Gantt Chart Library Documentation

Introduction

Welcome to the documentation for Gantt Chart Library. This library provides a powerful set of tools and functionalities to create interactive Gantt charts for project management. This documentation will guide you through the installation process, usage instructions, and available features of the library.

Table of Contents

1. Installation

To use the Gantt Chart Library, follow these steps:

1. Download the library files from our website or GitHub repository.
2. Include the library files (**`gantt.js`** and **`gantt.css`**) in your project.
3. Add the necessary dependencies, such as **jsPDF (V. `2.5.1`)** and **html2canvas (V. `1.4.1`)** if you are using the export for the pdf or PNG.
4. Link the library files in your HTML file.
5. You're now ready to start using the Gantt Chart Library!

2. Getting Started

To create a basic Gantt chart, follow these steps:

1. Create a container element in your HTML where you want to render the chart.

```
<div id="gantt_here"></div>
```

2. Initialize the Gantt Chart Library using JavaScript by targeting the container element.

```
let element = document.getElementById("gantt_here");
let gantt = new ztGantt(element);
```

3. Define your tasks, their durations, and any dependencies.
4. Render the chart by calling the appropriate function.

```
gantt.render();
```

5. Customize the chart's appearance and behavior as needed.

3. Chart Configuration

The Gantt Chart Library provides various configuration options to tailor the chart to your specific requirements. You can customize the chart's timeline, task bars, labels, colors, and more.

- ## Options

1. gantt.options.date_format

Format of the date you passed in data

```
gantt.options.date_format = "%m-%d-%Y";
```

2. gantt.options.localLang

Set the language as per your requirements
By default, language is English

```
gantt.options.localLang = "en";
```

3. gantt.options.startDate and gantt.options.endDate

startDate and endDate are date of the gantt range

```
gantt.options.startDate = "2023-05-01T11:46:17.775Z";
gantt.options.endDate = "2023-06-10T11:46:17.775Z";
```

4. gantt.options.columns

Columns of the left side grid

```
gantt.options.columns = [
        {name: "text",width: 245,
          min_width: 80,
          max_width: 300,
          tree: true,
          label: "Name",
          resize: true,
          template: (task) => {
            return `<span>${
              task.parent == 0 ? task.text : task.subject
            }</span>`;
          },
        },
        {name: "estimated_hours",width: 100, min_width: 80,
          tree: false,
          align: "center",
          label: "Planned Hour",
          resize: true,
          template: (task) => {
            return `<span>${task.estimated_hours || ""} </span>`;
          },
        },
      ];
```

name is name of the column present in the data,
width is the actual width of that column,
min_width is the minimum width of that column,
max_width is the maximum width of that column,
tree is the Boolean for that column should become tree or not,
label is the header label of that column,
resize is for the column should be resizable or not,
template is a function that return the html string which will displayed in column row

5. gantt.options.taskColor

To enable custom color box for all tasks, by default it is false
It can be a function or Boolean type

```
gantt.options.taskColor = true;
Or
gantt.options.taskColor = (task)=> {
        if (task.parent === 0) {
          return false;
        }
        return true;
      };
```

6. **gantt.taskOpacity**

You can customize the opacity of the task color with the help of task opacity

```
gantt.taskOpacity = 0.7;
```

