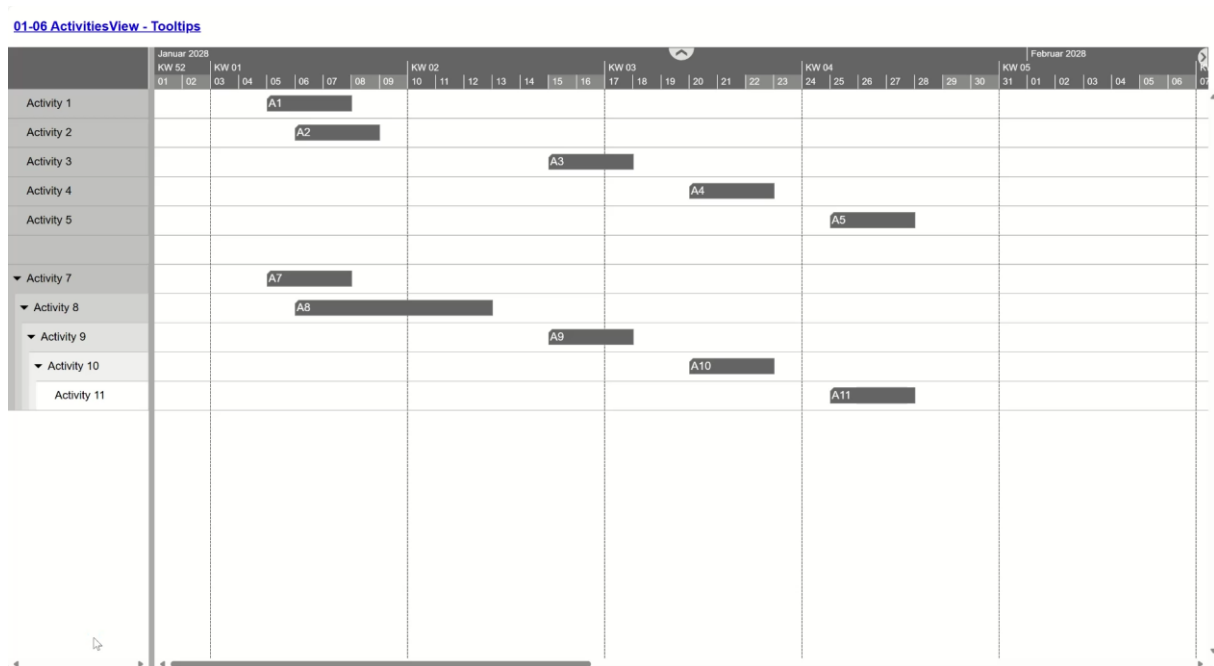


14-01 Activities view – Tooltips

This example shows how activity data can be displayed in a tooltip.



In essence, the frame of a tooltip window is provided without content. The handling of when the tooltip appears and when it disappears again is handled by the visual scheduling widget.

The content of the tooltip is best defined as an HTML table element. Here too, as is generally the case, the style should be separated from the actual content structure.

We have defined a style for the container, the header, the descriptor column field and the data column field:

```
.table-container {  
  width: 100%;  
  max-width: 400px;  
  border-collapse: collapse;  
}  
  
.tooltip-header {  
  padding: 2px 6px;  
  background-color: #ccc;  
  color: #444;  
  font-size: 13px;  
  font-weight: bold;  
}  
  
.table-rowHeader {  
  padding: 2px 6px;  
  background-color: #f1f1f1;  
  text-align: left;  
  border-bottom: 1px solid #ddd;  
  font-size: 12px;  
  line-height: 1.2;  
}  
  
.table-data {  
  padding: 2px 6px;  
  text-align: left;  
  border-bottom: 1px solid #ddd;  
}
```

```

font-size: 12px;
line-height: 1.2;
}

