# Gantt Charts

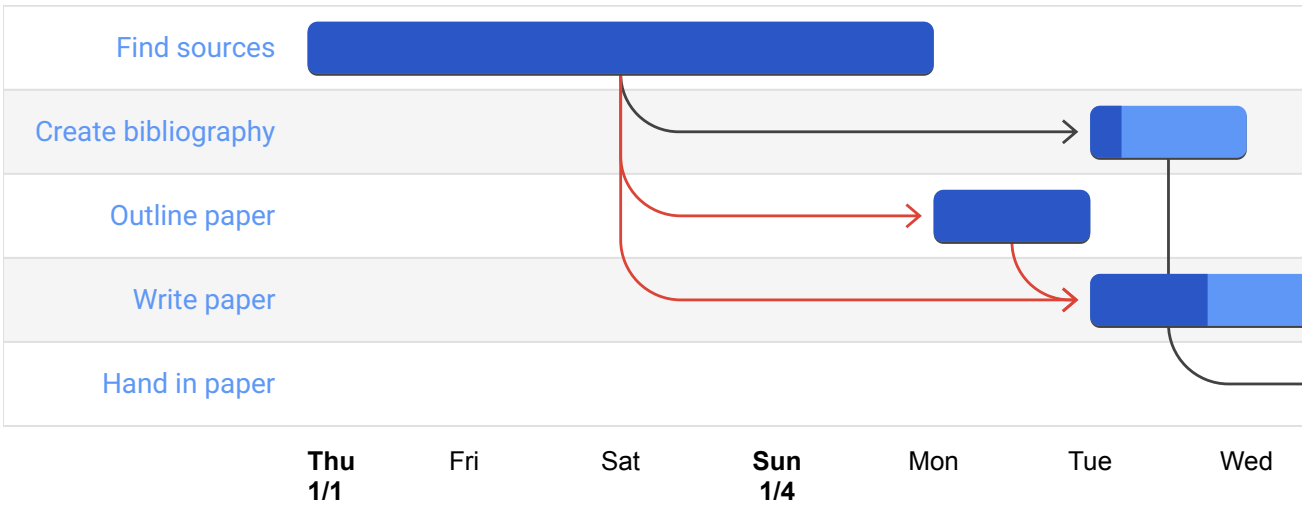
## Overview

A *Gantt chart* is a type of chart that illustrates the breakdown of a project into its component tasks. Google Gantt charts illustrate the start, end, and duration of tasks within a project, as well as any dependencies a task may have. Google Gantt charts are rendered in the browser using SVG (http://www.w3.org/Graphics/SVG/). Like all Google charts, Gantt charts display tooltips when the user hovers over the data.

**Note:** The Gantt chart is in **beta** and may be undergoing substantial revisions in future Google Charts releases.

**Note:** Gantt Charts will **not** work in old versions of Internet Explorer. (IE8 and earlier versions don't support SVG, which Gantt Charts require.)

