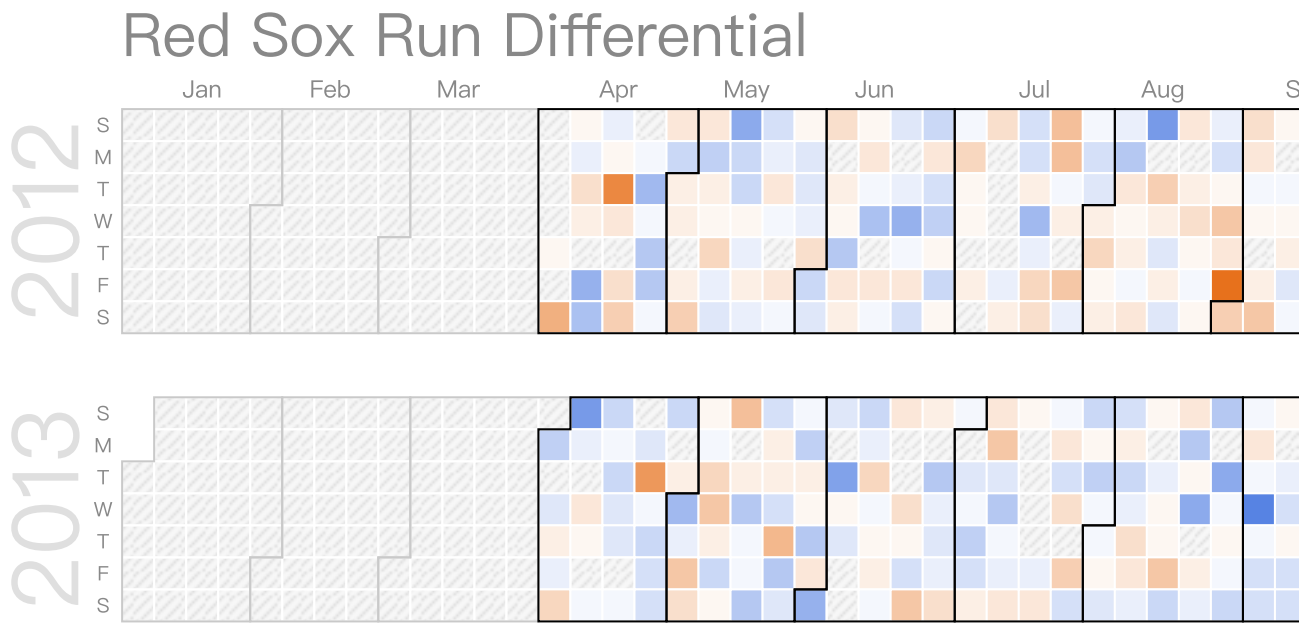
```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader
    <script type="text/javascript">
      google.charts.load("current", {packages:["calendar"]});
      google.charts.setOnLoadCallback(drawChart);

   function drawChart() {
       var dataTable = new google.visualization.DataTable();
       dataTable.addColumn({ type: 'date', id: 'Date' });
       dataTable.addColumn({ type: 'number', id: 'Won/Loss' });
       dataTable.addRows([
          [ new Date(2012, 3, 13), 37032 ],
          [ new Date(2012, 3, 14), 38024 ],
          [ new Date(2012, 3, 15), 38024 ],
          [ new Date(2012, 3, 16), 38108 ],
          [ new Date(2012, 3, 17), 38229 ],
          // Many rows omitted for brevity.
          [ new Date(2013, 9, 4), 38177 ],
          [ new Date(2013, 9, 5), 38705 ],
          [ new Date(2013, 9, 12), 38210 ],
```

```
          [ new Date(2013, 9, 13), 38029 ],
          [ new Date(2013, 9, 19), 38823 ],
          [ new Date(2013, 9, 23), 38345 ],
          [ new Date(2013, 9, 24), 38436 ],
          [ new Date(2013, 9, 30), 38447 ]
        ]);

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

        var options = {
          title: "Red Sox Attendance",
          height: 350,
        };

        chart.draw(dataTable, options);
    }
    </script>
  </head>
  <body>
    <div id="calendar_basic" style="width: 1000px; height: 350px;"></div>
  </body>
</html>
```

Days

Each square in a calendar chart represents a day. Currently, the color of the data cells can't be customized, although that will change in the next release of Google Charts.

