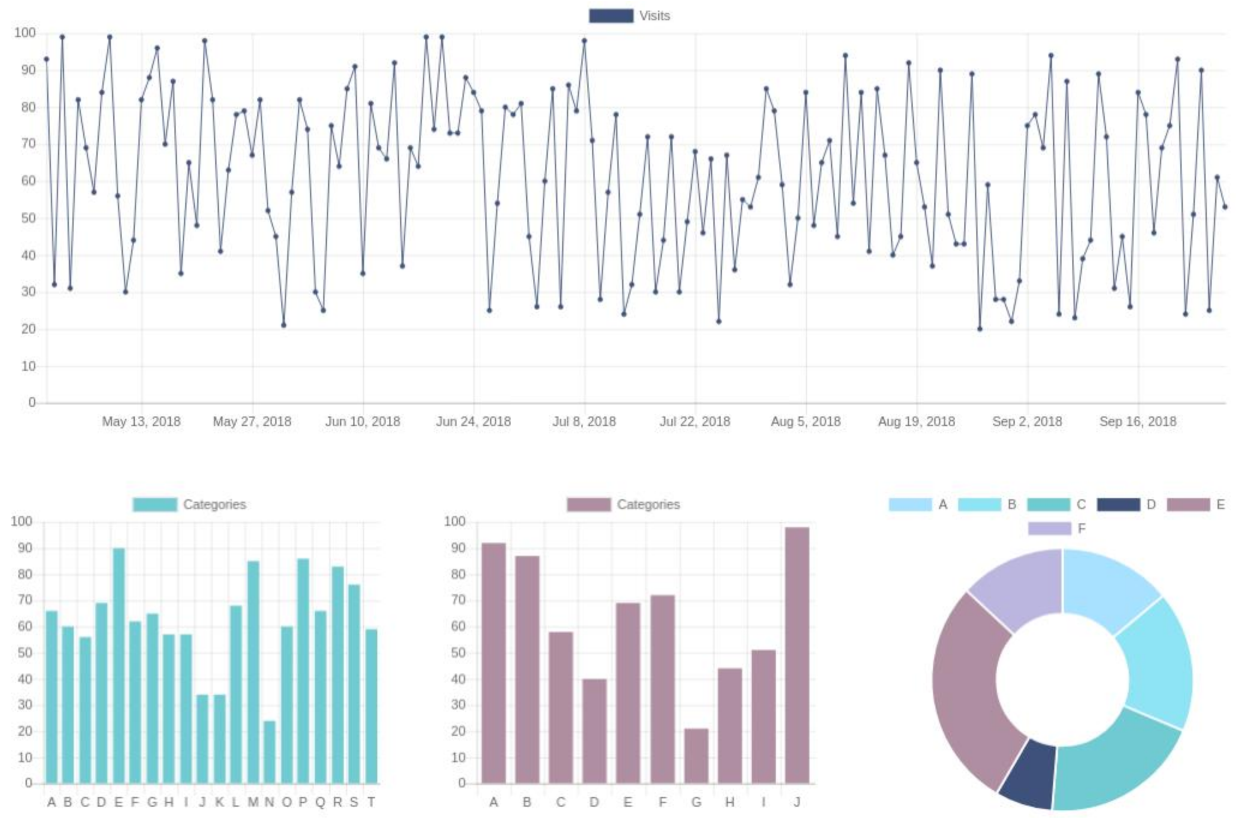


JavaScript Charting: A biblioteca Chart.js

Simple yet flexible JavaScript
charting for
designers & developers

Chart.js

Mário Pinto | ESMAD | Politécnico do Porto

1. Chart.js | Agenda

1. Getting Started
2. Chart Type
3. Chart Data
4. Chart Options
5. A Dashboard Example
6. 2-Axis Charts
7. Documentation
8. Chart.js API

Visits
100
90
80
70
60
50
40
30
20
10
0
May 13, 2018
May 27, 2018
Jun 10, 2018
Jun 24, 2018
Jul 8, 2018
Jul 22, 2018
Aug 5, 2018
Aug 19, 2018
Sep 2, 2018
Sep 16, 2018
Categories
A B C D E F G H I J K L M N O P Q R S T
Categories
A B C D E F G H I J
A
B
C
D
E
F

1. Chart.js | Getting Started

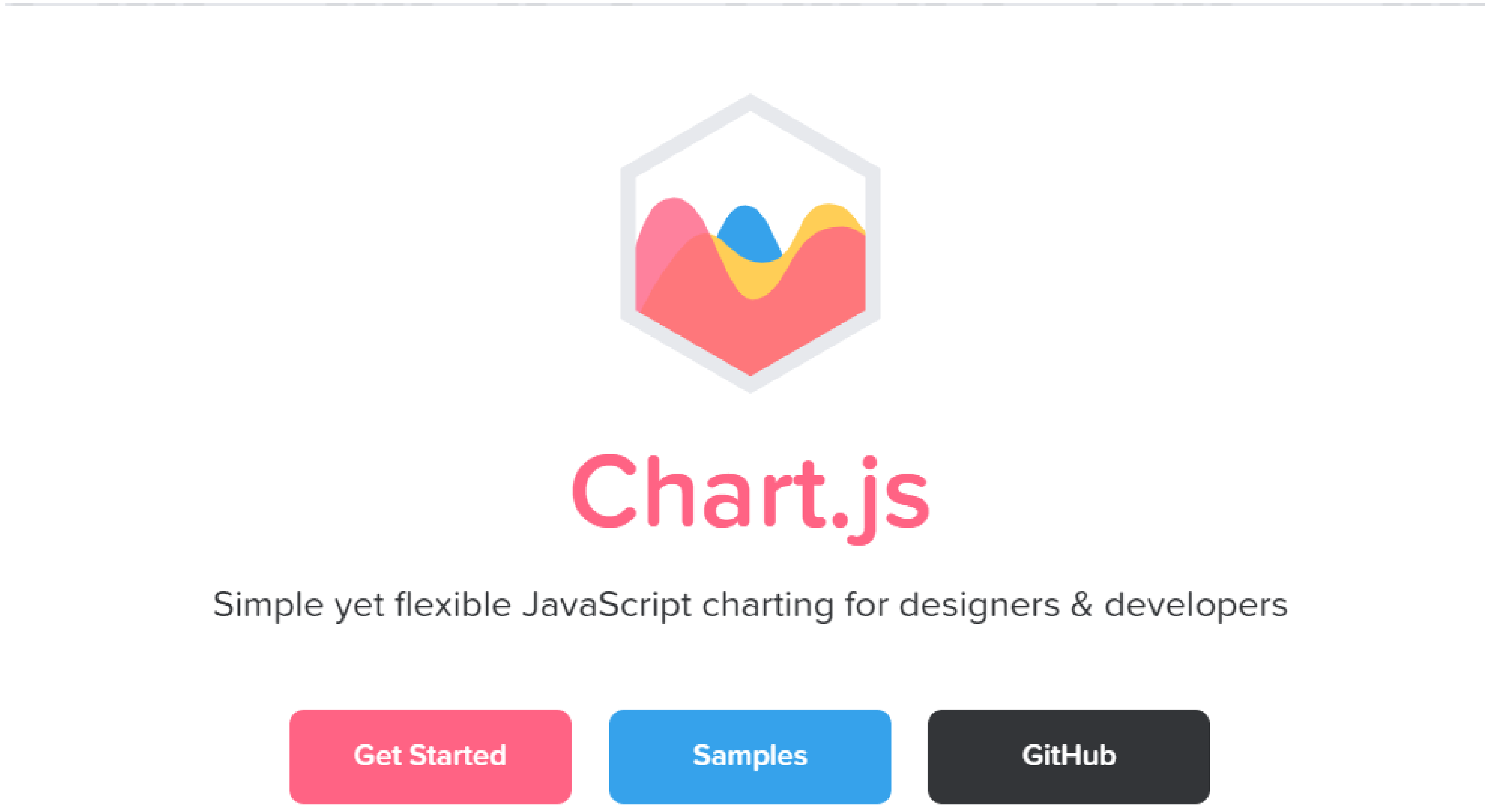
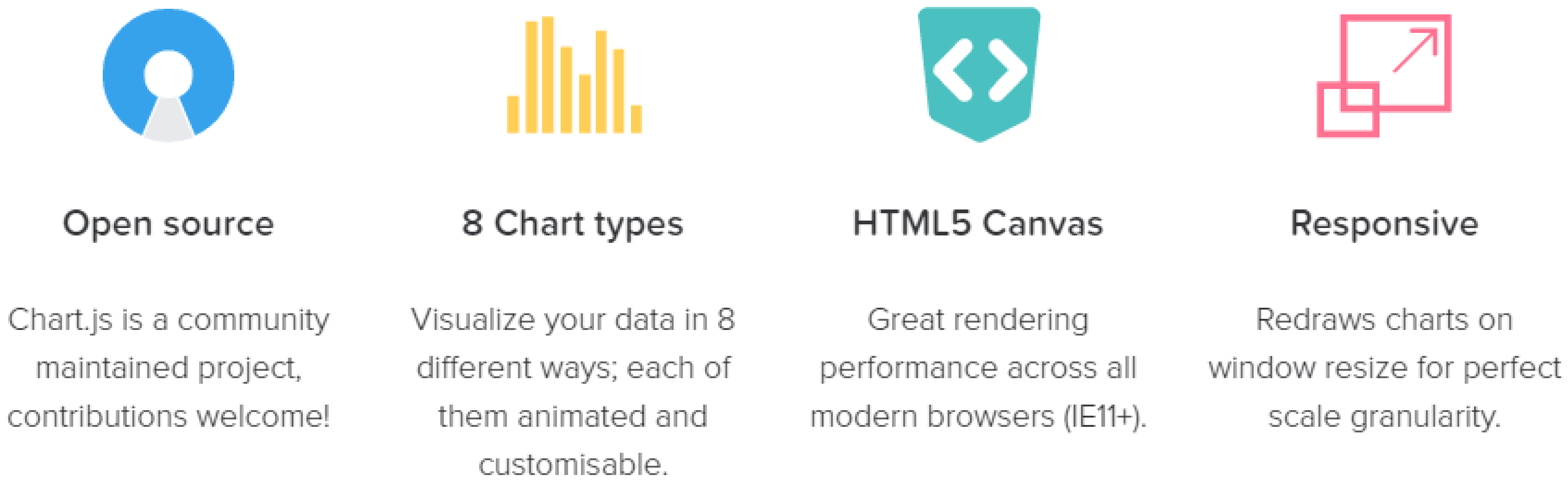


Chart.js

Simple yet flexible JavaScript charting for designers & developers

1. Chart.js | Getting Started



Open source

Chart.js is a community maintained project, contributions welcome!

8 Chart types

Visualize your data in 8 different ways; each of them animated and customisable.

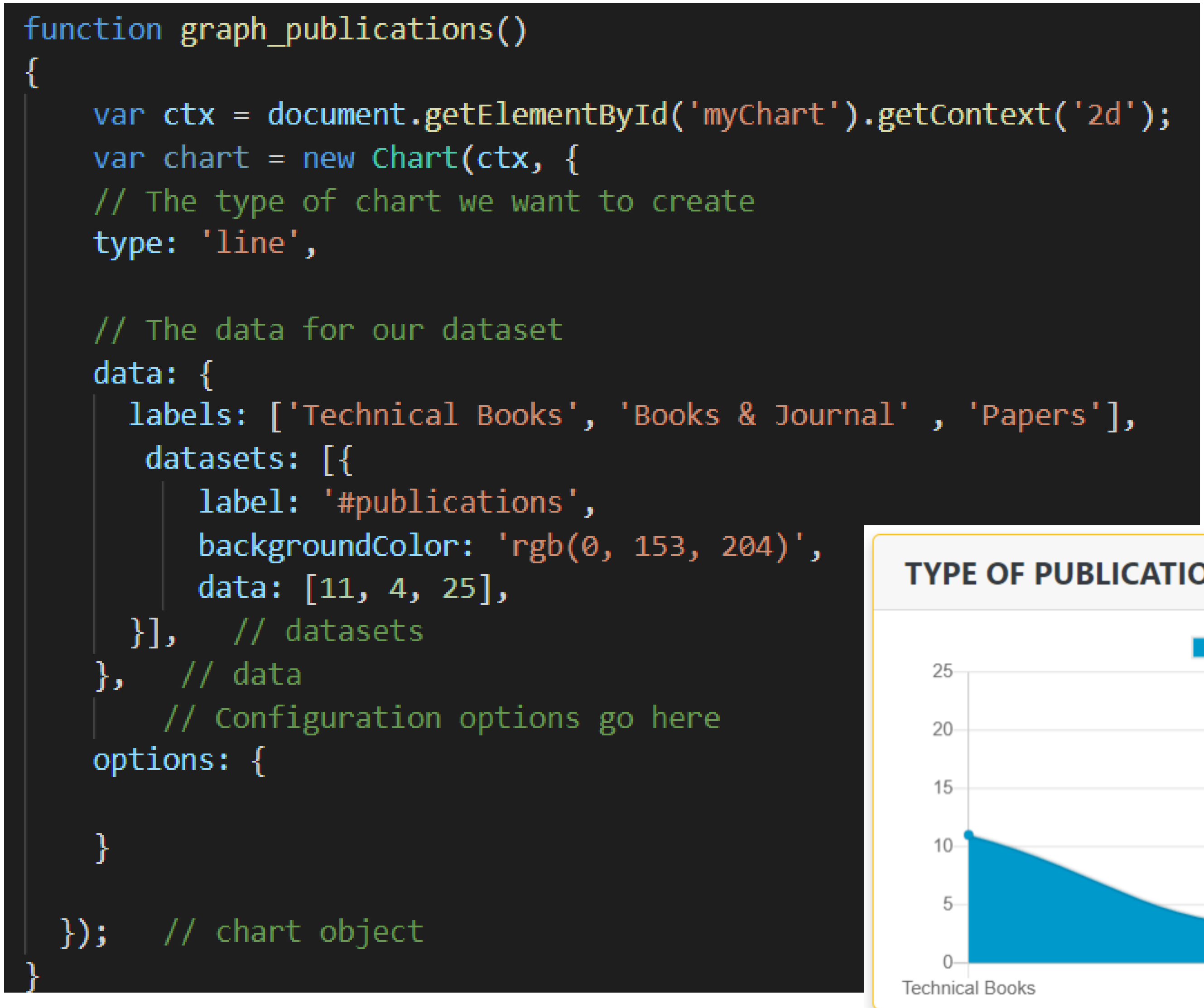
HTML5 Canvas

Great rendering performance across all modern browsers (IE11+).

Responsive

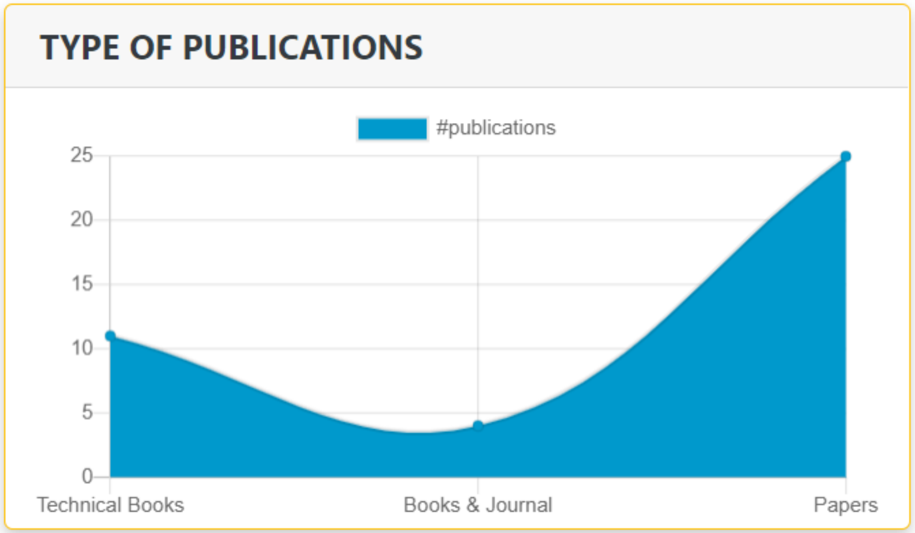
Redraws charts on window resize for perfect scale granularity.

