1 AmCharts V4

This page is a general introduction to the charts javascript library AmCharts in its version V4. An extensive documentaion and API can be found here, demos of all the charts that can be made can be found here and tips to customize chart can be found there.

1.1 Instantiating a chart

1.1.1 Modules

AmCharts allows to create several kind of charts. Depending on the one you wish to create, you will need to import several modules from the library. In all cases, you will need the main module `am4core` (which should always comes first) and an other module corresponding to the type of chart : either the `am4charts` for all the *regular* charts, or `am4maps` for map visualisations.

Some kind of charts requires additionnal plugins like force-directed graphs (`forceDirected`) or maps that need to import some geodata to display a background map (`worldLow`).

On top of that, you can import modules for setting assets like translations or themes that will define behaviors and color set of the charts.

```
<script src="https://www.amcharts.com/lib/4/core.js"></script>
<script src="https://www.amcharts.com/lib/4/charts.js"></script>
<script src="https://www.amcharts.com/lib/4/maps.js"></script>
<script src="https://www.amcharts.com/lib/4/geodata/worldLow.js"></script>
<script src="https://www.amcharts.com/lib/4/themes/animated.js"></script>
```

1.1.2 Creating a chart

The `create()` function is always first used to add new instances to your HTML page. It takes two arguments :

- The **tag id** of the `html` element in which is going to contain the chart (the element must already exist in your DOM, the function will not create it) ;
- The **type of chart**, *i.e.* the class reference.

```
var chart = am4core.create("chartdiv", am4charts.XYChart);
```

Combining multiple charts Is it possible to put several charts in the same `html` element. To do that you only need to create first a container in which you will create charts instances :

```
var container = am4core.create("chartdiv", am4core.Container);
var mapChart = container.createChild(am4maps.MapChart);
var XYChart = mapChart.chartContainer.createChild(am4charts.XYChart);
```

Configuration : object/JSON approachs There are two way of setting up the parameters and data for the charts : either a **JSON-based** approach, or an **object-based** approach. Object approach is more readable and less prone to bug whereas JSON approach, being fundamentally a string, can be stored in a database or load dynamically with AJAX.

The rest of this introduction will only give examples with the object approach but you can find more information here on the JSON way. Object-based only means that the chart is treated as an object whose properties we're going to change/call.

1.2 Anatomy of a chart

A chart is mainly composed of two elements : the "***frame***" and the "***shapes***" representing the data :

- The *frame* is, in most cases, composed of two `axes` defining the structure of the chart.
- The data can be displayed in different *shapes* depending of the type of chart : columns, bubbles, points, line, etc. Whatever its shape, data is an instance of one or several `series`.

1.2.1 Axes

Axes are the structure of the chart ; they are specially fundamental to XYChart and some other type of charts. The axes determines how the data is going to be spread out through out the chart.

The axes can be of mainly two types :

- **Category axis** : used to position data items on text-based categories. For example, for a dataset detailling the number of folio per manuscript, the category axis would represent all the manuscripts.
- **Value axis** : used to position data items on a numeric scale. In our previous example, it would represent the total number of folios.
- Other types of axes :
  - **Duration axis**
  - **Date axis**

And, in the case of XYCharts, axes can be horizontal – or circular in case of some charts like radar – (`xAxes`) or vertical (`yAxes`) ; any of them can be a category axis or value axis, and both of them can be of the same type.

Creating axes To associate axis to a chart, you need to push them to your existing chart like so :

```
var valueAxis = chart.yAxes.push(new am4charts.ValueAxis());
var categoryAxis = chart.xAxes.push(new am4charts.CategoryAxis());
```

Configuring axes Most of the appearence parameters and settings can be configured directly on the axis variable or with the `renderer` property :

```
// define a title for the value axis
valueAxis.title.text = "Number of folios";
// display the axis one the opposite side than the default one
categoryAxis.renderer.opposite = true;
// to set the background grid of the value axis to be transparent
valueAxis.renderer.grid.template.strokeOpacity = 0;
// set the font size of the label from the category axis to 20
categoryAxis.renderer.labels.template.fontSize = 20;
```

1.2.2 Series

A series is a collection of similar, logically grouped data points, it gathers the configuration defining how the data is going to be displayed.

One series aggregates data you want to have the same properties. Sometimes one series will be enough to display gigantic datasets, but sometimes you will need to provide each data its series in order to specify finely the settings for each one in particular. A series, in a column chart for instance, can be either — depending on what you want to show — a group of columns, a single column, a group of cells through out the columns or a single cell.

Each type of chart allows only certain kind of series : a `pieSeries` on a `mapChart` will never be displayed.

Series have two main purposes: - Setting appearance and behavior of a collection of chart items ; - Binding individual chart items to source data.