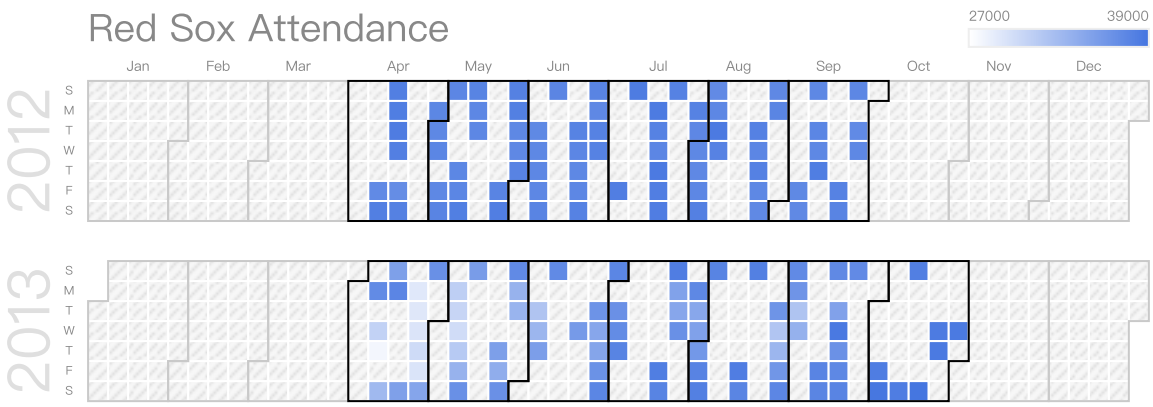
If the data values are all positive, the colors will range from white to blue, with the deepest blues indicating the highest values. If there are negative data values, they will appear orange, as shown below.



The code for this calendar is similar to the first, except that the rows look like this:

```
[ new Date(2013, 9, 4), 10 ],
[ new Date(2013, 9, 5), 3 ],
[ new Date(2013, 9, 7), -1 ],
[ new Date(2013, 9, 8), 2 ],
[ new Date(2013, 9, 12), -1 ],
[ new Date(2013, 9, 13), 1 ],
[ new Date(2013, 9, 15), 1 ],
[ new Date(2013, 9, 16), -4 ],
```

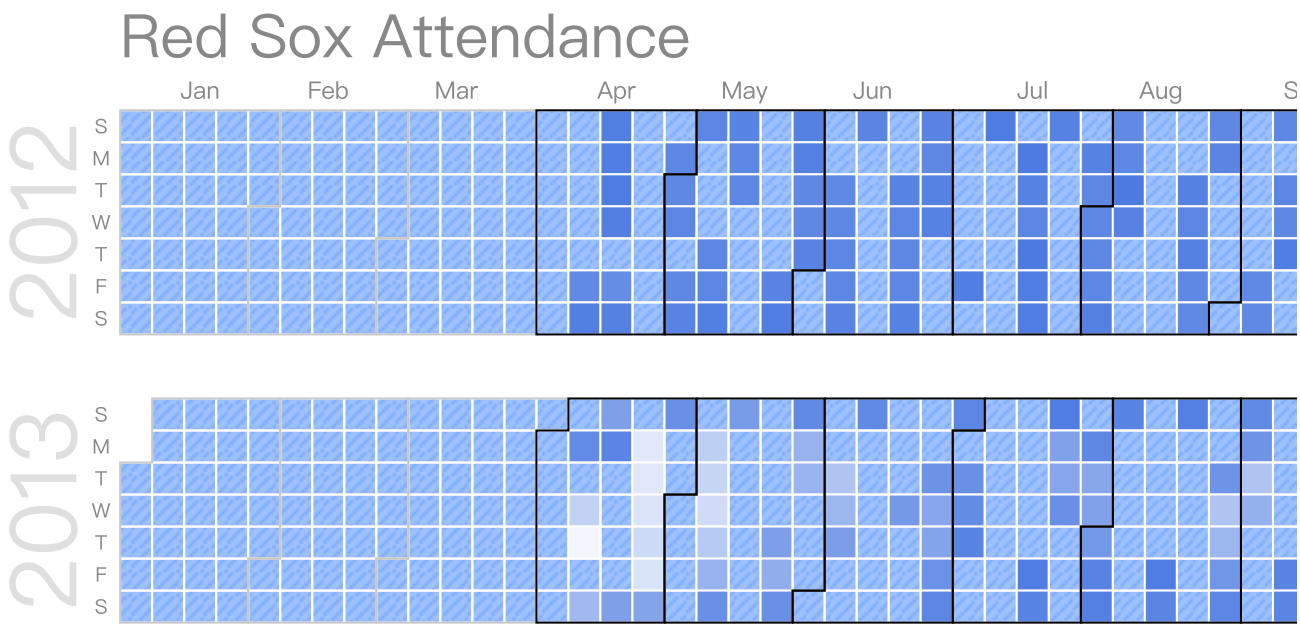
You can change the size of the days ("cells") with the `calendar.cellSize` option:



Here, we changed `calendar.cellSize` to 10, shrinking the days and therefore the chart as a whole.

```
var options = {
  title: 'Red Sox Attendance',
  calendar: { cellSize: 10 },
};
```

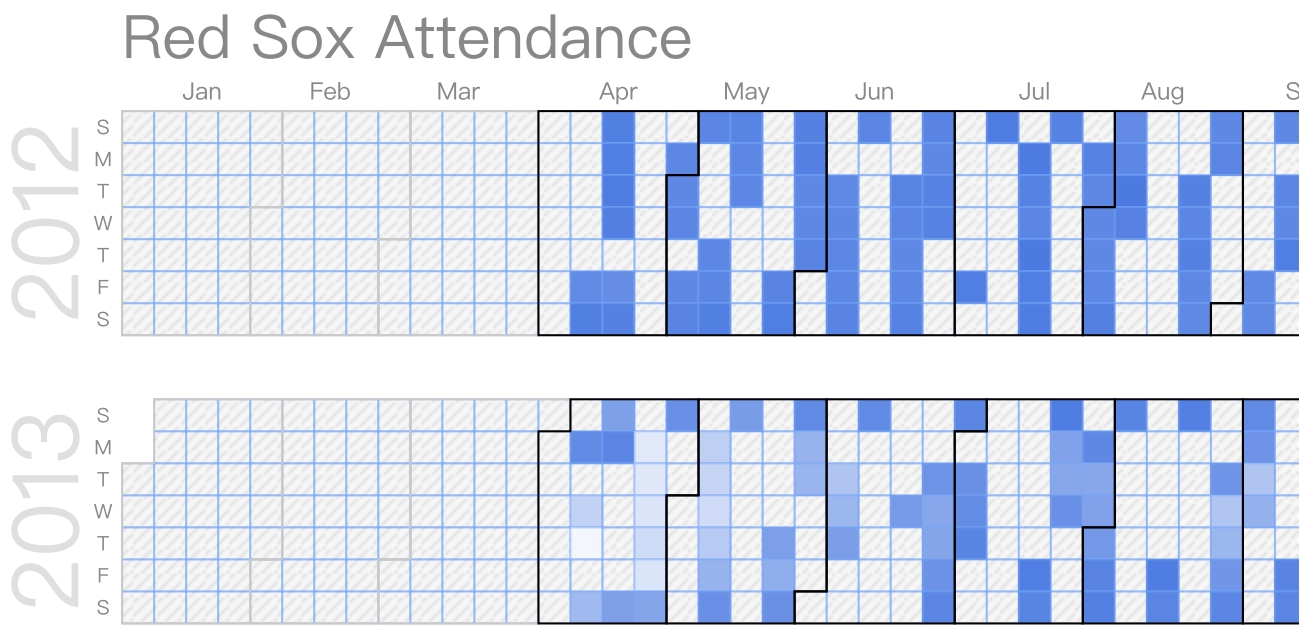
Days with no data values can be customized with the `noDataPattern` option:



Here, we set the `color` to a light blue and `backgroundColor` to a slightly darker shade:

```
var options = {
  title: "Red Sox Attendance",
  height: 350,
  noDataPattern: {
    backgroundColor: '#76a7fa',
    color: '#a0c3ff'
  }
};
```