1. Chart.js | Getting Started



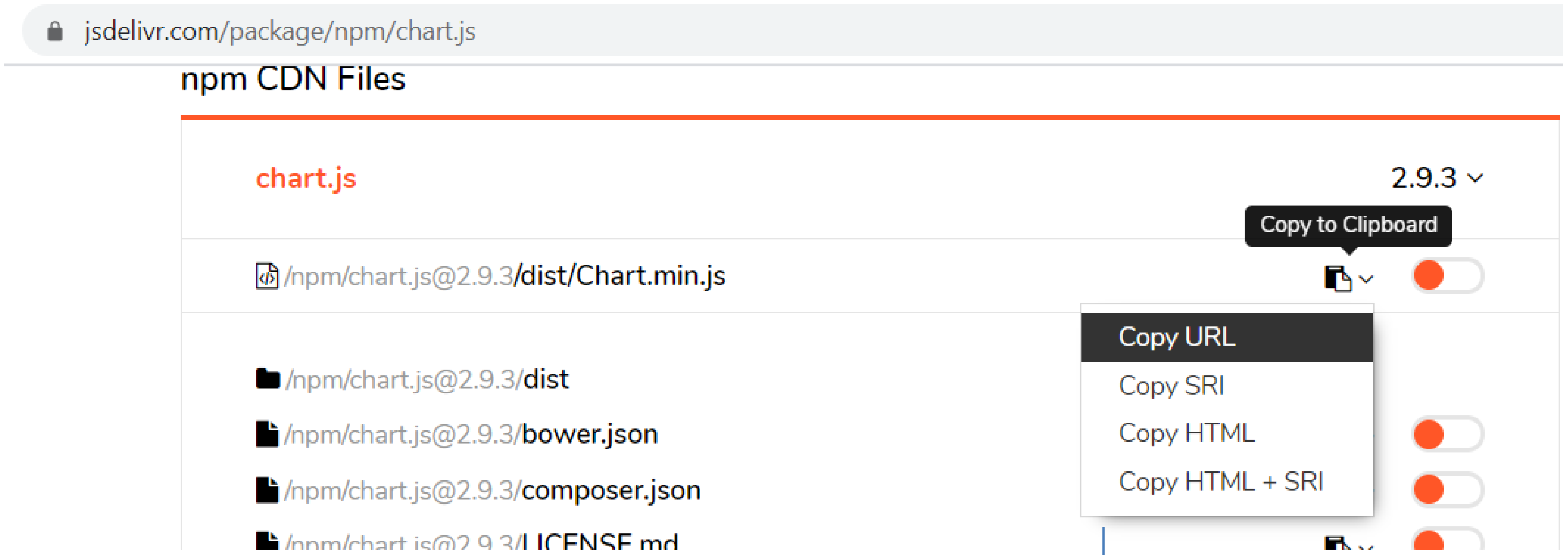
```
function graph_publications()
{
    var ctx = document.getElementById('myChart').getContext('2d');
    var chart = new Chart(ctx, {
    // The type of chart we want to create
    type: 'line',

    // The data for our dataset
    data: {
      labels: ['Technical Books', 'Books & Journal' , 'Papers'],
       datasets: [{
          label: '#publications',
          backgroundColor: 'rgb(0, 153, 204)',
          data: [11, 4, 25],
      }],   // datasets
    },   // data
        // Configuration options go here
    options: {

    }

  });   // chart object
}
```

- ❑ One easy example …



1. Chart.js | Getting Started

- ❑ You can download the latest version of Chart.js from the GitHub releases or ...
- ❑ Use a Chart.js CDN (content delivery network)



jsdelivr.com/package/npm/chart.js

npm CDN Files

chart.js 2.9.3

/npm/chart.js@2.9.3/dist/Chart.min.js

/npm/chart.js@2.9.3/dist

/npm/chart.js@2.9.3/bower.json

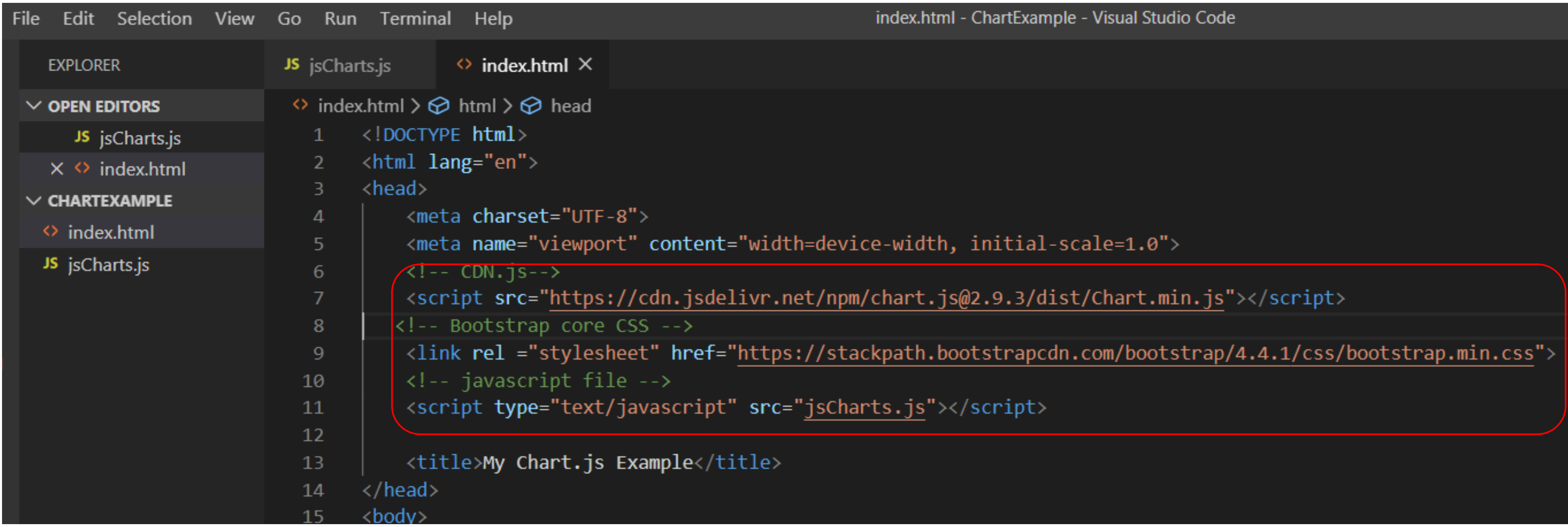
/npm/chart.js@2.9.3/composer.json

/npm/chart.js@2.9.3/LICENSE.md

Copy URL of Chart.min.js cdn...

1. Chart.js | Getting Started

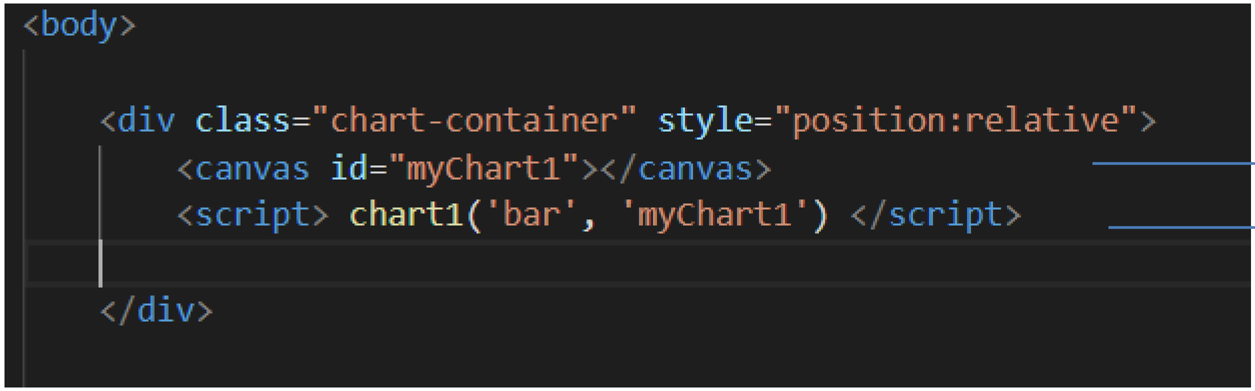
- ❑ Include in your HTML page:
  - ❑ Chart.js cdn
  - ❑ Bootstrap CSS (easy way to style charts)
  - ❑ JavaScript file source



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- CDN.js-->
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>
  <!-- Bootstrap core CSS -->
    <link rel ="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
    <!-- javascript file -->
    <script type="text/javascript" src="jsCharts.js"></script>

    <title>My Chart.js Example</title>
</head>
<body>
```

1. Chart.js | Getting Started

- ❑ Chart.js charts are rendered on user provided canvas elements
- ❑ All that's required is the script included in your page along with a single <canvas> element, to render the chart



```
<body>

    <div class="chart-container" style="position:relative">
        <canvas id="myChart1"></canvas>
        <script> chart1('bar', 'myChart1') </script>

    </div>
```

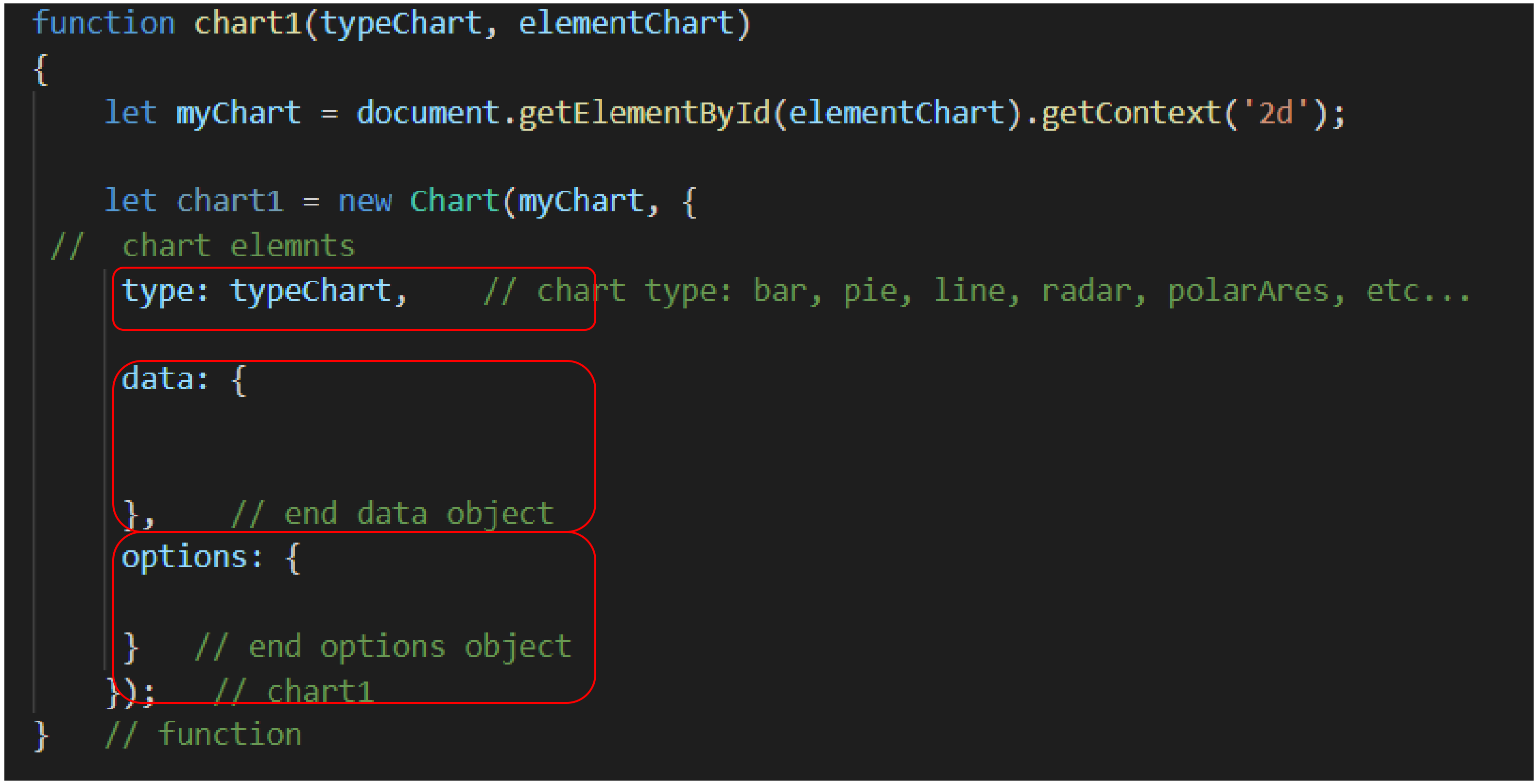
Canvas element

JavaScript function

1. Chart.js | Getting Started

- The 3 main properties in a chart object: **type**, **data**, **options**

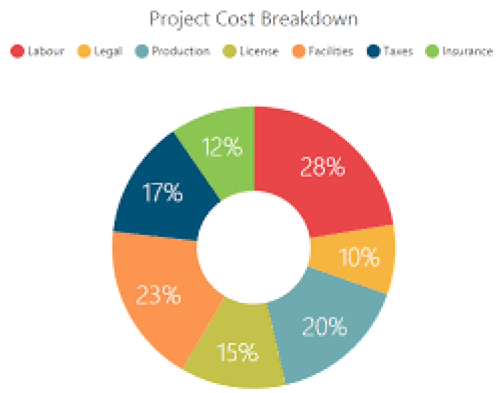
- type : line, bar, radar, doughnut, pie, polarArea, bubble, scatter
- data: labels, datasets, …
- options: title, legend, scales, …

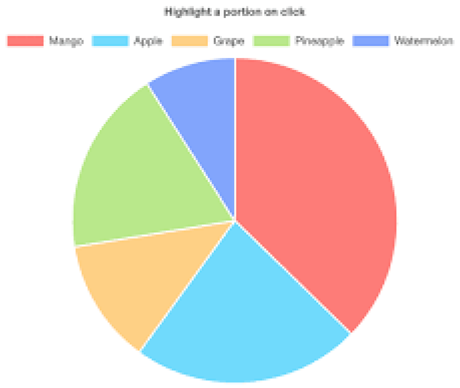


```
function chart1(typeChart, elementChart)
{
    let myChart = document.getElementById(elementChart).getContext('2d');

    let chart1 = new Chart(myChart, {
 //  chart elemnts
     type: typeChart,    // chart type: bar, pie, line, radar, polarAres, etc...

