Collabora

LibreOffice
The Document Foundation

SUSE

Re-using the Sidebar on phones

Collabora Office

OPENSUSE-LIBREOFFICE CONF'20

2020

By Szymon Kłos
Software Engineer at Collabora Productivity

Includes MySQL

LINUX MAGAZINE

datto

ADMIN Network & Security

fedora

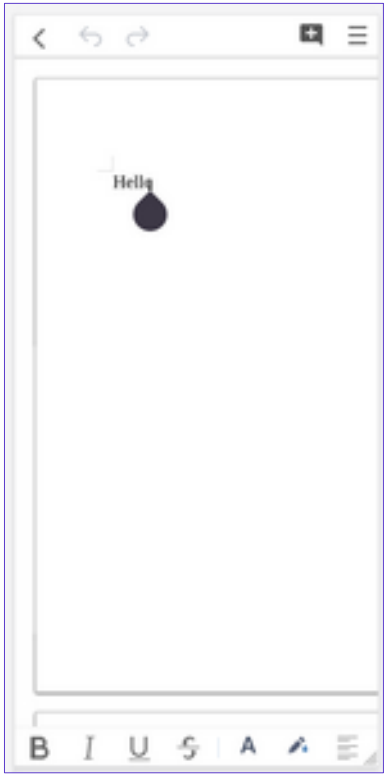
TUXEDO COMPUTERS

.riss applications

Required Magic
advanced technology

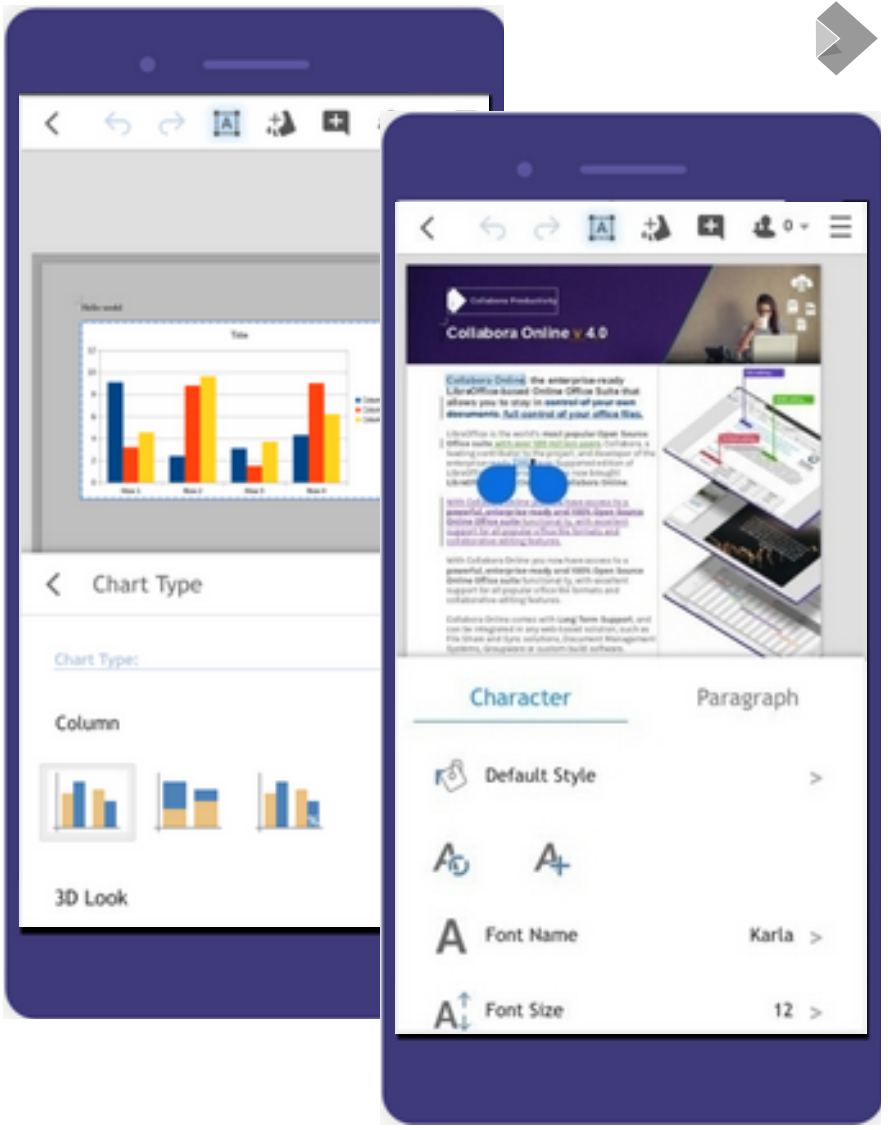
Online on the mobile phone in the past

- It was simple viewer app
- Only basic editing functionality like font color, bolding, align options, etc.
- Hamburger menu opened small menu on the top, not much comfortable on small mobile screen



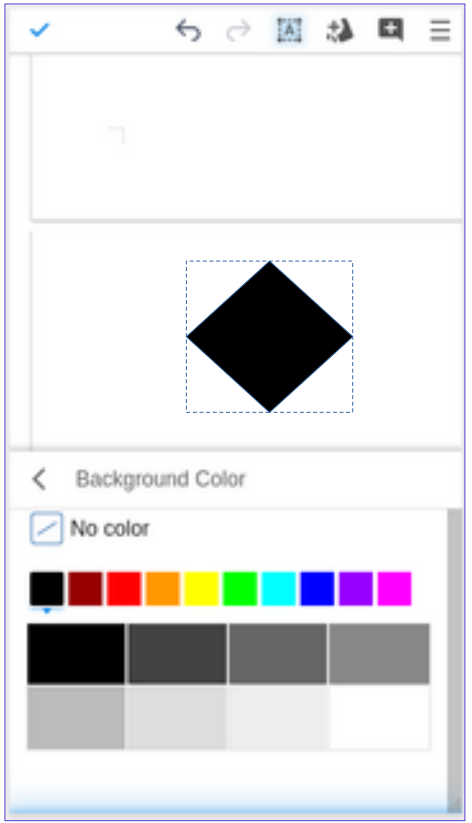
New menus

- New menus made Online a powerful fully-featured editor
- Increased productivity and user experience thanks to mobile friendly design