You can control the cell border color, border width, and opacity with `calendar.cellColor`:



You'll need to be careful to choose a stroke color that will be distinguished from the `monthOutlineColor`, or to choose a low opacity. Here are the options for the chart above:

```
var options = {
  title: 'Red Sox Attendance',
  height: 350,
  calendar: {
    cellColor: {
      stroke: '#76a7fa',
      strokeOpacity: 0.5,
      strokeWidth: 1,
    }
  }
};
```

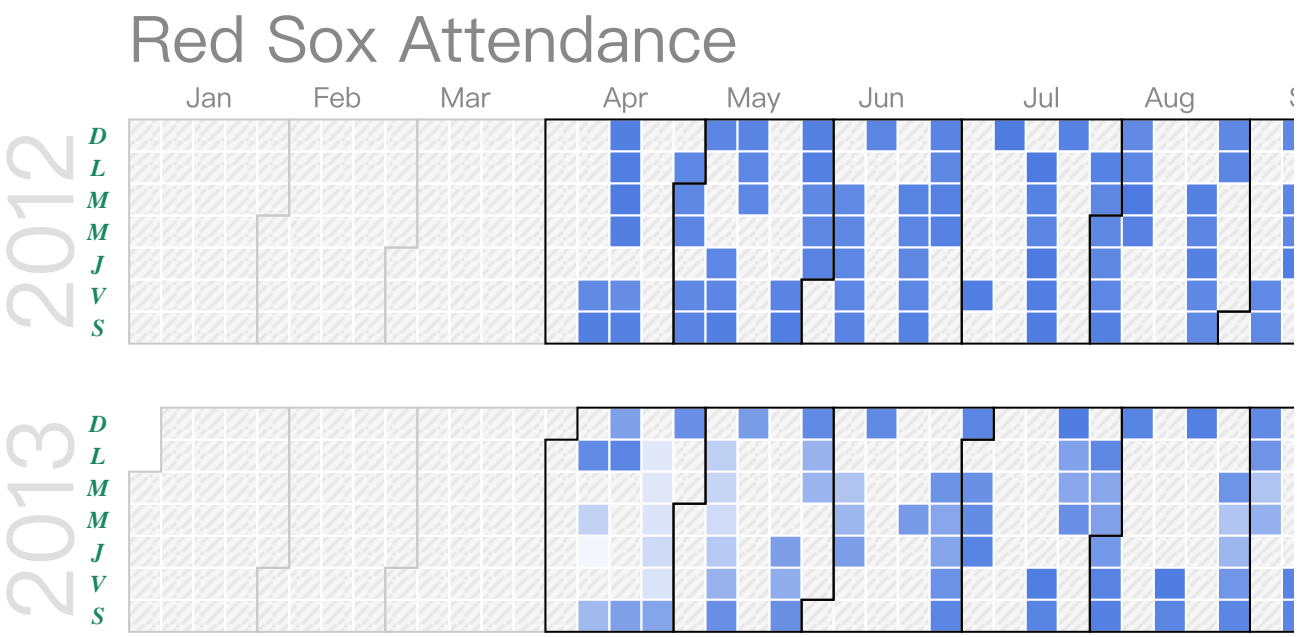
If you focus on a day in the above chart, the border will highlight in red. You can control that behavior with the `calendar.focusedCellColor` options:

```
var options = {
  title: 'Red Sox Attendance',
  height: 350,
  calendar: {
    focusedCellColor: {
      stroke: '#d3362d',
      strokeOpacity: 1,
      strokeWidth: 1,
    }
```

```
  }
};
```

Weeks

By default, the days of the week are labeled with the first letters of Sunday through Saturday. You can't change the ordering of the days, but you can change what letters are used with the `calendar.daysOfWeek` option. Also, you can control the padding between the days of the week and the chart with `calendar.dayOfWeekRightSpace`, and you can customize the text style with `calendar.dayOfWeekLabel`:



