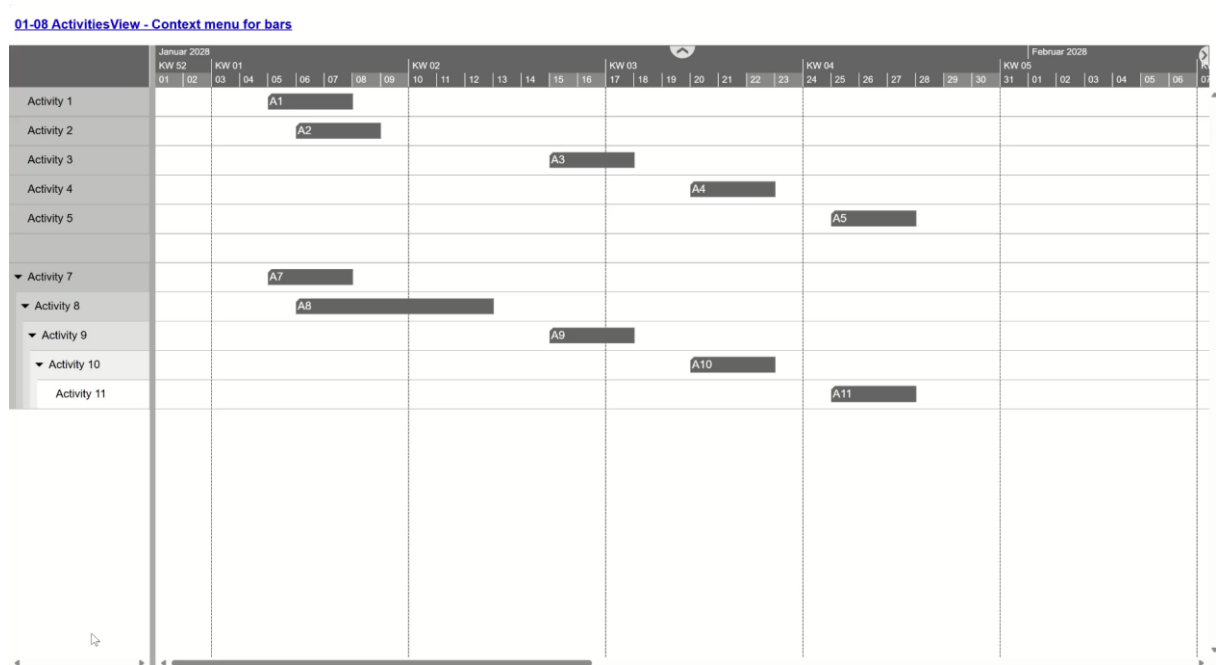


94-01 All views – Tooltips – Context menus

This example shows how to add a context menu for bars. You can use the commands in the content menu to assign random fill colours to the bars and select the bar shape in the submenu.



Context menus are not part of the visual planning widget library, so you must programme them yourself. Depending on the framework used, there may be special support for the structure of context menus. This example shows how it works without a framework, only with html 5.

The **onShowContextMenu** callback is triggered when the user right-clicks. This is the place where you insert your own code to make the context menu appear. The information provided in the callback makes it possible to decide what content should be displayed on the visual type that the mouse pointer hovers over.

```
if (args.visualType == VisualType.Bar)
```

```
...
```

The content context menu must disappear when the user clicks outside the content menu or when the user presses the **Esc** key. The two event listeners **keydown** and **click** are therefore required.

```
document.addEventListener("click", () => {
    onCloseContextMenu();
});

document.addEventListener("keydown", (event) => {
    if (event.key === "Escape")
        onCloseContextMenu();
});
```

Some style definitions are required for an appealing look.

```
.context-menu {
    display: block;
    position: absolute;
    background: white;
```

```

border: 1px solid #ccc;
box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2);
visibility: hidden;
padding: 2px 0;
border-radius: 4px;
min-width: 180px;
z-index: 1000;
}

.menu-item {
display: flex;
align-items: center;
padding: 5px 10px;
cursor: pointer;
user-select: none;
position: relative;
white-space: nowrap;
font-size: 12px;
}

.menu-item:hover {
background: #eaeaea;
}

.submenu {
position: absolute;
left: 100%;
top: 0;
background: white;
border: 1px solid #ccc;
box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2);
border-radius: 4px;
min-width: 160px;
opacity: 0;
transform: translateX(-10px);
transition: opacity 0.2s ease, transform 0.2s ease;
display: none;
}

.menu-item:hover > .submenu {
display: block;
opacity: 1;
transform: translateX(0);
}

.separator {
border-top: 1px solid #ddd;
margin: 3px 0;
}

.arrow {
margin-left: auto;
flex-shrink: 0;
}