7. **gantt.options.data**

   The task data of projects

You can create the nested task by assigning parent the child task parent will be the parent task id.

```
gantt.options.data = [
{"id":1, "text":"Project 1", parent: 0, progress: 50, taskColor: "#56a4fdf2"},
{"id":2, "text":"Task #1", "start_date":"05-05-2023", "end_date": "05-05-
2023","parent":1, progress: 60, taskColor: "#56a4fdf2"},
{"id":3, "text":"Task #2", "start_date":"05-05-2023", "end_date": "05-05-
2023","parent":1, progress: 30, taskColor: "#56a4fdf2"},
{"id":5, "text":"SubTask #1", "start_date":"05-05-2023", "end_date": "05-05-
2023","parent":3, progress: 10, taskColor: "#56a4fdf2"},
{"id":6, "text":"SubTask #2", "start_date":"05-05-2023", "end_date": "05-05-
2023","parent":3, progress: 80, taskColor: "#56a4fdf2"},
{id: 12, text: "Final Milestone", start_date: "06-17-2023", end_date: "06-17-
2023", parent: 8, type:"milestone", taskColor: "#56a4fdf2"},

{"id":7, "text":"SubTask #3", "start_date":"05-05-2023", "end_date": "05-05-
2023","parent":3, progress: 45, taskColor: "#56a4fdf2"},
{"id":4, "text":"Task #3","parent":1, progress: 15, taskColor: "#56a4fdf2"},
{"id":8, "text":"Project 2", "parent":0, progress: 55, taskColor: "#56a4fdf2"},
{"id":9, "text":"Project 3", "parent":0, progress: 65, taskColor: "#56a4fdf2"},
{"id":10, "text":"Project 4", "parent":0, progress: 75, taskColor:
"#56a4fdf2"},
{"id":11, "text":"Project 5", progress: 100, taskColor: "#56a4fdf2"},
{id: 13, text: "Next Milestone", start_date: "06-17-2023", end_date: "06-17-
2023", parent: 8, type:"milestone", taskColor: "#56a4fdf2"},

]
```

id is the task id,
text is the text for displaying text of task,
parent is the id of the parent task, if task is at top level then parent is 0,
start_date is the start date of the task,
end_date is the end date of the task,
progress is the percentage of task completion,
type is the type of the task, it can be **milestone** or **task**
taskColor is the color of the taskbar

8. **gantt.options.taskProgress**

To enable progress in taskbar
type: Boolean, byDefault it is false

```
gantt.options.taskProgress = true;
```

9. **gantt.options.rightGrid**

Columns for the right-side grid, it is optional.

```
gantt.options.rightGrid = [
        {
          name: "estimated_hours",
          label: "Total",
          width: 100,
          align: "center",
          resize: true,
          template: function (task) {
            var totalHours = 0;
            return `<b>${task.estimated_hours}</b>`;
          },
        },
        {
          name: "Stats",
          width: 100,
          label: "Stats",
          align: "center",
          resize: true,
          template: function (task) {
            return `<b>${task.estimated_hours}</b>`;
          },
        },
      ];
```

10. **gantt.options.scales**

Scales array of the timeline

```
gantt.options.scales= [
    {
        unit: "week",
        step: 1,
        format: (t) => {
        return "%d %F";
        },
    },
    {
        unit: "day", step: 1, format: "%d %D"
    },
];
```

unit is the unit of the scale column in which format you want the scale,
Here are 6 types of units: -
1. hour,
2. day,
3. week,
4. month,
5. quarter,
6. year

step is the number of steps you want to include in the column,
format it can be a string or function which return the string which is the format of the date

11. gantt.options.zoomLevel

To change the zoom level of the gantt timeline

```
gantt.options.zoomLevel = "day";
```

Here are 6 levels
1. hour
2. day
3. week
4. month
5. quarter
6. year

12. gantt.options.zoomConfig

The configuration for different levels of zoom
name is name of zoom level
scale_hight is hight of scale for the zoom level
min_col_width is min column width for the zoom level
scales is the scales for the zoom level

After setting zoomLevel and zoomConfig run gantt.zoomInit() for applying your current zoom level

```
gantt.options.zoomConfig = {
        levels: [
          {
            name: "day",
            scale_height: 27,
            min_col_width:80,
            scales: [{ unit: "day", step: 1, format: "%d %M" }],
          },
          {
            name: "week",
            scale_height: 50,
            min_col_width:50,
            scales: [
              {
                unit: "week",
                step: 1,
                format: function (date) {
                  var dateToStr = gantt.formatDateToString("%d %M");
                  var endDate = gantt.add(date, 6, "day");
                  var weekNum = gantt.formatDateToString("%W",date);
                  return (
                    "#" +
                    weekNum +
                    ", " +
                    gantt.formatDateToString("%d %M",date) +
                    " - " +
                    gantt.formatDateToString("%d %M",endDate)
                  );
                },
```

```
        },
        { unit: "day", step: 1, format: "%j %D" },
      ],
    },
    {
      name: "month",
      scale_height: 50,
      min_col_width:120,
      scales: [
        { unit: "month", format: "%F, %Y" },
        { unit: "week", format: "Week #%W" },
      ],
    },
    {
      name: "quarter",
      scale_height: 50,
      min_col_width:90,
      scales: [
        { unit: "month", format: "%F, %Y" },
        { unit: "week", format: "Week #%W" },
      ],
    },
    {
      name: "year",
      scale_height: 50,
      min_col_width: 30,
      scales: [{ unit: "year", step: 1, format: "%Y" }],
    },
  ],
```

```
    };
```