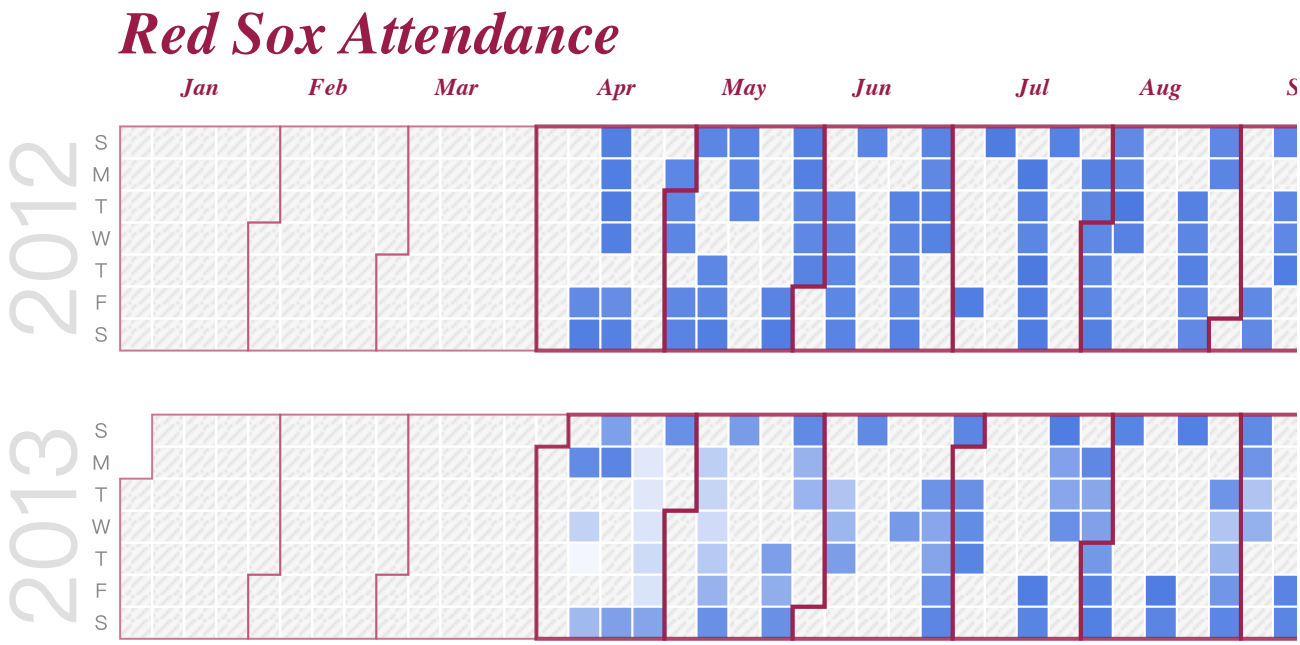
Here, we change the font of the week labels, put in a padding of 10 pixels between the labels and the chart data, and start weeks on Monday.

```
var options = {
  title: 'Red Sox Attendance',
  height: 350,
  calendar: {
    dayOfWeekLabel: {
      fontName: 'Times-Roman',
      fontSize: 12,
      color: '#1a8763',
      bold: true,
      italic: true,
    },
    dayOfWeekRightSpace: 10,
```

```
    daysOfWeek: 'DLMMJVS',
  }
};
```

Months

By default, months are identified by dark grey lines. You can use the `calendar.monthOutlineColor` option to control the borders, the `calendar.monthLabel` to customize the label font, and `calendar.underMonthSpace` to adjust the label padding:



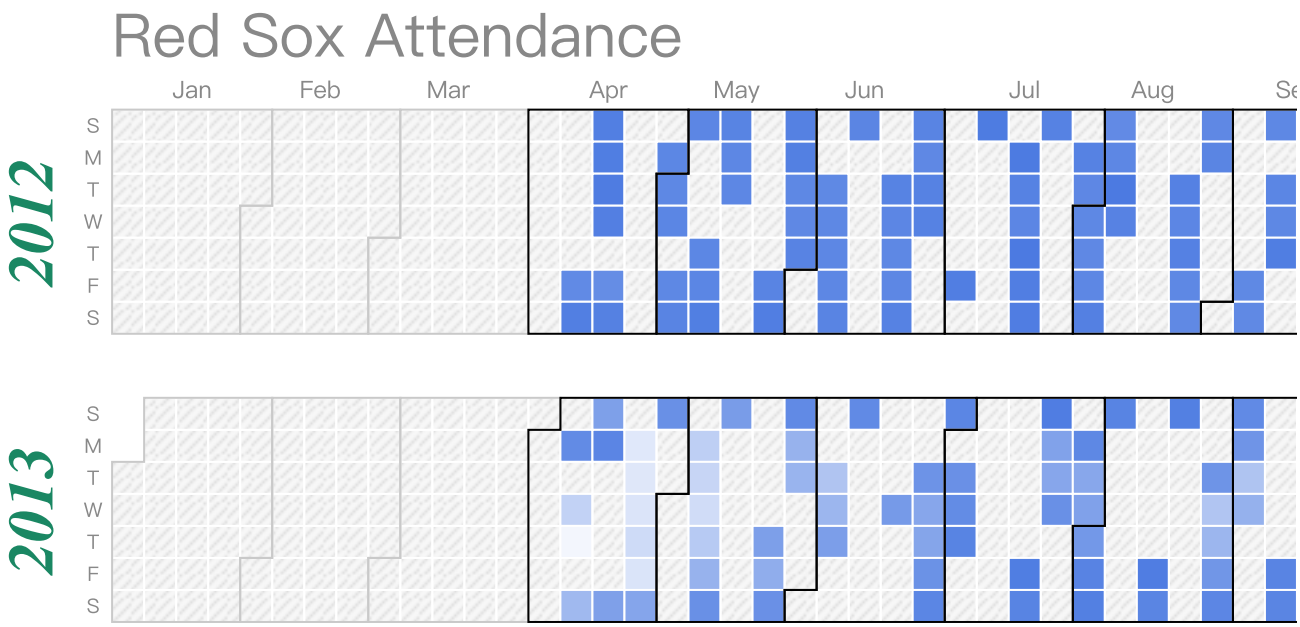
We set the label font to a deep red 12pt Times-Roman bold italic, set the outlines to the same color, and put in a padding of 16 pixels. The unused month outlines are set to a fainter color of the same hue.

```
var options = {
  title: 'Red Sox Attendance',
  height: 350,
  calendar: {
    monthLabel: {
      fontName: 'Times-Roman',
      fontSize: 12,
      color: '#981b48',
      bold: true,
      italic: true
    },
    monthOutlineColor: {
```

```
      stroke: '#981b48',
      strokeOpacity: 0.8,
      strokeWidth: 2
    },
    unusedMonthOutlineColor: {
      stroke: '#bc5679',
      strokeOpacity: 0.8,
      strokeWidth: 1
    },
    underMonthSpace: 16,
  }
};
```

Years

The years in calendar charts are always on the left edge of the chart, and can be customized with `calendar.yearLabel` and `calendar.underYearSpace`:



We set the year font to a dark green 32pt Times-Roman bold italic, and add ten pixels between the year labels and the bottom of the chart:

```
var options = {
  title: 'Red Sox Attendance',
  height: 350,
  calendar: {
    underYearSpace: 10, // Bottom padding for the year labels.
    yearLabel: {
```

```
      fontName: 'Times-Roman',
      fontSize: 32,
      color: '#1A8763',
      bold: true,
      italic: true
    }
  }
};
```

Loading

The `google.charts.load` package name is `"calendar"`:

```
google.charts.load("current", {packages: ["calendar"]});
```

The visualization's class name is `google.visualization.Calendar`:

```
var visualization = new google.visualization.Calendar(container);
```

Data Format

Rows: Each row in the table represents a date.