Creating series To create a new series for your chart, you must proceed as for the axes :

```
var series = chart.series.push(new am4charts.ColumnSeries());
```

Configuring series You can set configuration for a series either directly on the series variable or through its `template` property (more on templates later on) :

```
// setting width of the stroke
series.strokeWidth = 0;
// setting text appearing when hovering on columns
series.columns.template.tooltipText = "Number of folios in manuscript";
```

1.2.3 Additionnal components

Inside a chart, you can add several type components like **SVG shapes**, **labels**, etc. To create them, you need to push them in your chart container, then you can configure them as you like :

```
var rectangle = chart.chartContainer.createChild(am4core.Container);
var rectLabel = rectangle.createChild(am4core.Label);
```

1.3 Data

1.3.1 Data structure

All data, in order to be used by your chart, need it to be formatted as an **array of objects** :

```
var msData = [
  {
    "ms": "Latin 10264",
    "folios": 3
  },{
    "ms": "Bav. Lat. 1374",
    "folios": 14
  }
]
```

1.3.2 Binding data

In order to make your chart use your data, you first need to associate your array to the data property of your chart :

```
chart.data = msData;
```

With this line, when we will set data properties to our chart series, it will know automatically where to find the information it needs. You can also bind a dataset to a specific series for complex layouts.

1.3.3 Using `dataFields`

Now, the axes and the series that have been earlier create needs to be associated with the data from our array. The `dataFields` property is a simple object that binds specific field in series, to a data field in data objects : using `dataField` the series, as well as the category axis, will know what key in the dataset they are binded to.

```
series.dataFields.categoryX = "ms";
series.dataFields.valueY = "folios";
categoryAxis.dataFields.category = "ms";
```

What those lines means is : for each object in the array of data, the numerical value for the series will be found to the "folios" key and the string-based value will be found to the "ms" key. The value axis will generate itself according to the maximal value associated with the series to display, but the category axis needs to know which labels to show.

NB : for each kind of chart, the syntax might vary. You will find more details on that in the numerous examples in the demos.

1.3.4 Using `propertiesFields`

The `dataFields` property only holds information that is needed for the chart to be displayed, additionnal information can be binded to the chart to customize it further : `propertyFields` are the way to tie chart element's properties to data.

Let's say some fields were added to your dataset like that :

```
var msData = [
  {
    "ms": "Latin 10264",
    "folios": 3,
    "color": "#fcba03",
    "url": "https://archivesetmanuscrits.bnf.fr/ark:/12148/cc71999k"
  },{
    "ms": "Bav. Lat. 1374",
    "folios": 14,
    "color": "#fc3103",
    "url": "https://digi.ub.uni-heidelberg.de/diglit/bav_pal_lat_1374"
  }
];
```

Pretty much any part of your chart can be customize using information from your dataset with the `propertiesField` property :

```
// make column labels a link
categoryAxis.renderer.labels.template.propertyFields.url = "url";
// colour the columns
series.columns.template.propertyFields.fill = "color";
```

Using `dummyData` You can bind any data of your choice to a chart item using its `dummyData` property : it is a sort of storage container to put anything you want in it, without affecting the overall behavior of your chart.

It can be very useful when you will need to customize your chart creating tailored function using event listeners or adapters : it can become handy if you are dynamically generating data you want to access later on, for instance to bind behavior for two distinct events.

1.4 Appearance configuration

1.4.1 Templates

Practically all of the charts components have a `template` property : it is some kind of pattern for the layout and behavior of specific parts of your chart. Using templates, you can, for example, set that all filling colors of columns to be half transparent or for all axis labels to become green when clicked on :

```
// Half transparency of the columns
series.columns.template.fillOpacity = 0.5;
// Green labels
valueAxis.renderer.labels.template.events.on("hit", function(ev) {
  valueAxis.renderer.labels.template.fill = "green";
});
```

Templates can be modified dynamically using DOM events, which is not the case with data values (associated with `dataFields`) : to change the values your chart is using, you will need either to switch datasets or use incremental loading).

AmCharts is an incredibly customisable library : if you would like to change any aspects of your chart, by browsing through the properties of the element you want to modify in the API, you will surely find something to suit your desires.

1.4.2 Colors, percentage, etc.

To unlock all methods applicable to colors and percentages, you will need to use special Amcharts functions. The `color()` method takes only one argument of color and transform it into an object that can be later on modified. The same is true for `percent()` method. To define the color or the percentage of a property :

```
rectangle.width = am4core.percent(50);
rectangle.background.fill = am4core.color("orange");
```

Heat rules Heat rules defines a property of a chart item that will be bind to a numeric value. Any property is usable, as long as you can define a min and max state like so :

```
// The higher the value, the wider the column
series.heatRules.push({
    target: series.columns.template,
    property: "width",
    min: 80,
    max: 160
});
```

1.4.3 States

Any part of the chart can be associated with a `state` object that holds properties defining how the element will behave when on this particular state. States define alternative properties that will override "*normal state*" properties on certain occasions.

As many other objects in AmCharts, defining state is a two steps process : first **creating** the state for a specific element, then **configurating** it :

```
var hoverState = series.columns.template.states.create("hover");
hoverState.properties.fillOpacity = 0.9;
```

1.4.4 Formatting text in labels and tooltips

Font styling Text styling options are always contained in `[square brackets]`. It works like HTML tag so each opening bracket should be closed with a closing bracket (`[/]`), but an other opening bracket will act as a closing one for one opened earlier.