     data: {


     },    // end data object
     options: {

     }   // end options object
    });   // chart1
}   // function
```

2. Chart.js | Chart Type

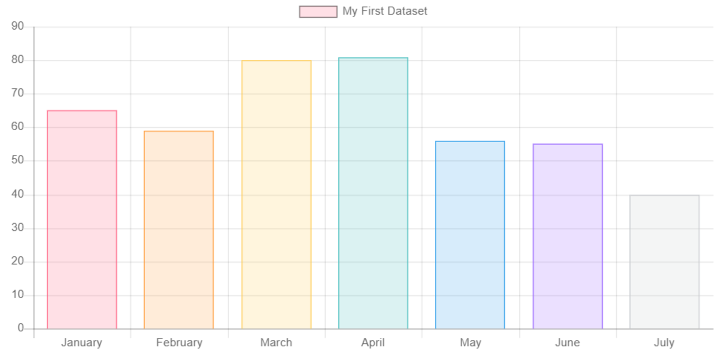
❑ Chart Type





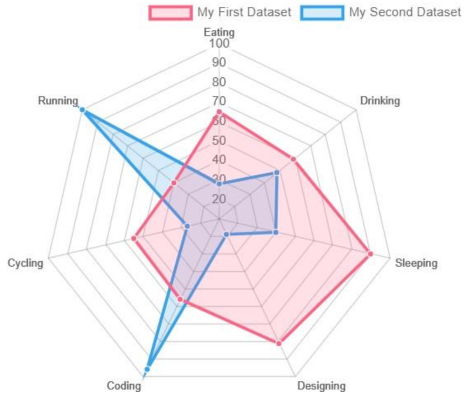
Bar

A bar chart provides a way of showing data values represented as vertical bars. It is sometimes used to show trend data, and the comparison of multiple data sets side by side.



A radar chart is a way of showing multiple data points and the variation between the

They are often useful for comparing the points of two or more different data sets.

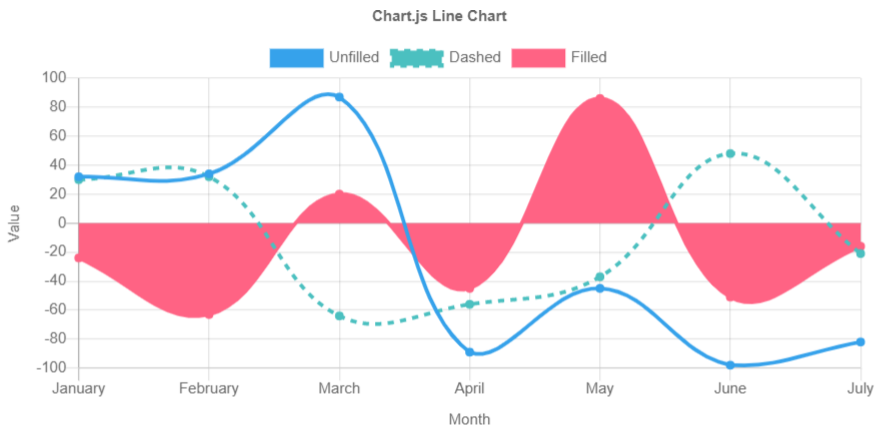


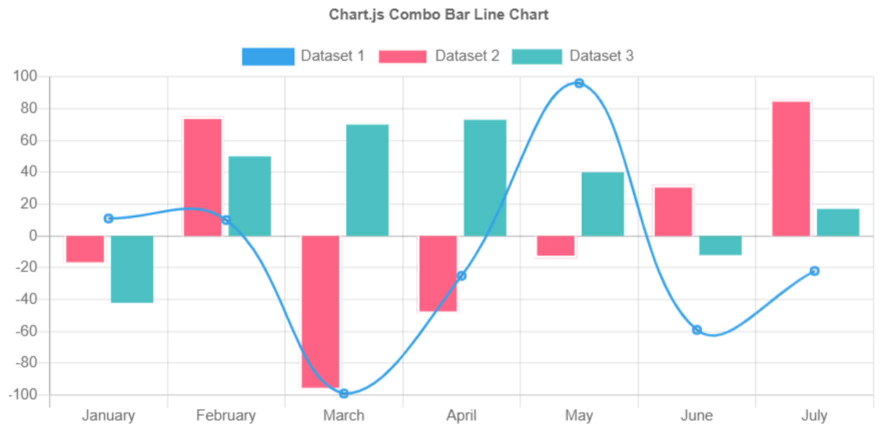
Charts

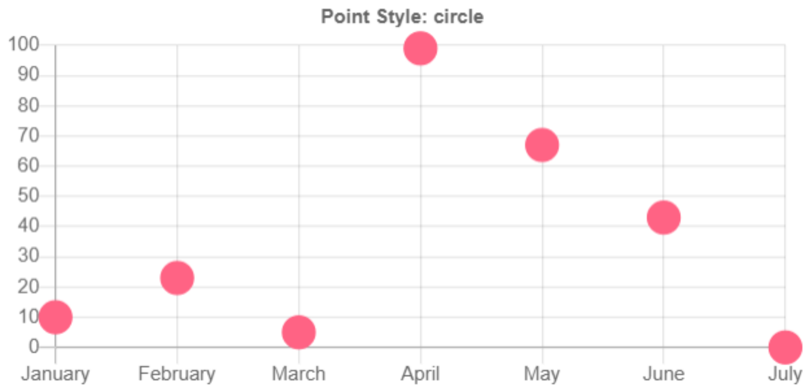
- Line
- Bar
- Radar
- Doughnut & Pie
- Polar Area
- Bubble
- Scatter
- Area
- Mixed

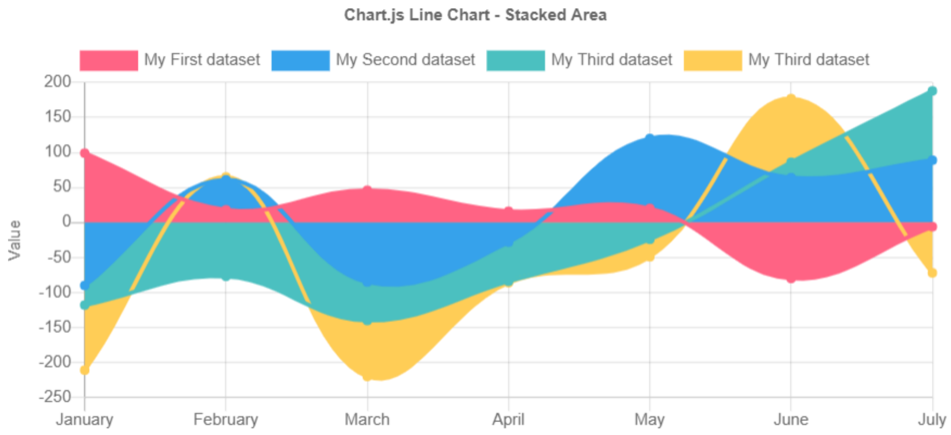
2. Chart.js | Chart Type

- ❑ **Chart Type: some examples…**



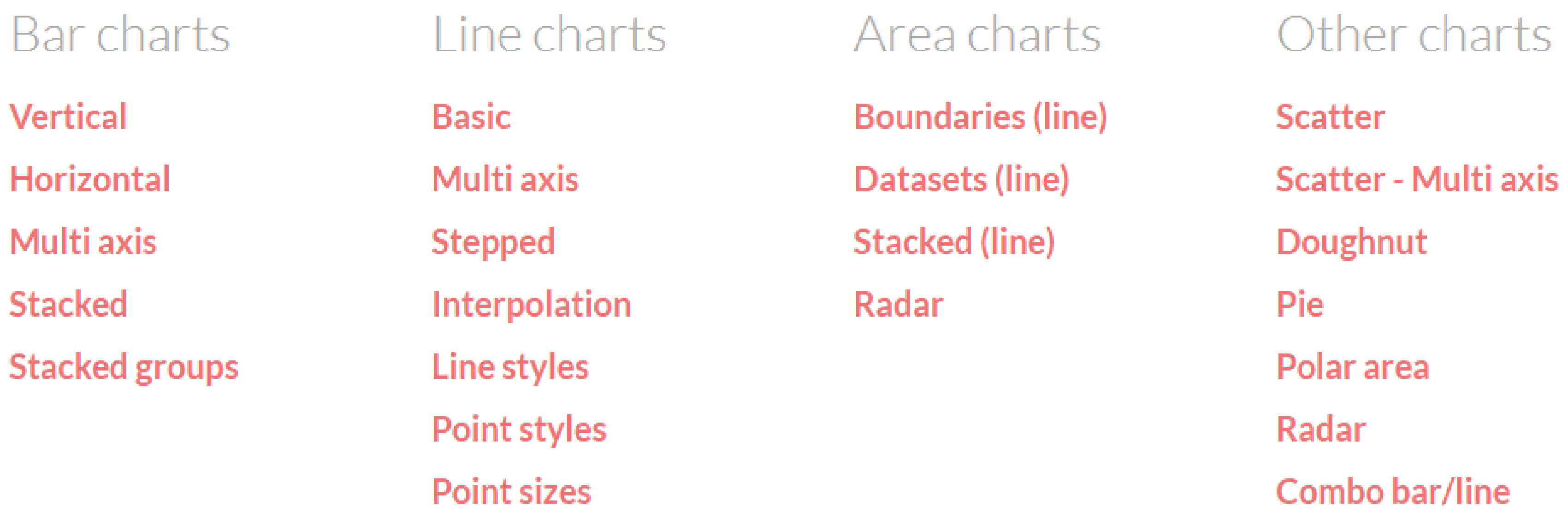






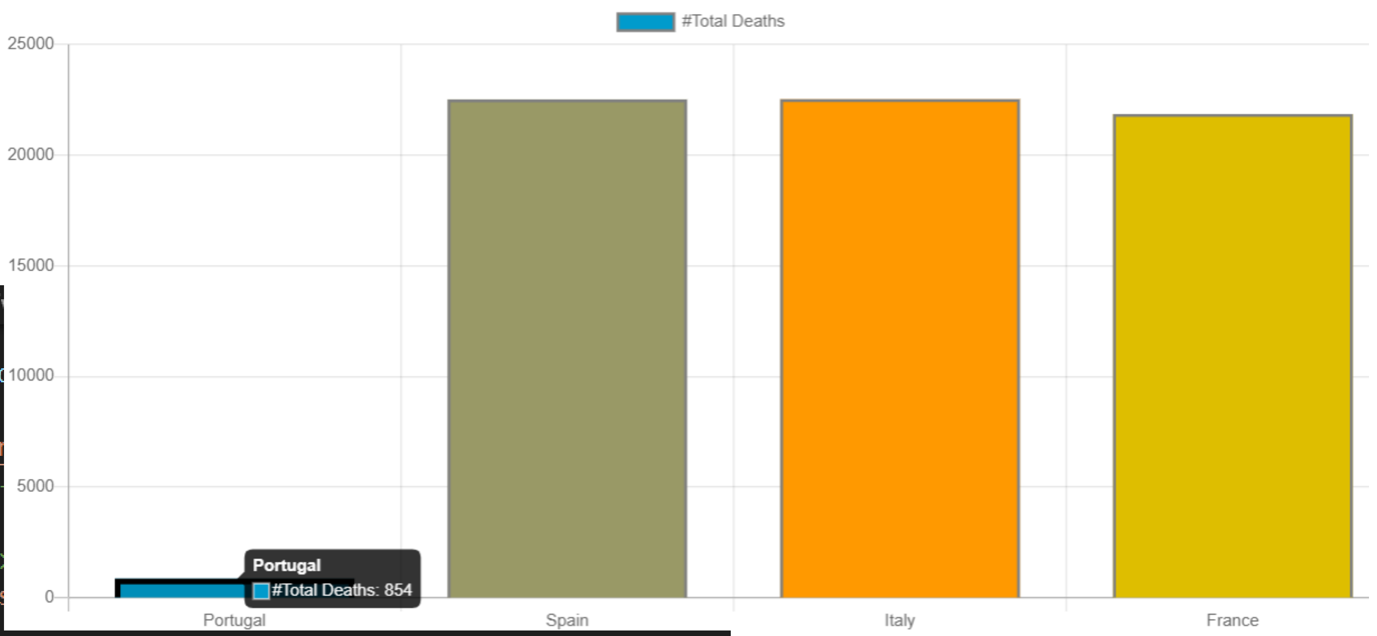
2. Chart.js | Chart Type

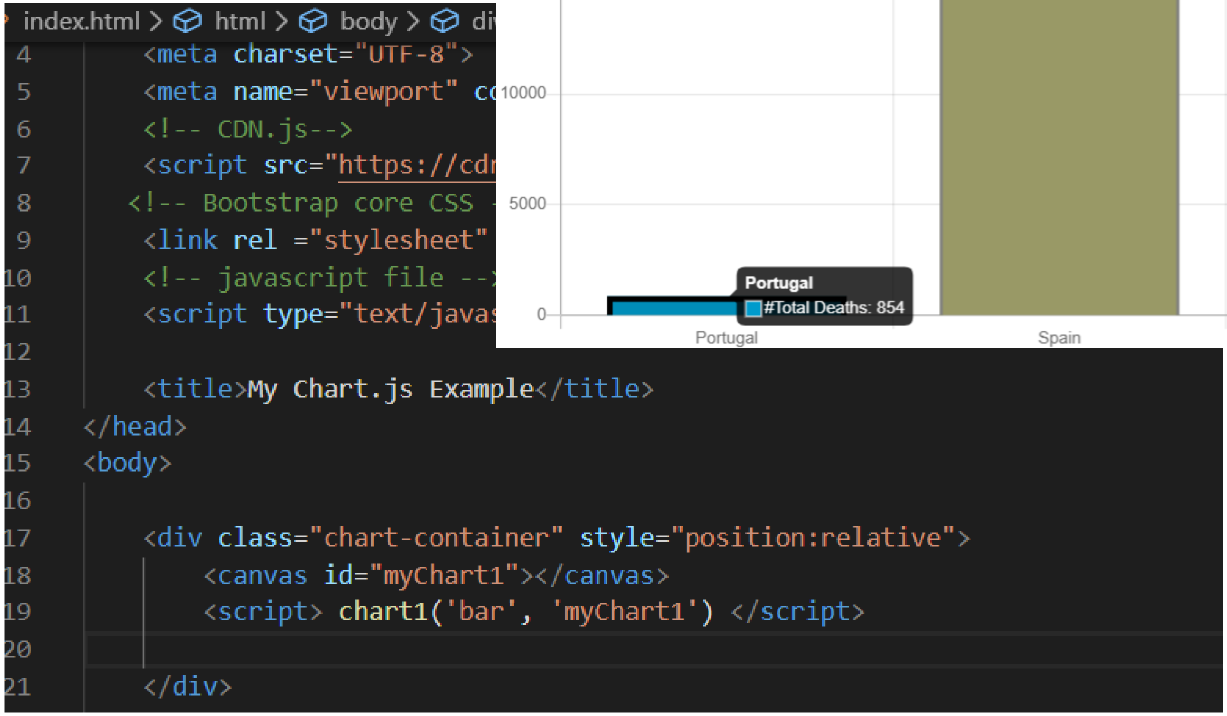
❑ **Chart Type**



Bar charts	Line charts	Area charts	Other charts
Vertical	Basic	Boundaries (line)	Scatter
Horizontal	Multi axis	Datasets (line)	Scatter - Multi axis
Multi axis	Stepped	Stacked (line)	Doughnut
Stacked	Interpolation	Radar	Pie
Stacked groups	Line styles		Polar area
	Point styles		Radar
	Point sizes		Combo bar/line

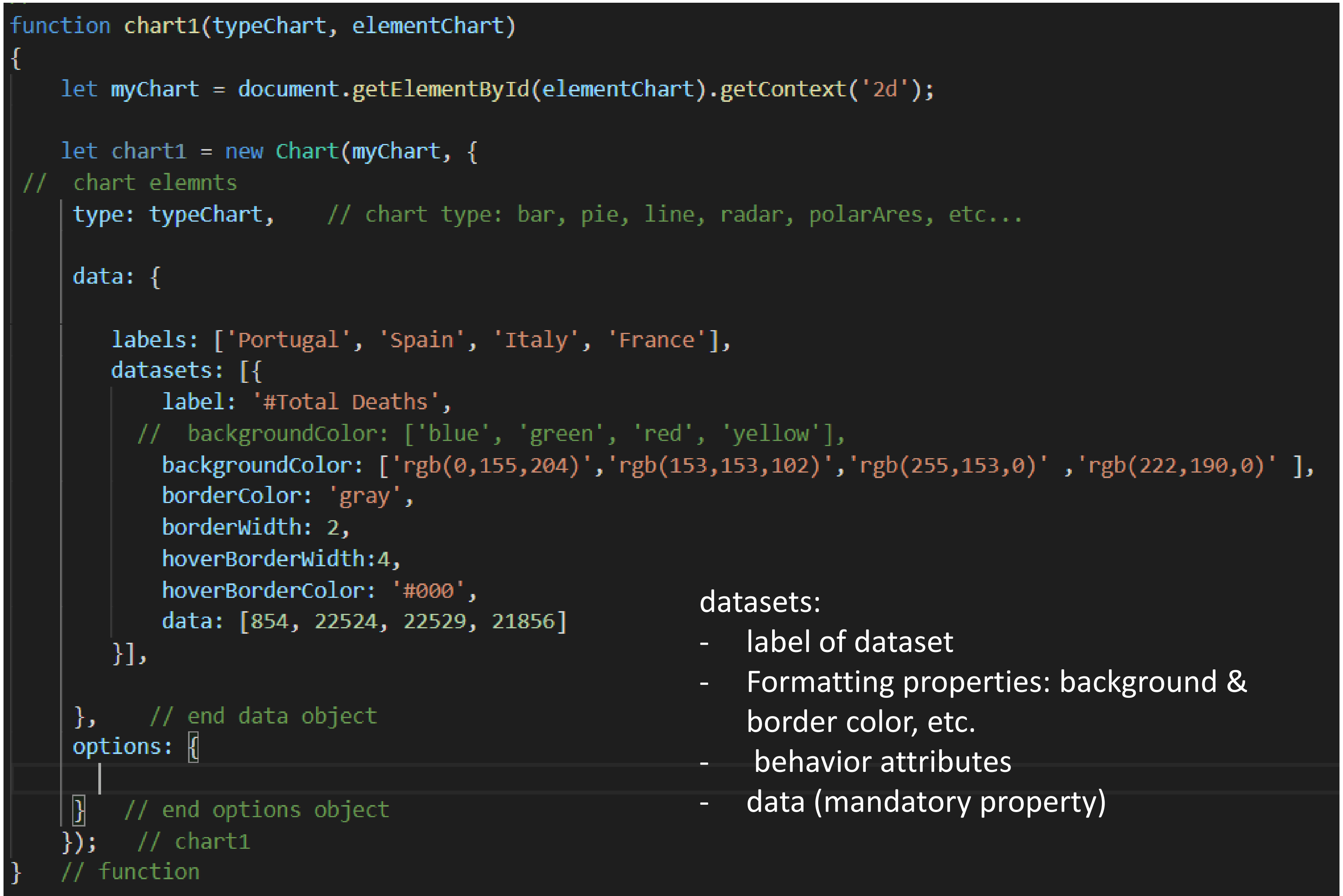
3. Chart.js | Data





```
    <meta charset="UTF-8">
    <meta name="viewport" c
    <!-- CDN.js-->
    <script src="https://cdr
   <!-- Bootstrap core CSS -
    <link rel ="stylesheet"
    <!-- javascript file --
    <script type="text/javas

    <title>My Chart.js Example</title>
</head>
<body>

    <div class="chart-container" style="position:relative">
        <canvas id="myChart1"></canvas>
        <script> chart1('bar', 'myChart1') </script>

    </div>
```

3. Chart.js | Data



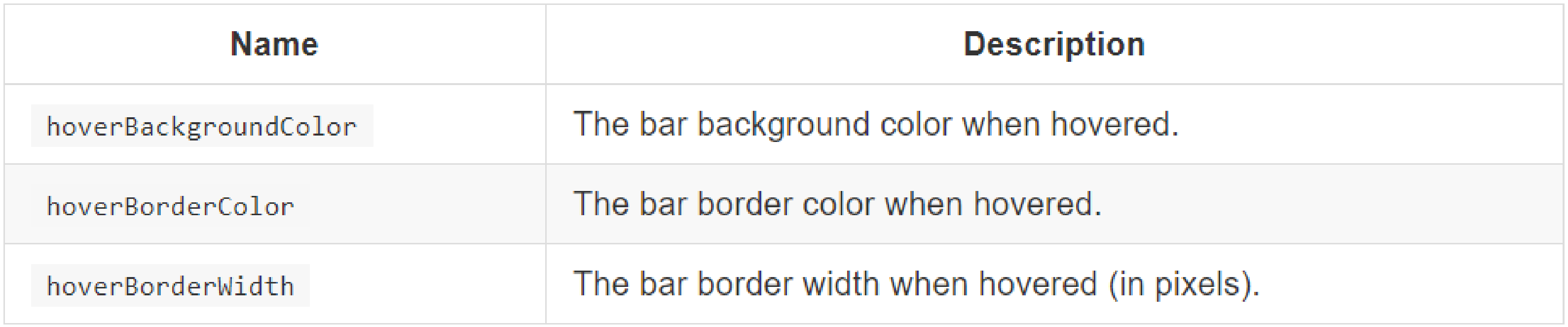
```
function chart1(typeChart, elementChart)
{
    let myChart = document.getElementById(elementChart).getContext('2d');

    let chart1 = new Chart(myChart, {
 //  chart elemnts
     type: typeChart,    // chart type: bar, pie, line, radar, polarAres, etc...

     data: {

        labels: ['Portugal', 'Spain', 'Italy', 'France'],
        datasets: [{
            label: '#Total Deaths',
          //  backgroundColor: ['blue', 'green', 'red', 'yellow'],
            backgroundColor: ['rgb(0,155,204)','rgb(153,153,102)','rgb(255,153,0)' ,'rgb(222,190,0)' ],
            borderColor: 'gray',
            borderWidth: 2,
            hoverBorderWidth:4,
            hoverBorderColor: '#000',
            data: [854, 22524, 22529, 21856]
        }],

