TIMETRAVEL JS GANTT READER

v1.0.0

Contents

Chapter 1

Getting started

First of all, I would like to thank you for the purchase of this **Gantt Reader** JavaScript library. It will allow you to **display a Gantt Model in a simple and nice way**. This is the first version of the library and I hope it will not be the last. Don't hesitate, **give your feeling** thanks to the market place comment system.

1.1 Simple use case

Before I give you deeper explanations, let's have a look at a simple use case. You will find the sample code below in the following location : *doc/examples/getting-started/index.html*.

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Getting Started</title>
  <link rel="stylesheet" href="../../../dist/css/ganttchart.css">
</head>
<body>
  <div id="gantt-reader" style="width: 500px; height: 500px" />
  <script src="../../../dist/timetraveljs-gantt-reader.min.js"></script>
  <script>
    var myGanttReader = new tt.ganttchart.GanttReader(document.getElementById('gantt-reader'));
    // define the model
    model = '  {                                ' +
            '   "name": "model1",               ' +
            '   "temporalData": [               ' +
            '    {                              ' +
            '      "type": "Task",              ' +
            '      "name": "Sample Task",       ' +
            '      "start": "2015-05-04",       ' +
            '      "finish": "2015-05-10",      ' +
            '      "percentComplete": 50        ' +
            '    }                              ' +
            '   ]                               ' +
            '  }                                ';
    // set the model
    myGanttReader.ganttModel = model;
  </script>
</body>
</html>
```

Here are described the significant lines of code :

N°	Description
6	Import Gantt Reader **css** file.
9	Define a **container** for our Gantt Reader component.
10	**Import** the minified version of the **library**. Please notice that the component **does not need jQuery to work**.
12	Create a **new instance of Gantt Reader** to be included in the `<div>` with `id` equals to `gantt-reader`. `tt` is the root namespace and stands for **t**ime**t**raveljs, `ganttchart` is the sub namespace where `GanttReader` class lives.
14 - 25	Define an inline **JSON Gantt Model**. This model has the name "model1" with a unique temporal data of `Task` type. The format of this model will be discussed later in this document.
27	**Set** the model.

If you launch the *index.html* page you will get this result :

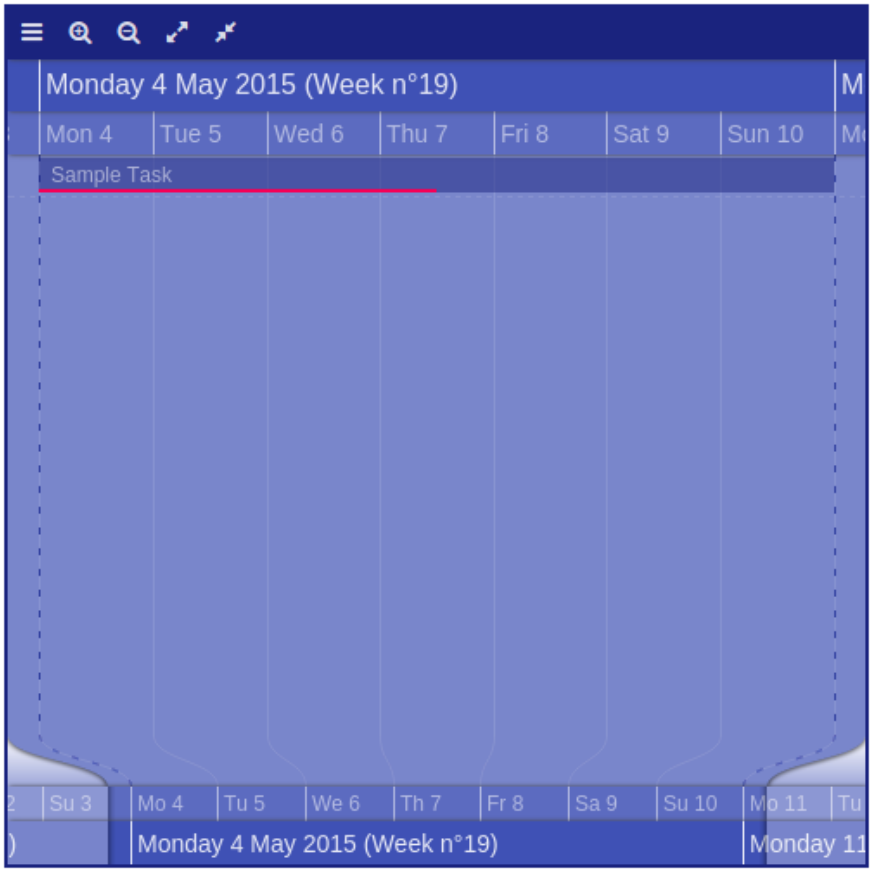

Figure 1.1: Result of getting started example

No more no less, you have simply displayed a tiny Gantt model. **You can play around with the component**. For example you could try some drag and drop on the tasks list, the timelines, click or shift click on different periods... All the controls will be later described on this document.

1.2 A more complex model

The Gantt Reader can display complex set of temporal data like the next example will demonstrate. You will find the sample code below in the following location : *doc/examples/more-complex-model/index.html*.

