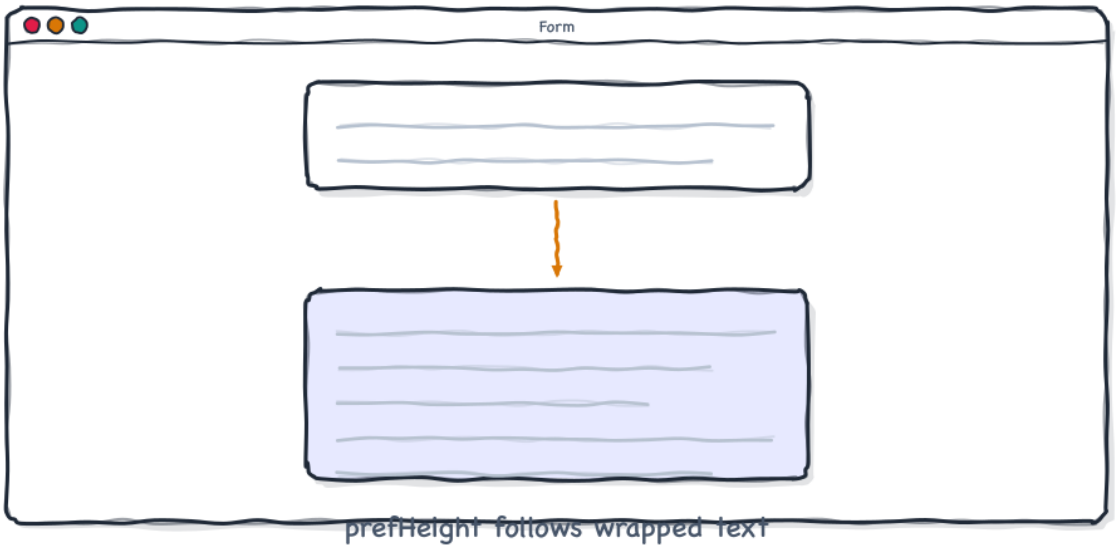
ExpandingTextArea

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



A generated cartoon overview of ExpandingTextArea.

ExpandingTextArea is a TextArea subclass that wraps text, suppresses scroll bars and binds its preferred height to the measured text height.

Table of Contents

1. Introduction

ExpandingTextArea is a customized `TextArea` for form layouts where all text should stay visible. It does not introduce new public properties; instead it configures and observes the standard text-area skin.

1.1 Key features

- Automatically calls `setWrapText(true)`.
- Forces the internal scroll pane horizontal and vertical scroll bars to `NEVER`.
- Binds `prefHeight` to the measured text node height plus insets.
- Keeps the normal TextArea API for text, prompt text and selection.

