LIBOCON19

Bringing the Sidebar Online

By Ashod Nakashian
Collabora Productivity
ash@collabora.com

Collabora Productivity

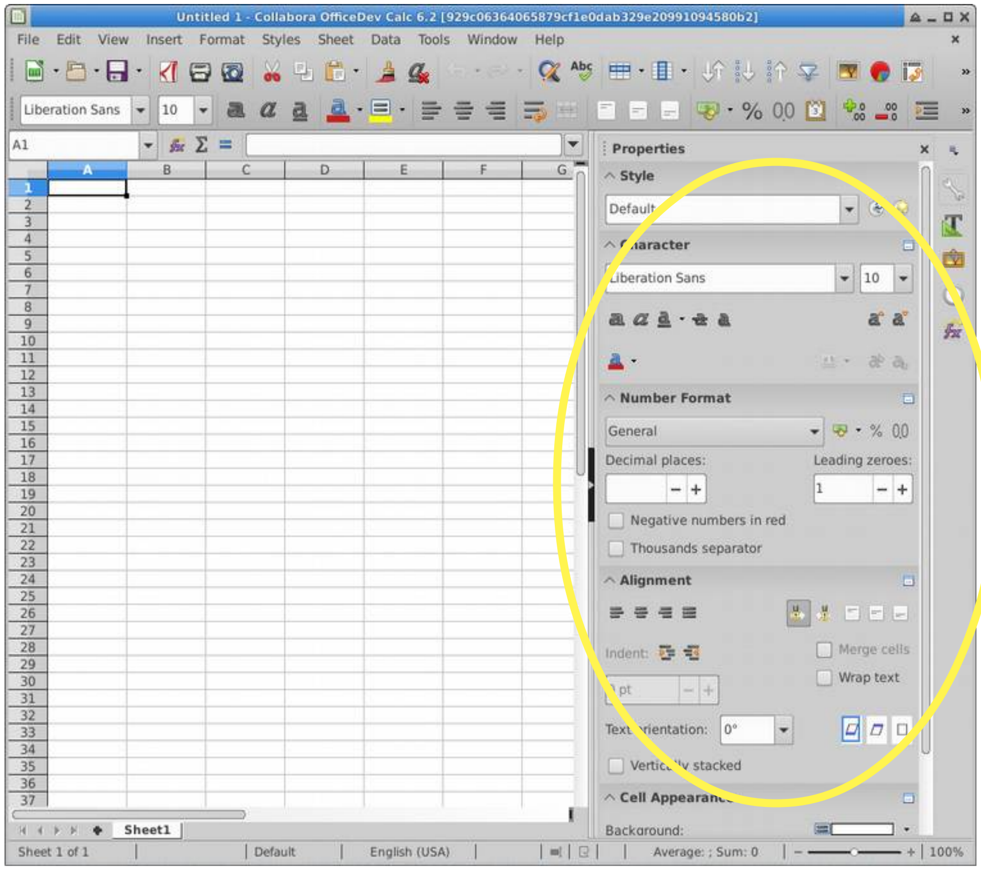
Overview

This talk in a nutshell

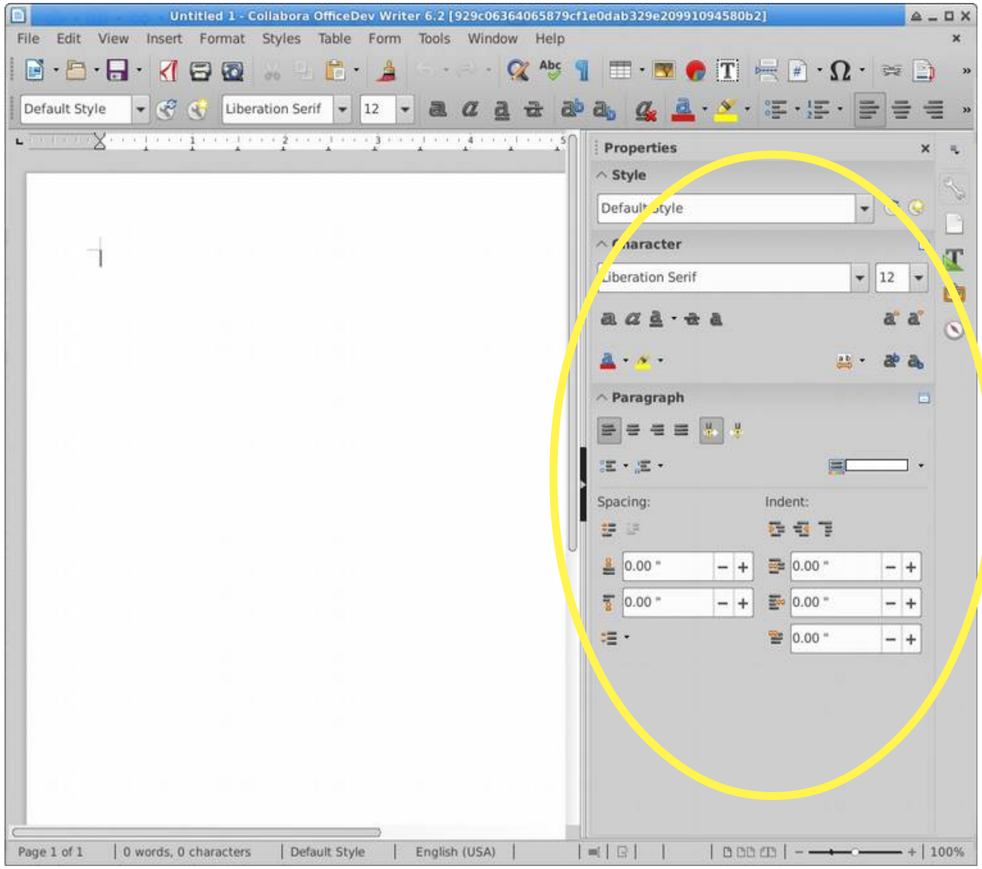
- No intro for your presenter today
- What’s Sidebar and why we need it
- What it takes to bring a UI feature to the web
- Bringing features to the web can be more challenging than it seems
- Technical details of the dialog tunnelling and Sidebar

Overview:
What is Sidebar?

Sidebar on the desktop


Untitled 1 - Collabora OfficeDev Calc 6.2 [929c06364065879cf1e0dab329e20991094580b2]
File Edit View Insert Format Styles Sheet Data Tools Window Help
Properties
Style
Character
Number Format
Alignment
Cell Appearance