13. **gantt.options.addLinks**

To show task relation through links in the Gantt

type: Boolean or can be a function which return Boolean, by default it is false.

```
gantt.options.addLinks = true;
Or
gantt.options.addLinks = (task)=> {
        if (task.parent === 0) {
          return false;
        }
        return true;
      };
```

14. **gantt.options.links**

Task relations array

type: Array

```
gantt.options.links = [
        {"id":1, "source":1, "target":2, "type": 0},
        {"id":2, "source":2, "target":3, "type": 1},
        {"id":3, "source":3, "target":4, "type": 2},
        {"id":4, "source":12, "target":15, "type": 3}
    ]
```

source is source id

target is target id

type is type of link

Links types can be of 4 types

0 is **finish_to_start**

1 is **start_to_start**

2 is **finish_to_finish**

3 is **start_to_finish**

By default, the link type is **0**

15. **gantt.options.collapse**

To make the tree initially collapse or open

type: Boolean, by default it is true.

```
gantt.options.collapse = false;
```

16. **gantt.options.fullWeek**

Show the full week or workdays.

type: Boolean, by default it is true.

```
gantt.options.fullWeek = true;
```

17. **gantt.options.todayMarker**

It adds a vertical marker at today's date column.
type: Boolean, by default it is true.

```
gantt.options.todayMarker = false;
```

18. **gantt.options.weekends**

type: Array, array of strings "Sat", "Sun", to set the weekends dynamically.

```
gantt.options.weekEnds = ["Sat", "Sun"];
```

19. **gantt.options.weekStart**

You can pass from 0 to 6.
type: Number, it set the start of the week, by default it is set to 1 means "Monday".

```
gantt.options.weekStart = 0;
```

20. **gantt.options.scale_height**

To set the height of the scale. You can pass the number value which will apply to all scales or you can pass the Array for different height for different scales respectively.
type: Number or Array, 30 || [20, 30], set the height of scales, by default, it is 30.

```
gantt.options.scale_height = [20, 30];
```

21. **gantt.options.row_height**

To set the height of the row.
type: Number, by default it is 50.

```
gantt.options.row_height = 60;
```

22. **gantt.options.addTaskOnDrag**

For selecting task start and end through drag
type: Boolean, by default it is false.

```
gantt.options.addTaskOnDrag = true;
```

- # Templates

1. **gantt.templates.tooltip_text**

   It is a function template which return the html for the tooltip

```
gantt.templates.tooltip_text = function (start, end, task) {
        return `<b>${task.parent === 0 ? "User" : "Task"}:</b>
            ${task.parent === 0 ? task.text : task.subject}
            <br/><b>Start date:</b>${start}
            <br/><b>End date:</b>${end}
            <br/>
            <b>Duration:</b> ${task.duration} ${
          task.duration > 1 ? "Days" : "Day"
        }`;
      };
```

2. **gantt.templates.taskbar_text**

It returns the html for the taskbars

```
gantt.templates.taskbar_text = function (start, end, task) {
        if (task.parent == 0) {
          return `User : ${task.text}`;
        } else {
          return `Task : ${task.subject}`;
        }
      };
```

3. **gantt.templates.task_drag**

It returns true or false, for allowing the task drag

```
gantt.templates.task_drag = (mode, task) => {
        if (task.parent == 0 || (task.children && task.children.length > 0)) {
          return false;
        }
        return true;
      };
```

4. **gantt.templates.grid_folder**

It returns the grid folder html

```
gantt.templates.grid_folder = (task) => {
        return `<div class="folder-class">Folder</div>`;
      };
```

