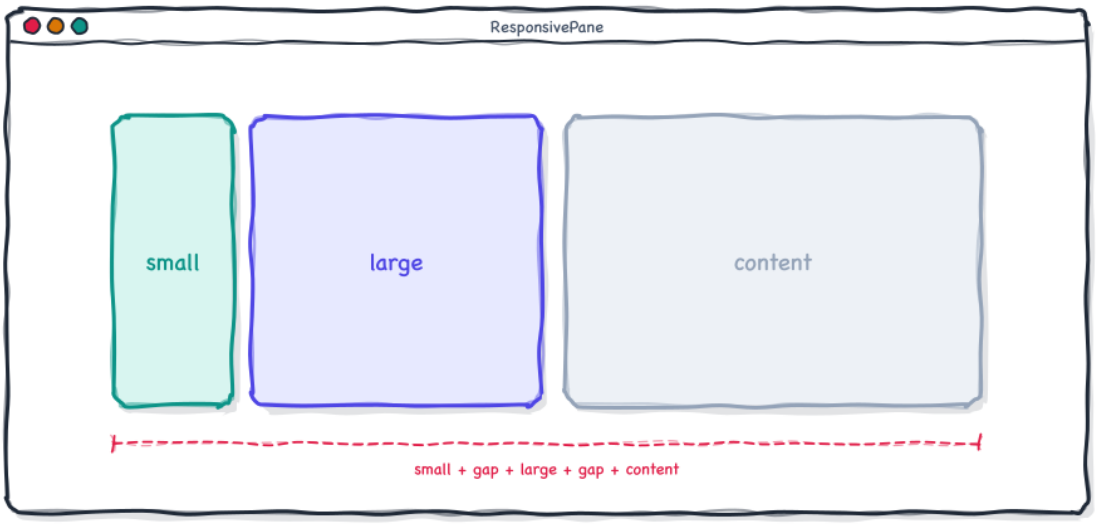
ResponsivePane

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



A generated cartoon overview of ResponsivePane.

ResponsivePane is a StackPane-based responsive layout that chooses no sidebar, a compact sidebar or a large sidebar from available size and preferred node sizes.

Table of Contents

1. Introduction

ResponsivePane chooses between no sidebar, a compact sidebar and a large sidebar from the available width or height. LEFT and RIGHT layouts compare widths; TOP and BOTTOM compare heights.

1.1 Key features

- Root style class `responsive-pane` and user agent stylesheet `responsive-pane.css`.
- Styleable `side` and `gap` properties.
- State pseudo classes indicate which sidebar is active.
- Forced large display uses an internal GlassPane that closes on click.
- Node properties keep the children list synchronized.