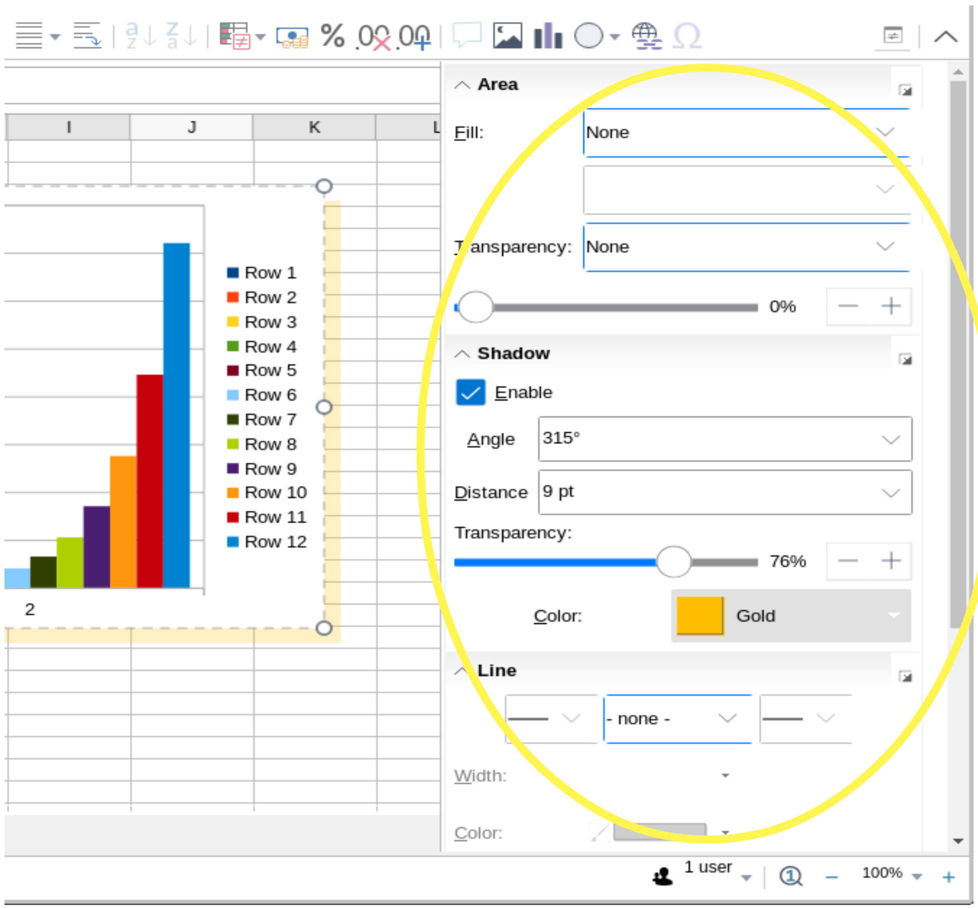
Liberation Sans 10
Default
General
Decimal places:
Leading zeroes: 1
Negative numbers in red
Thousands separator
Indent:
Merge cells
Wrap text
Text orientation: 0°
Vertically stacked
Background:
Sheet 1 of 1 | Default | English (USA) | Average: ; Sum: 0 | 100%


Untitled 1 - Collabora OfficeDev Writer 6.2 [929c06364065879cf1e0dab329e20991094580b2]
File Edit View Insert Format Styles Table Form Tools Window Help
Properties
Style
Character
Paragraph

Default Style
Liberation Serif 12
Spacing:
Indent:
0.00 "
0.00 "
0.00 "
0.00 "
0.00 "
Page 1 of 1 | 0 words, 0 characters | Default Style | English (USA) | 100%

Sidebar allows for quick-access to oft-used context-sensitive features.