Columns:

	Column 0	Column 1	...Column *N* (optiona
Purpose:	Dates	Values	... Optional roles
Data Type:	date, datetime, or timeofday	number	...
Role:	domain	data	...
Optional column roles (https://developers.google.com/chart/interactive/docs/roles):	*None*	*None*	... • tooltip (https://developer

Configuration Options

Name	

calendar.cellColor	The **`calendar.cellColor`** option lets you customize the border of t squares:
	
	`var options = {`
	`  calendar: {`
	`  cellColor: {`
	`  stroke: 'red',  // Color the border of th`
	`  strokeOpacity: 0.5,  // Make the borders half`
	`  strokeWidth: 2  // ...and two pixels thic`
	`  }`
	`  }`
	`};`
	
	Type: object
	Default: **`{ stroke: '#fff', strokeOpacity: 1, strokeWid`**

calendar.cellSize	The size of the calendar day squares: `var options = { calendar: { cellSize: 10 } };` **Type:** integer **Default:** 16

calendar.dayOfWeekLabel	Controls the font style of the week labels at the top of the chart:
	
	`var options = {`
	`  calendar: {`
	`  dayOfWeekLabel: {`
	`  fontName: 'Times-Roman',`
	`  fontSize: 12,`
	`  color: 'black',`
	`  bold: false,`
	`  italic: false`
	`  }`
	`  }`
	`};`

	Type: object **Default:** `{ fontName: 'sans-serif', color: '#888', bold italic: false }`
calendar.dayOfWeekRightSpace	The distance between the right edge of the week labels and the left ed day squares. **Type:** integer **Default:** 4
calendar.daysOfWeek	The single-letter labels to use for Sunday through Saturday. **Type:** string **Default:** `'SMTWTFS'`
calendar.focusedCellColor	When the user focuses (say, by hovering) over a day square, calendar highlight the square. `var options = { calendar: focusedCellColor: { stroke: 'red', strokeOpacity: 0.8, strokeWidth: 3 } } };` **Type:** object **Default:** `{ stroke: '#000', strokeOpacity: 1, strokeWid`
calendar.monthLabel	Style for the month labels, e.g.: `var options = { calendar: { monthLabel: { fontName: 'Times-Roman', fontSize: 16, color: 'green', bold: true, italic: false } } };` **Type:** object

	Default: `{ fontName: 'sans-serif', color: '#888', bold italic: false }`
calendar.monthOutlineColor	Months with data values are delineated from others using a border in `var options = { calendar: { monthOutlineColor: { stroke: 'blue', strokeOpacity: 0.8, strokeWidth: 2 } } };` (Also see **`calendar.unusedMonthOutlineColor`**.) **Type:** object **Default:** `{ stroke: '#000', strokeOpacity: 1, strokeWid`
calendar.underMonthSpace	The number of pixels between the bottom of the month labels and the squares: `var options = { calendar: { underMonthSpace: 12 } }` **Type:** integer **Default:** 6
calendar.underYearSpace	The number of pixels between the bottom-most year label and the bot chart: `var options = { calendar: { underYearSpace: 2 } };` **Type:** integer **Default:** 0
calendar.unusedMonthOutlineColor	Months *without* data values are delineated from others using a borde `var options = { calendar: { unusedMonthOutlineColor: { stroke: 'yellow', strokeOpacity: 0.8, strokeWidth: 2 } } };`

	(Also see **`calendar.monthOutlineColor`**.) **Type:** object **Default:** **`{ stroke: '#c9c9c9', strokeOpacity: 1, stroke`**
colorAxis	An object that specifies a mapping between color column values and gradient scale. To specify properties of this object, you can use objec as shown here: `{minValue: 0,  colors: ['#FF0000', '#00FF00']}` **Type:** object **Default:** null
colorAxis.colors	Colors to assign to values in the visualization. An array of strings, whe element is an HTML color string, for example: **`colorAxis: {color ['red','#004411']}`**. You must have at least two values; the gradi all your values, plus calculated intermediary values, with the first colo value, and the last color as the highest. **Type:** array of color strings **Default:** null
colorAxis.maxValue	If present, specifies a maximum value for chart color data. Color data value and higher will be rendered as the last color in the **`colorAxis.`** **Type:** number **Default:** Maximum value of color column in chart data
colorAxis.minValue	If present, specifies a minimum value for chart color data. Color data value and lower will be rendered as the first color in the **`colorAxis.`** **Type:** number **Default:** Minimum value of color column in chart data
colorAxis.values	If present, controls how values are associated with colors. Each value with the corresponding color in the **`colorAxis.colors`** array. These the chart color data. Coloring is done according to a gradient of the va here. Not specifying a value for this option is equivalent to specifying maxValue]. **Type:** array of numbers **Default:** null
forceIFrame	Draws the chart inside an inline frame. (Note that on IE8, this option is charts are drawn in i-frames.) **Type:** boolean **Default:** false

height	Height of the chart, in pixels. **Type:** number **Default:** height of the containing element

noDataPattern	Calendar charts use a striped diagonal pattern to indicate that there is particular day. Use the **`noDataPattern.backgroundColor`** and **`noDataPattern.color`** options to override the grayscale defaults, e
	