5. **gantt.templates.grid_file**

It returns the grid file html

```
gantt.templates.grid_file = (task) => {
        return `<div class="file-class">File</div>`;
      };
```

6. **gantt.templates.grid_header_class**

It returns the class for the left grid header
You can add multiple classes by separating them with space.

```
gantt.templates.grid_header_class = (columns,index) => {
        return "my-header-class header-class"
      }
```

7. gantt.templates.grid_row_class

It returns the class for the row of left grid
You can add multiple classes by separating them with space.

```
gantt.templates.grid_row_class = (start, end, task) => {
        return "my-grid-row-class"
      }
```

8. **gantt.templates.task_class**

   It returns classes for task

```
gantt.templates.task_class = (start, end, task) => {
        return "my-task-class"
      }
```

9. **gantt.templates.task_row_class**

   It returns classes for the task row

```
gantt.templates.task_row_class = (start, end, task) => {
        return "my-task-row-class"
      }
```

10. **gantt.templates.scale_cell_class**

    It returns classes for the timeline scales

```
gantt.templates.scale_cell_class = (date, scale, scaleIndex) => {
        return "my-scale-class"
      }
```

11. gantt.templates.**grid_cell_class**

It returns classes for the left grid cell

```
gantt.templates.grid_cell_class = (col, task) => {
        return "my-grid-cell-class"
      }
```

12. **gantt.templates.timeline_cell_class**
    It returns classes for the timeline cell

```
gantt.templates.timeline_cell_class = (task, date) => {
        return "my-task-cell-class"
      }
```

13. **gantt.templates.showLightBox**
    It returns the html for the popup modal open on dblclick

```
gantt.templates.showLightBox = (task)=>{
return `<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>`
}
```

4. Methods

1. Format Date to String

```
gantt.formatDateToString(format, date);
```

formateDateToString is for formatting date in required format
Here format is the format in which we want the date and ***date*** is the date which we want to formate.

While specifying the format for dates you can use any character from the following list:

- **%y** - the year as a two-digit number (00 *to 99)*;
- **%Y** - the year as a four-digit number (1900-*9999)*;
-
- **%m** - the month as a number with a leading zero (01 *to 12)*;
- **%n** - the month as a number without a leading zero (1 *to 12)*;
- **%M** - the month as an abbreviation (Jan *to Dec)*;
- **%F** - the month as a full name (January *to December)*;

- **%W** - the week number of the year. Weeks start on Monday;

- **%d** - the day as a number with a leading zero (01 *to 31)*;
- **%j** - the day as a number without a leading zero (1 *to 31)*;

- **%D** - the day as an abbreviation (Sun *to Sat)*;
- **%l** - the day as a full name (Sunday *to Saturday)*;

- **%h** - the hour based on the 12-hour clock (00 *to 11)*;
- **%H** - the hour based on the 24-hour clock (00 *to 23)*;
- **%g** - the hour based on the 12-hour clock without a leading zero (1 *to 12)*;
- **%G** - the hour based on the 24-hour clock without a leading zero (0 *to 23)*;

- **%i** - the minute as a number with a leading zero (00 *to 59)*;
- **%s** - the second as a number with a leading zero (00 *to 59)*;
- **%a** - displays **am** (for times from midnight until noon) and **pm** (for times from noon until midnight);
- **%A** - displays **AM** (for times from midnight until noon) and **PM** (for times from noon until midnight).

For example, if you want 20th June 2023 as 20/06/2023, you should specify "%d/%m/%Y".

2. Get the current zoom level scale config

Function to get the current level zoom scale

```
gantt.getScale();
```

3. Set Language

To set or change the language

```
gantt.setLocalLang("hi");
```

4. Initialize zoom level

To initialize the zoom level, call this method

```
gantt.zoomInit();
```

5. Add days, week, month, year, hour, and minute to a date

```
gantt.add(date, number, unit)
```

- ***date*** - (*Date*) the date object that you need to add a time to/subtract a time from
- ***number*** - (*number*) the number of units to add. If this number is positive - the time will be added to the date, if negative - the time will be subtracted
- ***unit*** - (*string*) the time unit. Values: 'minute', 'hour', 'day', 'week', 'month', 'year'.