     },    // end data object
     options: {

     }   // end options object
    });   // chart1
}   // function
```

datasets:
- label of dataset
- Formatting properties: background & border color, etc.
- behavior attributes
- data (mandatory property)

3. Chart.js | Data

Some dataset properties:

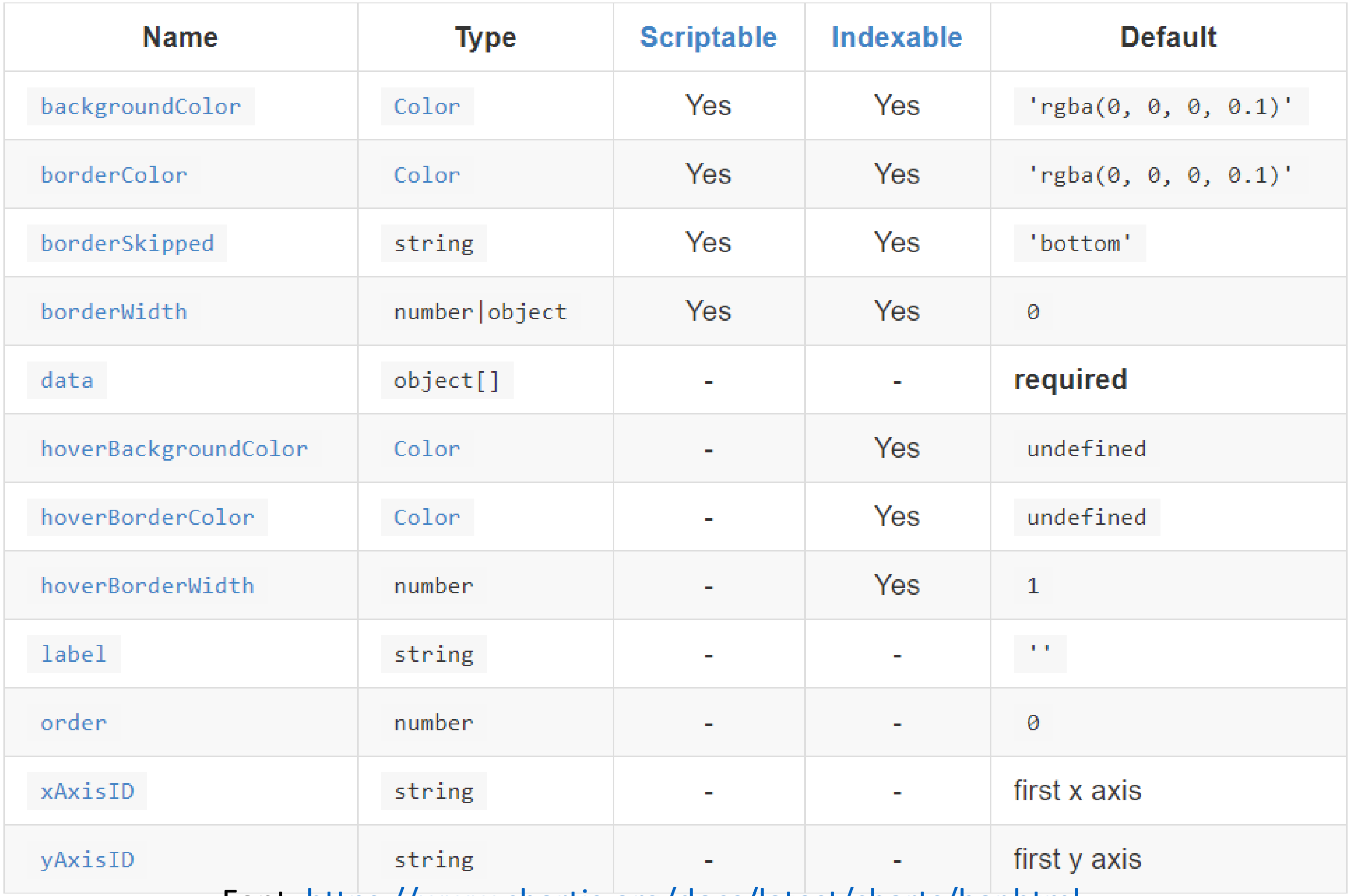
- backgroundColor & borderColor:
  - You can specify the color as a string in **hexadecimal**, **RGB**, or **HSL** notations.
  - If a color is needed, but not specified, Chart.js will use the global default color. This color is stored at Chart.defaults.global.defaultColor. It is initially set to 'rgb(0, 0, 0,)'.
- Data: an object with input data for the chart
- Interactions:



Name	Description
`hoverBackgroundColor`	The bar background color when hovered.
`hoverBorderColor`	The bar border color when hovered.
`hoverBorderWidth`	The bar border width when hovered (in pixels).

3. Chart.js | Data

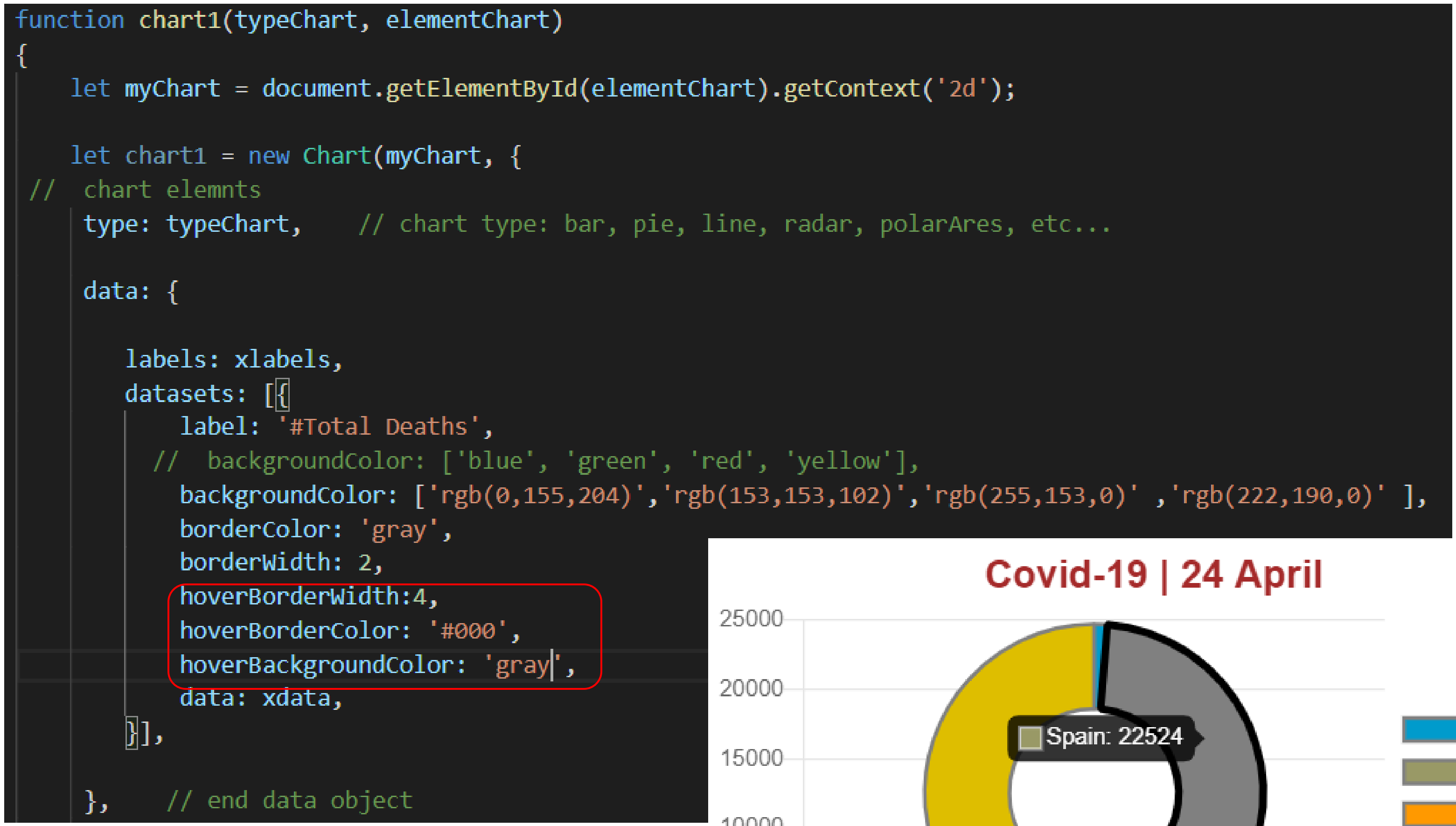
- Each chart type has a set of dataset properties, such as:



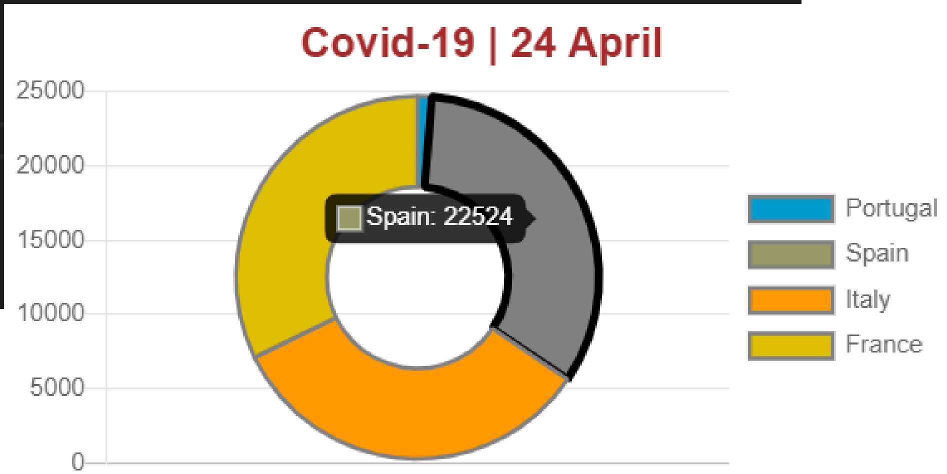
Name	Type	Scriptable	Indexable	Default
backgroundColor	Color	Yes	Yes	'rgba(0, 0, 0, 0.1)'
borderColor	Color	Yes	Yes	'rgba(0, 0, 0, 0.1)'
borderSkipped	string	Yes	Yes	'bottom'
borderWidth	number|object	Yes	Yes	0
data	object[]	-	-	**required**
hoverBackgroundColor	Color	-	Yes	undefined
hoverBorderColor	Color	-	Yes	undefined
hoverBorderWidth	number	-	Yes	1
label	string	-	-	''
order	number	-	-	0
xAxisID	string	-	-	first x axis
yAxisID	string	-	-	first y axis

Font: https://www.chartjs.org/docs/latest/charts/bar.html

3. Chart.js | Data



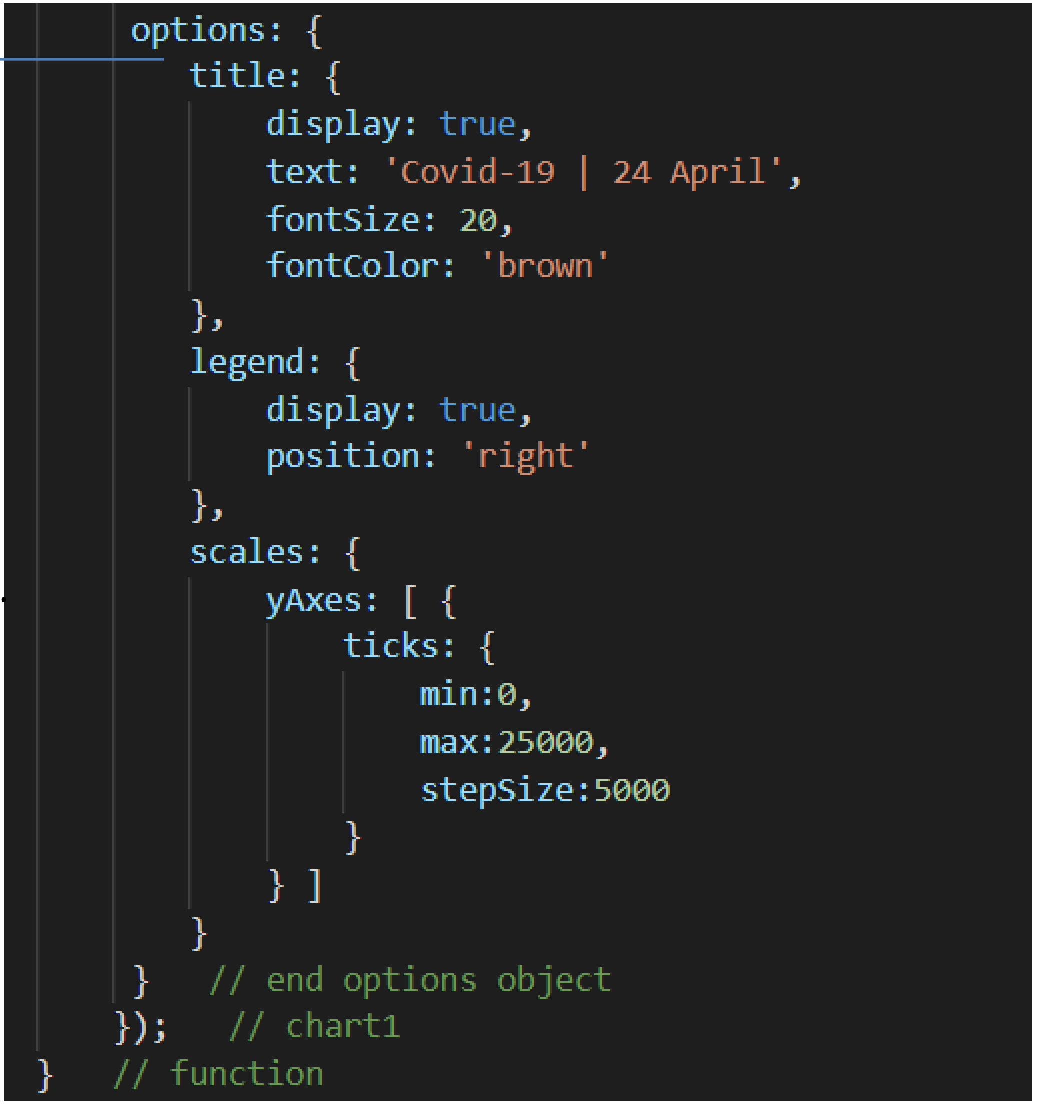
```
function chart1(typeChart, elementChart)
{
    let myChart = document.getElementById(elementChart).getContext('2d');

    let chart1 = new Chart(myChart, {
 //  chart elemnts
     type: typeChart,    // chart type: bar, pie, line, radar, polarAres, etc...

     data: {

        labels: xlabels,
        datasets: [{
            label: '#Total Deaths',
          //  backgroundColor: ['blue', 'green', 'red', 'yellow'],
            backgroundColor: ['rgb(0,155,204)','rgb(153,153,102)','rgb(255,153,0)' ,'rgb(222,190,0)' ],
            borderColor: 'gray',
            borderWidth: 2,
            hoverBorderWidth:4,
            hoverBorderColor: '#000',
            hoverBackgroundColor: 'gray',
            data: xdata,
        }],