## A simple example



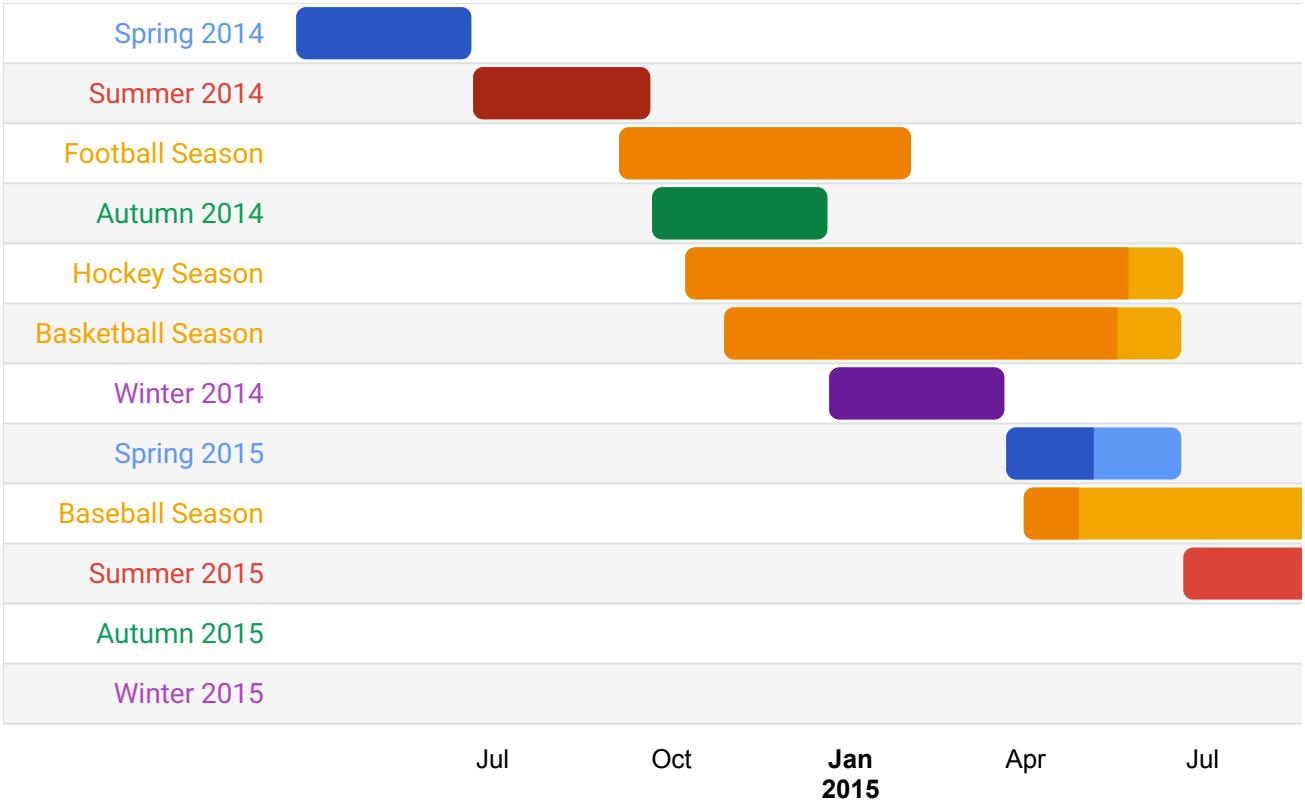

CODE IT YOURSELF ON JSFIDDLE

```
<html>
<head>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.j
  <script type="text/javascript">
```

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

    function daysToMilliseconds(days) {
      return days * 24 * 60 * 60 * 1000;
    }

    function drawChart() {

      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Task ID');
      data.addColumn('string', 'Task Name');
      data.addColumn('date', 'Start Date');
      data.addColumn('date', 'End Date');
      data.addColumn('number', 'Duration');
      data.addColumn('number', 'Percent Complete');
      data.addColumn('string', 'Dependencies');

      data.addRows([
        ['Research', 'Find sources',
         new Date(2015, 0, 1), new Date(2015, 0, 5), null,  100,  null],
        ['Write', 'Write paper',
         null, new Date(2015, 0, 9), daysToMilliseconds(3), 25, 'Research,Outl
        ['Cite', 'Create bibliography',
         null, new Date(2015, 0, 7), daysToMilliseconds(1), 20, 'Research'],
        ['Complete', 'Hand in paper',
         null, new Date(2015, 0, 10), daysToMilliseconds(1), 0, 'Cite,Write'],
        ['Outline', 'Outline paper',
         null, new Date(2015, 0, 6), daysToMilliseconds(1), 100, 'Research']
      ]);

      var options = {
        height: 275
      };