```
1  <!DOCTYPE html>
2  <head>
3    <meta charset="utf-8">
4    <meta http-equiv="X-UA-Compatible" content="IE=edge">
5    <title>Getting Started</title>
6    <link rel="stylesheet" href="../../../dist/css/ganttchart.css">
7  </head>
8  <body>
9  <div id="gantt-reader" style="width: 500px; height: 500px" />
10 <script src="js/jquery-2.1.3.js"></script>
11 <script src="../../../dist/timetraveljs-gantt-reader.min.js"></script>
12 <script>
13   var myGanttReader = new tt.ganttchart.GanttReader(document.getElementById('gantt-reader'));
14   $.ajax('more-complex-model.json').done(function(data) {
15     myGanttReader.ganttModel = data;
16   });
17 </script>
18 </body>
19 </html>
```

Here are described the significant lines of code :

N°	Description
6	Import Gantt Reader **css** file.
9	Define a **container** for our Gantt Reader component.
10	**Import jQuery** to easily do the ajax call. The Gantt Reader library itself does not rely on jQuery to work.
11	**Import** the minified version of the **library**.
13	Create a **new instance** of Gantt Reader to be included in the `<div>` with `id` equals to `gantt-reader`.
14 - 16	Make an ajax call to load the file *more-complex-model.json* and set the return JSON object to the Gantt Reader component.

If you launch the *index.html* page you will get this result :

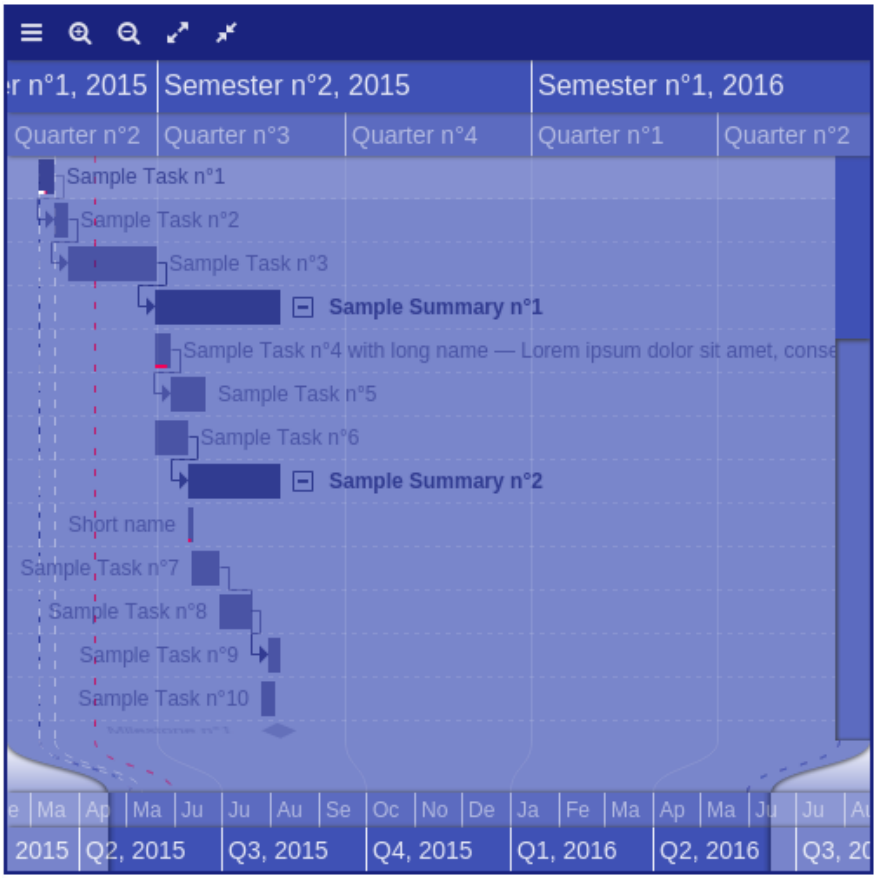

Figure 1.2: Result of a more complex model

Try to browse the model, hover or collapse summaries.

Below is displayed the start of the JSON file. You can take a deeper look at it at this location: *doc/examples/more-complex-model/more-complex-model.json*.