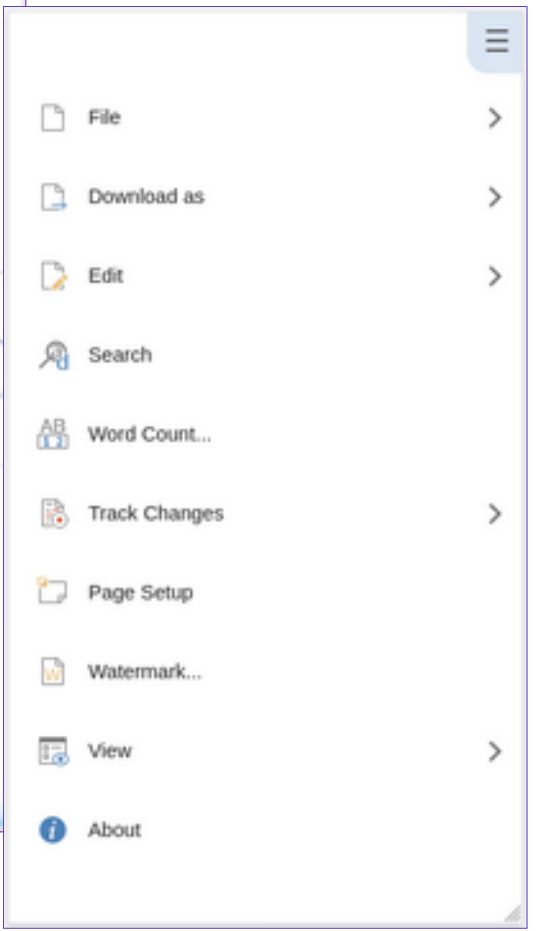


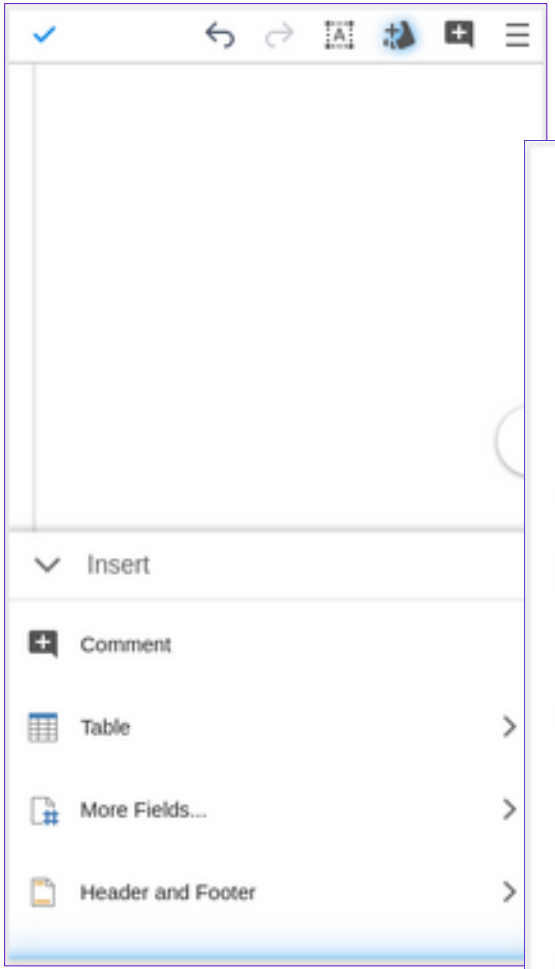
New menus

Color picker



Hamburger menu

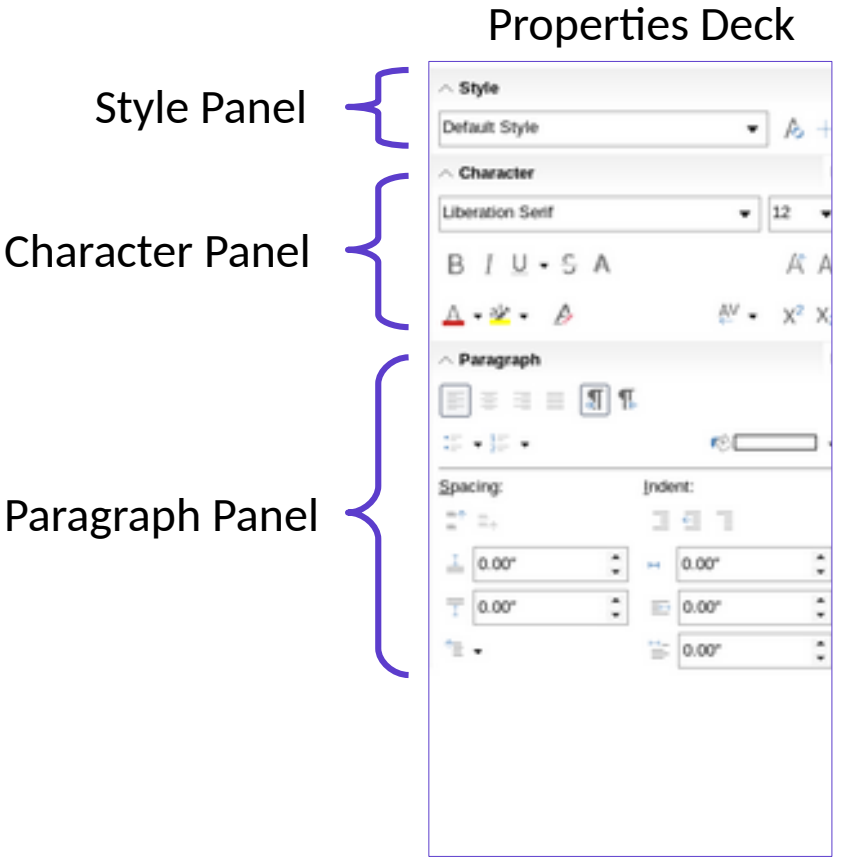




Insert menu

Sidebar

- Context dependant tools palette
- Organized in Decks and Panels