      var chart = new google.visualization.Gantt(document.getElementById('cha

      chart.draw(data, options);
    }
  </script>
</head>
<body>
  <div id="chart_div"></div>
</body>
</html>
```

## No dependencies

To create a Gantt chart that has no dependencies, make sure that the last value for each row in your DataTable is set to `null`.




CODE IT YOURSELF ON JSFIDDLE

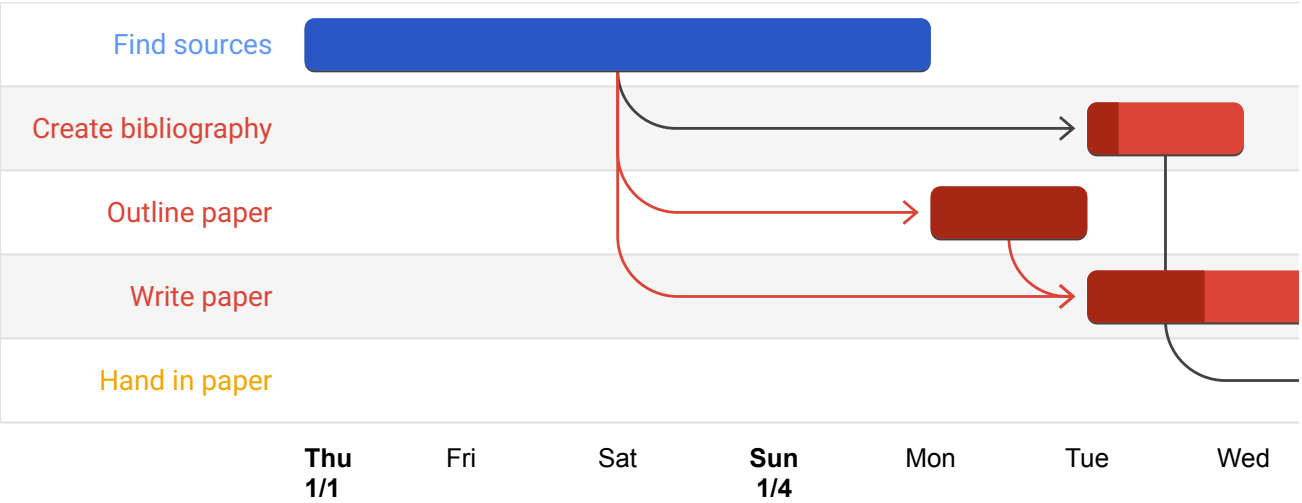
```
<html>
<head>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.j
  <script type="text/javascript">
    google.charts.load('current', {'packages':['gantt']});
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {

      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Task ID');
      data.addColumn('string', 'Task Name');
      data.addColumn('string', 'Resource');
      data.addColumn('date', 'Start Date');
      data.addColumn('date', 'End Date');
      data.addColumn('number', 'Duration');
      data.addColumn('number', 'Percent Complete');
      data.addColumn('string', 'Dependencies');
```

```
      data.addRows([
        ['2014Spring', 'Spring 2014', 'spring',
         new Date(2014, 2, 22), new Date(2014, 5, 20), null, 100, null],
        ['2014Summer', 'Summer 2014', 'summer',
         new Date(2014, 5, 21), new Date(2014, 8, 20), null, 100, null],
        ['2014Autumn', 'Autumn 2014', 'autumn',
         new Date(2014, 8, 21), new Date(2014, 11, 20), null, 100, null],
        ['2014Winter', 'Winter 2014', 'winter',
         new Date(2014, 11, 21), new Date(2015, 2, 21), null, 100, null],
        ['2015Spring', 'Spring 2015', 'spring',
         new Date(2015, 2, 22), new Date(2015, 5, 20), null, 50, null],
        ['2015Summer', 'Summer 2015', 'summer',
         new Date(2015, 5, 21), new Date(2015, 8, 20), null, 0, null],
        ['2015Autumn', 'Autumn 2015', 'autumn',
         new Date(2015, 8, 21), new Date(2015, 11, 20), null, 0, null],
        ['2015Winter', 'Winter 2015', 'winter',
         new Date(2015, 11, 21), new Date(2016, 2, 21), null, 0, null],
        ['Football', 'Football Season', 'sports',
         new Date(2014, 8, 4), new Date(2015, 1, 1), null, 100, null],
        ['Baseball', 'Baseball Season', 'sports',
         new Date(2015, 2, 31), new Date(2015, 9, 20), null, 14, null],
        ['Basketball', 'Basketball Season', 'sports',
         new Date(2014, 9, 28), new Date(2015, 5, 20), null, 86, null],
        ['Hockey', 'Hockey Season', 'sports',
         new Date(2014, 9, 8), new Date(2015, 5, 21), null, 89, null]
      ]);