```
{
  "name": "model1",
  "temporalData": [
    {
      "type": "Task",
      "uid": 1,
      "name": "Sample Task n°1",
      "start": "2015-05-04",
      "finish": "2015-05-10",
      "percentComplete": 50
    }, {
      "type": "Task",
      "uid": 2,
      "name": "Sample Task n°2",
      "start": "2015-05-11",
      "finish": "2015-05-17",
      "predecessorLinks": [
        {
          "predecessorUID": 1
        }
      ]
    }, {
      "type": "Task",
      "uid": 3,
      "name": "Sample Task n°3",
```

```
      "start": "2015-05-18",
      "finish": "2015-06-30",
      "predecessorLinks": [
        {
          "predecessorUID": 2
        }
      ]
    }, {
      "type": "Summary",
      "uid": 1000,
      "name": "Sample Summary n°1",
      "temporalData": [
        {
          "type": "Task",
          "uid": 100,
          "name": "Sample Task n°4 with long name   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
               tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation
               ullamco laboris nisi ut aliquip ex ea commodo consequat.",
          "start": "2015-06-30",
          "finish": "2015-07-07",
          "percentComplete": 75
        }

        ...
```

Not so complex, isn't it ?

After a little paragraph about the story of the component, you will discover **all the features** included in the library.

Chapter 2

Library presentation

2.1 Genesis

For one of my clients, I created with **Mathurin Body**, a great colleague, several time based components. Among these, there was a Gantt Chart. We created these components with the **Adobe Flex** technology. We enjoyed to develop these applications with these tools and the client was very satisfied of the brought solutions.

Sadly, Adobe decided not to support the Flex SDK anymore and gave the project to the Apache Foundation. In the same time, Apple and Microsoft decided not to accept plugins like **Flash** in several of their products anymore. That's why, despite the power of Flash Technologies, I decided to focus on historical web solutions, that is to say the **HTML/JavaScript** couple. These technologies have some resemblances with the **MXML/ActionScript** couple but there is still some work to do to fill the gap.

The port of the Flex library was done, not without difficulty. I had to deal with different JavaScript/Dom engines and at different versions for each of them, anyway this is web development tradition, but the result is here : **a nice, localized, theme-able, mobile compatible Gantt Reader**.

2.2 Technical point of view

The HTML technology I used for drawing most of the parts of the component is **Canvas**, that's why you will not be able to use it on old browsers like **IE8**. I did not code it directly in JavaScript but in **CoffeeScript** which improves the code readability a lot, but also gives an easy way to handle **classes in the prototype** world of JavaScript. Don't worry, the transpiled JavaScript code is still very **readable** and with the bonus to be **optimized** for free.

The tasks list has also been optimized so that you can easily set a model with more than **1000 tasks**. Only the visible tasks are rendered.

Now lets discover the component's features.

Chapter 3

Features

3.1 Controls

3.1.1 Horizontal menu bar:

Figure 3.1: Horizontal menu bar

- **Menu** button : opens a **vertical menu** with some actions like **configuration**, **locale management** ... (described later)
- **Horizontal** period **zoom in/out** buttons : allows you to zoom in/out on different ranges of periods.
- **Vertical** temporal data **zoom in/out** buttons: allows you to select different **level of details** for the displayed temporal data. Today, 3 levels are supported :
  - Compact : **basic** display,

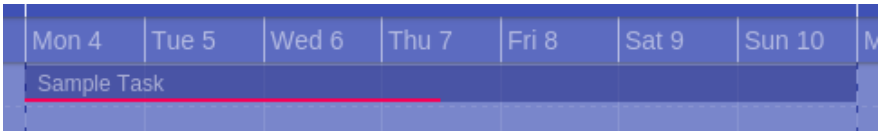

Figure 3.2: Compact mode

  - Normal : displays **date boundaries**,

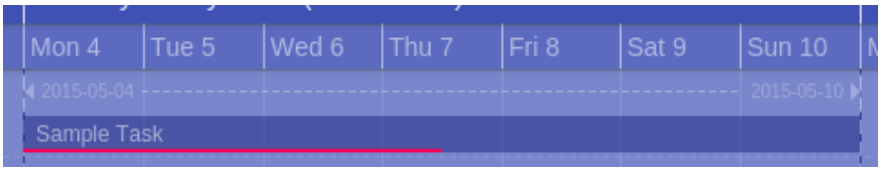

Figure 3.3: Normal mode

– Extended : same as Normal but with **larger font and graphics**.

Figure 3.4: Extended mode

3.1.2 Vertical menu bar:

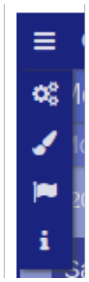

Figure 3.5: Vertical menu bar

3.1.2.1 Configuration panel

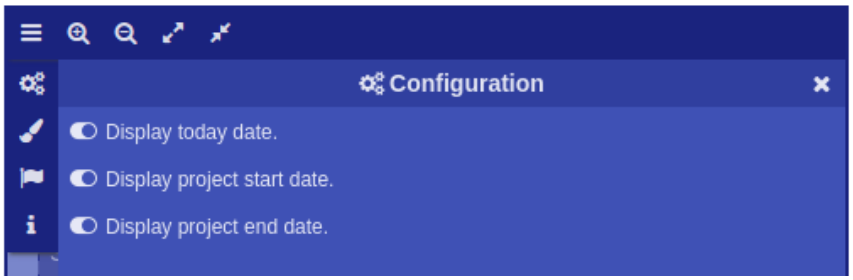

Figure 3.6: Configuration panel

- Display **today date** : enable the display of the tick of the today date on the tasks list background,