- On desktop (in Online) transferred as an image – doesn't fit mobile experience



UI defined by the server

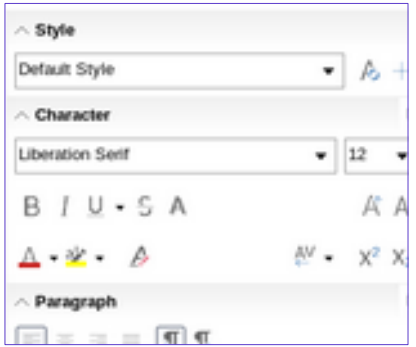
SERVER

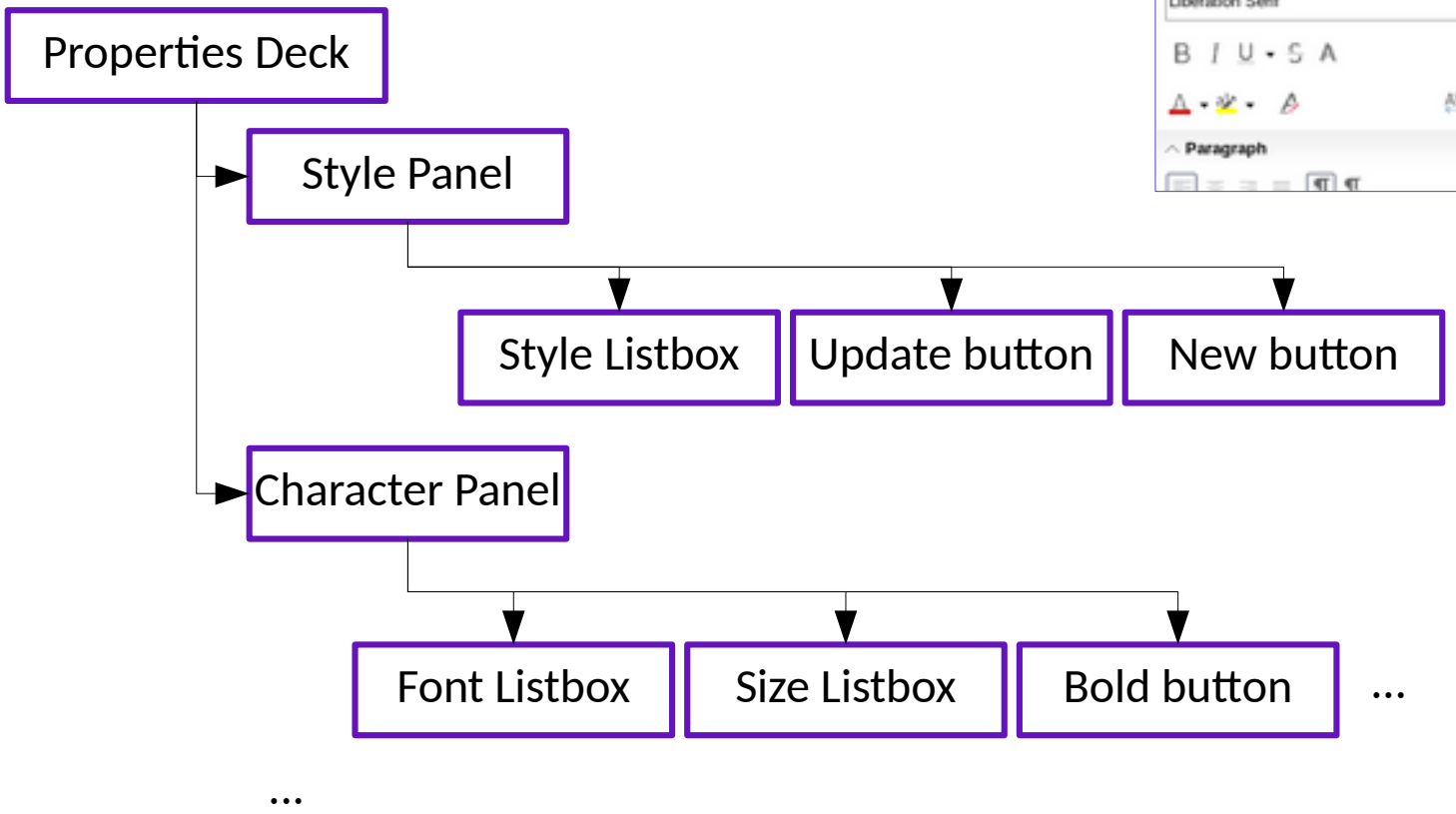
CLIENT

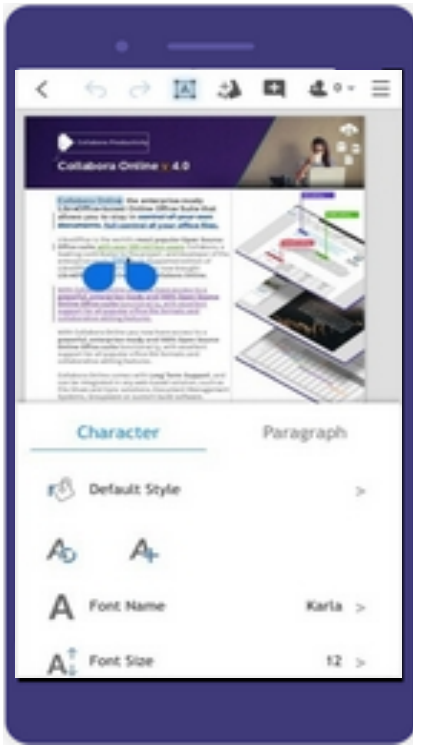
JSON (UI description)