      var options = {
        height: 400,
        gantt: {
          trackHeight: 30
        }
      };

      var chart = new google.visualization.Gantt(document.getElementById('cha

      chart.draw(data, options);
    }
  </script>
</head>
<body>
  <div id="chart_div"></div>
</body>
</html>
```

(https://developers.google.com/chart/interactive/docs/gallery/Resources)

## Grouping resources

Tasks that are similar in nature can be grouped together using resources. Add a column of type `string` to your data (after the `Task ID` and `Task Name` columns), and make sure any tasks that should be grouped into a resource have the same resource ID. Resources will be grouped by color.




CODE IT YOURSELF ON JSFIDDLE

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

    function daysToMilliseconds(days) {
      return days * 24 * 60 * 60 * 1000;
    }

    function drawChart() {

      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Task ID');
      data.addColumn('string', 'Task Name');
      data.addColumn('string', 'Resource');
      data.addColumn('date', 'Start Date');
      data.addColumn('date', 'End Date');
      data.addColumn('number', 'Duration');
```

```
      data.addColumn('number', 'Percent Complete');
      data.addColumn('string', 'Dependencies');

      data.addRows([
        ['Research', 'Find sources', null,
         new Date(2015, 0, 1), new Date(2015, 0, 5), null,  100,  null],
        ['Write', 'Write paper', 'write',
         null, new Date(2015, 0, 9), daysToMilliseconds(3), 25, 'Research,Outl
        ['Cite', 'Create bibliography', 'write',
         null, new Date(2015, 0, 7), daysToMilliseconds(1), 20, 'Research'],
        ['Complete', 'Hand in paper', 'complete',
         null, new Date(2015, 0, 10), daysToMilliseconds(1), 0, 'Cite,Write']
        ['Outline', 'Outline paper', 'write',
         null, new Date(2015, 0, 6), daysToMilliseconds(1), 100, 'Research']
      ]);

      var options = {
        height: 275
      };

      var chart = new google.visualization.Gantt(document.getElementById('cha

      chart.draw(data, options);
    }
  </script>
</head>
<body>
  <div id="chart_div"></div>
</body>
</html>
```

(https://developers.google.com/chart/interactive/docs/gallery/ComputingTime)