```

The structure of the context menu is as follows:

```

<div id="contextMenu" class="context-menu">
  <div class="menu-item" onclick="executeCommand('RandomColor')">Random Color</div>
  <div class="menu-item" onclick="executeCommand('DefaultColor')">Default Color</div>
  <div class="separator"></div>
  <div class="menu-item">
    <span class="menu-text">Bar Shape</span>
    <span class="arrow">▶</span>
    <div class="submenu">
      <div class="menu-item" onclick="executeCommand('Shape1')">Regular</div>
      <div class="menu-item" onclick="executeCommand('Shape2')">Rectangle</div>
      <div class="menu-item" onclick="executeCommand('Shape3')">Summary</div>
      <div class="menu-item" onclick="executeCommand('Shape4')">Diamond</div>
    </div>
  </div>
  <div class="separator"></div>
</div>

```

```

    <div class="menu-item" onclick="executeCommand('Settings')">Settings ...</div>
</div>

```

A separate **executeCommand** function is provided to execute the commands.

```

<!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>
    .context-menu {
      display: block;
      position: absolute;
      background: white;
      border: 1px solid #ccc;
      box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2);
      visibility: hidden;
      padding: 2px 0;
      border-radius: 4px;
      min-width: 180px;
      z-index: 1000;
    }

    .menu-item {
      display: flex;
      align-items: center;
      padding: 5px 10px;
      cursor: pointer;
      user-select: none;
      position: relative;
      white-space: nowrap;
      font-size: 12px;
    }

    .menu-item:hover {
      background: #eaeaea;
    }

    .submenu {
      position: absolute;
      left: 100%;
      top: 0;
      background: white;
      border: 1px solid #ccc;
      box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.2);
      border-radius: 4px;
      min-width: 160px;
      opacity: 0;
      transform: translateX(-10px);
      transition: opacity 0.2s ease, transform 0.2s ease;
      display: none;
    }

    .menu-item:hover > .submenu {
      display: block;
      opacity: 1;
      transform: translateX(0);
    }

    .separator {
      border-top: 1px solid #ddd;
    }
  </style>

```

```

        margin: 3px 0;
    }

    .arrow {
        margin-left: auto;
        flex-shrink: 0;
    }
</style>
</head>
<body>
    <div id="toolbar"></div>
    <div id="VSWidget"></div>
    <div id="contextMenu" class="context-menu">
        <div class="menu-item" name="RandomColor">
            Random Color
        </div>
        <div class="menu-item" name="DefaultColor">
            Default Color
        </div>
        <div class="separator"></div>
        <div class="menu-item">
            <span class="menu-text">Bar Shape</span>
            <span class="arrow">▶</span>
            <div class="submenu">
                <div class="menu-item" name="Shape1">
                    Regular
                </div>
                <div class="menu-item" name="Shape2">
                    Rectangle
                </div>
                <div class="menu-item" name="Shape3">
                    Summary
                </div>
                <div class="menu-item" name="Shape4">
                    Diamond
                </div>
            </div>
        </div>
        <div class="separator"></div>
        <div class="menu-item" name="Settings">
            Settings...
        </div>
    </div>
</div>

<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, BarShape, RowDesigns, SnapTargets, VisualType } = netronic.nVSW;

        document.querySelectorAll(".menu-item").forEach((item) => {
            item.addEventListener("click", (event) => {
                const command = event.currentTarget.getAttribute("name");
                if (command)
                    executeCommand(command);
            });
        });

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

        const hideContextMenu = () => {
            const contextMenu = document.getElementById("contextMenu");
            contextMenu.style.visibility = "hidden";
            currentActivity = null;
        };

        const 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.None,
defaultActivityBarSelectable: false,
defaultActivityRowSelectable: false,
multipleSelectionEnabled: false,

onShowContextMenu: (args) => {
    if (args.visualType === VisualType.Bar) {
        currentActivity = args.object;

        const contextMenu = document.getElementById("contextMenu");
        let menuWidth = contextMenu.offsetWidth;
        let menuHeight = contextMenu.offsetHeight;

        let posX = args.event.clientX;
        let posY = args.event.clientY;
        if (posX + 2 * menuWidth > window.innerWidth)
            posX = window.innerWidth - 2 * menuWidth;
        if (posY + 1.2 * menuHeight > window.innerHeight)
            posY -= window.innerHeight - 1.2 * menuHeight;

        contextMenu.style.left = `${posX}px`;
        contextMenu.style.top = `${posY}px`;
        contextMenu.style.visibility = "visible";
        contextMenu.style.opacity = 1;

        args.promise = new Promise((resolve, _reject) => {
            // create an observer instance
            const observer = new MutationObserver(
                (mutationList, observer) => {
                    for (const mutation of mutationList) {
                        // resolve promise when context menu gets invisible
                        if (mutation.attributeName === "style" &&
                            contextMenu.style.visibility === "hidden") {
                            observer.disconnect();
                            resolve();
                        }
                    }
                }
            );

            // start observing the target node for configured mutations
            observer.observe(contextMenu, { attributes: true });
        });
    }
},

onCloseContextMenu: () => window.setTimeout(hideContextMenu, 100),
};

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

const activities = createActivities();
vsWidget.addActivities(activities);
vsWidget.render();

document.addEventListener("click", (event) => {
    // close context menu if the user clicked besides it
    const target = document.elementFromPoint(event.clientX, event.clientY);
    if (!target.closest("#contextMenu"))
        window.setTimeout(hideContextMenu, 100);
});

document.addEventListener("keydown", (event) => {
    if (event.key === "Escape")
        hideContextMenu();
});