Interface builder

Core side: JSON generation







Core side: JSON generation

- **Every control type has *DumpAsPropertyTree* method which dumps specific properties for a given type of a control with current values**
- **It is a tree-like structure describing whole dialog / sidebar**
- **Container's children have additional *top* and *left* properties with coordinates defining the placement**

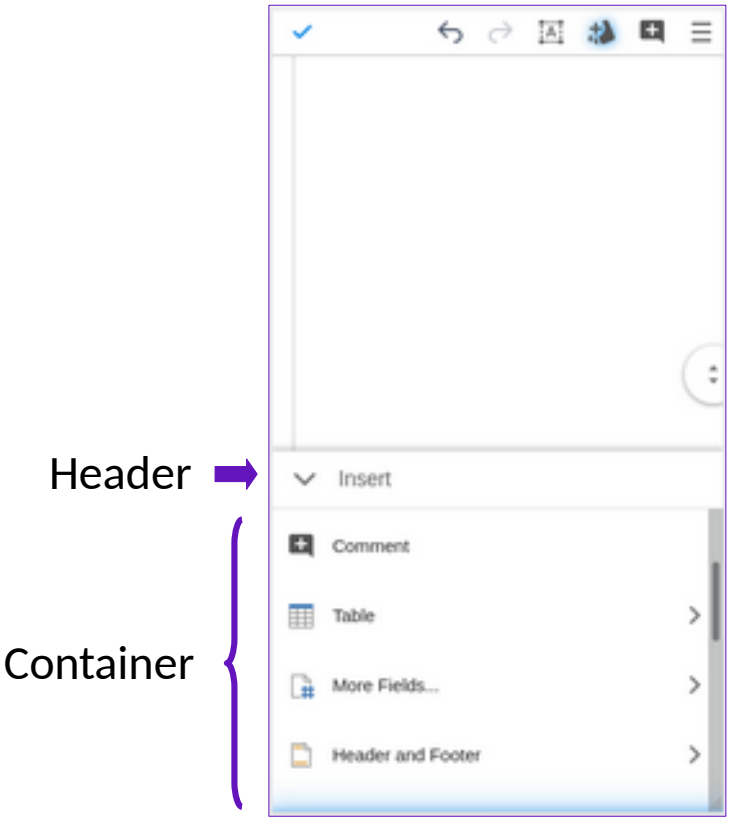
```
{
    "id": "box",
    "type": "container",
    "children": [
        {
            "id": "confirm",
            "type": "checkbox",
            "enabled": "true",
            "text": "I confirm",

            "left": "0", // position
            "top": "1" // in the container
        },
        ...
    ]
}
```

Core side: JSON generation

- In the implementation of SidebarDockingWindow there is a special timer which triggers JSON generation in the idle time (SidebarNotifyIdle)
- JSON is sent only if changed the state from the last time
- It uses LOK Callback to send the message: LOK_CALLBACK_JSDIALOG with an id of the sender window
- Performance issue: boost::property_tree::ptree replaced with JSONWriter (Noel Grandin)

Online side – code perspective

- **Code split into container (Control.MobileWizard.js) and builder (Control.JSDialogBuilder.js)**
- **Container code handles the header (shows back or exit buttons, title or tabs)**
- **Container handles traversal of the menu structure (*goLevelDown*, *goLevelUp* functions)**



Navigation

- **Handled back button on mobile devices – History API**
- **Opening the menu or entering the next level pushes new state to the history**
- **On back button we go up one level or we close the panel**

Online side – code perspective

- **Builder has the main function: *build* which parses the JSON and inserts widgets, each in a new row**
- **Has a register with handler functions for supported widget types**
- **Also binds custom handlers for particular uno commands**

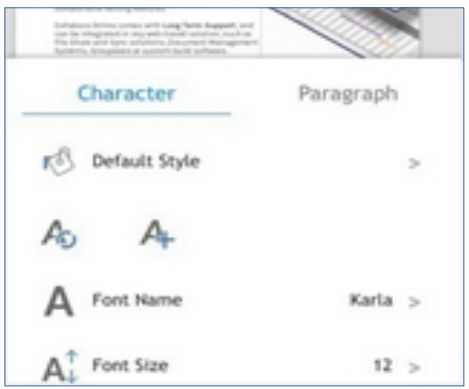
Control.JSDialogBuilder.js

```
_setup: function(options) {
        this._clearColorPickers();
        this.wizard = options.mobileWizard;
        this.map = options.map;
        this.callback = options.callback ? options.callback : this._defaultCallbackHandler;

        this._controlHandlers['radiobutton'] = this._radiobuttonControl;
        this._controlHandlers['checkbox'] = this._checkboxControl;
        this._controlHandlers['spinfield'] = this._spinfieldControl;
        this._controlHandlers['metricfield'] = this._metricfieldControl;
        this._controlHandlers['formattedfield'] = this._formattedfieldControl;
        this._controlHandlers['edit'] = this._editControl;
                        ...
        this._toolitemHandlers['.uno:XLineColor'] = this._colorControl;
        this._toolitemHandlers['.uno:SelectWidth'] = this._lineWidthControl;
        this._toolitemHandlers['.uno:FontColor'] = this._colorControl;
        this._toolitemHandlers['.uno:BackColor'] = this._colorControl;
        this._toolitemHandlers['.uno:CharBackColor'] = this._colorControl;
        this._toolitemHandlers['.uno:BackgroundColor'] = this._colorControl;
        this._toolitemHandlers['.uno:FrameLineColor'] = this._colorControl;
        this._toolitemHandlers['.uno:Color'] = this._colorControl;
        this._toolitemHandlers['.uno:FillColor'] = this._colorControl;
        this._toolitemHandlers['.uno:ResetAttributes'] = this._clearFormattingControl;
        this._toolitemHandlers['.uno:SetDefault'] = this._clearFormattingControl;

        this._toolitemHandlers['.uno:InsertFormula'] = function () {};
        this._toolitemHandlers['.uno:SetBorderStyle'] = function () {};
        this._toolitemHandlers['.uno:TableCellBackgroundColor'] = function () {};

        this._currentDepth = 0;
},
```