- Display **project start date** : enable the display of the tick of the project start date on the tasks list background,
- Display **project end date** : enable the display of the tick of the project finish date on the tasks list background.

3.1.2.2 Theme chooser panel

Themes are made of **2 set of shades**, one for **primary colors** and the other for **accent colors**. I pick up these colors from **Google Material Design recommendations**.

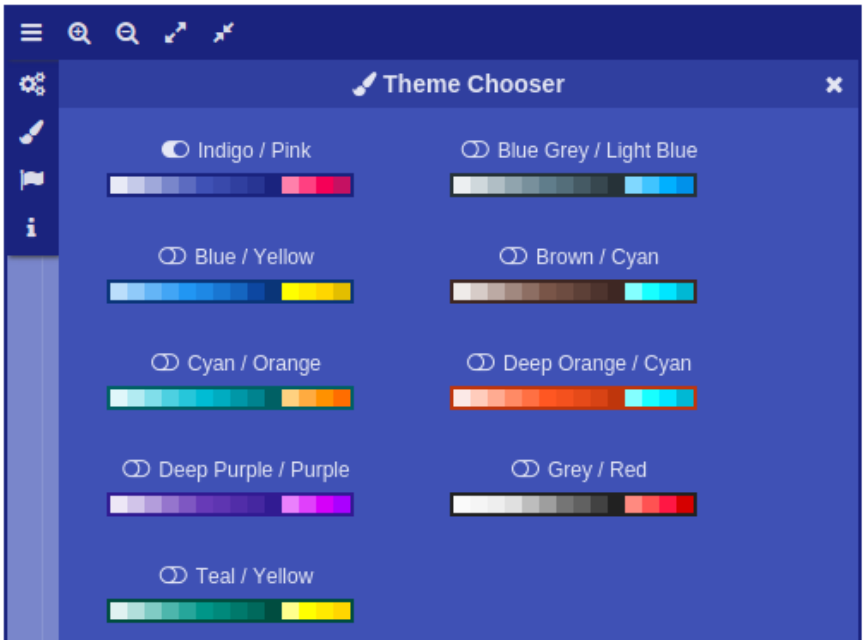

Figure 3.7: Theme chooser panel

Here are all the **available themes** :

Name	Primary color shades	Accent color shades
`indigo-pink` (*default*)	Indigo	Pink
`bluegrey-lightblue`	Blue grey	Light blue
`blue-yellow`	Blue	Yellow
`brown-cyan`	Brown	Cyan
`cyan-orange`	Cyan	Orange
`deeporange-cyan`	Deep Orange	Cyan
`deeppurple-purple`	Deep Purple	Purple
`grey-red`	Grey	Red
`teal-yellow`	Teal	Yellow

3.1.2.3 Language chooser panel

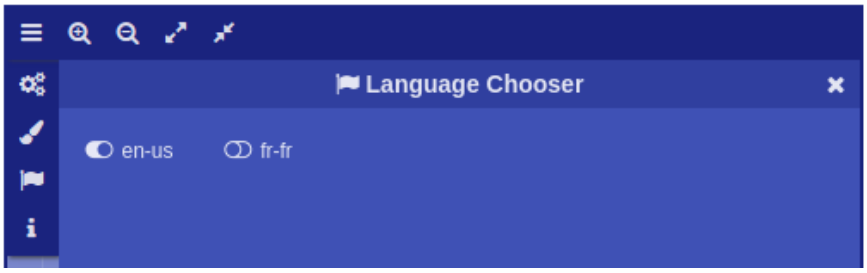

Figure 3.8: Locale chooser panel

In this panel you can **choose the language** used in the component. For the time being two locales have been specified:

- English-USA
- French-France

3.1.3 Top timeline :

Figure 3.9: Top timeline

- Scroll zoom in/out : use the **mouse wheel** on the timeline to **zoom in/out** on the displayed period.
- Drag move period : drag and move on the timeline to **slide** the current displayed period.
- On click period selection: click on one displayed interval to **zoom into it**.
- On timeline **shift click**: shift click on the timeline to **zoom out**.

3.1.4 Bottom timeline period resizer :

Figure 3.10: Bottom timeline

- Scroll zoom in/out : The **center** of this timeline corresponds to the **current displayed period** of time. Use the mouse wheel on the timeline to zoom in/out on the displayed period.

- Resize period by the right/left: the **lighter sides** are here to choose the boundaries of the currently displayed period. Drag and move these sides to change the boundaries.
- Drag move displayed period: drag and move the **center interval to slide the current displayed period** without changing its duration.
- On **click** period selection: as the top timeline you can choose to quickly **jump into a clicked interval of time**.
- On timeline **shift click**: shift click on the timeline to **zoom out**.

3.1.5 Task list viewport:

- **Drag move scrollbar**: drag and move up/down the scrollbar to **browse the temporal data**. The use of the scrollbar **is preferred on low profile configuration** such as cheap smartphones because the tasks are less likely to be redrawn. **The horizontal scrolling of the tasks implies the full redraw of them** which is **CPU intensive**. So use the scrollbar whenever time displacement is not necessary to **prevent lagging on low profile configurations**.
- Scroll up/down list: use the **mouse wheel** on the tasks list to **browse the temporal data**.
- Drag move tasks list up/down/left/right: drag and move onto the tasks list to **browse the temporal data and/or slide the current period**.
- Summary collapsing: **click** on a summary to **collapse/expand** it.

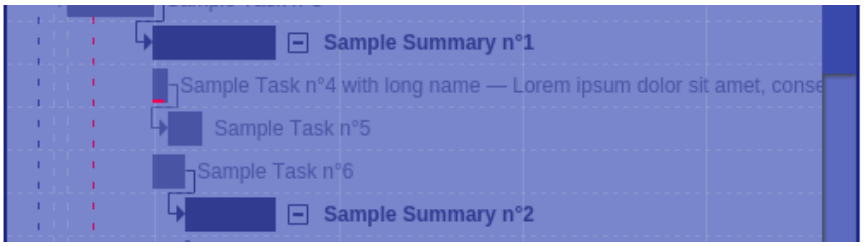

Figure 3.11: Summary collapsing

3.2 Visual effects

- **High Density Screen support**: on high density pixel screens, the resolution of the canvas is increased to prevent blurry issues.
- Icons from Fontello/Font Awesome: I build a set of **font icons** from the awesome Fontello website.
- Timelines:
  - **Period change animation** using TweenJS by the CreateJS team.
  - **Selectable period hover animation**.
- Time bender: **bends tick of the top timeline to the bottom timeline**.
- Panels:
  - Animate open/close action.

- Buttons:
  - Hover animation.
- Colors:
  - **Luminance detection** to determine the **best foreground colors** in function of the selected theme.
- Temporal data list:
  - Summary **hover animation** and all the children.
  - Task / Milestone hover animation.
  - Task **percent complete animation**.
  - Temporal data hover period **feedback animation**.
  - Bottom temporal data **perspective/rotation/opacity**: when the last displayed temporal data is about to disappear a rotation effect is applied. (this only works on modern browsers)

3.3 Browser support

- Firefox
- Opera
- Chrome
- IE >= 9
- Chrome and Firefox mobile touch support.

Chapter 4

Developer guide

4.1 Package content

Minified files are excluded for readability.

