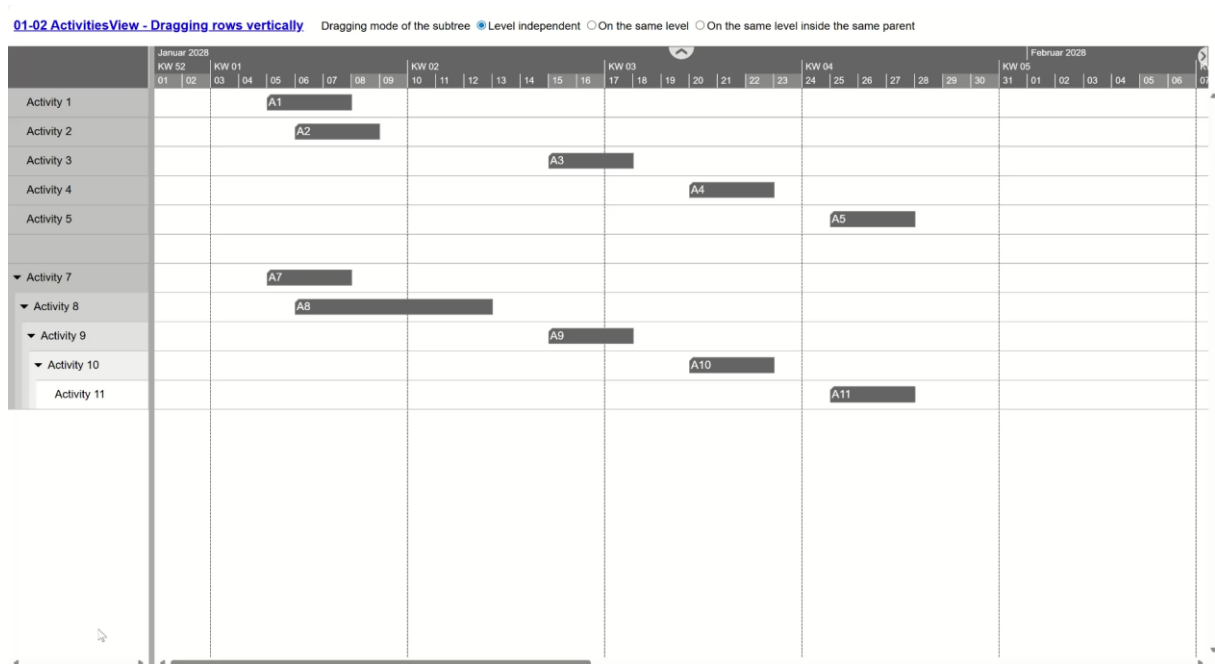


12-01 Activities view – Interactions – Moving rows via drag&drop

This example shows how moving a row changes the position of the associated subtree.

To allow rows to be dragged, the **defaultActivityAllowedRowDragModes** option must be set for the **RowDragModes.DragVertically** enum value.

In addition, the **activityRowSortMode** option must be set to the enum value **RowSortMode.Ascending** so that the row sequence can be managed.



The way in which the movement affects the subtree can be selected.

Dragging mode of the subtree ☒ Level independent ☐ On the same level ☐ On the same level inside the same parent

- **Level independent**

The **defaultActivityAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically**. A subtree can be inserted into an existing hierarchy at any level and at any position. However, it is not possible to insert it into your own subtree. If a subtree is to be assigned as a child of an activity that does not yet have any children, the **Shift** key must be held down when dragging.

- **On the same level**

The **defaultActivityAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically | Enum.RowDragModes.DragOnSameLevelOnly**. A subtree can be inserted into the same level at any position.

- **On the same level inside the same parent**

The **defaultActivityAllowedRowDragModes** option must be set to the enum value **RowDragModes.DragVertically | Enum.RowDragModes.DragInSameTableParentOnly**. A subtree can be inserted into the same level inside the same parent at any position.

The **processOnDrop()** method, which is called within the **onDrop** callback function, updates the activity objects. The method simplifies application development if no additional changes need to be made to the activity objects by the application.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>01-02 NETRONIC VSW SE</title>
  <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">
    <h3>
      <a href="docs/12-01_ActivitiesView-Interactions-Moving_rows_via_drag&drop.pdf"
        target="_blank">12-01_Activities view - Interactions - Moving rows via drag&drop</a>
    </h3>
    <label style="margin-left: 20px;">Dragging mode of the subtree</label>
    <label>
      <input id="levelIndependent" type="radio" name="moveMode" value="levelIndependent" checked/>
      Level independent
    </label>
    <label>
      <input id="onSameLevel" type="radio" name="moveMode" value="onSameLevel"/>
      On the same level
    </label>
    <label>
      <input id="inSameTableParentOnly" type="radio" name="moveMode" value="inSameTableParentOnly"/>
      On the same level inside the same parent
    </label>
  </div>
  <div id="VSWidget"></div>
</script>
  Miscellaneous.ready(() => {
    const { RowDesigns, BarDragModes, RowDragModes, RowSortMode } = netronic.nVSW;

    document.getElementById('levelIndependent').addEventListener('click', changeRowDragDropOptions);
    document.getElementById('onSameLevel').addEventListener('click', changeRowDragDropOptions);
    document.getElementById('inSameTableParentOnly').addEventListener('click', changeRowDragDropOptions);

    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,

      defaultCalendarID: "Cal1",
      defaultActivityExpandedRowDesign: RowDesigns.Bars,
      defaultActivityCollapsedRowDesign: RowDesigns.Bars |
        RowDesigns.BarsInHiddenDescendantRows |
        RowDesigns.BarsStacked,
      ignoreCalendarOnActivityBarInteractions: true,
      defaultActivityAllowedBarDragModes: BarDragModes.None,
      defaultActivityAllowedRowDragModes: RowDragModes.DragVertically,
      activityRowSortMode: RowSortMode.Ascending,
      defaultActivityBarSelectable: false,
      defaultActivityRowSelectable: false,
    };
  });
}
```

```

        multipleSelectionEnabled: false,

        onDrop: (args) => { vsWidget.processOnDrop(args); },
    };

    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 changeRowDragDropOptions(event) {
    const optionBox = event.target;

    if (optionBox.value === "levelIndependent")
        vsWidget.option("defaultActivityAllowedRowDragModes", RowDragModes.DragVertically);

    else if (optionBox.value === "onSameLevel")
        vsWidget.option("defaultActivityAllowedRowDragModes", RowDragModes.DragVertically |
                                                                    RowDragModes.DragOnSameLevelOnly);

    else // if (optionBox.value === "inSameTableParentOnly")
        vsWidget.option("defaultActivityAllowedRowDragModes", RowDragModes.DragVertically |
                                                                    RowDragModes.DragInSameTableParentOnly);
}
});
</script>
</body>
</html>

```