Sidebar in Online



Area

Fill: None

Transparency: None

0%

Shadow

✓ Enable

Angle 315°

Distance 9 pt

Transparency: 76%

Color: Gold

Line

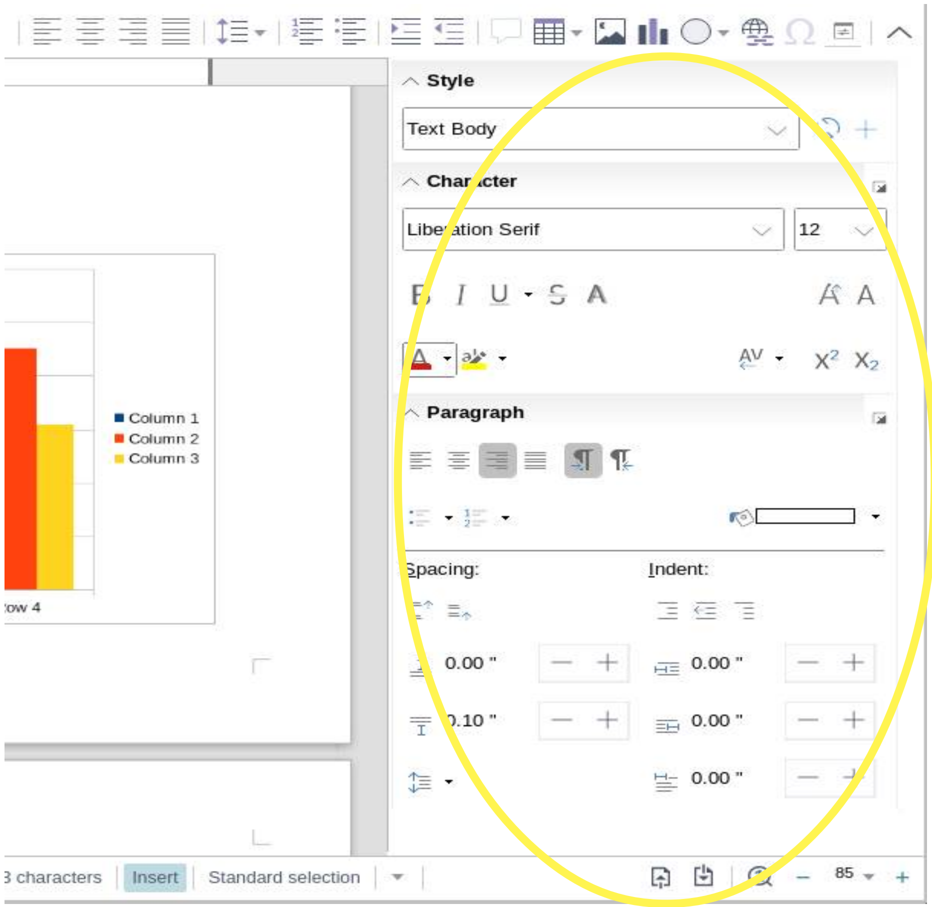
- none -

Width:

Color:

1 user

100%



Style

Text Body

Character

Liberation Serif 12

Paragraph

Spacing: 0.00 " 0.10 "

Indent: 0.00 " 0.00 " 0.00 "

3 characters | Insert | Standard selection

85

Sidebar allows for quick-access to oft-used context-sensitive features.

Thanks to Collabora partners

Sidebar in Online: how hard can it be?

How hard can it be?

Challenges

- Superficially, the Sidebar is a type of dialog;
    - But one that is persistent;
        - Unless the user dismisses.
- And being context-sensitive, auto-updates on selection change;
    - Which may change its height;
        - Which needs overflow handling.
- Unlike dialogs, it has to resize with the window, as it’s embedded in it;
    - And when visible, it needs to push the contents to the left;
        - And maximize content area when hidden.

Tunnelling

Dialog Tunnelling: an introduction

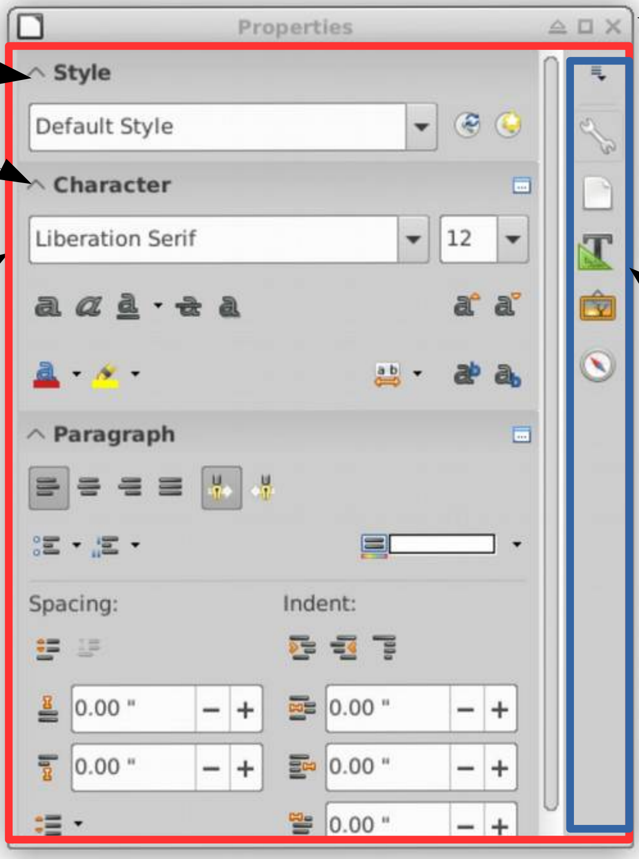
- Each dialog gets its unique ID at creation
- Dialog activity notified via callbacks to the client
  - Callbacks are translated into ‘window’ messages to the client
- Mouse and keyboard input are sent as events to Core;
  - These generate new notifications, such as invalidation of the UI
- The client reacts to the notifications by updating UI elements
- The client requests ‘windowpaint’ to get the dialog as an image
  - The image (PNG) is rendered on the screen