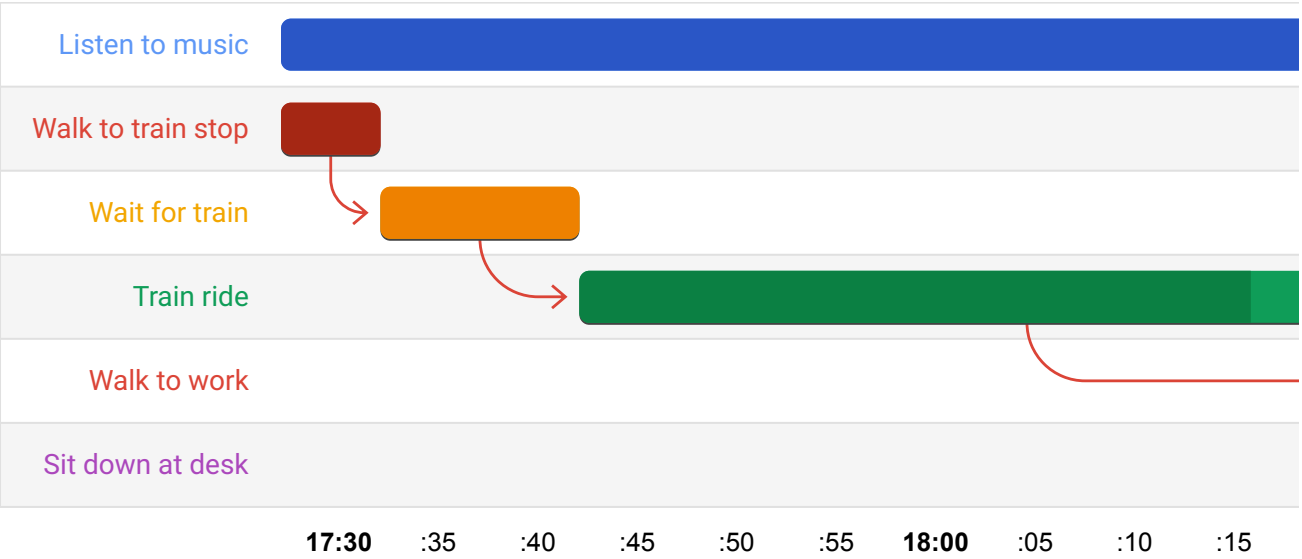
## Computing start/end/duration

Gantt charts accept three values relating to the duration of the task: a start date, an end date, and a duration (in milliseconds). If, for example, there is no start date, the chart can calculate the missing time based on the end date and the duration. The same goes for calculating the end date. If both the start and end date are given, the duration can be calculated between the two.

See the below table for a list of how Gantt handles the presence of start, end, and duration in different circumstances.

| Start | End | Duration | Result |
|---|---|---|---|
| Present | Present | Present | Check that duration is consistent with start/end times. Throws error if inconsistent. |
| Present | Present | Null | Computes duration from start and end times. |
| Present | Null | Present | Computes end time. |
| Present | Null | Null | Throws error as unable to compute duration or end time. |
| Null | Present | Present | Computes start time. |
| Null | Null | Present | Computes start time based on dependencies. In conjunction with `defaultStartDate`, enables chart to be drawn using only durations. |
| Null | Present | Null | Throws error as unable to calculate start time or duration. |
| Null | Null | Null | Throws error as unable to calculate start time, end time, or duration. |

With the above in mind, you can create a chart laying out a typical commute to work using only the duration of each task.



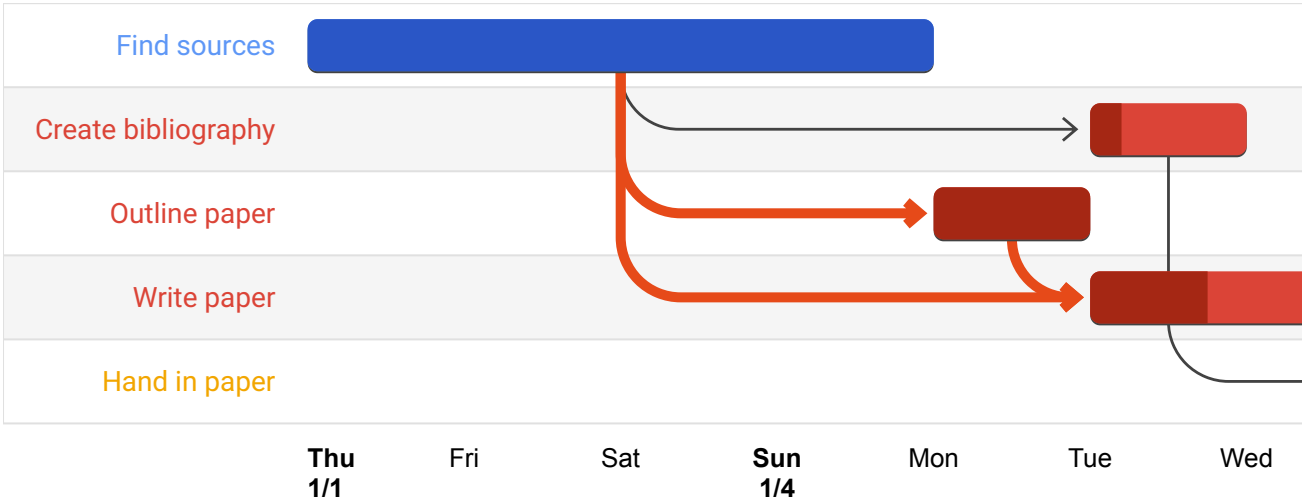

CODE IT YOURSELF ON JSFIDDLE

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

    function toMilliseconds(minutes) {
```

```
      return minutes * 60 * 1000;
    }

    function drawChart() {

      var otherData = new google.visualization.DataTable();
      otherData.addColumn('string', 'Task ID');
      otherData.addColumn('string', 'Task Name');
      otherData.addColumn('string', 'Resource');
      otherData.addColumn('date', 'Start');
      otherData.addColumn('date', 'End');
      otherData.addColumn('number', 'Duration');
      otherData.addColumn('number', 'Percent Complete');
      otherData.addColumn('string', 'Dependencies');