	`noDataPattern: {`
	`  backgroundColor: '#76a7fa',`
	`  color: '#a0c3ff'`
	`}`
	
	Type: object
	Default: null

tooltip.isHtml	Set to **`false`** to use SVG-rendered (rather than HTML-rendered) toolti Customizing Tooltip Content (https://developers.google.com/chart/interactive/docs/customizing_ for more details. ★ **Note:** customization of the HTML tooltip content via the tooltip colum (https://developers.google.com/chart/interactive/docs/roles#tooltip supported by the Pie Chart (https://developers.google.com/chart/interactive/docs/gallery/piech Chart (https://developers.google.com/chart/interactive/docs/gallery/ visualizations. **Type:** boolean **Default:** true
width	Width of the chart, in pixels. **Type:** number **Default:** width of the containing element

Methods

Method	
`draw(data, options)`	Draws the chart. The chart accepts further method calls only after the (#Events)event is fired. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#vis

	Return Type: none

`getBoundingBox(id)`	Returns an object containing the left, top, width, and height of chart ele
	The format for **`id`** isn't yet documented (they're the return values of eve
	(https://developers.google.com/chart/interactive/docs/events)), but h
	some examples:
	
	`var cli = chart.getChartLayoutInterface();`
	
	Height of the chart area
	
	`cli.getBoundingBox('chartarea').height`
	
	Width of the third bar in the first series of a bar or column cl
	
	`cli.getBoundingBox('bar#0#2').width`
	
	Bounding box of the fifth wedge of a pie chart
	
	`cli.getBoundingBox('slice#4')`
	
	Bounding box of the chart data of a vertical (e.g., column) c
	
	`cli.getBoundingBox('vAxis#0#gridline')`
	
	Bounding box of the chart data of a horizontal (e.g., bar) cha
	
	`cli.getBoundingBox('hAxis#0#gridline')`
	
	Values are relative to the container of the chart. Call this *after* the chart
	
	Return Type: object

`getSelection()`	Returns an array of the selected chart entities. Selectable entities are b entries and categories. A bar corresponds to a cell in the data table, a l to a column (row index is null), and a category to a row (column index i this chart, only one entity can be selected at any given moment. Exter `description` (https://developers.google.com/chart/interactive/docs/reference#vis . **Return Type:** Array of selection elements
`setSelection()`	Selects the specified chart entities. Cancels any previous selection. Se entities are bars, legend entries and categories. A bar corresponds to a data table, a legend entry to a column (row index is null), and a categor (column index is null). For this chart, only one entity can be selected at `Extended description`

	(https://developers.google.com/chart/interactive/docs/reference#vis . **Return Type:** none
`clearChart()`	Clears the chart, and releases all of its allocated resources. **Return Type:** none

Events

Name	
`error`	Fired when an error occurs when attempting to render the chart. **Properties:** id, message
`onmouseover`	Fired when the user mouses over a visual entity. Passes back the row index and date value of the entity. If there is no data table element for the entity, the value returned for the row index is **`undefined`**. **Properties:** row, date
`onmouseout`	Fired when the user mouses away from a visual entity. Passes back the row index and date value of the entity. If there is no data table element for the entity, the value returned for the row index is **`undefined`**. **Properties** row, date
`ready`	The chart is ready for external method calls. If you want to interact with the chart, and call methods after you draw it, you should set up a listener for this event *before* you call the **`draw`** method, and call them only after the event was fired. **Properties:** none
`select`	Fired when the user clicks a visual entity. To learn what has been selected, call `getSelection()` (#Methods). **Properties:** none

Data Policy

All code and data are processed and rendered in the browser. No data is sent to any server.

上次更新日期：二月 23, 2017