     },    // end data object
```



4. Chart.js | Options

Options:
- title
- legend
- scales
- animations

Other properties (title):
- *position*: ‘top’, ‘bottom’, ‘left’, ‘right’
- *fontFamily*: 'Helvetica Neue’,
  'Helvetica', 'Arial', ‘sans-serif’,…
- *fontStyle*: ‘bold’

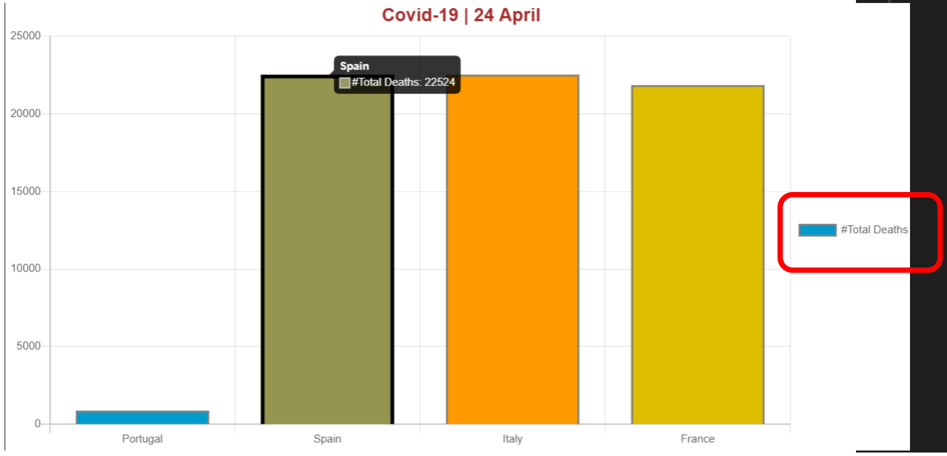


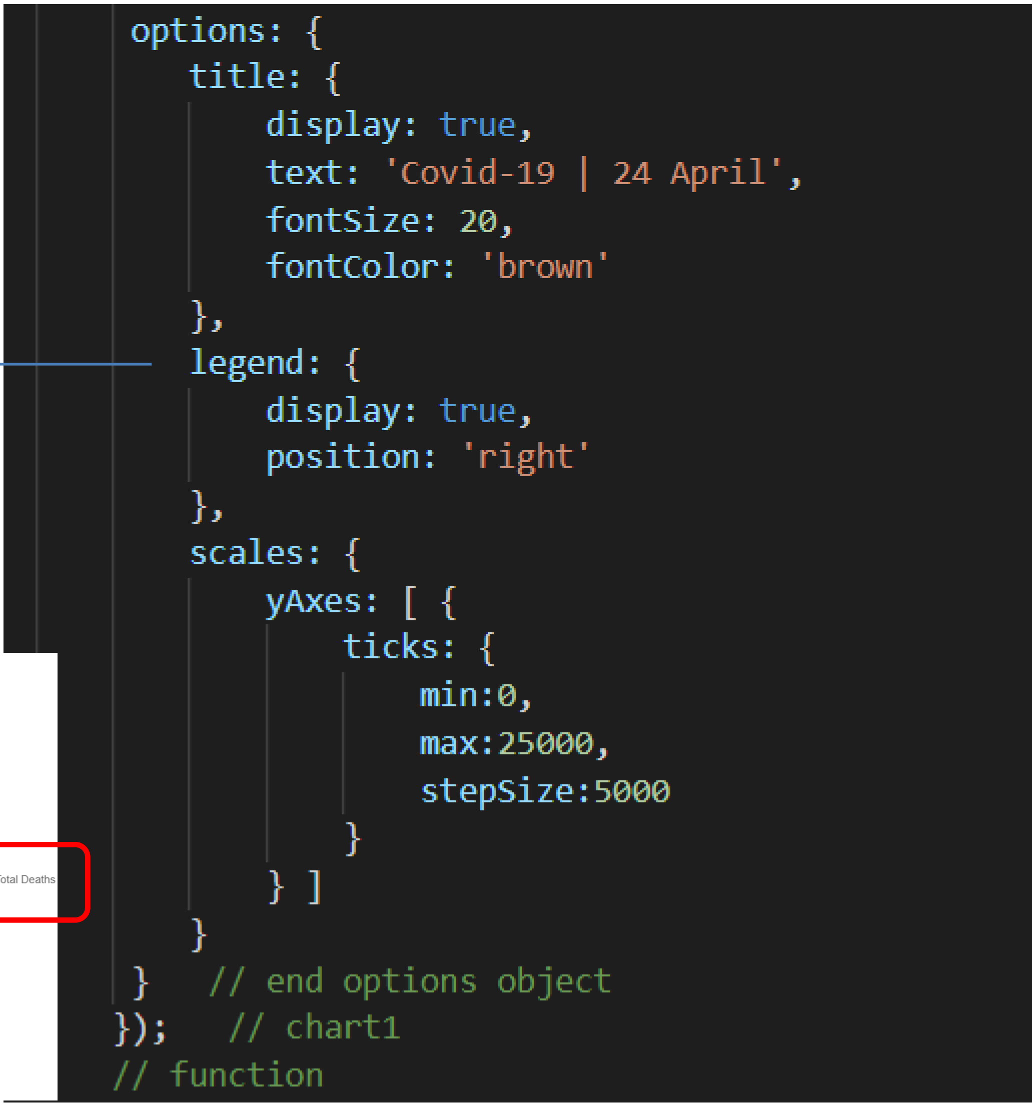
```
        options: {
            title: {
                display: true,
                text: 'Covid-19 | 24 April',
                fontSize: 20,
                fontColor: 'brown'
            },
            legend: {
                display: true,
                position: 'right'
            },
            scales: {
                yAxes: [ {
                    ticks: {
                        min:0,
                        max:25000,
                        stepSize:5000
                    }
                } ]
            }
        }   // end options object
    });   // chart1
}   // function
```

4. Chart.js | Options

Options:
- title
- legend
- scales
- animations

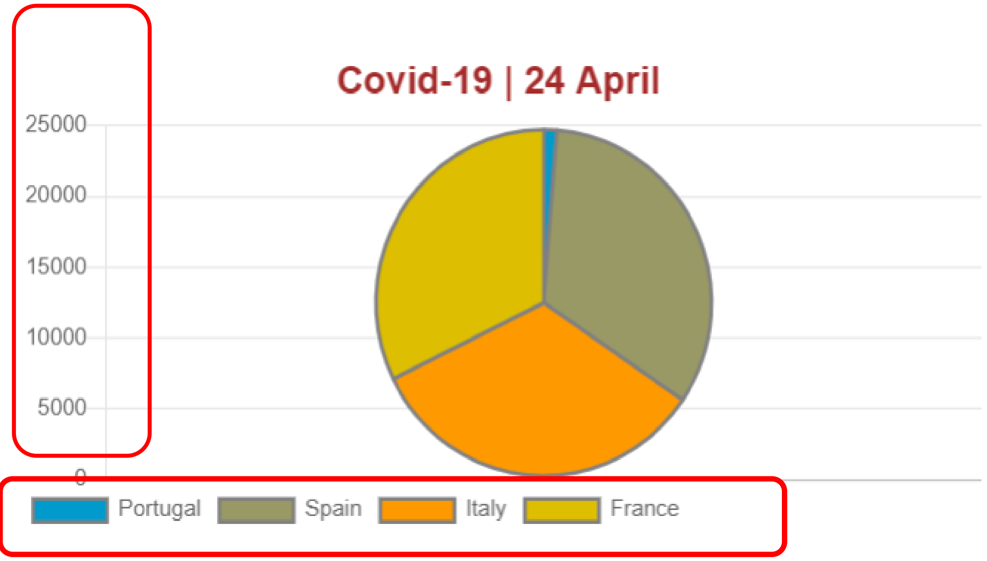
Other properties (legend):
- *position*: 'top', 'bottom', 'left', 'right'
- fontFamily: 'Helvetica Neue', 'Helvetica', 'Arial', 'sans-serif',…
- *align*: 'start', 'center', 'end'

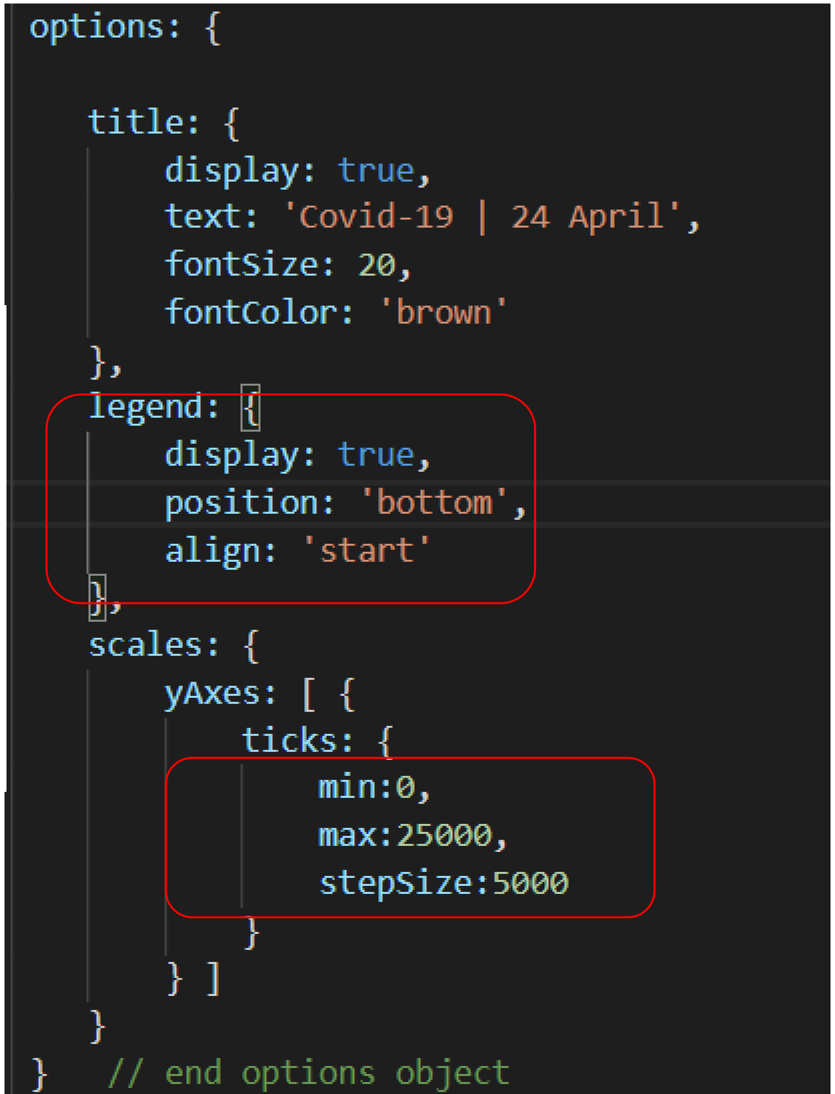




```
options: {
  title: {
    display: true,
    text: 'Covid-19 | 24 April',
    fontSize: 20,
    fontColor: 'brown'
  },
  legend: {
    display: true,
    position: 'right'
  },
  scales: {
    yAxes: [ {
      ticks: {
        min:0,
        max:25000,
        stepSize:5000
      }
    } ]
  }
}  // end options object
});  // chart1
// function
```

4. Chart.js | Options





```
options: {

    title: {
        display: true,
        text: 'Covid-19 | 24 April',
        fontSize: 20,
        fontColor: 'brown'
    },
    legend: {
        display: true,
        position: 'bottom',
        align: 'start'
    },
    scales: {
        yAxes: [ {
            ticks: {
                min:0,
                max:25000,
                stepSize:5000
            }
        } ]
    }
}   // end options object
```

4. Chart.js | Options

```
options: {
    scales: {
        yAxes: [{
            ticks: {
                beginAtZero: true
            }
        }]
    }
}
```

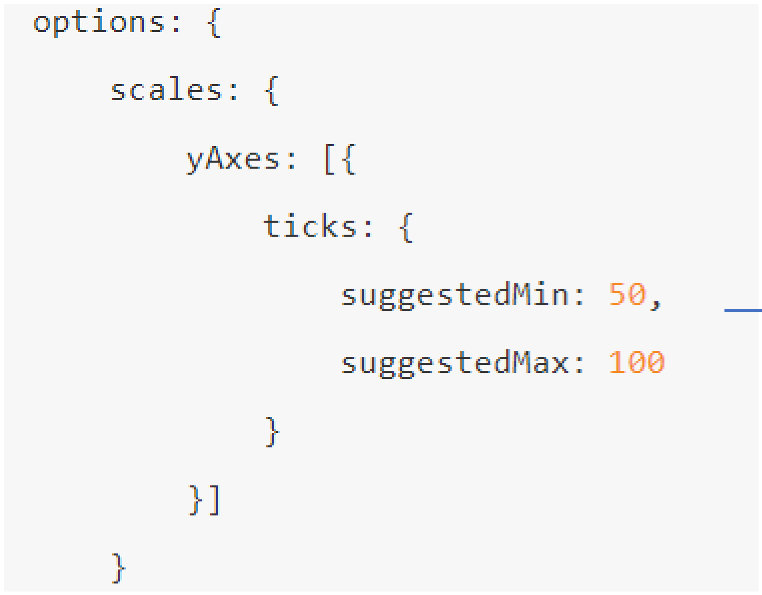
Given the number of axis range settings, it is important to understand how they all interact with each other.

The `suggestedMax` and `suggestedMin` settings only change the data values that are used to scale the axis. These are useful for extending the range of the axis while maintaining the auto fit behaviour.

```
let minDataValue = Math.min(mostNegativeValue, options.ticks.suggestedMin);
let maxDataValue = Math.max(mostPositiveValue, options.ticks.suggestedMax);
```

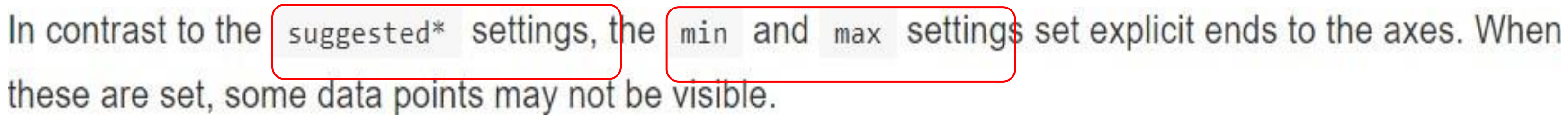
4. Chart.js | Options

Options



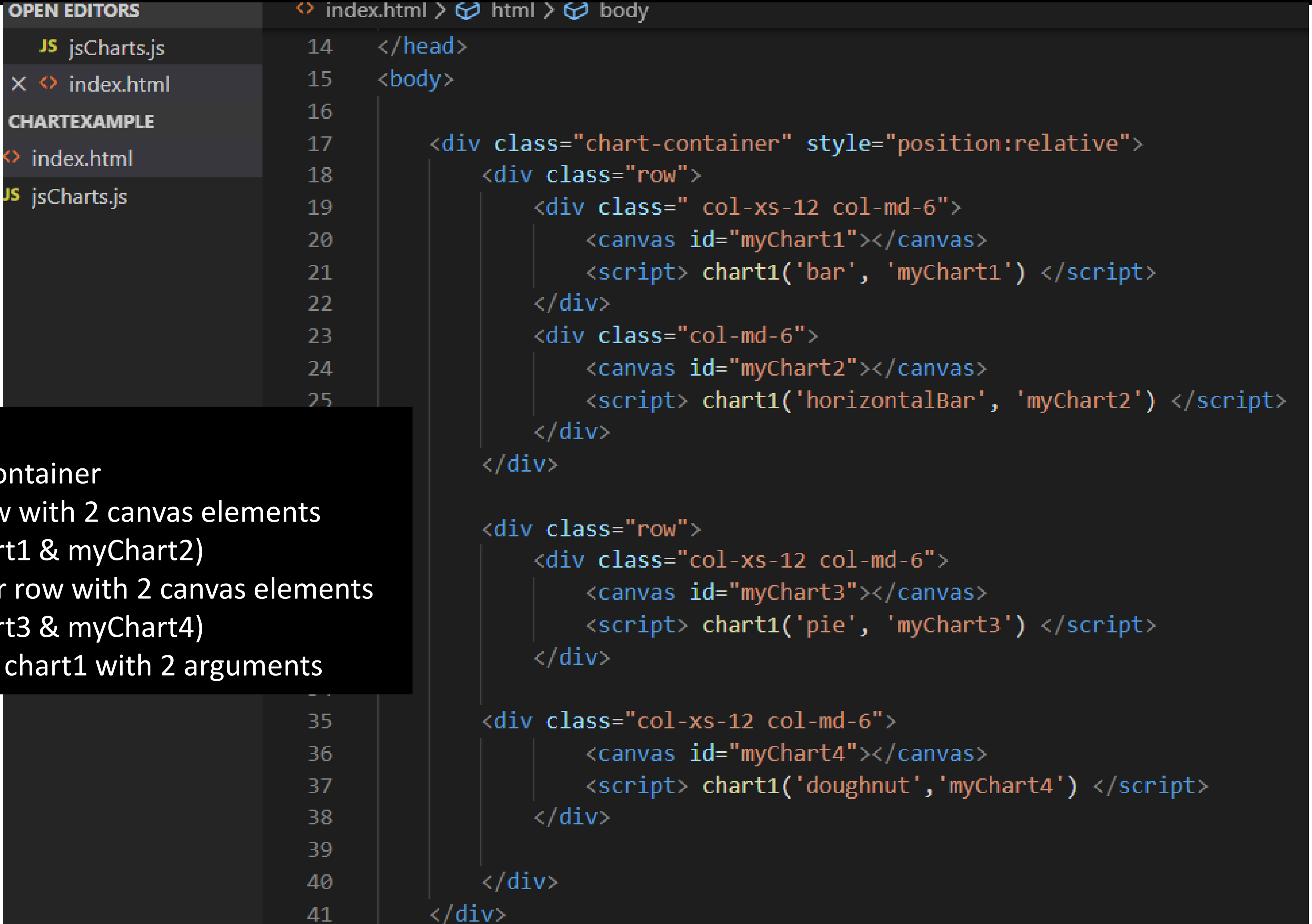
```
options: {
    scales: {
        yAxes: [{
            ticks: {
                suggestedMin: 50,
                suggestedMax: 100
            }
        }]
    }
```

Axis Range Settings



In contrast to the `suggested*` settings, the `min` and `max` settings set explicit ends to the axes. When these are set, some data points may not be visible.

5. Chart.js | A dashboard example

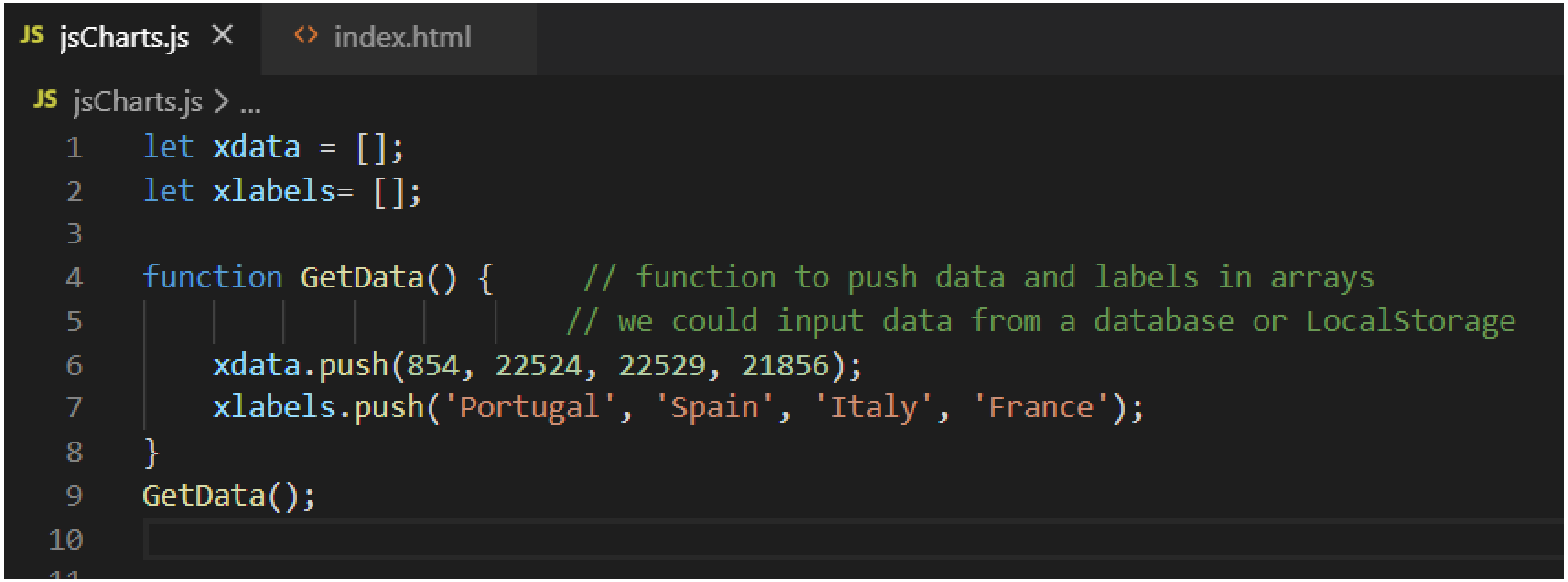