      otherData.addRows([
        ['toTrain', 'Walk to train stop', 'walk', null, null, toMilliseconds(5
        ['music', 'Listen to music', 'music', null, null, toMilliseconds(70),
        ['wait', 'Wait for train', 'wait', null, null, toMilliseconds(10), 100
        ['train', 'Train ride', 'train', null, null, toMilliseconds(45), 75,
        ['toWork', 'Walk to work', 'walk', null, null, toMilliseconds(10), 0,
        ['work', 'Sit down at desk', null, null, null, toMilliseconds(2), 0,

      ]);

      var options = {
        height: 275,
        gantt: {
          defaultStartDateMillis: new Date(2015, 3, 28)
        }
      };

      var chart = new google.visualization.Gantt(document.getElementById('cha

      chart.draw(otherData, options);
    }
  </script>
</head>
<body>
  <div id="chart_div"></div>
</body>
</html>
```

(https://developers.google.com/chart/interactive/docs/gallery/Critical)Critical path

The *critical path* in a Gantt chart is the path, or paths, that directly affect the finish date. The critical path in Google Gantt charts is colored red by default, and can be customized using the `criticalPathStyle` options. You can also turn off the critical path by setting `criticalPathEnabled` to `false`.

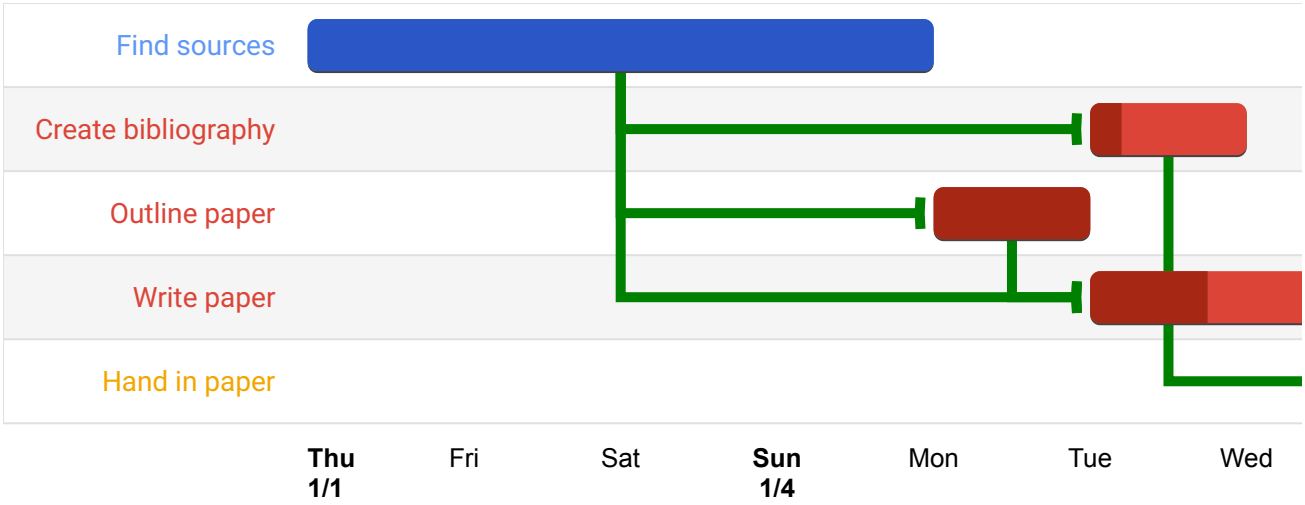



CODE IT YOURSELF ON JSFIDDLE