```

```

function createActivities() {
  const activities = [
    {
      ID: "Act1",
      TableText: "Activity 1",
      BarText: "A1",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act2",
      TableText: "Activity 2",
      BarText: "A2",
      Start: new Date(2028, 0, 6),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act3",
      TableText: "Activity 3",
      BarText: "A3",
      Start: new Date(2028, 0, 15),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act4",
      TableText: "Activity 4",
      BarText: "A4",
      Start: new Date(2028, 0, 20),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act5",
      TableText: "Activity 5",
      BarText: "A5",
      Start: new Date(2028, 0, 25),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act6",
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act7",
      TableText: "Activity 7",
      BarText: "A7",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
      ParentID: "",
      Color: "",
      BarShape: BarShape.Regular,
    },
    {
      ID: "Act8",
      TableText: "Activity 8",
      BarText: "A8",
      Start: new Date(2028, 0, 6),
      my_WorkingDays: 7,
    }
  ]
}

```

```

        ParentID: "Act7",
        Color: "",
        BarShape: BarShape.Regular,
    },
    {
        ID: "Act9",
        TableText: "Activity 9",
        BarText: "A9",
        Start: new Date(2028, 0, 15),
        my_WorkingDays: 3,
        ParentID: "Act8",
        Color: "",
        BarShape: BarShape.Regular,
    },
    {
        ID: "Act10",
        TableText: "Activity 10",
        BarText: "A10",
        Start: new Date(2028, 0, 20),
        my_WorkingDays: 3,
        ParentID: "Act9",
        Color: "",
        BarShape: BarShape.Regular,
    },
    {
        ID: "Act11",
        TableText: "Activity 11",
        BarText: "A11",
        Start: new Date(2028, 0, 25),
        my_WorkingDays: 3,
        ParentID: "Act10",
        Color: "",
        BarShape: BarShape.Regular,
    },
    ],
];

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 executeCommand(command) {
    document.querySelector("#contextMenu").style.visibility = "hidden";

    setTimeout(() => {
        if (command === "RandomColor") {
            currentActivity.Color = getRandomHexColor();
            vsWidget.updateActivities([currentActivity]);
            vsWidget.render();
        }
        else if (command === "DefaultColor") {
            delete currentActivity["Color"];
            vsWidget.updateActivities([currentActivity]);
            vsWidget.render();
        }
        else if (command === "Shape1") {
            currentActivity.BarShape = BarShape.Regular;
            vsWidget.updateActivities([currentActivity]);
            vsWidget.render();
        }
        else if (command === "Shape2") {
            currentActivity.BarShape = BarShape.Rectangle;
            vsWidget.updateActivities([currentActivity]);
            vsWidget.render();
        }
        else if (command === "Shape3") {
            currentActivity.BarShape = BarShape.Summary;
            vsWidget.updateActivities([currentActivity]);
            vsWidget.render();
        }
    });
}

```

```

    }
    else if (command === "Shape4") {
        currentActivity.BarShape = BarShape.Diamond;
        vsWidget.updateActivities([currentActivity]);
        vsWidget.render();
    }
    else
        alert(`${command} not implemented!`);
}, 10);
}

function getRandomHexColor() {
    return `#${Math.floor(Math.random() * 16777215).toString(16).padStart(6, "0")}`;
}
});
</script>
</body>
</html>

```