```html
</head>
<body>

    <div class="chart-container" style="position:relative">
        <div class="row">
            <div class=" col-xs-12 col-md-6">
                <canvas id="myChart1"></canvas>
                <script> chart1('bar', 'myChart1') </script>
            </div>
            <div class="col-md-6">
                <canvas id="myChart2"></canvas>
                <script> chart1('horizontalBar', 'myChart2') </script>
            </div>
        </div>

        <div class="row">
            <div class="col-xs-12 col-md-6">
                <canvas id="myChart3"></canvas>
                <script> chart1('pie', 'myChart3') </script>
            </div>

        <div class="col-xs-12 col-md-6">
                <canvas id="myChart4"></canvas>
                <script> chart1('doughnut','myChart4') </script>
            </div>

        </div>
    </div>
```

Notes:
- chart-container
- One row with 2 canvas elements (myChart1 & myChart2)
- Another row with 2 canvas elements (myChart3 & myChart4)
- Function chart1 with 2 arguments

5. Chart.js | A dashboard example

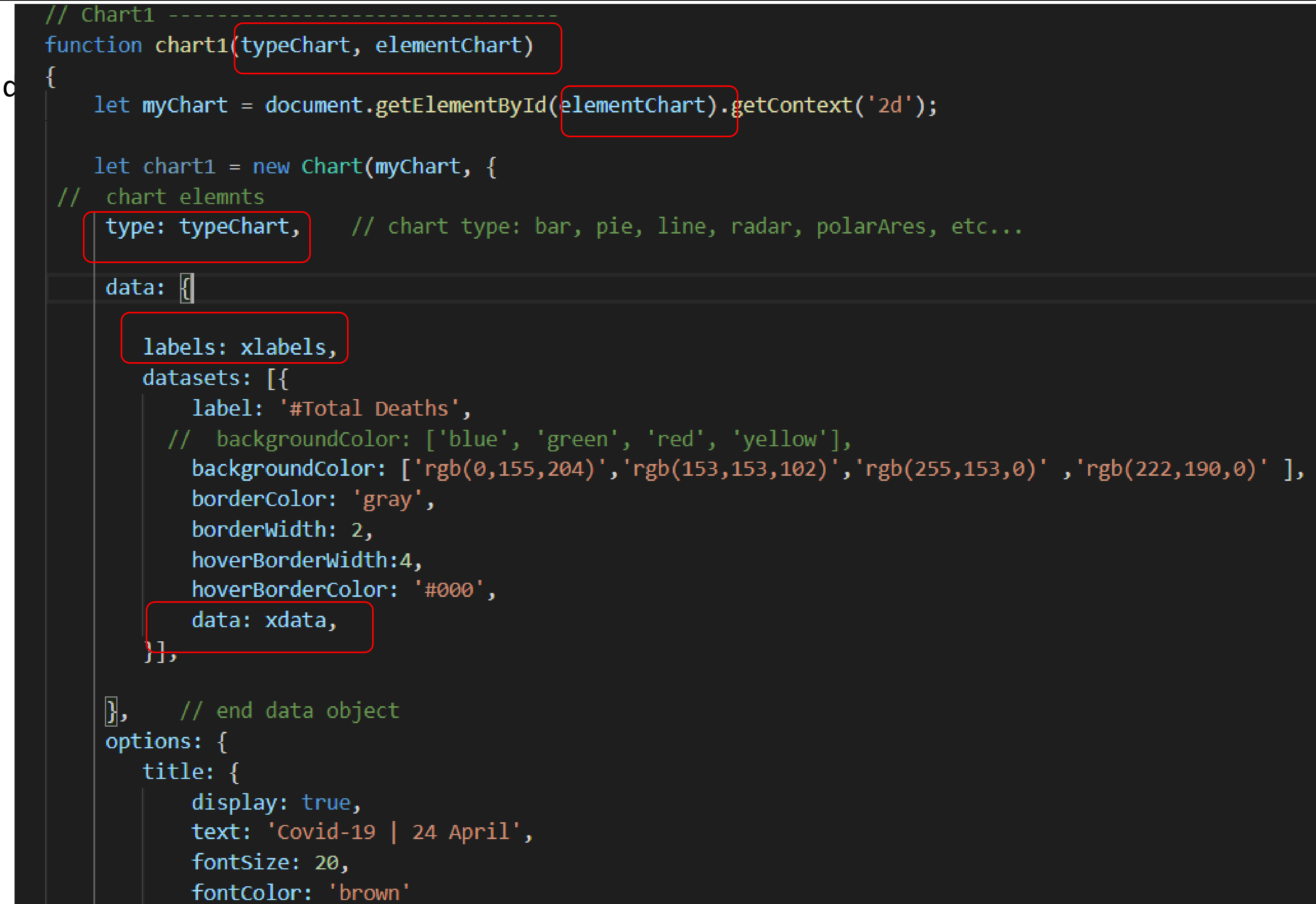
- A dashboard example



```javascript
let xdata = [];
let xlabels= [];

function GetData() {      // function to push data and labels in arrays
                         // we could input data from a database or LocalStorage
    xdata.push(854, 22524, 22529, 21856);
    xlabels.push('Portugal', 'Spain', 'Italy', 'France');
}
GetData();
```

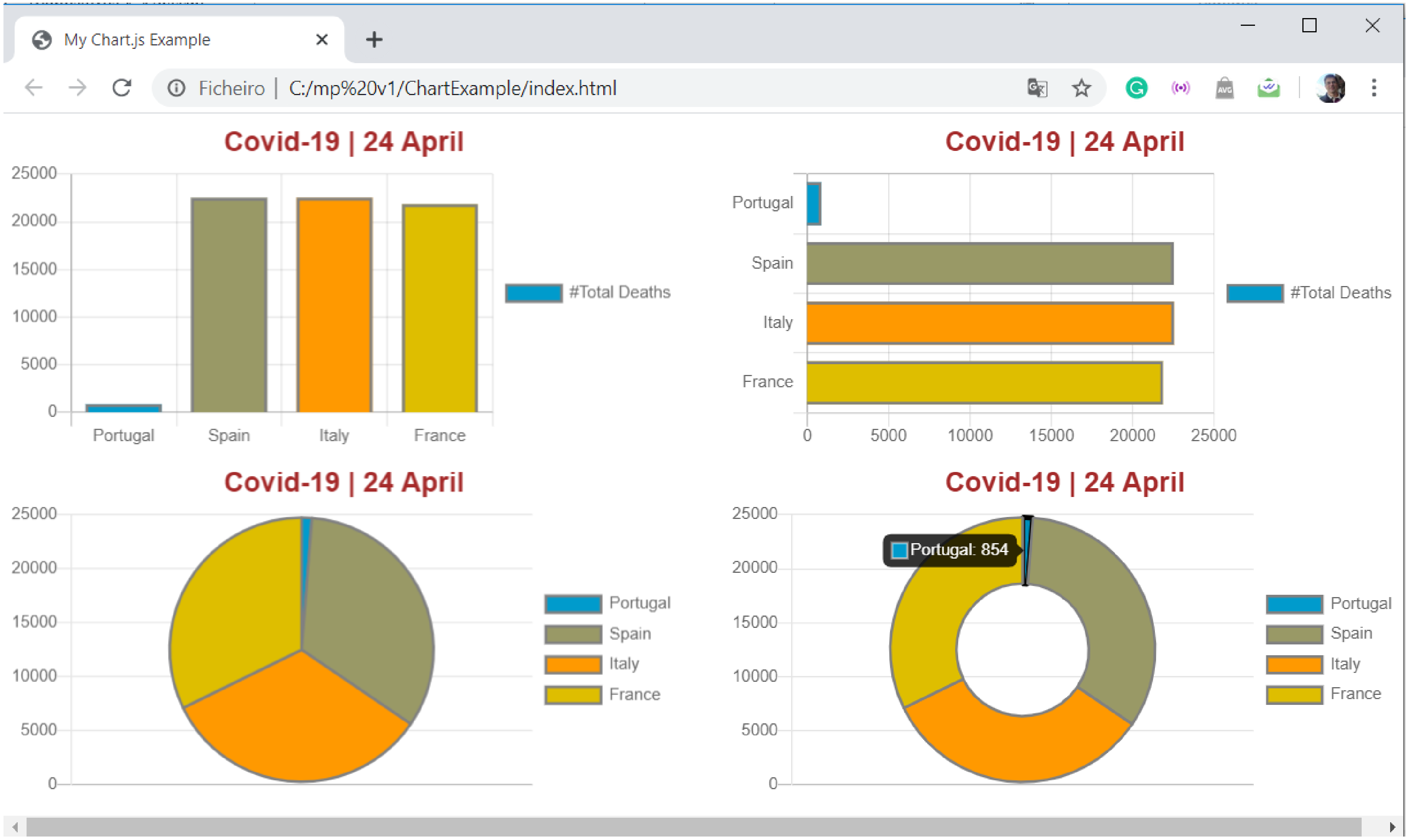
5. Chart.js | A dashboard example

- A c



```
// Chart1 ---------------------------------
function chart1(typeChart, elementChart)
{
    let myChart = document.getElementById(elementChart).getContext('2d');

    let chart1 = new Chart(myChart, {
 //  chart elemnts
     type: typeChart,    // chart type: bar, pie, line, radar, polarAres, etc...

     data: {

        labels: xlabels,
        datasets: [{
            label: '#Total Deaths',
          //  backgroundColor: ['blue', 'green', 'red', 'yellow'],
            backgroundColor: ['rgb(0,155,204)','rgb(153,153,102)','rgb(255,153,0)' ,'rgb(222,190,0)' ],
            borderColor: 'gray',
            borderWidth: 2,
            hoverBorderWidth:4,
            hoverBorderColor: '#000',
            data: xdata,
        }],

     },    // end data object
     options: {
        title: {
            display: true,
            text: 'Covid-19 | 24 April',
            fontSize: 20,
            fontColor: 'brown'
```

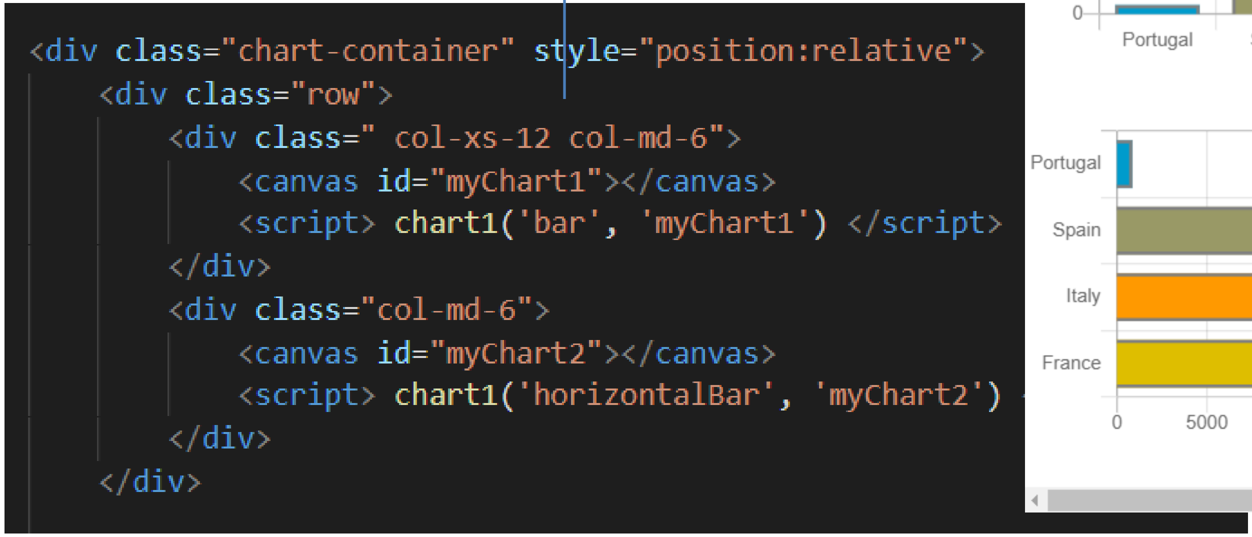
5. Chart.js | A dashboard example



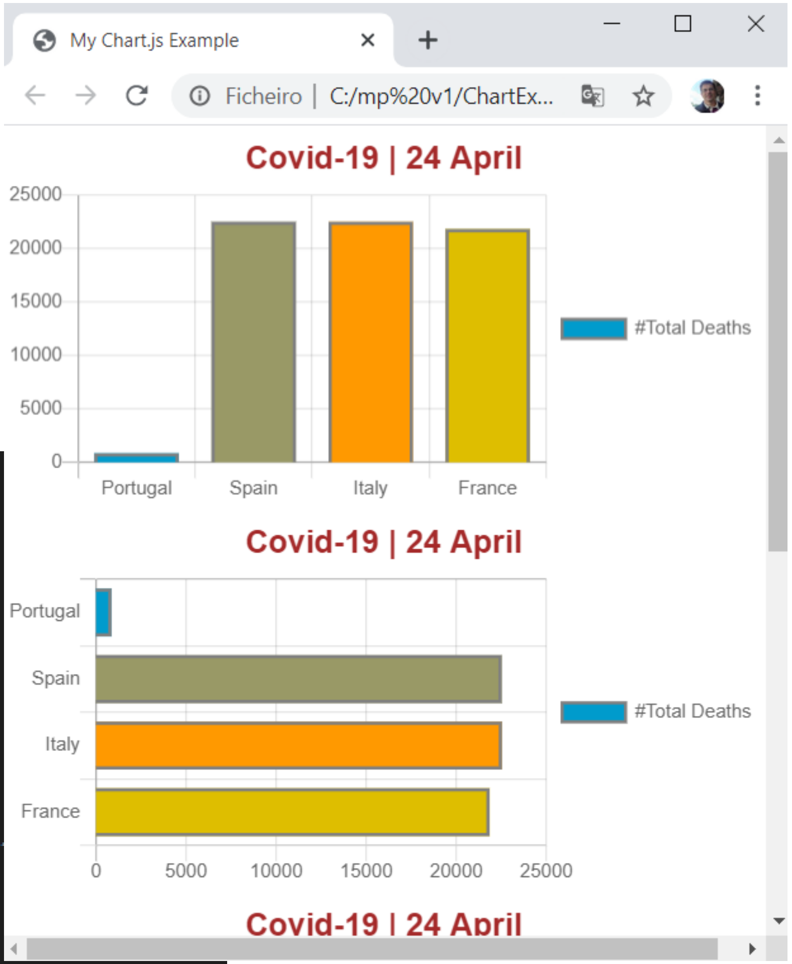
5. Chart.js | A dashboard example

- ❑ Responsive charts!

- ❑ Remember in this example:
- Extra small screens (xs), 12 grid columns
- Medium screens (md), 6 grid columns

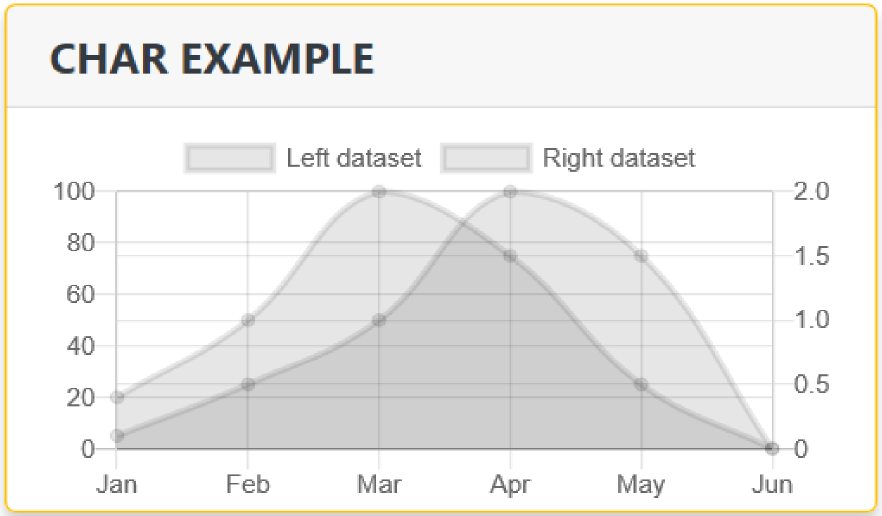