6. Request FullScreen

for view gantt in fullscreen

```
gantt.requestFullScreen();
```

7. Exit FullScreen

for exiting fullScreen

```
gantt.exitFullScreen();
```

8. Expand All

for expanding all tasks

```
gantt.expandAll();
```

9. Collapse All

for collapsing all tasks

```
gantt.collapseAll();
```

10. Get Task

for getting task by id

```
gantt.getTask(id)
```

11. Filter Tasks

```
gantt.filterTask(condition, isFilter);
```

condition is a function in which return the condition of the filter
isFilter is type of `Boolean` which state that filter should apply or not.

```
gantt.filterTask((task) => {
          if (task.parent === 0) {
            return task.text
              .toLowerCase()
              .includes("string".toLowerCase());
          } else {
            return task.subject
              .toLowerCase()
              .includes("string".toLowerCase());
          }
        }, true);
      }
```

12.Add Custom Marker

```
let marker = {
    start_date: gantt.add(new Date(), 1, "day"), //a Date object that sets the
marker's date
    css: "tomorrow", //a CSS class applied to the marker
    text: "Tomorrow", //the marker title
    title: gantt.formatDateToString("%d %M %y",gantt.add(new Date(), 1,"day")),
}

gantt.addMarker(marker);
```

start_date is the start date of the marker
css is the classes applied to the marker it could be multiple separated by space
text is the text added to the marker
title is the title added to the marker

13.Add Today Marker

```
gantt.addTodayFlag();
```

It adds a today marker to the gantt

14. Remove Today Marker

```
gantt.removeTodayFlag();
```

15. Add Task

```
gantt.addTask(task);
```

here task is the new task to add

16. Update Task

```
gantt.updateTaskData(task);
```

here **task** is the updated task

17. Delete Task

```
gantt.deleteTask(id);
```

here id is the id of the task

18. Loop through all tasks

It iterates through all tasks

```
gantt.eachTask((task)=>{
console.log(task);
})
```

19. Open Task

Used for opening a perticular task tree

```
gantt.openTask(id);
```

20. Add Data to existing data

```
gantt.parse(data);
```

using this you can add new data to the existing data in gantt
here **data** is array which contain new data

21. Get the position of the date cell from timeline start

```
gantt.posFromDate(date);
```

22. Clear All

It clears the old data

use it only when you change the data from your side.

```
gantt.clearAll();
```

23. Delete Link

function to delete a link

```
gantt.deleteLink(id);
```

24. Export To PNG

for exporting gantt to PNG

```
gantt.exportToPNG(name);
```

here name is for the exported file name, it's optional

25. Export To PDF

for exporting gantt to PDF

```
gantt.exportToPDF(name);
```

here name is for the exported file name, it's optional

26. Export To Excel

for exporting gantt to Excel

```
gantt.exportToExcel(name);
```

here name is for the exported file name, it's optional

27. Render Gantt

```
gantt.render(element);
```

Here element is the div where you want to append gantt,
element is optional if you are not calling render function first time in your code;

28. Destroy Gantt

```
gantt.destroy();
```

To destroy the gantt chart

29. Auto Scheduling

```
gantt.autoScheduling();
```

Call this method to automatically schedule your tasks based on the relations between them.

5. Events

1. On Task DblClick

Triggered when Double clicked on the task

```
gantt.attachEvent("onTaskDblClick", (event) => {
        console.log("onTaskDblClick: ", event);
      });
```

2. On Link DblClick

Triggered when Double clicked on the link

```
gantt.attachEvent("onLinkDblClick", (event) => {
        console.log("onLinkDblClick: ", event);
      });
```

3. On Before Link Add

Triggered before the link added

```
gantt.attachEvent("onBeforeLinkAdd", (event) => {
        console.log("onBeforeLinkAdd: ", event);
      });
```

4. On Link Add

Triggered when link added

```
gantt.attachEvent("onLinkAdd", (event) => {
        console.log("onLinkAdd: ", event);
      });
```

5. On Delete Link

Triggered when link deleted

```
gantt.attachEvent("onDeleteLink", (event) => {
        console.log("onDeleteLink: ", event);
      });
```