1.2 Maven dependency

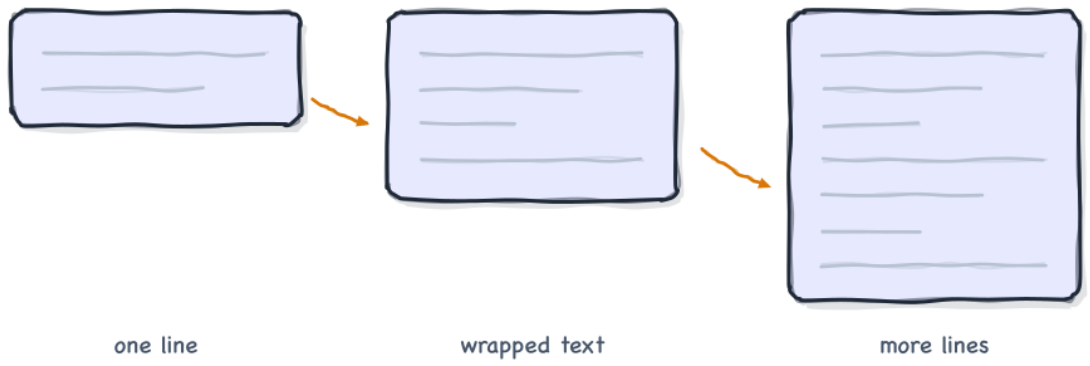
```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

Maven coordinates for the GemsFX control library.

2. Getting started

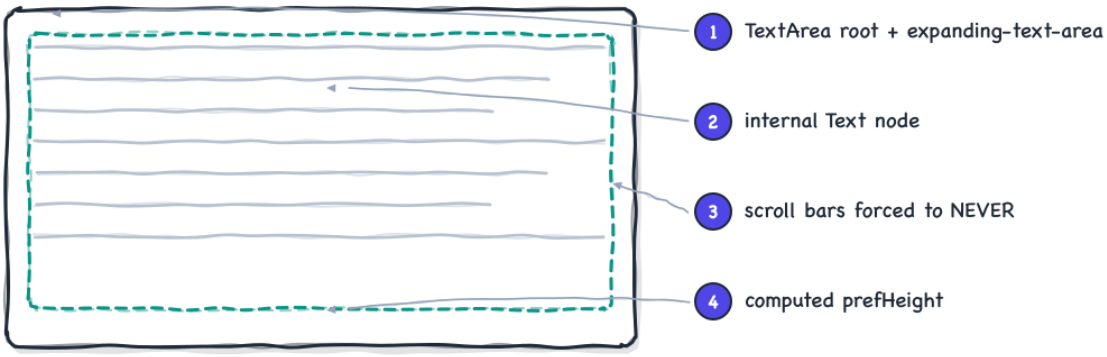
```
ExpandingTextArea area = new ExpandingTextArea();
area.setPromptText("Notes");
area.setMaxWidth(400);
area.textProperty().addListener((obs, old, text) -> saveDraft(text));
```

A complete expanding text area setup.



The preferred height increases as wrapped text needs more lines.

3. Anatomy



Parts involved in height computation.

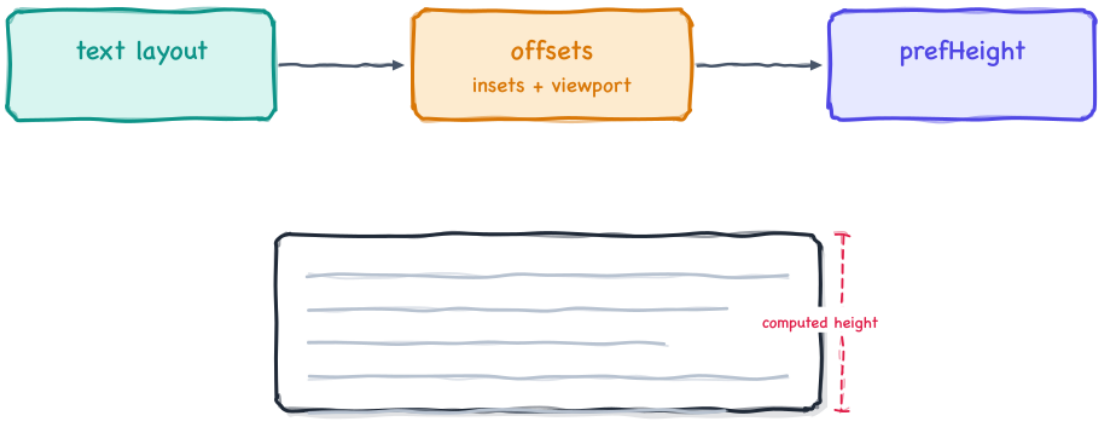
Part	Source	Description
Root	`.expanding-text-area`	Style class added by the constructor.
Scroll pane	lookup `.scroll-pane`	Its bars are disabled.
Viewport / content	lookup nodes	Insets are added to the height calculation.
Text node	`Text` whose parent is a `Group`	Its layout bounds drive the preferred height.

4. Control API

The control adds no new JavaFX properties. Use the inherited `TextArea` API.

Property	Type	Default	Description
text	StringProperty	empty string	Inherited text content.
promptText	StringProperty	empty string	Inherited prompt shown when text is empty.
wrapText	BooleanProperty	true	Set to true by the constructor.
prefHeight	DoubleProperty	bound after skin setup	Computed from the internal text node and insets.

5. Height computation

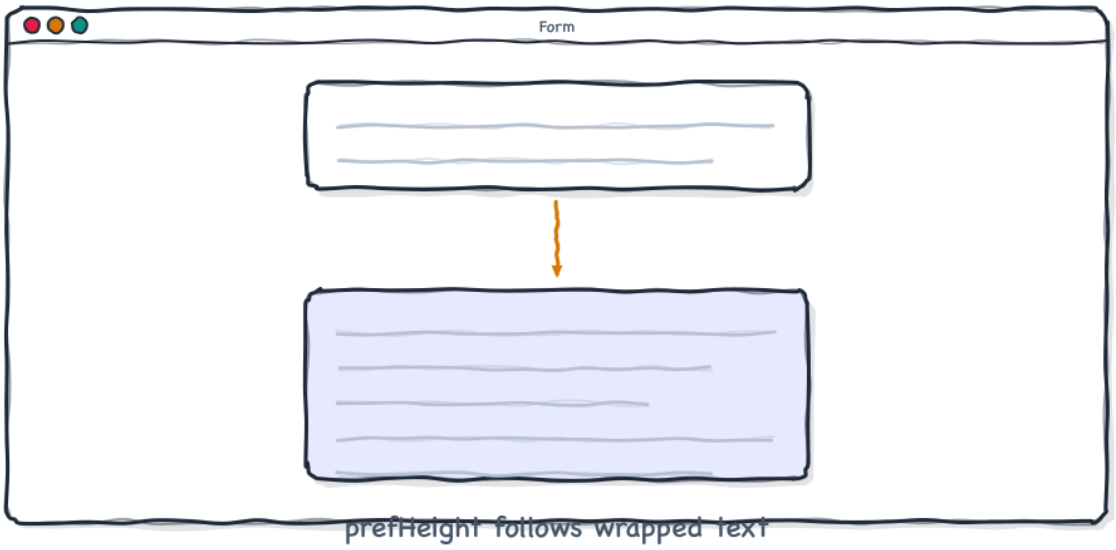


How text layout becomes preferred height.

Binding starts when both a scene and a skin are available. The implementation then waits for the scroll pane skin, finds the real content text node and binds `prefHeight` to a computation using `localToScreen(text.getLayoutBounds())` plus top and bottom offsets.

> **Careful.** Because `prefHeight` becomes bound by the control, do not bind or set that property for sizing. Put the area in a scroll pane or constrain its parent instead.

6. Layout guidance



The area grows vertically instead of showing scroll bars.

The control works best in forms where width is constrained and height can grow. Since wrapping is enabled in the constructor, the chosen width directly determines how many visual lines the text occupies and

therefore the computed preferred height.

Container	Guidance
`VBox`	Let the VBox use the area preferred height and avoid setting a fixed row height.
`ScrollPane`	Put the whole form in a scroll pane when long text can make the page taller than the viewport.
Grid layouts	Avoid binding prefHeight; use row constraints or surrounding scroll panes instead.