How it look in online

- If we have only two Panels → shows tabs, in onther cases we show explorable entries

- For mobile friendly usage we position every element in a separate row, only simple buttons are stacked in the rows



Tabs

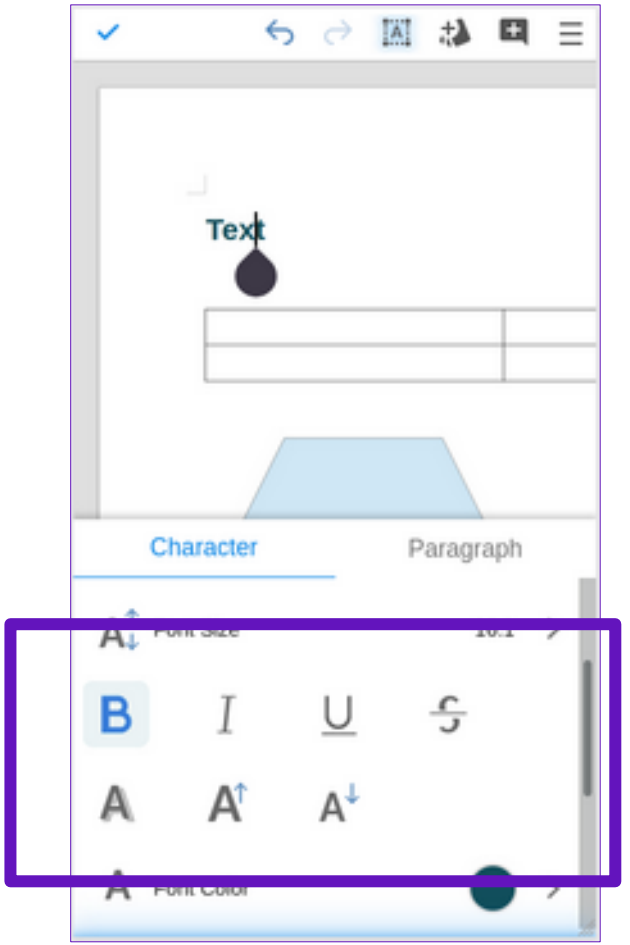
Exploring Panels (Area, Line, Pos. and Size)

UNO buttons

- **Selection for activated buttons – follows UNO command status**

- **Sends UNO command on click**

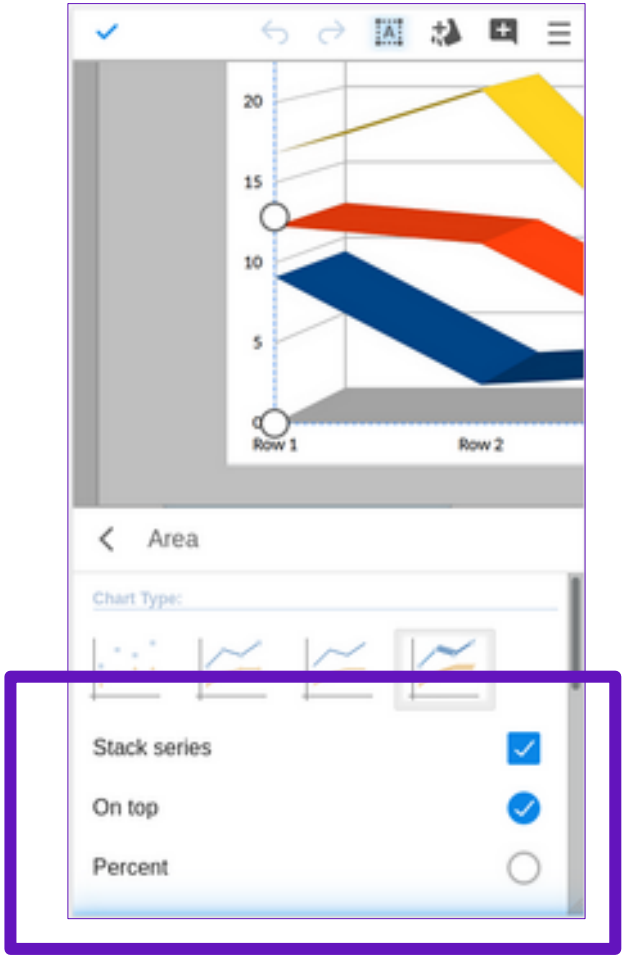
```
{
        id: xxxxxx,
        type: “toolitem”,
        text: “XXXXXXX”,
        command: “.uno:XXXXXX”
}
```