```
var options = {
  height: 275,
  gantt: {
    criticalPathEnabled: true,
    criticalPathStyle: {
      stroke: '#e64a19',
      strokeWidth: 5
    }
  }
};
```

## (https://developers.google.com/chart/interactive/docs/gallery/Arrows)Styling arrows

You can style the dependency arrows between tasks with the `gantt.arrow` options:

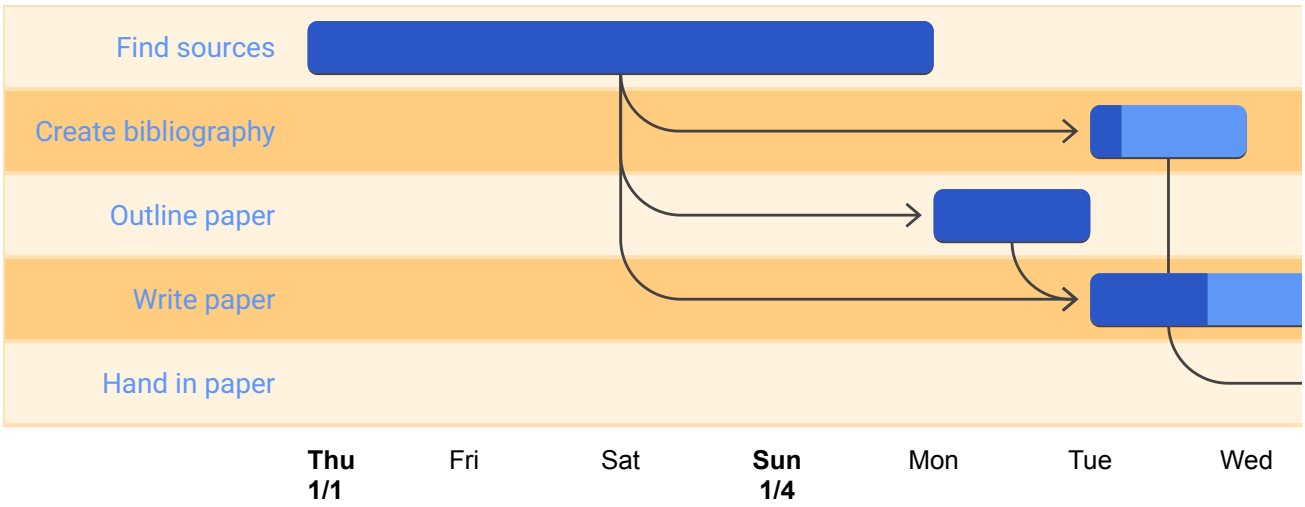



CODE IT YOURSELF ON JSFIDDLE

```
var options = {
  height: 275,
  gantt: {
    criticalPathEnabled: false, // Critical path arrows will be the sa
    arrow: {
      angle: 100,
      width: 5,
      color: 'green',
      radius: 0
    }
  }
};
```

## (https://developers.google.com/chart/interactive/docs/gallery/Tracks)Styling tracks

Grid styling is handled by a combination of `innerGridHorizLine`, `innerGridTrack`, and `innerGridDarkTrack`. By setting only the `innerGridTrack`, the chart will calculate a darker color for the `innerGridDarkTrack`, but by setting only the `innerGridDarkTrack`, the `innerGridTrack` will use its default color and will not calculate a lighter color.




CODE IT YOURSELF ON JSFIDDLE

```
var options = {
  height: 275,
  gantt: {
    criticalPathEnabled: false,
    innerGridHorizLine: {
      stroke: '#ffe0b2',
      strokeWidth: 2
    },
    innerGridTrack: {fill: '#fff3e0'},
    innerGridDarkTrack: {fill: '#ffcc80'}
  }
};
```

## Loading

The `google.charts.load` package name is `"gantt"`.

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

The visualization's class name is `google.visualization.Gantt`.