```

The structure of the table is as follows:

```

<table class="table-container">
  <tr>
    <td class="tooltip-header" colspan="2">Activity</td>
  </tr>
  <tr>
    <td class="table-rowHeader">ID</td>
    <td class="table-data">{{ID}}</td>
  </tr>
  <tr>
    <td class="table-rowHeader">Name</td>
    <td class="table-data">{{TableText}}</td>
  </tr>
  <tr>
    <td class="table-rowHeader">Start</td>
    <td class="table-data">{{Start:date:dateTimeFmt}}</td>
  </tr>
  <tr>
    <td class="table-rowHeader">End</td>
    <td class="table-data">{{End:date:dateTimeFmt}}</td>
  </tr>
  <tr>
    <td class="table-rowHeader">Working days</td>
    <td class="table-data">{{WorkingDays}}</td>
  </tr>
  <tr>
    <td class="table-rowHeader">Running days</td>
    <td class="table-data">{{RunningDays}}</td>
  </tr>
</table>

```

The double-curly entry contains the data field whose content is to be output in the template.

```
{{my_WorkingDays}}
```

If the data field contains a date, the identifier for a formatting rule can be entered using a colon:

```
{{Start:date:dateTimeFmt}}
```

The format is defined with the **intlDateTimeFormatOptionsMap** option.

```

intlDateTimeFormatOptionsMap: {
  dateTimeFmt: {
    year: "numeric",
    month: "2-digit",
    day: "2-digit",
    hour: "2-digit",
    minute: "2-digit",
  },
},

```

Finally, the widget must know which tooltip template is to be used. To do this, the tooltip templates are added to the **vsWidget** using the **addTooltipTemplates** method

```
vsWidget.addTooltipTemplates(tooltips);
```

and the **defaultActivityBarTooltipTemplateID** option is set with the corresponding **ID**.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
<link href="../../VSWidget/nwaf-apptools.min.css" rel="stylesheet"/>
<link href="../../VSWidget/nwaf-table.min.css" rel="stylesheet"/>
<link href="../../VSWidget/nwaf-gantt.min.css" rel="stylesheet"/>
<link href="../../VSWidget/nwaf-rab.min.css" rel="stylesheet"/>
<script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8/hammer.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/d3@7.9.0/dist/d3.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/tinycolor2@1.6.0/cjs/tinycolor.min.js"></script>
<script src="../../VSWidget/nwaf-apptools.min.js"></script>
<script src="../../VSWidget/nwaf-table.min.js"></script>
<script src="../../VSWidget/nwaf-gantt.min.js"></script>
<script src="../../VSWidget/nwaf-rab.min.js"></script>
<script src="../../LicenseKey.js"></script>
<script src="Miscellaneous.js"></script>
<style>
    .table-container {
        width: 100%;
        max-width: 400px;
        border-collapse: collapse;
    }

    .tooltip-header {
        padding: 2px 6px;
        background-color: #ccc;
        color: #444;
        font-size: 13px;
        font-weight: bold;
    }

    .table-rowHeader {
        padding: 2px 6px;
        background-color: #f1f1f1;
        text-align: left;
        border-bottom: 1px solid #ddd;
        font-size: 12px;
        line-height: 1.2;
    }

    .table-data {
        padding: 2px 6px;
        text-align: left;
        border-bottom: 1px solid #ddd;
        font-size: 12px;
        line-height: 1.2;
    }
</style>
</head>
<body>
<div id="toolbar"></div>
<div id="VSWidget"></div>
<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, RowDesigns, SnapTargets } = netronic.nVSW;

        const timeAreaStart = new Date(2028, 0, 1);
        const timeAreaEnd = new Date(2028, 3, 1);

        let options = {
            licenseKey: window.VSW_LICENSE_KEY,

            start: timeAreaStart,
            end: timeAreaEnd,

            ignoreCalendarOnActivityBarInteractions: true,
            defaultActivitySnapTargetsForStart: SnapTargets.None,
            defaultActivitySnapTargetsForEnd: SnapTargets.None,

            defaultActivityExpandedRowDesign: RowDesigns.Bars,
            defaultActivityCollapsedRowDesign: RowDesigns.Bars |
                RowDesigns.BarsInHiddenDescendantRows |
                RowDesigns.BarsStacked,

            defaultActivityAllowedBarDragModes: BarDragModes.DragHorizontally,

            defaultActivityBarSelectable: false,

```

```

defaultActivityRowSelectable: false,
multipleSelectionEnabled: false,

intlDateTimeFormatOptionsMap: {
  dateTimeFmt: {
    year: "numeric",
    month: "2-digit",
    day: "2-digit",
    hour: "2-digit",
    minute: "2-digit",
  },
},

defaultActivityBarTooltipTemplateID: "Tooltip1",
defaultActivityRowTooltipTemplateID: "Tooltip1",