Checkboxes and Radio Buttons

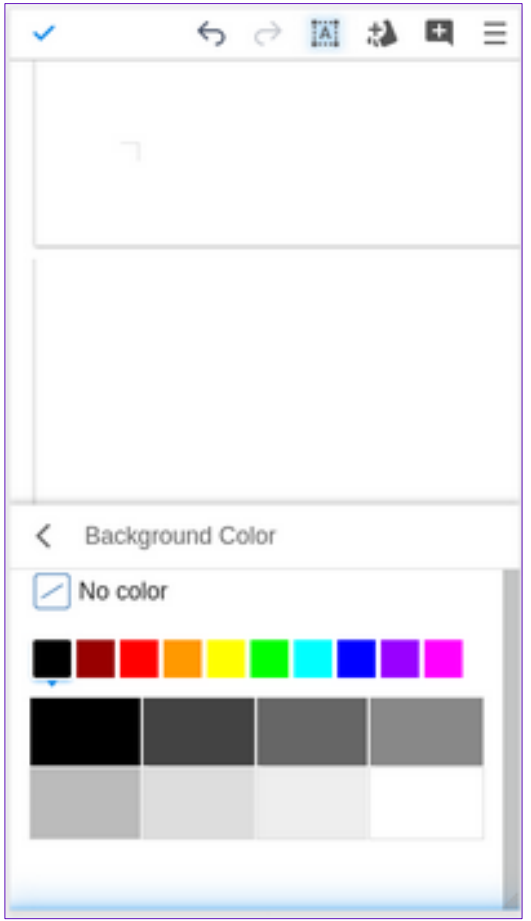
- **Checkboxes and radio buttons in mobile style**
- **Easy to use with one hand**

```
{
        id: xxxxxx,
        type: “checkbox”,
        checked: “true”
}
```



Color Picker

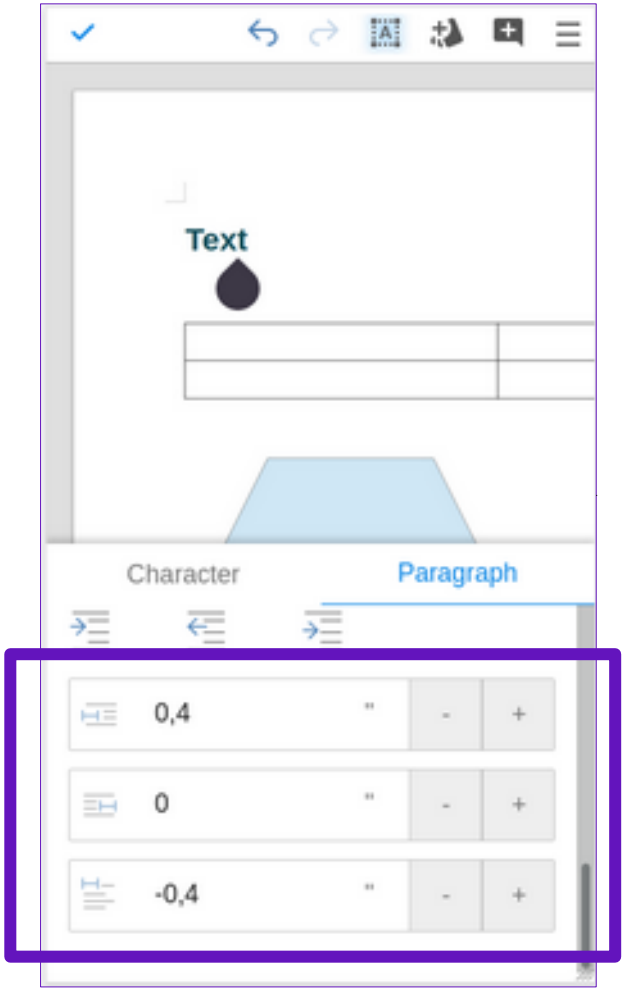
- Easy to tap on touch screen
- As a result sends associated uno command with the new value
- Replaces split buttons with dropdowns for picking the color



Metricbox

- **Input field with icon**
- **Has unit**
- **Manipulation buttons to increase and decrease values**

```
{
        id: xxxxxx,
        type: “listbox”,
        value: “0.4”,
        unit: “””,
        min: “0”,
        max: “10”
}
```



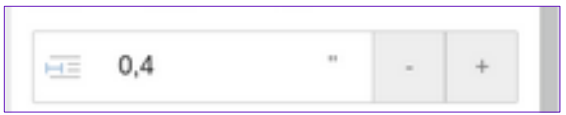
Listboxes

- **Listboxes converted into explorable entries**

```
{
        id: xxxxxx,
        type: “listbox”,
        entries: [
                {id: N, name: “yyyyyy”}
        ],
        selectedEntries: [
                N
        ]
}
```

Two way comunication

- **HTML events are catched and sent to the server**



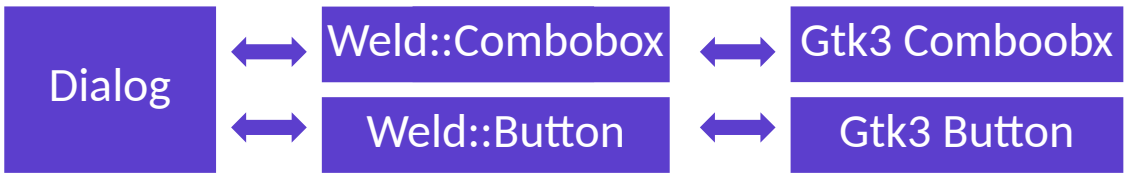
window id

```
dialogevent 3 {
        "id":"XXXXXX",
        "cmd": "plus",
        "data": "...",
        "type": "spinfield"
}
```

- **Backend uses TestUI wrappers to execute actions (also used in Python UI tests)**
- **After event is processed JSON generation is triggered and sent back to the client**
- **This approach has disatvantage: we don’t receive updates if something happens asynchronously**