```
<div class="chart-container" style="position:relative">
    <div class="row">
        <div class=" col-xs-12 col-md-6">
            <canvas id="myChart1"></canvas>
            <script> chart1('bar', 'myChart1') </script>
        </div>
        <div class="col-md-6">
            <canvas id="myChart2"></canvas>
            <script> chart1('horizontalBar', 'myChart2')
        </div>
    </div>
```



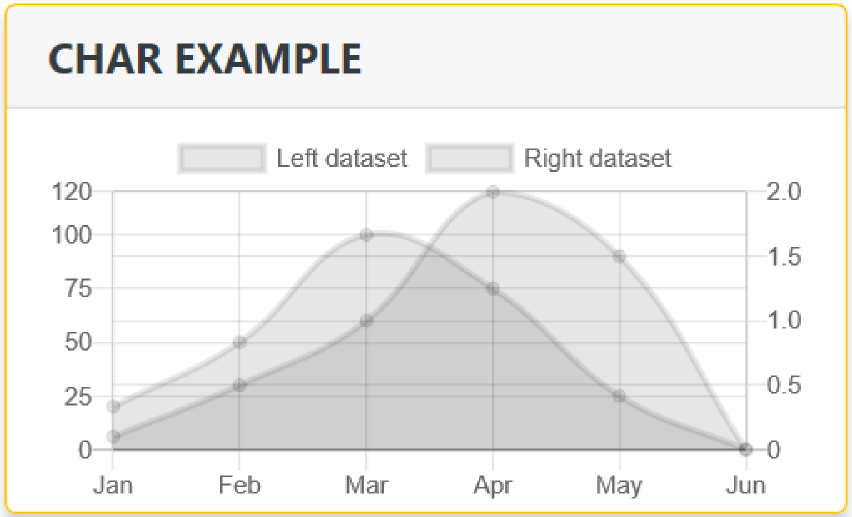
6. Chart.js | 2-axis charts

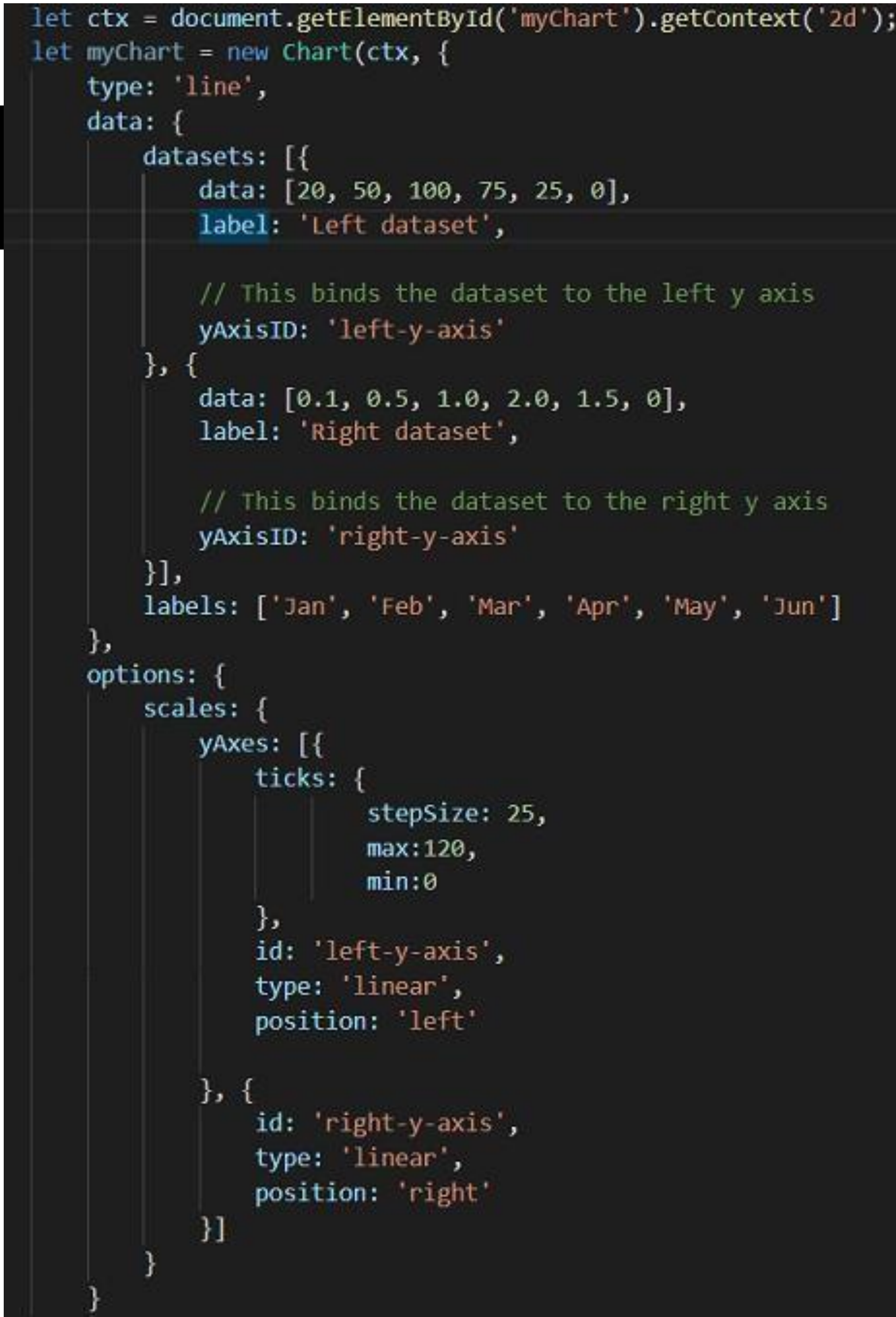
```
options: {
    scales: {
        yAxes: [{
            id: 'left-y-axis',
            type: 'linear',
            position: 'left'
        }, {
            id: 'right-y-axis',
            type: 'linear',
            position: 'right'
        }]
    }
}
```



Creating 2 Axes

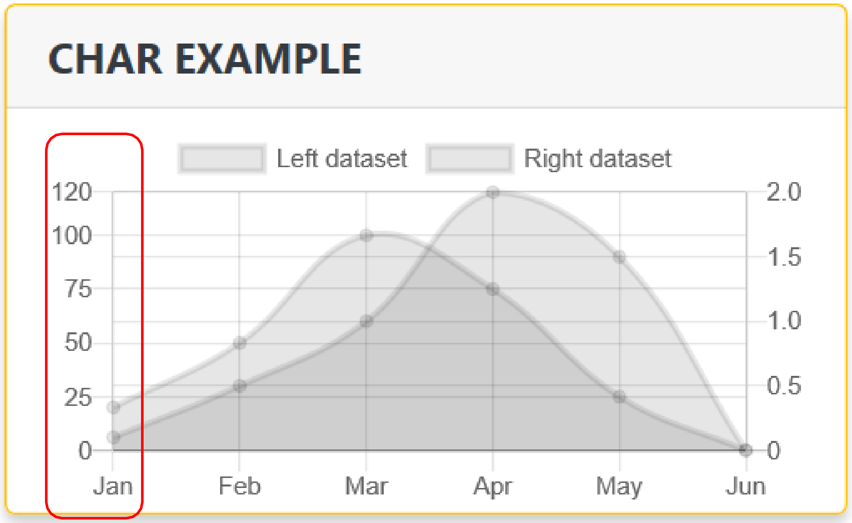
6. Chart.js |

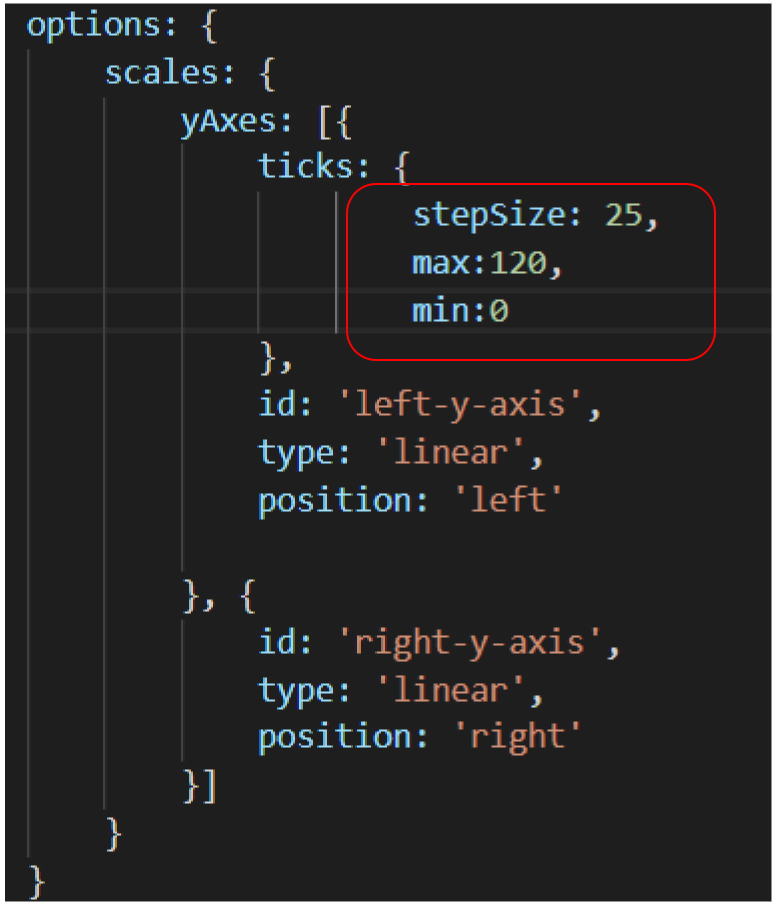




```
let ctx = document.getElementById('myChart').getContext('2d');
let myChart = new Chart(ctx, {
    type: 'line',
    data: {
        datasets: [{
            data: [20, 50, 100, 75, 25, 0],
            label: 'Left dataset',

            // This binds the dataset to the left y axis
            yAxisID: 'left-y-axis'
        }, {
            data: [0.1, 0.5, 1.0, 2.0, 1.5, 0],
            label: 'Right dataset',

            // This binds the dataset to the right y axis
            yAxisID: 'right-y-axis'
        }],
        labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                        stepSize: 25,
                        max:120,
                        min:0
                },
                id: 'left-y-axis',
                type: 'linear',
                position: 'left'

            }, {
                id: 'right-y-axis',
                type: 'linear',
                position: 'right'
            }]
        }
    }
```

6. Chart.js | 2-axis charts





```
options: {
    scales: {
        yAxes: [{
            ticks: {
                    stepSize: 25,
                    max:120,
                    min:0
            },
            id: 'left-y-axis',
            type: 'linear',
            position: 'left'