```
gantt-reader                                    // root folder
|-- dist                                        // library content folder
|   |-- css
|   |   `-- ganttchart.css                      // css to always import
|   |-- image
|   |   `-- gantt-reader.svg                    // branding image
|   |
|   |-- locale                                  // locales folder
|   |   `-- fr-fr.js
|   |-- theme                                   // theme folder
|   |   |-- BluegreyLightblue.js
|   |   |-- BlueYellow.js
|   |   |-- BrownCyan.js
|   |   |-- CyanOrange.js
|   |   |-- DeeporangeCyan.js
|   |   |-- DeeppurplePurple.js
|   |   |-- GreyRed.js
|   |   `-- TealYellow.js
|   |-- timetraveljs-gantt-reader-all-included.js  // import this file if you want to all themes and
|   |                                           // locales included
|   |
|   |
|   |-- timetraveljs-gantt-reader.js            // import this file to only load default theme and
|   |                                           // locale
|   |
|   |
|   |-- timetraveljs-gantt-reader-with-locales.js  // import this file to get all locales included
|   |
|   |
|   `-- timetraveljs-gantt-reader-with-themes.js   // import this file to get all themes included
|
|
|-- demo                                        // demo folder
|-- doc
|   `-- examples                                // documentation examples
|
|-- CHANGELOG.txt                               // history of the versions
|
`-- README.txt                                  // to read at first
```

4.2 Gantt Model

Here is an example of Gantt model:

```
{
  "name": "model1",
  "temporalData": [  // list of temporal date nodes
    {
      "type": "Task",                // Task definition
      "uid": 1,
      "name": "Sample Task n°1",
      "start": "2015-05-04",
      "finish": "2015-05-10",
      "percentComplete": 50          // Percent complete definition
    }, {
      "type": "Task",                // Task definition
      "name": "Sample Task n°2",
      "start": "2015-05-18",
      "finish": "2015-06-30",
      "predecessorLinks": [
        {
          "predecessorUID": 1       // Predecessor definition
        }
      ]
    }, {
      "type": "Summary",           // Summary definition
      "name": "Sample Summary n°1",
      "temporalData": [
        {
          "type": "Task",
          "uid": 100,
          "name": "Sample Task n°4 with long name   Lorem ipsum dolor si...",
          "start": "2015-06-30",
          "finish": "2015-07-07",
          "percentComplete": 75
        }, {
          "type": "Task",
          "name": "Sample Task n°5",
          "start": "2015-07-08",
          "finish": "2015-07-24",
          "predecessorLinks": [
            {
              "predecessorUID": 100
            }
          ]
        }, {
          "type": "Summary",             // Summary inside a Summary
          "name": "Sample Summary n°2",
          "temporalData": [
            {
              "type": "Task",
              "name": "Short name",
              "start": "2015-07-16",
              "finish": "2015-07-18",
              "percentComplete": 50
            }
          ]
        }
      ]
    }
  ]
}
```

Now, lets see the format specification.

4.2.1 Root node

Field	Type	Description
name	`string`	the name of the project (*for future use*).
temporalData	`array`	(*optional*) the first level of temporal data to display, they can be of type `Task`, `Summary` or `Milestone`.

Here is a sample root node representation:

```
{
  "name": "model1",
  "temporalData": [
    {
      "type": "Task"
      ...
    }, {
      "type": "Milestone"
      ...
    }
  ]
}
```

4.2.2 Task

Field	Type	Description
uid	`number`	(*optional*) the **unique identifier** of the task. Only useful if you need to **reference the task** elsewhere in the document. For example when using the task as the **predecessor** of another temporal data.
name	`string`	the name of the task.
predecessorLinks	`array`	(*optional*) an array of the predecessors of the task. The elements can be of any temporal data type.
start	`string`	the start date of the task. It must respect the format `YYYY-MM-DD`.
finish	`string`	the finish date of the task. It must respect the format `YYYY-MM-DD`. **Important to notice**: this will the end of day of this date that will be taken for display. For example: '2015-10-30' will become 2015-10-30 + 23h 59m 59s 999ms.
percentComplete	`number`	(*optional*) the percentage of accomplished work.

Here is a sample `Task` type node representation:

```
{
  "type": "Task",
  "uid": 1,
  "name": "Sample Task n°1",
  "start": "2015-05-04",
  "finish": "2015-05-10",
  "percentComplete": 50
}
```

4.2.3 Summary

Field	Type	Description
uid	`number`	(*optional*) the **unique identifier** of the summary. Only useful if you need to **reference the summary** elsewhere in the document. For example when using the summary as the **predecessor** of another temporal data.
name	`string`	the name of the summary
predecessorLinks	`array`	(*optional*) an array of the predecessors of the summary. The elements can be of any temporal data type.

Field	Type	Description
temporalData	`array`	*(optional)* an array of temporal data of any type to be included in the summary. Of course, a **summary can contain another summary** to construct a tree of task.

Important to notice: the start and finish date of the summary are computed from its underlying content.

A sample of `Summary` node:

```
{
  "type": "Summary",
  "name": "Sample Summary n°1",
  "temporalData": [
    {
      "type": "Task",
      "name": "Sample Task n°4",
      "start": "2015-06-30",
      "finish": "2015-07-07",
      "percentComplete": 75
    }, {
      "type": "Task",
      "name": "Sample Task n°5",
      "start": "2015-07-08",
      "finish": "2015-07-24",
      ]
    }
  ]
}
```

4.2.4 Milestone

Field	Type	Description
uid	`number`	*(optional)* the **unique identifier** of the milestone. Only useful if you need to **reference the milestone** elsewhere in the document. For example when using the milestone as the **predecessor** of another temporal data.
name	`string`	the name of the milestone
predecessorLinks	`array`	*(optional)* an array of the predecessors of the milestone. It can be any type of temporal data.
start	`string`	The date of the milestone. It must respect the format `YYYY-MM-DD`. **Important to notice**: this will the end of day of this date that will be taken for display. For example: '2015-10-30' will become 2015-10-30 + 23h 59m 59s 999ms.

A sample of `Milestone` node:

```
{
  "type": "Milestone",
  "uid": 10001,
  "name": "Milestone n°1",
  "start": "2015-08-28"
}
```

4.2.5 Predecessor Link

Field	Type	Description
predecessorUID	`number`	the uid of the temporal data that precedes the temporal data owning this node.

An example of `predecessorLinks` use:

```
{
  "type": "Task",
  "uid": 49,
  "name": "Sample Task n°7",
  "start": "2015-07-18",
  "finish": "2015-07-31"
}, {
  "type": "Task",
  "uid": 50,
  "name": "Sample Task n°8",
  "start": "2015-08-01",
  "finish": "2015-08-15"
}, {
  "type": "Task",
  "uid": 51,
  "name": "Sample Task n°9",
  "start": "2015-08-25",
  "finish": "2015-08-30",
  "predecessorLinks": [
    {
      "predecessorUID": 49
    }, {
      "predecessorUID": 50
    }
  ]
}
```

4.3 Import theme

To import one theme, simply import the corresponding JavaScript file after the file `timetraveljs-gantt-reader[.min].js`. For example, to import the *Blue - Yellow* and the *Cyan - Orange* themes, you can type (see *doc/examples/import-theme/index.html*):

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Getting Started</title>
  <link rel="stylesheet" href="../../../dist/css/ganttchart.css">
</head>
<body>
<div id="gantt-reader" style="width: 500px; height: 500px" />
<script src="../../../dist/timetraveljs-gantt-reader.min.js"></script>
<script src="../../../dist/theme/BlueYellow.min.js"></script>
<script src="../../../dist/theme/CyanOrange.min.js"></script>
<script>
  var myGanttReader = new tt.ganttchart.GanttReader(document.getElementById('gantt-reader'));

  // choose the theme
  myGanttReader.theme = 'blue-yellow';
</script>
</body>
</html>
```

Here are the available theme names:

- `indigo-pink` (*default*)
- `bluegrey-lightblue`
- `blue-yellow`
- `brown-cyan`
- `cyan-orange`
- `deeporange-cyan`
- `deeppurple-purple`
- `grey-red`
- `teal-yellow`

If you don't explicitly set the theme, the last imported theme wins.

If you want all the themes to be included, you must import either `timetraveljs-gantt-reader-with-themes[.min].js` or `timetraveljs-gantt-reader-all-included[.min].js` which also includes all the languages.

4.4 Import locale

In the same manner as the previous title, to import one language, simply import the corresponding JavaScript file after the file `timetraveljs-gantt-reader[.min].js`. For example, to import the *fr-fr* theme, you can type (see *doc/examples/import-locale/index.html*):

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Getting Started</title>
  <link rel="stylesheet" href="../../../dist/css/ganttchart.css">
</head>
<body>
<div id="gantt-reader" style="width: 500px; height: 500px" />
<script src="../../../dist/timetraveljs-gantt-reader.min.js"></script>
<script src="../../../dist/locale/fr-fr.min.js"></script>
<script>
  var myGanttReader = new tt.ganttchart.GanttReader(document.getElementById('gantt-reader'));
</script>
</body>
</html>
```

Here are the available locale names:

- `en-us` (*default*)
- `fr-fr`

If you don't explicitly set the locale, the last imported locale wins.

If you want all the locales to be included, you must import either `timetraveljs-gantt-reader-with-locales[.min].js` or `timetraveljs-gantt-reader-all-included[.min].js` which also includes all the themes.

4.5 Gantt Reader API

The `tt.timechart.GanttReader` class expose several public methods. Feel free to use them to adapt the component to your application.

- **Methode name:** `set ganttModel`
- **Attributes:**
  - *string | object* `ganttModel`: a Gantt model in JSON string or JSON object
- **Description:**: this is the most important method of the component. It sets a GanttModel to the component and display it when the next draw call occurs.
- **Example:**

