

## 10-01 Activities view - Basics - Activities

This example shows how activities are added to the ActivitiesView and appear as bars in the time area.

Each activity object requires a unique identifier, which is saved in the **ID** property. This allows activity objects to be distinguished from one another.

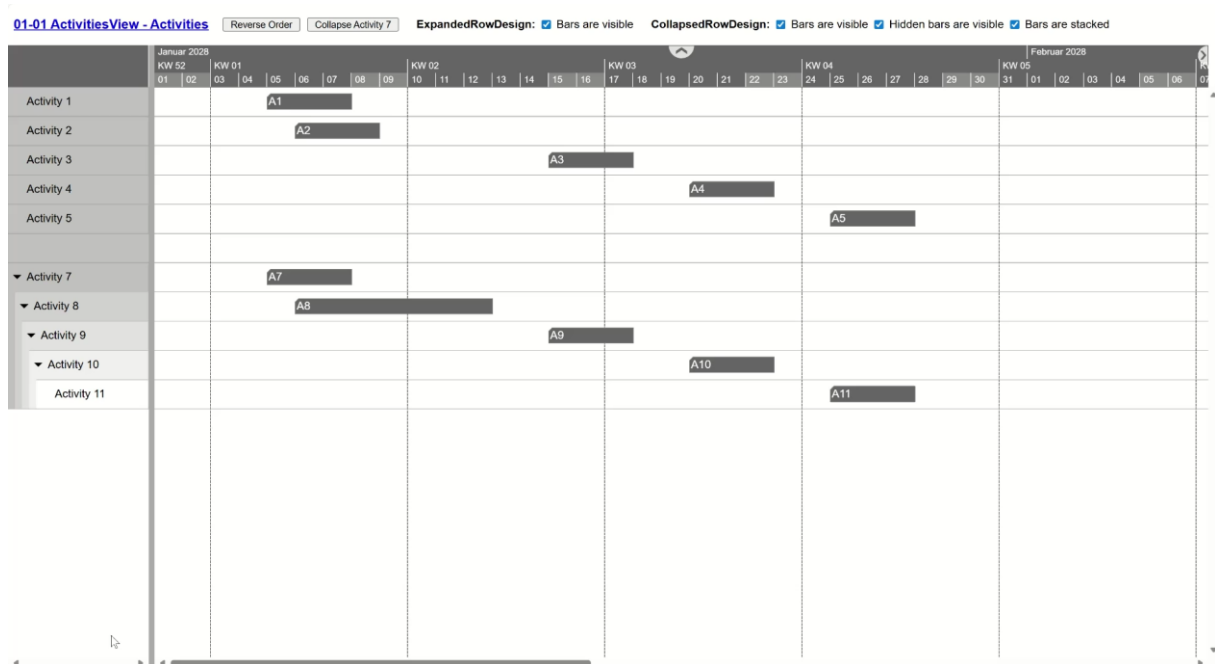
The start date (**Start**) and end date (**End**) are required to display an activity bar.

A text assignment for the table column can be made with **TableText** and for the labelling within the bar with **BarText**.

The **ParentID** property is required for hierarchical displays.

In the VSWidget, the principle applies that all objects are created and managed by the user of the library. If information with the specified designations is available in the object, it is used. All other entries are irrelevant for VSWidget. The VSWidget does not change any entries.

VSWidget is informed about the existing objects by the **AddActivities()** method. Exactly one row is created in the activity diagram for each object entry. The display sequence corresponds to the specified sequence. Hierarchical subordinate activities are lined up directly behind ParentRows.



Bars within the rows are only ever visible if several conditions are met:

- The activity object must have a valid entry for **Start** and **End**.
- The time range for the bar must lie at least partially within the time range selected for the display.
- The **ExpandedRowDesign** and the **CollapseRowDesign** must be set to display bars.

There are special features for collapsed rows. Here you can set whether all bars of the non-visible rows should be displayed within the collapsed row:

## BarsInHiddenDescendantRows



If overlaps occur, you can also set whether these bars should be displayed in their own sub-rows without overlaps:

## BarsStacked



The **activityBarSortModeForStackedRowDesign** option defines how the bars are arranged. The **BarSortMode** is set to **StartAndEnd** by default: The bars are sorted according to the start date. If the start date is the same, the longer bar is taken first.