6. On Before Task Drag

Triggered before the dragging of the task

```
gantt.attachEvent("onBeforeTaskDrag", (event) => {
        console.log("onBeforeTaskDrag: ", event);
        if (event.task.children.length !== 0) {
          return false;
        } else {
          return true;
        }
      });
```

7. On Task Drag

Triggered on the dragging of the task

```
gantt.attachEvent("onTaskDrag", (event) => {
        console.log("onTaskDrag: ", event);
      });
```

8. On Before Task Drop

Triggered before task drop

```
gantt.attachEvent("onBeforeTaskDrop", (event) => {
        console.log("onBeforeTaskDrop: ", event);
        if (event.parentTask.id == 12) {
          return false;
        }
      });
```

9. On After Task Drag

Triggered after the dragging of the task

```
gantt.attachEvent("onAfterTaskDrag", (event) => {
        console.log("onAfterTaskDrag: ", event);
      });
```

10. On Task Delete

Triggered when the task deleted

```
gantt.attachEvent("onTaskDelete", (event) => {
        console.log("onTaskDelete: ", event);
      });
```

11. On After Task Update

Triggered after the task updated

```
gantt.attachEvent("onAfterTaskUpdate", (event) => {
        console.log("onAfterTaskUpdate: ", event);
      });
```

12. On Scroll

Triggered when you scroll gantt

```
gantt.attachEvent("onScroll", (event) => {
        console.log("onScroll: ", event);
      });
```

13. On Resize

Triggered on window resize

```
gantt.attachEvent("onResize", (event) => {
        console.log(" onResize: ", event);
      });
```

14. On Timeline cell click

Triggered when clicked on the timeline cell

```
gantt.attachEvent("onCellClick", (event) => {
        console.log("onCellClick: ", event);
      });
```

15. On Expand

Triggered when requested Fullscreen

```
gantt.attachEvent("onExpand", (event) => {
        console.log("onExpand: ", event);
      });
```

16. On Collapse

Triggered when exited Fullscreen

```
gantt.attachEvent("onCollapse", (event) => {
        console.log("onCollapse: ", event);
      });
```

17. On Color Change

Triggered after task progress drag

```
gantt.attachEvent("onColorChange", (event) => {
        console.log("onColorChange: ", event);
      });
```

18. Add Task on Drag

Select the start & end date of the new Task through Drag

```
gantt.attachEvent("addTaskOnDrag", (event) => {

        gantt.addTask({

          id: 12,

          start_date: event.task.startDate,

          end_date: event.task.endDate,

          parent: event.task.parent,

          text: "Task Added"

        })

        gantt.render();

      });
```

19. On After Progress Drag

Triggered after task progress drag

```
gantt.attachEvent("onAfterProgressDrag", (event) => {
        console.log("onAfterProgressDrag: ", event);
      });
```

20. On Before Progress Drag

Triggered before task progress drag

```
gantt.attachEvent("onBeforeProgressDrag", (event) => {
        console.log("onBeforeProgressDrag: ", event);
        if(event.task.parent === 0){
          return false;
        }else{
          return true;
        }
 });
```

6. Multilanguage Support

Supported Languages

The Gantt Library currently supports the following languages:

Language	Language Code
Arabic	ar
Belarusian	be
English	en
Catalan	ca
Chinese	cn
Croatian	hr
Czech	cs
Danish	da
Dutch	nl
Finnish	fi
French	fr
German	de
Greek	el
Hebrew	he
Hindi (India)	hi
Hungarian	hu
Indonesian	id
Italian	it
Japanese	jp

Korean	kr
Norwegian	no
Persian	fa
Polish	pl
Portuguese	pt
Romanian	ro
Russian	ru
Slovak	sk
Slovenian	si
Spanish	es
Swedish	sv
Turkish	tr
Ukrainian	ua

7. Theme

Dark Theme for Gantt chart

To apply a dark theme to your Gantt chart, simply incorporate the "dark.css" file located in the theme folder. By doing so, the Gantt chart will seamlessly adopt the visually appealing dark theme.

8. Example

For further details, please consult this example. ztGantt