Welding in vcl

- Way to use native controls on desktop, implemented by Caolán McNamara
- Currently gtk3 only implementation
- Adds an abstraction level which is a bridge between native UI and LibreOffice dialogs code



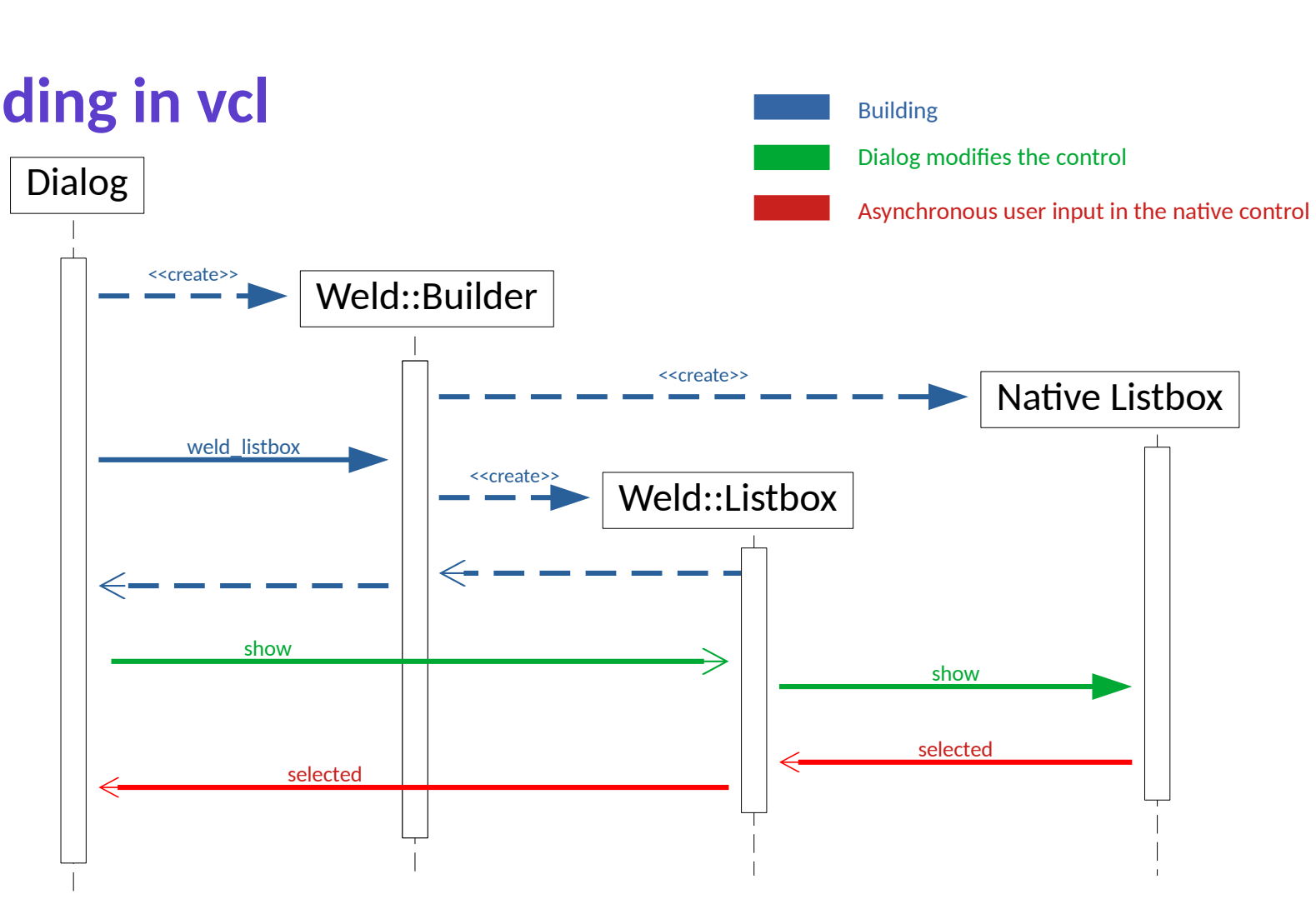
Welding in vcl

Advantages of using welded wrappers

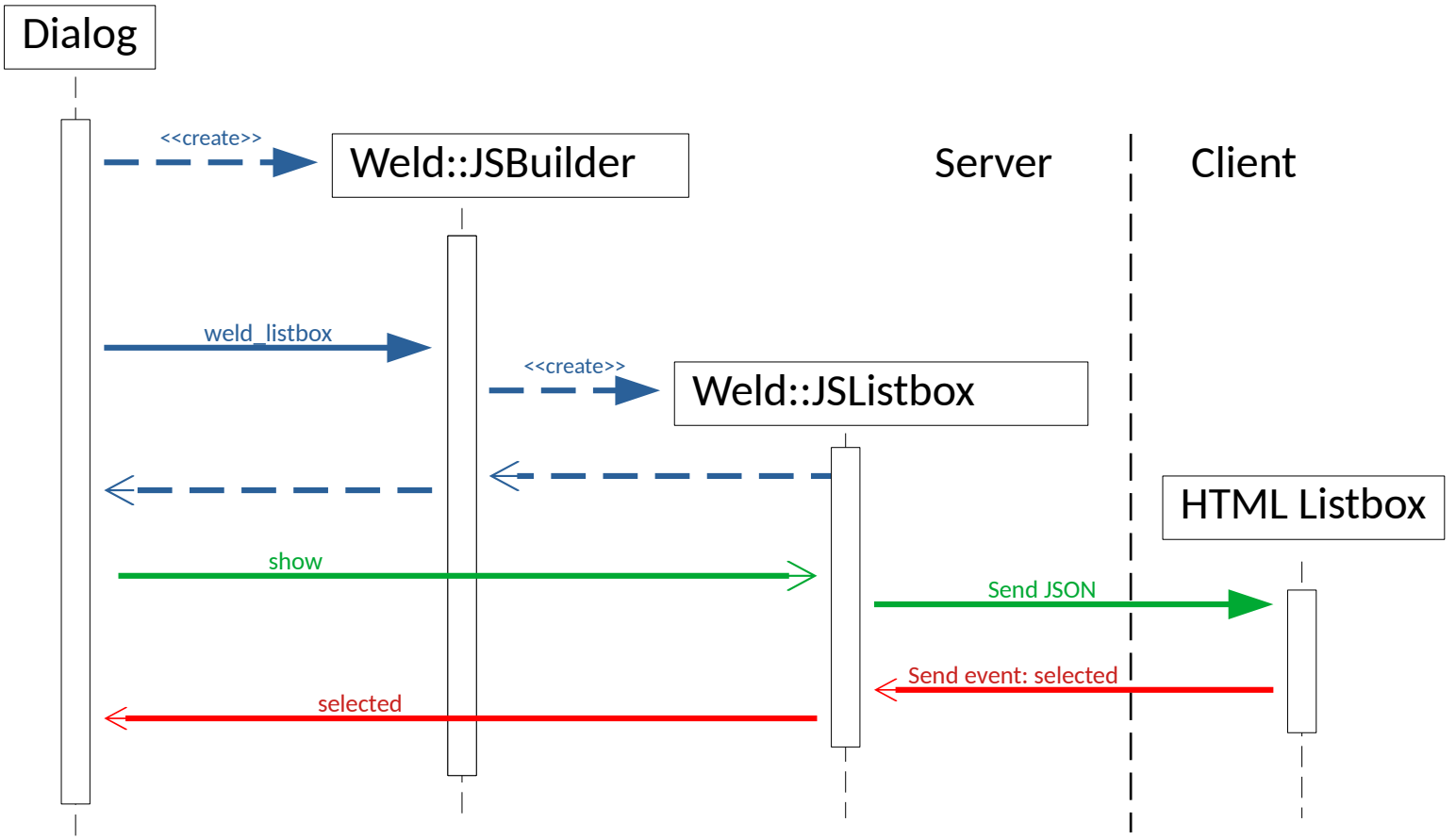
- We receive information about every modification of a control
- We can inform parent dialog about input performed by user eg. mouse press

```
namespace weld
{

class VCL_DLLPUBLIC Widget
{
protected:
    ...
    Link<const MouseEvent&, bool> m_aMousePressHdl;
    ...

public:
    virtual void set_sensitive(bool sensitive) = 0;
    virtual bool get_sensitive() const = 0;