Sidebar as a special kind of dialog

Reuse and extend dialog infrastructure in Online

- When creating Sidebar, use a different ‘type’ of window creation
- In Online, flag Sidebar window to differentiate from dialogs
    - Don’t close Sidebar automatically when otherwise dialogs close
- Sidebars are visually docked on the right (currently fixed)
- Handle long Sidebars by overflowing the rendered image
- Notify and handle browser resize by notifying LO Core

Anatomy of Sidebar



Panels

Floating

TabBar
(hidden for Online)

SidebarDockingWindow
- SidebarController
  - Decks
    - Panels
  - TabBar

Of course SidebarChildWindow is the *parent* of SidebarDockingWindow

Which ‘window’ is the Sidebar?

Finding the right level to tunnel

- Since the Sidebar is really a set of Decks, first try was to tunnel the Decks
- Turned out this wasn’t ideal because Decks are toggled
- Transitioning between Decks had to be handled in JavaScript
- Nightmare to stay in lock-step with Core
- Leaves us with the artefact of having Sidebar window type as ‘deck’
- Tunnel `SidebarDockingWindow` instead

Implementation Design

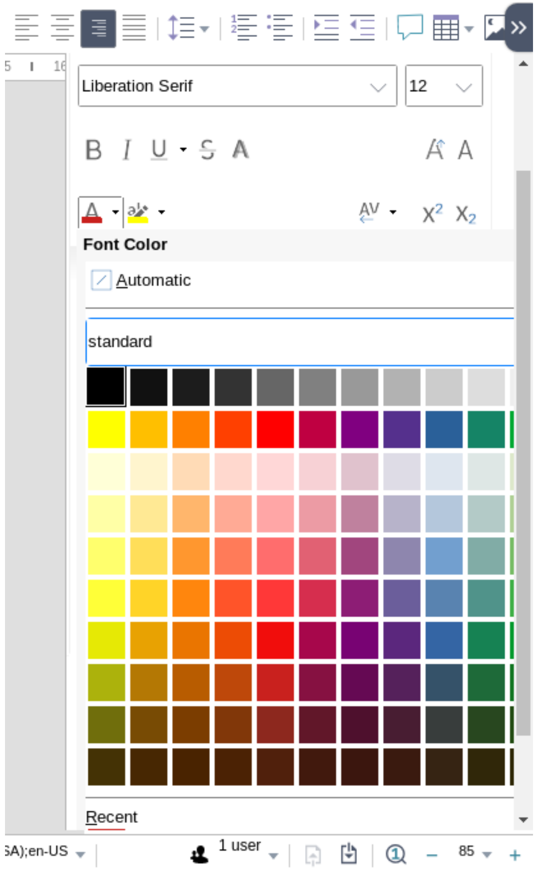
Technical details

- To support resizing (primarily height) we detach the Sidebar (float)
- Implement a new LoKit API to support resizing ‘window’ objects
  - Possibility to have the width resized via the UI in the future
- Hide TabBar: we control the visible Deck via `.uno` commands.
  - Account for the lack of TabBar when layouting
  - On Deck change, we notify the state of the hidden/shown Decks
- Maximize the height to scroll in the browser (more soon)

Child windows

Handling context menues and drop-down lists

- Unique IDs for each child window
- Child windows refer to their ‘parent’ window
- But the child window has its own HTML div node
- Child windows are auto-close; identical to desktop