1.2 Maven dependency

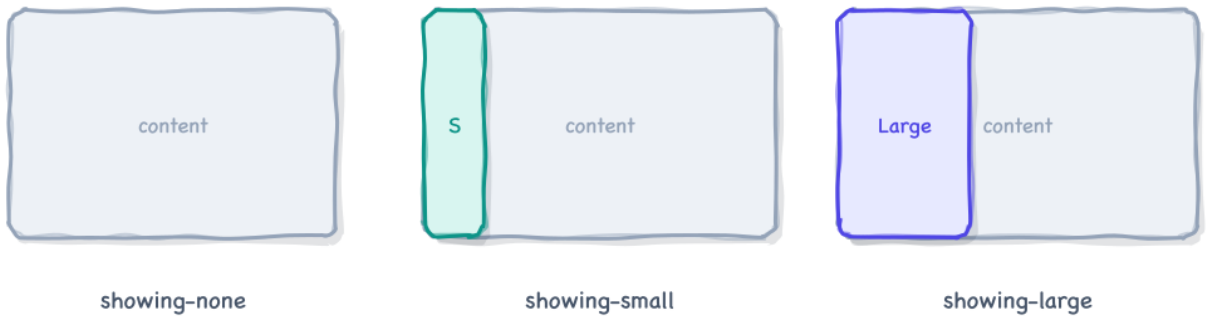
```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

Maven coordinates for the GemsFX control library.

2. Getting started

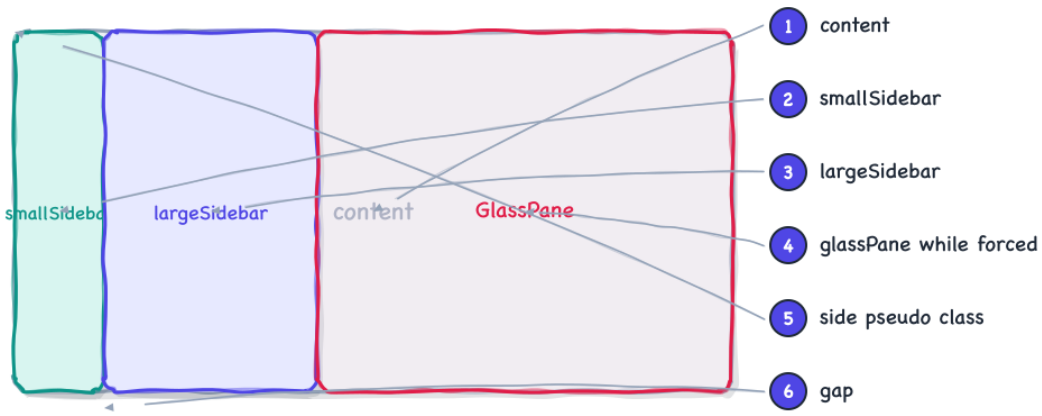
```
ResponsivePane pane = new ResponsivePane();
pane.setContent(contentView);
pane.setSmallSidebar(compactRail);
pane.setLargeSidebar(fullNavigation);
pane.setSide(Side.LEFT);
pane.setGap(10);
```

A content area with small and large sidebars.



The pane selects none, small or large from available space.

3. Anatomy



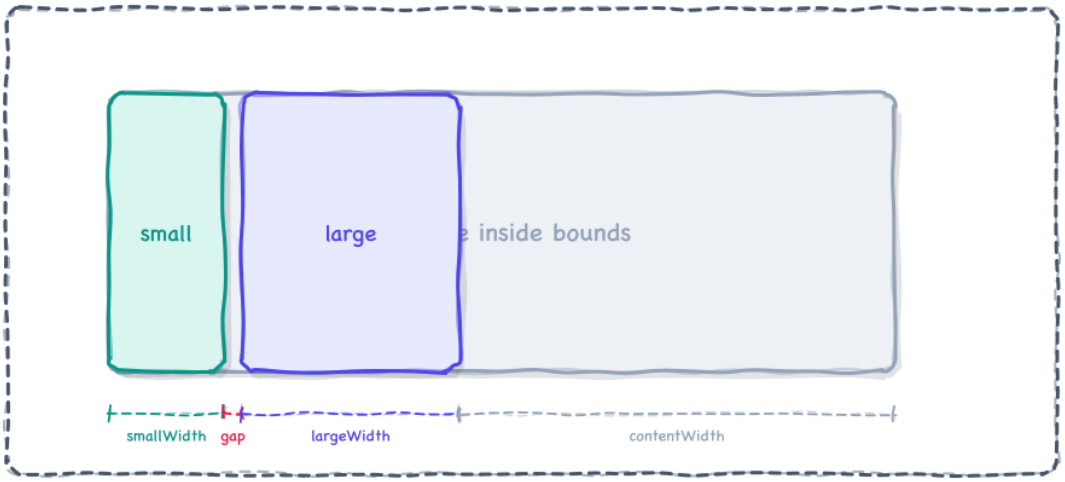
Content, sidebars and the glass pane.

Part	Property / node	Description
Content	`content`	Main node resized into remaining space.
Small sidebar	`smallSidebar`	Compact node used when the large sidebar does not fit.
Large sidebar	`largeSidebar`	Full sidebar used when it fits or is forced.
Glass pane	GlassPane	Internal overlay over content while large sidebar is forced.

4. Control API

Property	Type	Default	Description
content	ObjectProperty<Node>	null	Main content.
smallSidebar	ObjectProperty<Node>	null	Compact sidebar.
largeSidebar	ObjectProperty<Node>	null	Large sidebar.
side	ObjectProperty<Side>	Side.LEFT	Sidebar side; styleable with -fx-side.
gap	DoubleProperty	0	Gap between sidebar and content; styleable with -fx-gap and clamped to non-negative in layout.
forceLargeSidebarDisplay	BooleanProperty	false	Forces large sidebar when small sidebar is active.
largeSidebarCoversSmall	BooleanProperty	false	Controls whether forced large covers or sits next to small.

5. Breakpoint geometry



Width and gap measurements in a LEFT layout.

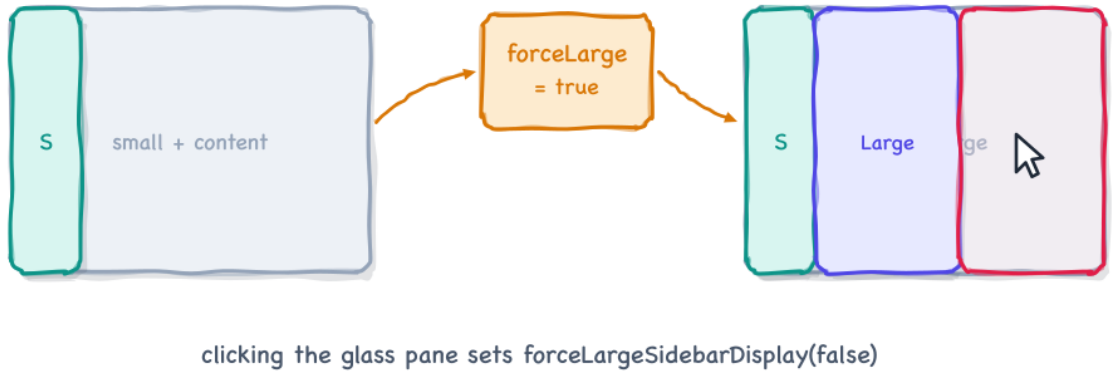
For horizontal sides the pane tests `insideWidth` against preferred content width, sidebar width and gap. For vertical sides it repeats the same logic with heights.