    virtual void show() = 0;
    virtual void hide() = 0;

...
    virtual void connect_mouse_press(...)
```

Welding in vcl

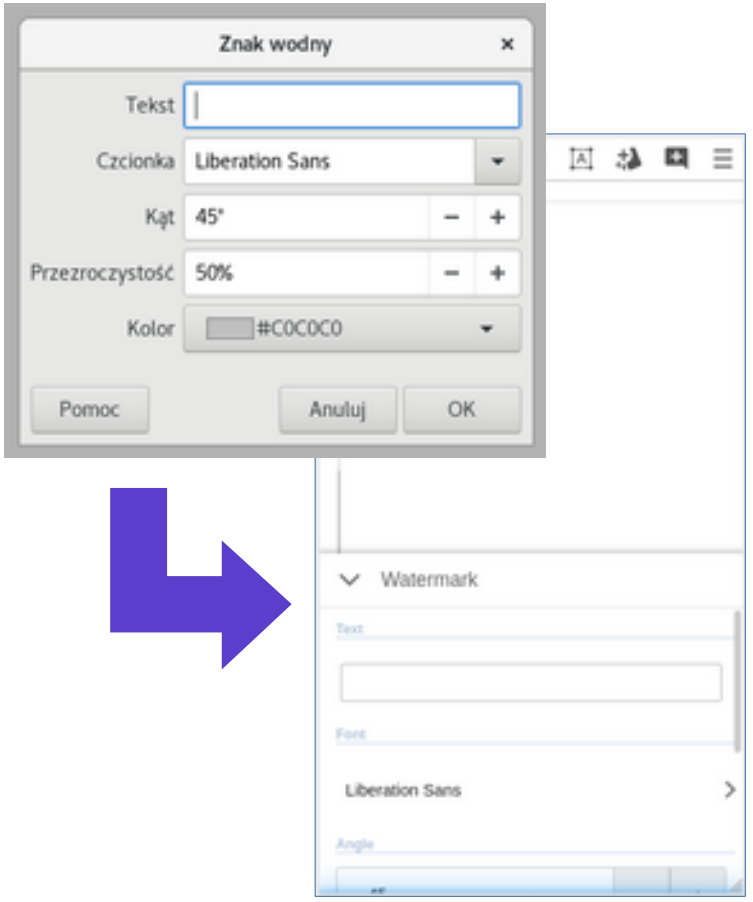


Using welding in online



Dialogs on mobile using welded widgets

- Code in **vcl/jsdialog**
- Implemented weld::Builder and control wrappers
- JSInstanceBuilder is injected only for few dialogs we want in online, only on mobile devices.
- Wrappers send JSON on control change
- **vcl/jsdialog/executor.cxx** code executes commands received from a client eg. when something is selected in a listbox



Collabora Productivity

Thank you!

- You can check new UI on mobile on your own:
- Download “Collabora Office” form Apple / Play Store

By Szymon Kłos
szymon.klos@collabora.com