In this example, the standard marking of bars has been prevented by setting the corresponding options:

**defaultActivityBarSelectable: false,**  
**defaultActivityRowSelectable: false,**  
**multipleSelectionEnabled: false,**

The option to move bars has also been deactivated:

**defaultActivityAllowedBarDragModes: BarDragModes.None,**

```
<!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>
</head>
<body>
  <div id="toolbar">
    <input id="reverseOrder" type="button" value="Reverse Order"
      style="margin-left: 15px; height: 20px; width: 110px"/>
    <input id="collapse" type="button" value="Collapse Activity 7"
      style="margin-left: 5px; height: 20px; width: 130px"/>
    <label style="margin-left: 20px;">
      <b>ExpandedRowDesign:</b>
    </label>
    <input type="checkbox" id="expandBarVisibility" checked/>
    <label for="expandBarVisibility">Bars are visible</label>
    <label style="margin-left: 20px;">
      <b>CollapsedRowDesign:</b>
    </label>
    <input type="checkbox" id="collapseBarVisibility" checked/>
    <label for="collapseBarVisibility">Bars are visible</label>
    <input type="checkbox" id="collapseHiddenBarVisibility" checked/>
    <label for="collapseHiddenBarVisibility">Hidden bars are visible</label>
    <input type="checkbox" id="collapseStackedBarVisibility" checked/>
    <label for="collapseStackedBarVisibility">Bars are stacked</label>
  </div>
```

```

<div id="VSWidget"></div>
<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, BarSortMode, CollapseState, RowDesigns } = netronic.nVSW;

        document.getElementById('reverseOrder').addEventListener('click', reverseOrder);
        document.getElementById('collapse').addEventListener('click', collapseOrExpandActivity7);
        document.getElementById('expandBarVisibility').addEventListener('click', updateRowDesigns);
        document.getElementById('collapseBarVisibility').addEventListener('click', updateRowDesigns);
        document.getElementById('collapseHiddenBarVisibility').addEventListener('click', updateRowDesigns);
        document.getElementById('collapseStackedBarVisibility').addEventListener('click', updateRowDesigns);

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

        const options = {
            licenseKey: window.VSW_LICENSE_KEY,

            start: timeAreaStart,
            end: timeAreaEnd,

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

            activityBarSortModeForStackedRowDesign: BarSortMode.StartAndEnd,

            defaultActivityAllowedBarDragModes: BarDragModes.None,
            defaultActivityBarSelectable: false,
            defaultActivityRowSelectable: false,
            multipleSelectionEnabled: false,

            onCollapseStateChanged: function (args) {
                if (args.object.ID === "Act7") {
                    const btn = document.getElementById('collapse');
                    btn.value = args.newCollapseState === CollapseState.Expanded ?
                        "Collapse Activity 7" : "Expand Activity 7";

                    const activity = args.object;
                    activity.CollapseState = args.newCollapseState;
                    vsWidget.updateActivities([activity]);
                    vsWidget.render();
                }
            },
        };

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

        const activities = createActivities();
        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);
    }

return activities;

```

```

    }

    function reverseOrder() {
        vsWidget.removeActivities(activities);
        vsWidget.addActivities(activities.reverse());
        vsWidget.render();
    }

    function collapseOrExpandActivity7() {
        const activity = activities.find(activity => activity.ID === "Act7");
        activity.CollapseState = activity.CollapseState === CollapseState.Collapsed ?
            CollapseState.Expanded : CollapseState.Collapsed;
        vsWidget.updateActivities([activity]);
        vsWidget.render();
    }

    function updateRowDesigns() {
        let expandedRowDesign = RowDesigns.None;
        let collapsedRowDesign = RowDesigns.None;

        if (document.getElementById("expandBarVisibility").checked)
            expandedRowDesign = RowDesigns.Bars;

        if (document.getElementById("collapseBarVisibility").checked)
            collapsedRowDesign = collapsedRowDesign | RowDesigns.Bars;

        if (document.getElementById("collapseHiddenBarVisibility").checked)
            collapsedRowDesign = collapsedRowDesign | RowDesigns.BarsInHiddenDescendantRows;

        if (document.getElementById("collapseStackedBarVisibility").checked)
            collapsedRowDesign = collapsedRowDesign | RowDesigns.BarsStacked;

        vsWidget.option({
            defaultActivityExpandedRowDesign: expandedRowDesign,
            defaultActivityCollapsedRowDesign: collapsedRowDesign,
        });
    }
    });
</script>
</body>
</html>

```