Some styling have an already implemented syntax in AmCharts, but you can use any CSS property :

```
rectLabel.text = "[bold]Work diffusion[/] :\n the example of the [font-style: italic]Tabule Magne
    [/]";
```

Accessing data values Is it possible to display data in labels with the `{curly brackets}` notation. Depending on what you write in your brackets, the parser (or the "formatter" to use AmCharts taxonomy) will try to find matching data in your dataset. You can either use the name of a key in your dataset, or the class reference of the data :

```
series.columns.template.tooltipText = "Numbers of [{color}]folios[/] in {ms} : \n {valueY}";
```

You can even perform some data computing on your numeric values with the formatter using modifiers such as `sum` or `count`.

Using HTML You can also display any HTML elements in your tooltips and labels using the `html` or `tooltipHTML` property :

```
rectangle.tooltipHTML = '<span style="font-variant: small-caps;">Regarding folio quantity in
    manuscript</span>';
```

1.5 Additionnal features

Satellite elements can be instantiated by creating a new object of their class, they will behave accordingly to their normal use but you can completely divert their utility

1.5.1 Cursors

Cursors are only available for XY charts and radar charts : they allow to add interactivity to charts as well as providing zoom modalities.

```
chart.cursor = new am4charts.XYCursor();
```

By default, cursors, when moving on the chart, display two crossing lines ending with tooltips to better pinpoint and understand the data, but all of this can be disabled or modified. You can as well, add some cursor behaviors to allow zooming and panning :

```
chart.cursor.behavior = "selectX";
```

1.5.2 Scrollbar

A scrollbar is associated to an axis of the chart and allows to zoom in it. Is it possible to push series in it to display a preview of the value of your chart, even to treat it as a chart in itself.

```
var scrollbarX = new am4charts.XYChartScrollbar();
scrollbarX.series.push(series);
chart.scrollbarX = scrollbarX;
```

1.5.3 Legends

A legend, when created, will automatically display the series that are associated to your chart. When clicking of the items of the legend, corresponding series disappear, giving possibilities to vary the visualization to the user.

In the case of our manuscript chart, adding a legend is not very insightful, because our data is contained in an unique series representing all the columns, which colors were assigned individually. If we wanted to display a matching legend, we would need to build a custom legend.

```
chart.legend = new am4charts.Legend();
```

1.5.4 Events listeners

Event listeners allows to bind custom functions to any action perform on a chart element. Each part of the chart has a list of events tied to variables that can be accessed and modified when the event is triggered.

Two kind of functions can be associated to the events :

- `on()` : any time the event happens ;
- `once()` : the first time the event happens.s

```
chart.cursor.events.on("selectended", function(ev) {
  var range = ev.target.xRange;
  var axis = ev.target.chart.xAxes.getIndex(0);
  var from = axis.getPositionLabel(axis.toAxisPosition(range.start));
  var to = axis.getPositionLabel(axis.toAxisPosition(range.end));
  alert("Selected from " + from + " to " + to);
});
```

The syntax for setting custom functions is always the same :
`{chartName}.{elementName}.events.[on/once]("{eventName}", function(args) {...});`
Some events are universal to any type of element, as `hit`, when the user is clicking or tapping on a touch screen.

1.6 Demos

1.6.1 Our manuscript chart

Here is the chart that was made during this introduction : Number of folios in manuscript.

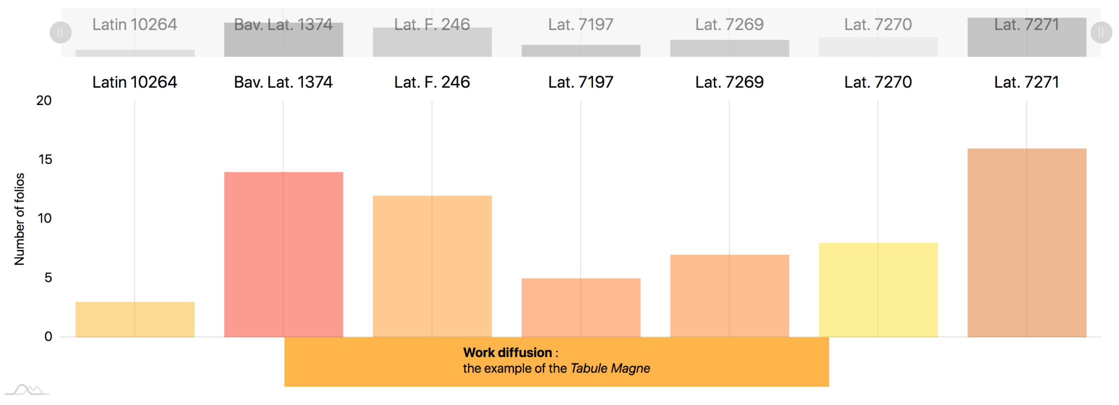

Figure 1: Chart made during the AmCharts tutorial

1.6.2 Examples in DISHAS

- Primary source : dynamic bar chart
- Historical sources : historical map