```
if insideWidth >= prefContentWidth + largeSidebarWidth + gap:
    active = largeSidebar
elif insideWidth > prefContentWidth + smallSidebarWidth + gap:
    active = smallSidebar
else:
    active = null
```

Simplified horizontal breakpoint logic.

> **Note.** The source uses >= for the large-sidebar threshold and > for the small-sidebar threshold.

6. Forced large sidebar



Forced display adds the large sidebar and a glass pane.

Forced display only has an effect while the small sidebar is the active sidebar. The large sidebar is made visible, moved to the front and accompanied by a glass pane over the content area.

largeSidebarCoversSmall	Placement
false	Large sidebar is placed next to the small sidebar.
true	Large sidebar starts at the same edge and covers the small sidebar.

7. Styling

Selector	Description
`.responsive-pane:left/right/top/bottom`	side state
`:showing-none`	no sidebar active
`:showing-small`	small sidebar active
`:showing-large`	large sidebar active
`:forced`	forced large visible
`:covering`	forced large covers small

CSS property	Type	Default
`-fx-side`	Side	LEFT
`-fx-gap`	number	0

```
.responsive-pane {
    -fx-side: left;
    -fx-gap: 10;
}
.responsive-pane:forced .overview {
    -fx-effect: dropshadow(gaussian, rgba(0,0,0,.26), 10, 0, 5, 0);
}
```

8. Child ownership and z-order

ResponsivePane stores content and sidebars in custom node properties. Assigning a property removes the old node from `getChildren()` and adds the new node. Removing a node directly from the children list also clears the matching property.

Operation	Effect
`setContent(node)`	Old content is removed; new content is added.
Remove content from children	The content property is set to null.
Layout content	Content is sent to the back when necessary.
Forced large sidebar	Large sidebar is brought to the front.

The internal glass pane is added by the constructor before application nodes. During layout it is resized over the content area and its hide property follows whether a forced large sidebar needs display.

```
glassPane.relocate(contentStartX, contentStartY);
glassPane.resize(contentWidth, contentHeight);
```

9. Vertical side layouts

When `side` is TOP or BOTTOM the same selection algorithm uses heights. The active sidebar stretches across the inside width; content height is reduced by sidebar height plus gap.

Side	Active sidebar placement	Content adjustment
TOP	x = left inset, y = top inset	contentStartY += sidebarHeight + gap
BOTTOM	x = left inset, y = bottom edge - sidebarHeight	contentHeight -= sidebarHeight + gap

Forced large display also supports top and bottom. If `largeSidebarCoversSmall` is false, the large sidebar sits next to the small sidebar along the vertical axis; otherwise it covers it from the same edge.

```
pane.setSide(Side.BOTTOM);
pane.setForceLargeSidebarDisplay(true);
pane.setLargeSidebarCoversSmall(false);
```

10. Designing breakpoints deliberately

ResponsivePane does not use hard-coded breakpoint numbers. The breakpoint is an emergent result of content preferred size, sidebar preferred size and gap. This makes the pane robust when fonts, localization or user scaling change preferred sizes.

Concern	Recommended practice
Preferred sizes	Set explicit preferred sizes for important children so the layout maths has stable inputs.
Insets and padding	Remember that pane insets are part of the available area calculation or child placement.
Managed state	Only managed children should be expected to participate in layout decisions.
Runtime changes	Property invalidation calls requestLayout, so batch related changes where possible.

If a sidebar appears too late or too early, adjust preferred size of the content or sidebar rather than adding external width listeners. The source selection logic already re-runs during layout.

```
content.setPrefWidth(420);
smallSidebar.setPrefWidth(56);
largeSidebar.setPrefWidth(220);
pane.setGap(12);
```

11. Recipes

11.1 Open overlay navigation

```
pane.setForceLargeSidebarDisplay(true);
```

11.2 Vertical sidebars

```
pane.setSide(Side.TOP);
pane.setGap(8);
```

11.3 Checklist

1. Set preferred sizes on content and both sidebars.
2. Use side to choose width-driven or height-driven breakpoints.
3. Style pseudo classes instead of adding width listeners.
4. Let the glass pane click close forced display.

12. See also

- Demo application: `com.dlsc.gemsfx.demo.ResponsivePaneApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.ResponsivePaneApp`)
- `HiddenSidesPane` - mouse edge overlays.
- `PowerPane` - application shell composition.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/