```
var chart = new google.visualization.Gantt(container);
```

## Data format

**Rows:** Each row in the table represents a task.

**Columns:**

| | Column 0 | Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|---|---|
| **Purpose:** | Task ID | Task Name | Resource ID (optional) | Start | End | Duration (in milliseconds) | Percent Complete | Dependencies |
| **Data Type:** | string | string | string | date | date | number | number | string |
| **Role:** | domain | data | data | data | data | data | data | data |

## Configuration options

| Name | Type | Default | Description |
|---|---|---|---|
| backgroundColor.fill | string | 'white' | The chart fill color, as an HTML color string. |
| gantt.arrow | object | null | For Gantt Charts (https://developers.google.com/chart/inter , `gantt.arrow` controls the various propert tasks. |
| gantt.arrow.angle | number | 45 | The angle of the head of the arrow. |
| gantt.arrow.color | string | '#000' | The color of the arrows. |
| gantt.arrow.length | number | 8 | The length of the head of the arrow. |
| gantt.arrow.radius | number | 15 | The radius for defining the curve of the arro |
| gantt.arrow.spaceAfter | number | 4 | The amount of whitespace between the hea which it points. |
| gantt.arrow.width | number | 1.4 | The width of the arrows. |
| gantt.barCornerRadius | number | 2 | The radius for defining the curve of a bar's c |
| gantt.barHeight | number | null | The height of the bars for tasks. |
| gantt.criticalPathEnabled | boolean | true | If `true` any arrows on the critical path will b |

| | | | |
|---|---|---|---|
| gantt.criticalPathStyle | object | null | An object containing the style for any critica |
| gantt.criticalPathStyle.stroke | string | null | The color of any critical path arrows. |
| gantt.criticalPathStyle.strokeWidth | number | 1.4 | The thickness of any critical path arrows. |
| gantt.defaultStartDate | date/number | null | If the start date cannot be computed from th start date will be set to this. Accepts a **`date`** **`D`**)) or a number, which is the number of mill |
| gantt.innerGridHorizLine | object | null | Defines the style of the inner horizontal grid |
| gantt.innerGridHorizLine.stroke | string | null | The color of the inner horizontal grid lines. |
| gantt.innerGridHorizLine.strokeWidth | number | 1 | The width of the inner horizontal grid lines. |
| gantt.innerGridTrack.fill | string | null | The fill color of the inner grid track. If no **`inn`** specified, this will be applied to every grid tra |
| gantt.innerGridDarkTrack.fill | string | null | The fill color of the alternate, dark inner grid |
| gantt.labelMaxWidth | number | 300 | The maximum amount of space allowed for |
| gantt.labelStyle | object | null | An object containing the styles for task labe<br>`labelStyle: {`<br>`  fontName: Roboto2,`<br>`  fontSize: 14,`<br>`  color: '#757575'`<br>`},` |
| gantt.percentEnabled | boolean | true | Fills the task bar based on the percentage c |
| gantt.percentStyle.fill | string | null | The color of the percentage completed porti |
| gantt.shadowEnabled | boolean | true | If set to **`true`**, draws a shadow under each t dependencies. |
| gantt.shadowColor | string | '#000' | Defines the color of the shadows under any dependencies. |
| gantt.shadowOffset | number | 1 | Defines the offset, in pixels, of the shadows dependencies. |
| gantt.trackHeight | number | null | The height of the tracks. |
| width | number | *width of the containing element* | Width of the chart, in pixels. |

| | | | |
|---|---|---|---|
| height | number | *height of the containing element* | height of the chart, in pixels. |

## Methods

| Method | Description |
|---|---|
| `draw(data, options)` | Draws the chart. The chart accepts further method calls only after the `ready` (#Events)event is fired. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#visdraw).<br><br>**Return Type:** none |
| `getSelection()` | Returns an array of the selected chart entities. Selectable entities are bars, legend entries and categories. For this chart, only one entity can be selected at any given moment. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#visgetselection).<br><br>**Return Type:** Array of selection elements |
| `setSelection()` | Selects the specified chart entities. Cancels any previous selection. Selectable entities are bars, legend entries and categories. For this chart, only one entity can be selected at a time. `Extended description` (https://developers.google.com/chart/interactive/docs/reference#vissetselection).<br><br>**Return Type:** none |
| `clearChart()` | Clears the chart, and releases all of its allocated resources.<br><br>**Return Type:** none |

## Events

| Event | Description |
|---|---|
| `click` | Fired when the user clicks inside the chart. Can be used to identify when the title, data elements, legend entries, axes, gridlines, or labels are clicked.<br><br>**Properties:** targetID |
| `error` | Fired when an error occurs when attempting to render the chart. |

| | |
|---|---|
| | **Properties:** id, message |
| `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.<br><br>**Properties:** none |
| `select` | Fired when the user clicks a visual entity. To learn what has been selected, call `getSelection()` (#Methods).<br><br>**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*