Challenges

Challenges

Fun and unexpected behavioural challenges

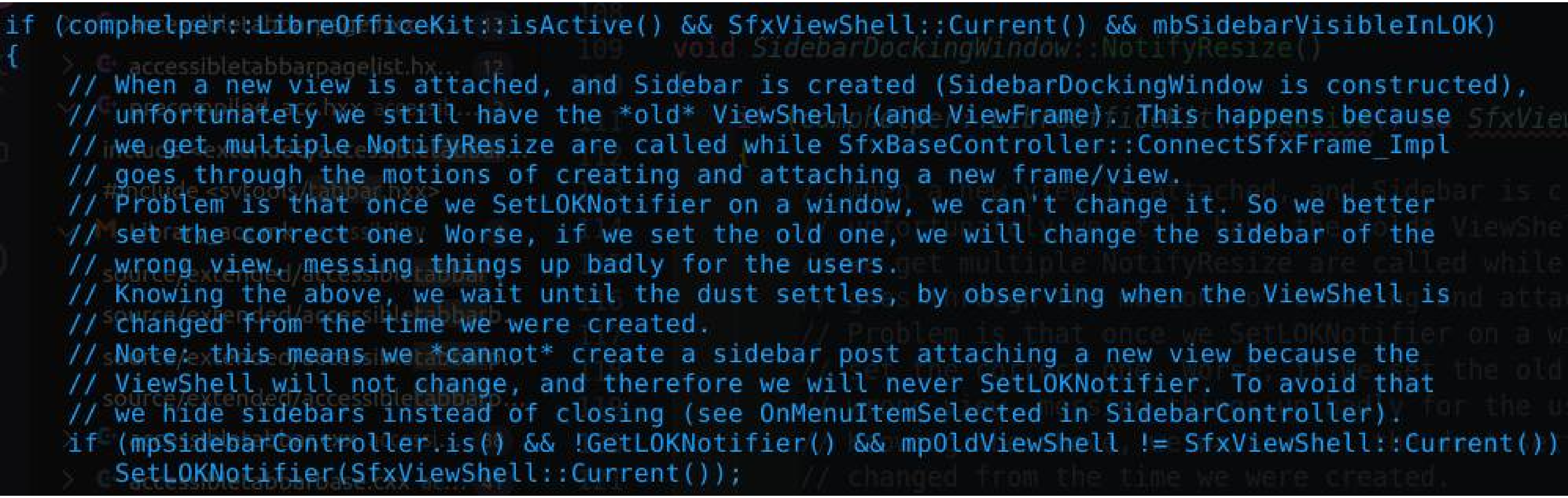
- Order of events from Core can be inverted
  - e.g. Window ‘invalidate’ issued before ‘created’
  - So we issue ‘created’ from NotifyResize()
- Window dimensions change many times before it settles;
  - Multiple ‘created’ events created; must avoid UI flicker etc.
- Sidebar can steal the input focus, since it’s not dismissed
- Impress has a different initialization workflow than Writer and Calc
  - Continued...

Challenges

ViewShell, FrameView, and LOKNotifier

- In Impress the `ViewShell` and `FrameView` change after `SidebarDockingWindow` is created;
  - `SidebarDockingWindow` is created using the previous user's `ViewShell`
- Calc and Writer don't have this oddity
- We need to support multiple-views, each view with its own Sidebar
- The notifier of the current view is set on the `ViewShell`;
  - So having the wrong `ViewShell` means the wrong user will see the updates of another user