onDrop: (args) => {
  const activity = args.object;
  activity.Start = args.newStart;
  activity.End = args.newEnd;
  vsWidget.updateActivities([activity]);
  vsWidget.render();
},
};

const vsWidget = Miscellaneous.instantiateVSWidget(document.querySelector("#VSWidget"), options);

const activities = createActivities();
const tooltips = createTooltips();
vsWidget.addTooltipTemplates(tooltips);
vsWidget.addActivities(activities);
vsWidget.render();

function createActivities() {
  const activities = [
    {
      ID: "Act1",
      TableText: "Activity 1",
      BarText: "A1",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
      ParentID: "",
    },
    {
      ID: "Act2",
      TableText: "Activity 2",
      BarText: "A2",
      Start: new Date(2028, 0, 6),
      my_WorkingDays: 3,
      ParentID: "",
    },
    {
      ID: "Act3",
      TableText: "Activity 3",
      BarText: "A3",
      Start: new Date(2028, 0, 15),
      my_WorkingDays: 3,
      ParentID: "",
    },
    {
      ID: "Act4",
      TableText: "Activity 4",
      BarText: "A4",
      Start: new Date(2028, 0, 20),
      my_WorkingDays: 3,
      ParentID: "",
    },
    {
      ID: "Act5",
      TableText: "Activity 5",
      BarText: "A5",
      Start: new Date(2028, 0, 25),
      my_WorkingDays: 3,
      ParentID: "",
    },
  ];
}

```

```

    },
    {
      ID: "Act6",
      ParentID: "",
    },
    {
      ID: "Act7",
      TableText: "Activity 7",
      BarText: "A7",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
      ParentID: "",
    },
    {
      ID: "Act8",
      TableText: "Activity 8",
      BarText: "A8",
      Start: new Date(2028, 0, 6),
      my_WorkingDays: 7,
      ParentID: "Act7",
    },
    {
      ID: "Act9",
      TableText: "Activity 9",
      BarText: "A9",
      Start: new Date(2028, 0, 15),
      my_WorkingDays: 3,
      ParentID: "Act8",
    },
    {
      ID: "Act10",
      TableText: "Activity 10",
      BarText: "A10",
      Start: new Date(2028, 0, 20),
      my_WorkingDays: 3,
      ParentID: "Act9",
    },
    {
      ID: "Act11",
      TableText: "Activity 11",
      BarText: "A11",
      Start: new Date(2028, 0, 25),
      my_WorkingDays: 3,
      ParentID: "Act10",
    },
  ],

  const millisecondsPerDay = 24 * 60 * 60 * 1000;
  for (const activity of activities)
    if (activity.Start) {
      activity.End = new Date(activity.Start.getTime() +
        (activity.my_WorkingDays ?? 1) * millisecondsPerDay);
      activity.my_RunningDays = activity.my_WorkingDays;
    }

  for (let i = 0; i < 5; i++) {
    activities[i].my_DocUrl = "pdf/01-06_ActivitiesView_Tooltips.pdf";
    activities[i].my_DocName = `Activity ${i + 1} Document`;
  }

  return activities;
}

function createTooltips() {
  const tooltips = [
    {
      ID: "Tooltip1",
      IsInteractive: true,
      HTMLFormat: `
        <div style="max-height: 120px; overflow:auto;">
          <table class="table-container">
            <tr>
              <td class="tooltip-header" colspan="2">Activity</td>
            </tr>

```

```

        <tr>
            <td class="table-rowHeader">ID</td>
            <td class="table-data">{{ID}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">Name</td>
            <td class="table-data">{{TableText}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">Start</td>
            <td class="table-data">{{Start:date:dateTimeFmt}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">End</td>
            <td class="table-data">{{End:date:dateTimeFmt}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">Working days</td>
            <td class="table-data">{{my_WorkingDays}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">Running days</td>
            <td class="table-data">{{my_RunningDays}}</td>
        </tr>
        <tr>
            <td class="table-rowHeader">Document</td>
            <td class="table-data">
                <a href="{{my_DocUrl}}" target="_blank">{{my_DocName}}</a>
            </td>
        </tr>
    </table>
</div>`,
    },
];

return tooltips;
}
});
</script>
</body>
</html>

```