```
ExpandingTextArea notes = new ExpandingTextArea();
notes.setMaxWidth(420);
VBox.setVgrow(notes, Priority.NEVER);
```

Note. The implementation depends on skin lookup of standard TextArea internals. Test custom themes that replace the standard skin structure.

7. Styling

There is no dedicated user agent CSS file and no custom CSS properties. Style it like a normal JavaFX text area plus the root style class below.

Selector	Description
`.expanding-text-area`	root text area style class
standard TextArea selectors	inherited from JavaFX / application stylesheets

```
.expanding-text-area {
    -fx-font-size: 14px;
}

.expanding-text-area .content {
    -fx-padding: 8px;
}
```

8. Accessibility

The constructor sets `AccessibleRole.TEXT_AREA`. No generated accessible text is bound; screen readers use the normal text-area semantics.

9. Recipes

9.1 Use in a scrollable form

```
VBox form = new VBox(10, nameField, new ExpandingTextArea());
ScrollPane page = new ScrollPane(form);
page.setFitToWidth(true);
```

9.2 Limit horizontal growth

```
area.setMaxWidth(400);
parent.setFillWidth(false);
```

9.3 Checklist

1. Do not expect scroll bars inside the control.
2. Use inherited `TextArea` selection and text APIs.
3. Constrain width so wrapping creates predictable height.

10. See also

- Demo application: `com.dlsc.gemsfx.demo.ExpandingTextAreaApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.ExpandingTextAreaApp`)
- `ResizableTextArea` - manual user resizing instead of automatic height.
- `LimitedTextArea` - text area with length feedback.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/