Challenges



```
if (comphelper::LibreOfficeKit::isActive() && SfxViewShell::Current() && mbSidebarVisibleInLOK)
{
    // When a new view is attached, and Sidebar is created (SidebarDockingWindow is constructed),
    // unfortunately we still have the *old* ViewShell (and ViewFrame). This happens because
    // we get multiple NotifyResize are called while SfxBaseController::ConnectSfxFrame_Impl
    // goes through the motions of creating and attaching a new frame/view.
    // Problem is that once we SetLOKNotifier on a window, we can't change it. So we better
    // set the correct one. Worse, if we set the old one, we will change the sidebar of the
    // wrong view, messing things up badly for the users.
    // Knowing the above, we wait until the dust settles, by observing when the ViewShell is
    // changed from the time we were created.
    // Note: this means we *cannot* create a sidebar post attaching a new view because the
    // ViewShell will not change, and therefore we will never SetLOKNotifier. To avoid that
    // we hide sidebars instead of closing (see OnMenuItemSelected in SidebarController).
    if (mpSidebarController.is() && !GetLOKNotifier() && mpOldViewShell != SfxViewShell::Current())
        SetLOKNotifier(SfxViewShell::Current());
```

Challenges

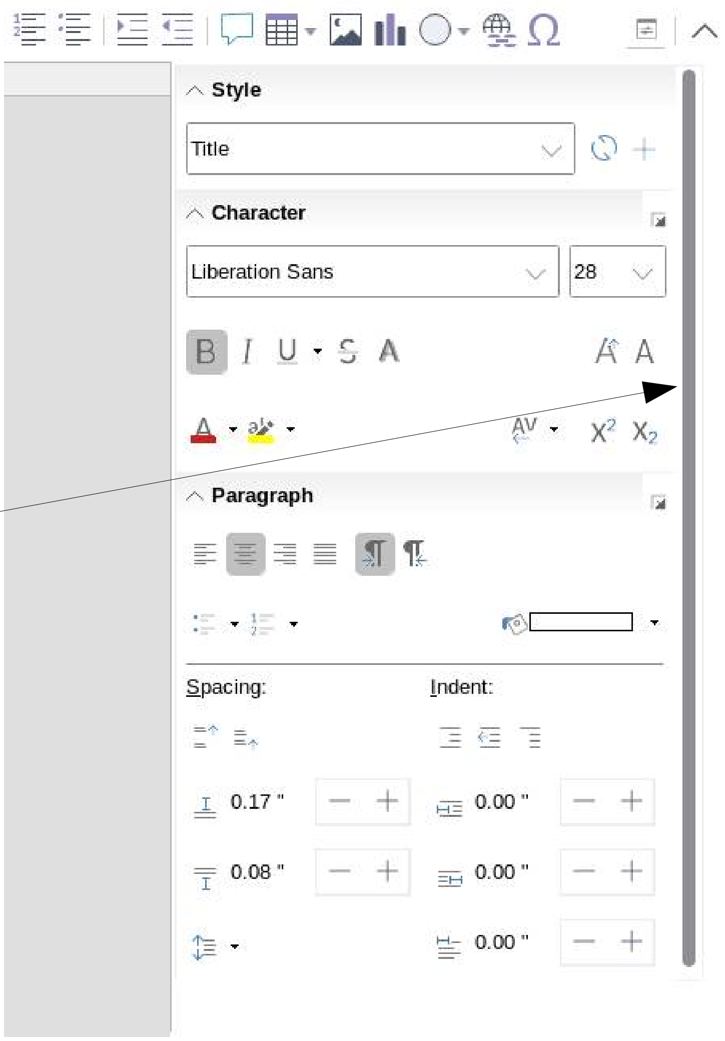
Vertical Scrollbar

- Scrolling in Core is extremely slow and inefficient
- To avoid it, make the Sidebar large enough to avoid scrollbars
- Render the complete Sidebar and overflow in the browser
- But how large should the Sidebar be to avoid scrollbars?
  - Greedy Panels resize to fill all available space!
  - Edge cases mean the scrollbar can rear its UN-beautiful head
  - Multi-pass layouting is needed to avoid this
  - And we need to cap the height for Decks with greedy Panels

Challenges

Vertical Scrollbar

Useless—but stubborn—scrollbar that can move only a few pixels



Collabora Productivity

Thank you!

<Your Question Here>

By Ashod Nakashian

ash@collabora.com