        }, {
            id: 'right-y-axis',
            type: 'linear',
            position: 'right'
        }]
    }
}
```

7. Chart.js | Documentation

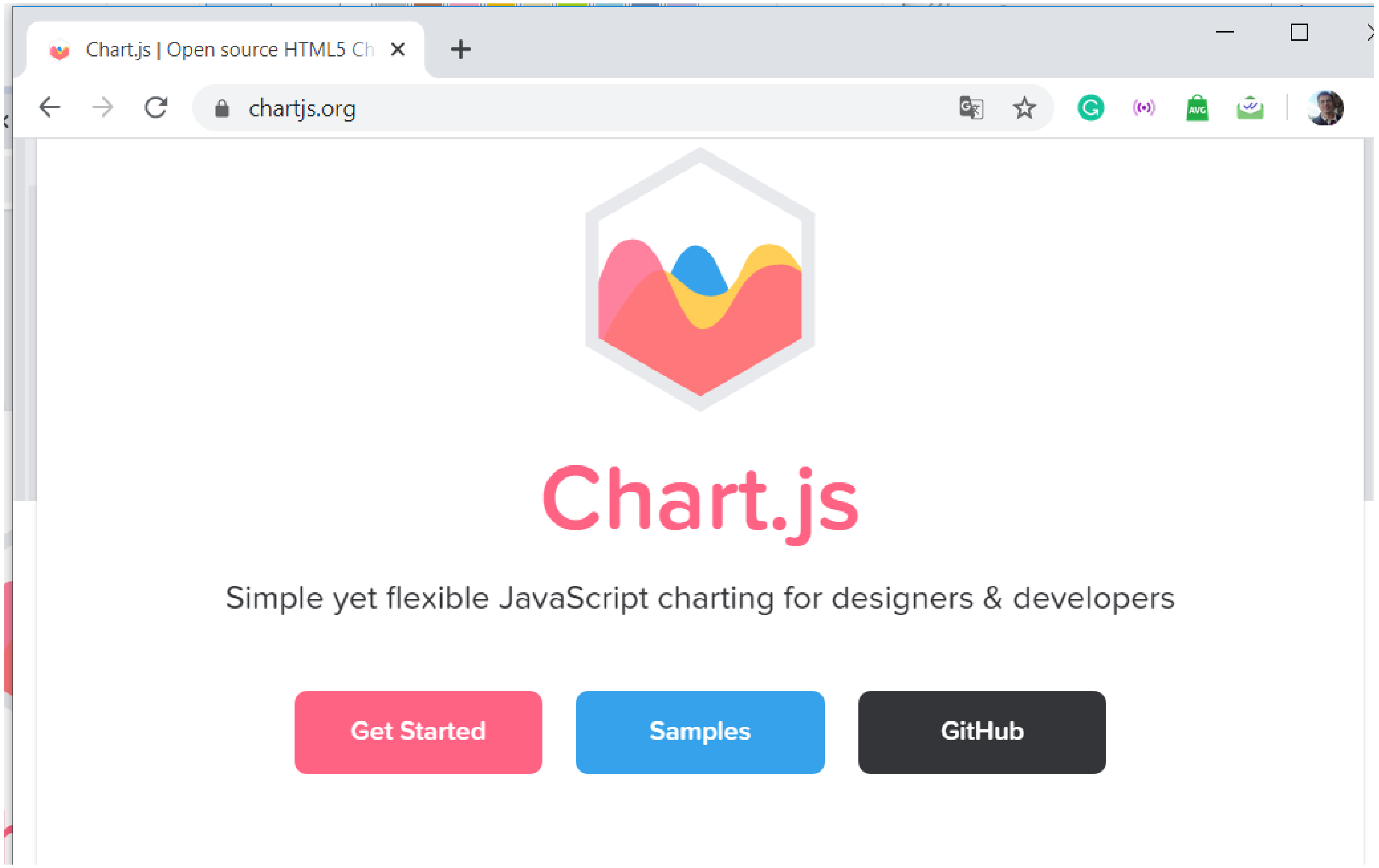


Chart.js | Open source HTML5 Ch
chartjs.org
Chart.js
Simple yet flexible JavaScript charting for designers & developers
Get Started
Samples
GitHub

7. Chart.js | Documentation

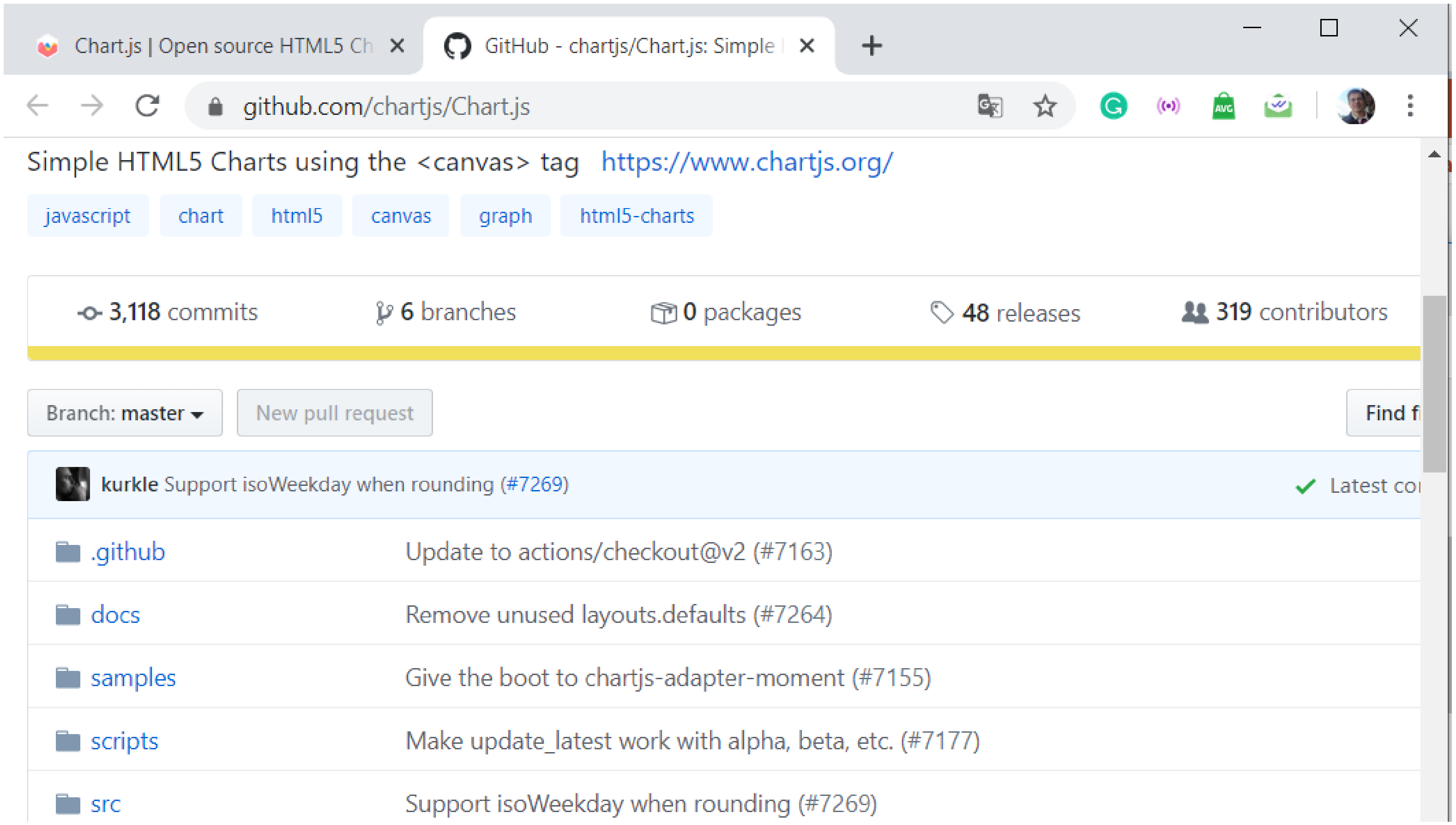


Chart.js | Open source HTML5 Ch
GitHub - chartjs/Chart.js: Simple

github.com/chartjs/Chart.js

Simple HTML5 Charts using the <canvas> tag https://www.chartjs.org/

javascript chart html5 canvas graph html5-charts

3,118 commits | 6 branches | 0 packages | 48 releases | 319 contributors

Branch: master | New pull request | Find fi

kurkle Support isoWeekday when rounding (#7269) ✓ Latest co

.github	Update to actions/checkout@v2 (#7163)
docs	Remove unused layouts.defaults (#7264)
samples	Give the boot to chartjs-adapter-moment (#7155)
scripts	Make update_latest work with alpha, beta, etc. (#7177)
src	Support isoWeekday when rounding (#7269)

7. Chart.js | Documentation

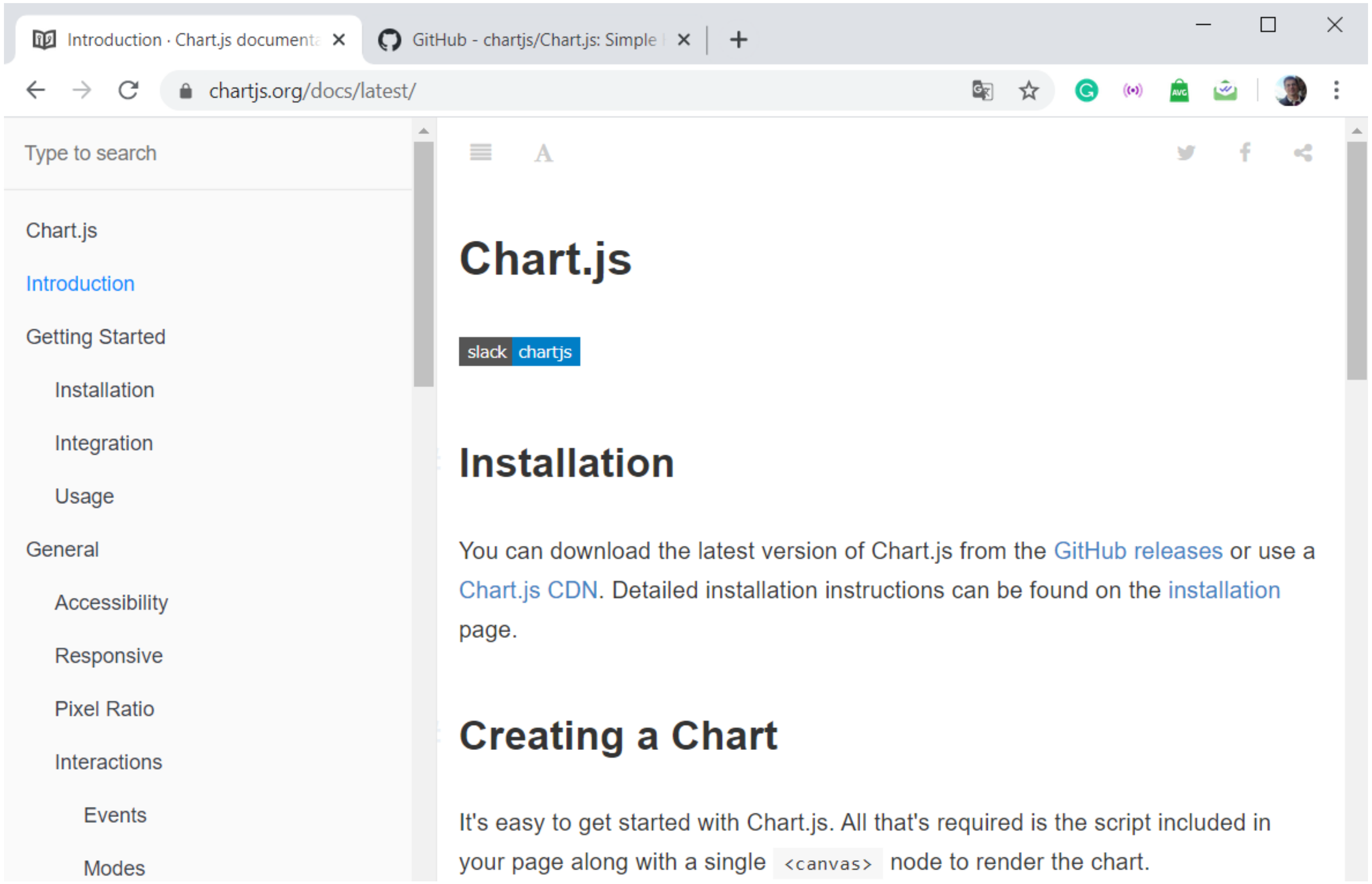

Introduction · Chart.js documenta
GitHub - chartjs/Chart.js: Simple
chartjs.org/docs/latest/
Type to search

Chart.js
Introduction
Getting Started
Installation
Integration
Usage
General
Accessibility
Responsive
Pixel Ratio
Interactions
Events
Modes

Chart.js

slack chartjs

Installation

You can download the latest version of Chart.js from the GitHub releases or use a Chart.js CDN. Detailed installation instructions can be found on the installation page.

Creating a Chart

It's easy to get started with Chart.js. All that's required is the script included in your page along with a single `<canvas>` node to render the chart.

8. Chart.js | API

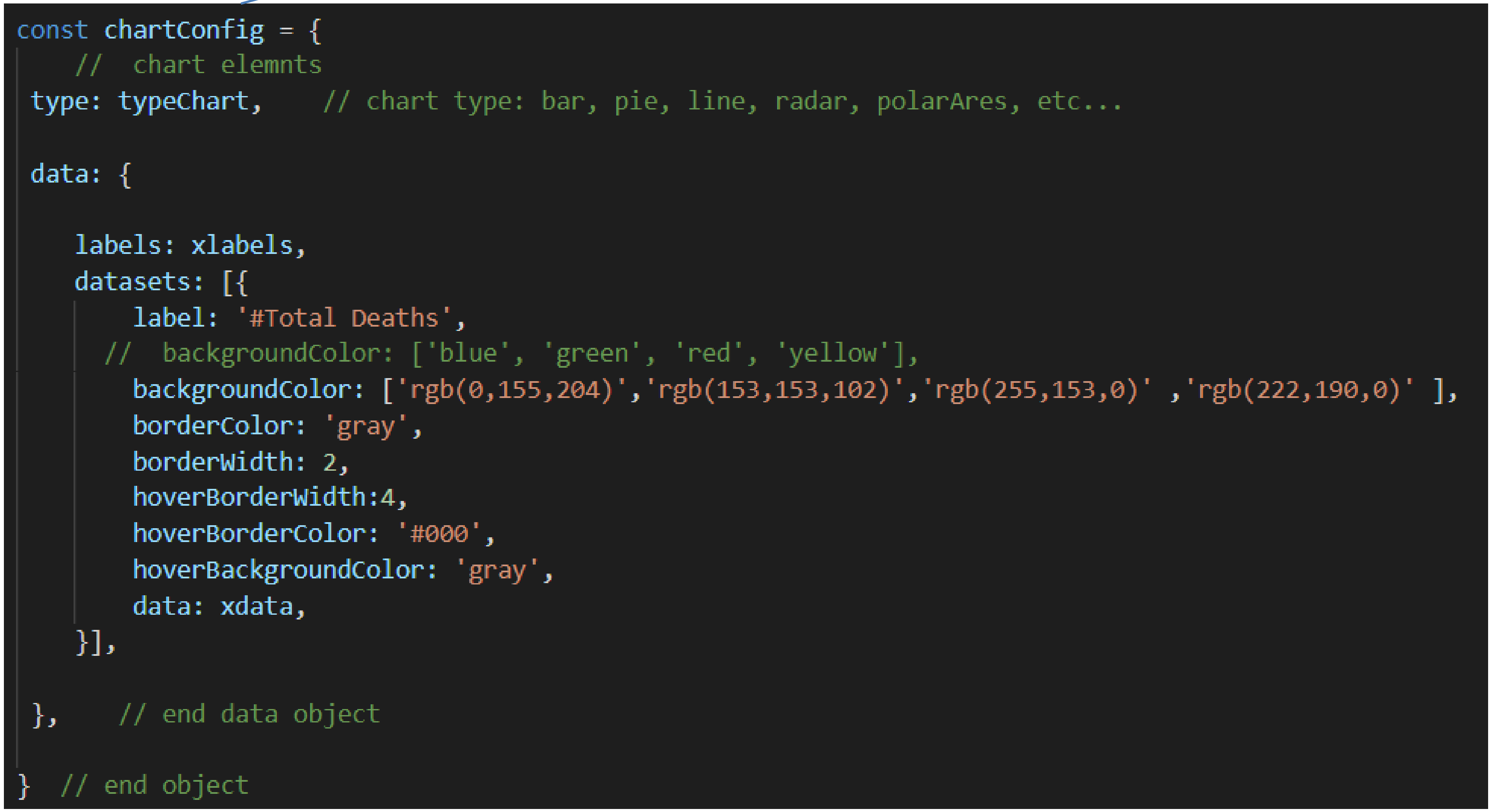
Chart.js provides a set of methods, useful to manipulate charts.
Some methods:

methods	Description
.render()	Redraw of all chart elements. Note, this does not update elements for new data. Use .update() in that case
.update()	This will update all scales, legends, and then re-render the chart.
.clear()	Will clear the chart canvas. Used extensively internally between animation frames, but you might find it useful.
.reset()	Reset the chart to it's state before the initial animation. A new animation can then be triggered using update
.resize()	Use this to resize the canvas element

8. Chart.js | API

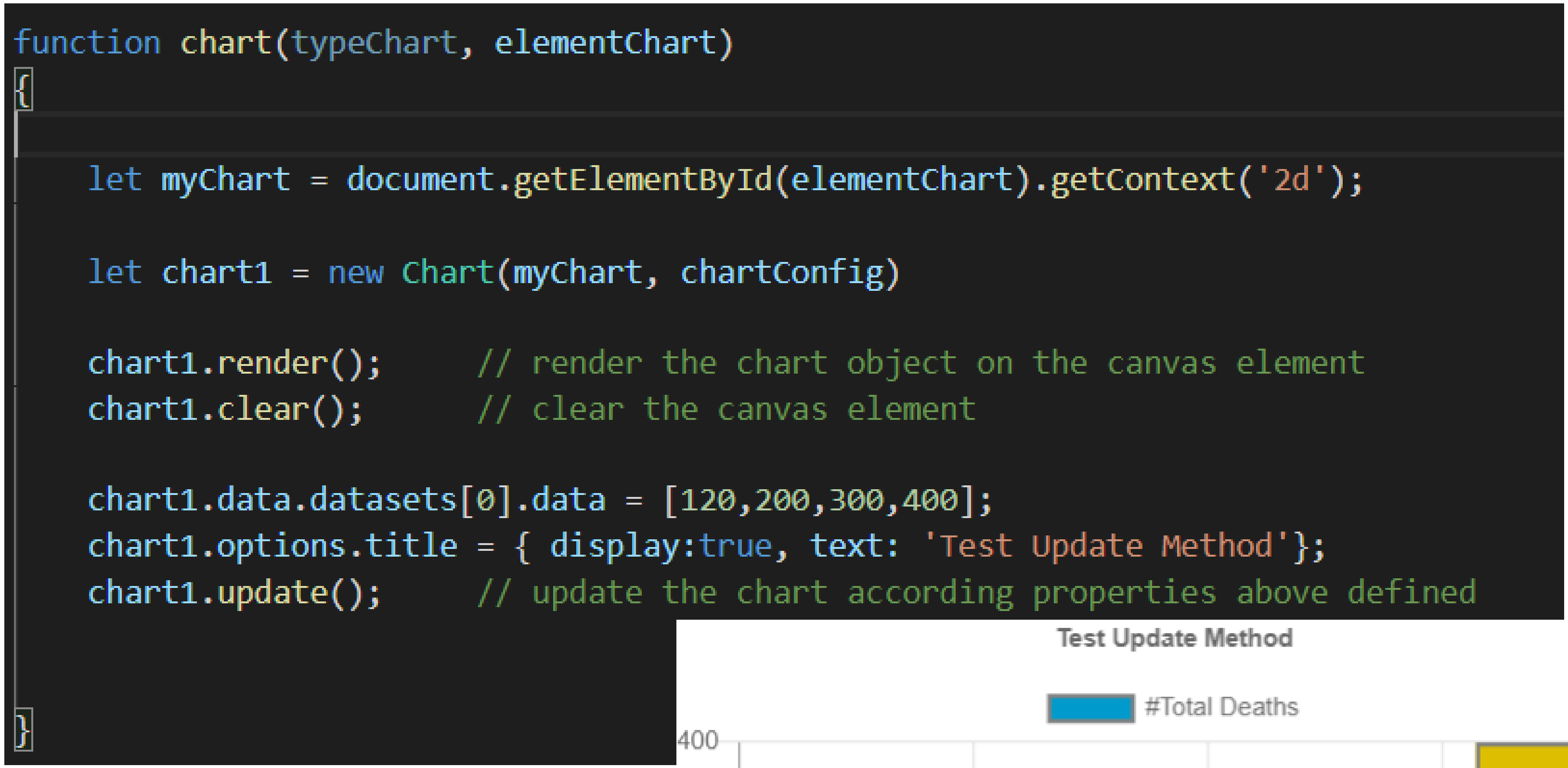
A simple example…

const with chart element description



```
const chartConfig = {
    //  chart elemnts
 type: typeChart,    // chart type: bar, pie, line, radar, polarAres, etc...

 data: {

    labels: xlabels,
    datasets: [{
        label: '#Total Deaths',
      //  backgroundColor: ['blue', 'green', 'red', 'yellow'],
        backgroundColor: ['rgb(0,155,204)','rgb(153,153,102)','rgb(255,153,0)' ,'rgb(222,190,0)' ],
        borderColor: 'gray',
        borderWidth: 2,
        hoverBorderWidth:4,
        hoverBorderColor: '#000',
        hoverBackgroundColor: 'gray',
        data: xdata,
    }],

 },    // end data object

}  // end object
```

8. Chart.js | API



```
function chart(typeChart, elementChart)
{

    let myChart = document.getElementById(elementChart).getContext('2d');

    let chart1 = new Chart(myChart, chartConfig)

    chart1.render();      // render the chart object on the canvas element
    chart1.clear();       // clear the canvas element

    chart1.data.datasets[0].data = [120,200,300,400];
    chart1.options.title = { display:true, text: 'Test Update Method'};
    chart1.update();      // update the chart according properties above defined


}
```

Chart object instance

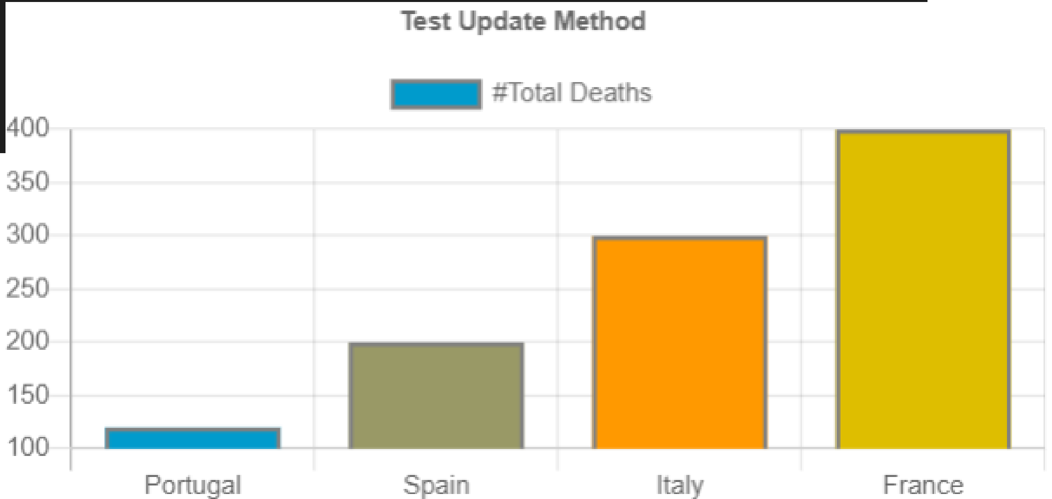
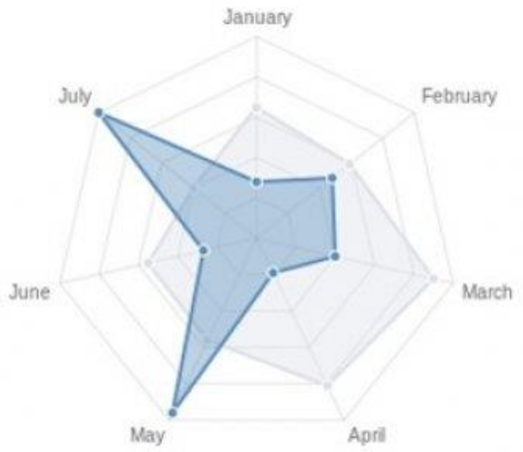
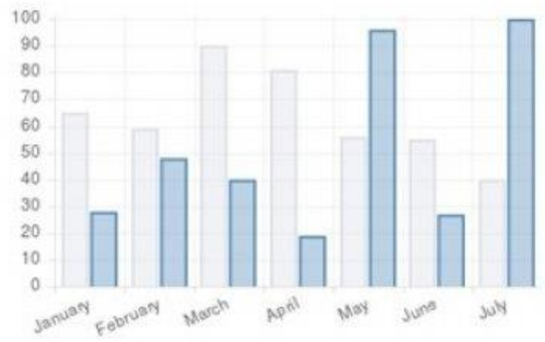
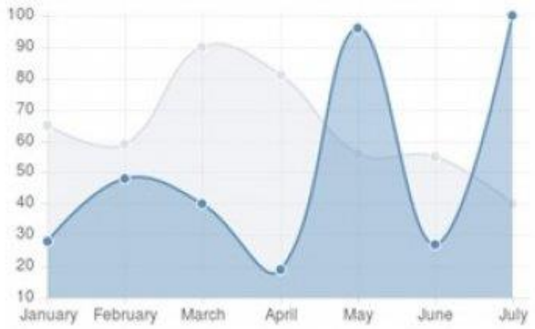
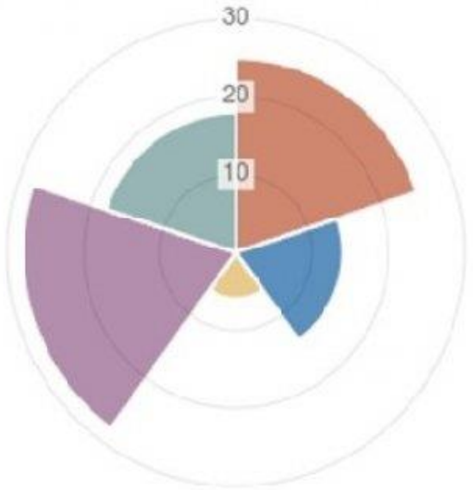


Chart.js









mariopinto@esmad.ipp.pt

Mariopinto

Thank You!