```
var myGanttChart = ...;
// sets a JSON string to the component
myGanttChart.ganttModel = '  {                              ' +
                          '   "name": "model1",              ' +
                          '   "temporalData": [              ' +
                          '    {                             ' +
                          '      "type": "Task",             ' +
                          '      "name": "Sample Task",      ' +
                          '      "start": "2015-05-04",      ' +
                          '      "finish": "2015-05-10",     ' +
                          '      "percentComplete": 50       ' +
                          '    }                             ' +
                          '   ]                              ' +
                          '  }                               ';
```

- **Methode name:** `set temporalDataRendererFactoryClass`
- **Attributes:**
  - *class* `temporalDataRendererFactoryClass` : the class responsible to render the temporal data in the list.
- **Description:** setter that let you choose the class responsible to render the temporal data in the list. 3 classes are available:
  - `tt.ganttchart.CompactTemporalDataRendererFactory` : for compact display,
  - `tt.ganttchart.NormalTemporalDataRendererFactory` : for normal display (boundaries date are displayed),
  - `tt.ganttchart.ExtendedTemporalDataRendererFactory` : like normal mode but with larger font and graphics.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.temporalDataRendererFactoryClass = tt.ganttchart.CompactTemporalDataRendererFactory;
```

- **Methode name:** `set theme`
- **Attributes:**
  - *string* `theme` : the theme name
- **Description:** allow you to change the theme of the component.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.theme = 'grey-red';
```

- **Methode name:** `set locale`
- **Attributes:**
  - *string* `locale` : the locale name
- **Description:** allow you to change the locale of the component.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.locale = 'fr-fr';
```

- **Methode name:** `set period`
- **Attributes:**
  - *tt.core.Period* `period` : the period to display
- **Description:** allow to change the displayed period. The class `Period` is inside the namespace `tt.core`.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.period = new tt.core.Period(
  // start date
  new Date(2015, 0, 1),
  // end date
  new Date(2015, 11, 31)
);
```

- **Methode name:** `set maxPeriod`
- **Attributes:**

  - *tt.core.Period* `maxPeriod` : the biggest period the component is able to display

- **Description:** allow to change the biggest period the component is able to display. The class `Period` is inside the namespace `tt.core`.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.maxPeriod = new tt.core.Period(
  // start date
  new Date(2015, 0, 1),
  // end date
  new Date(2015, 11, 31)
);
```

- **Methode name:** `zoomModel`
- **Attributes:** *none*
- **Description:** zoom to fit the computed start and end date of the project. A little margin is added.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.zoomModel();
```

- **Methode name:** `zoomToMaxPeriod`
- **Attributes:** *none*
- **Description:** zoom to the biggest period the component is able to display, that is to say the value of `maxPeriod`.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.zoomToMaxPeriod();
```

- **Methode name:** `zoomIn`
- **Attributes:**

  - (*optional*) *number* `percent` (*default* = 0.5): the percentage of zoom from `0` to `1`.

- **Description:** zoom in the current displayed period.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.zoomIn();
```

```
var myGanttChart = ...;
myGanttChart.zoomIn(0.75);
```

- **Methode name:** `zoomOut`
- **Attributes:**
  - (*optional*) *number* `percent` (*default* = *1*): the percentage of zoom from `0` to `1`.
- **Description:** zoom out the current displayed period.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.zoomOut();
```

```
var myGanttChart = ...;
myGanttChart.zoomOut(0.75);
```

- **Methode name:** `verticalZoomIn`
- **Attributes:** (*none*)
- **Description:** zoom in to the next available temporalDataRendererFactoryClass.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.verticalZoomIn();
```

- **Methode name:** `verticalZoomOut`
- **Attributes:** (*none*)
- **Description:** zoom out to the next available temporalDataRendererFactoryClass.
- **Example:**

```
var myGanttChart = ...;
myGanttChart.verticalZoomOut();
```

- **Methode name:** `endIsNow`
- **Attributes:** (*none*)
- **Description:** slide the current period until its start date equals to now
- **Example:**

```
1 var myGanttChart = ...;
2 myGanttChart.endIsNow();
```

- **Methode name:** `endToStart`
- **Attributes:** (*none*)
- **Description:** slide, if possible, the current period `p1` to the next period `p2`, that is to say until `p2.start == p1.end`, `p1` and `p2` having the same duration.
- **Example:**

```
1 var myGanttChart = ...;
2 myGanttChart.endToStart();
```

- **Methode name:** `displayToday`
- **Attributes:**
  - *boolean* `display` : enable today tick date display.
- **Example:**

```
1 var myGanttChart = ...;
2 myGanttChart.displayToday();
```

- **Methode name:** `displayStart`
- **Attributes:**
  - *boolean* `display` : enable project start tick date display.
- **Example:**

```
1 var myGanttChart = ...;
2 myGanttChart.displayStart();
```

- **Methode name:** `displayFinish`
- **Attributes:**
  - *boolean* `display` : enable project end tick date display.
- **Example:**

```
1 var myGanttChart = ...;
2 myGanttChart.displayFinish();
```
