Collabora LibreOffice The Document Foundation SUSE

Collabora Office on iOS Autumn 2020 update

Collabora Office

OPENSUSE-LIBREOFFICE CONF'20

Tor Lillqvist

Software Engineer at Collabora Productivity

@TorLillqvist

Includes MySQL LINUX MAGAZINE datto ADMIN Network & Security fedora TUXEDO COMPUTERS .riessapplications Required Magic advanced technology

Introduction

I am Tor Lillqvist. I work for Collabora as a contractor

- I have worked on LibreOffice and its predecessor for over ten years.
- I started hacking on cross-compilation of LibreOffice to iOS and Android back in 2011, I think it was.
- I enjoy travelling by train and photography. And beer.
- I am an atheist and a feminist.
- I live in Helsinki, Finland, with family, including a dog.

The app

Collabora Office has been available in the iOS App Store since May 2019

- Initially developed for iPad (because that is what our customer used) and intentionally did not even install on iPhone.
- Later was changed to install also on iPhone.
- Thanks to Adfinis for funding part of this work.

The iOS app shares most of the UI of Collabora Online. Also the C++ code of the server part of Collabora Online is used. Plus most of the LibreOffice core code.

- The app thus benefits from all improvements to the UI of Collabora Online, and from improvements to upstream LibreOffice, like import and export filter improvements.
- Currently using a 6.2-based vendor branch, soon 6.4.
- Lots of careful cherry-picks from more recent upstream, of course.

Need to point out again that despite the app being based on Collabora Online, there is no "online" aspect to it. All editing happens locally on the device. There is no server, as there is for Collabora Online.

- The documents to be edited can be located either locally on the device, or in iCloud, or on a server through an iOS file provider extension like that of NextCloud.
- (The Nextcloud iOS app is a quite different scenario. In that you are using “normal” Collabora Online embedded in the Nextcloud app’s WebView.)

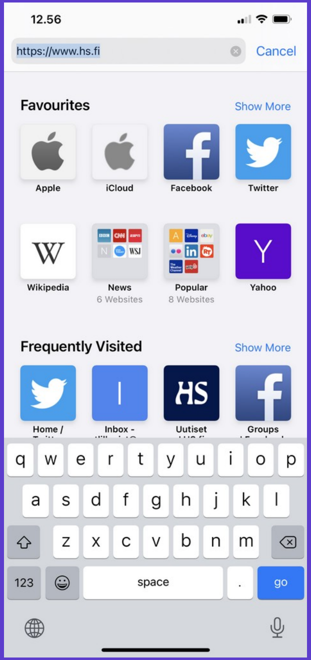
A few things I have worked personally on recently
1. Keyboard issues

Keyboard issues

- Especially in an iOS app that is based on WebView and most of the UI thus is in JavaScript, handling the keyboard can be a royal pain.
- Solution: CollaboraOnlineWebViewKeyboardManager (COKbdMgr).

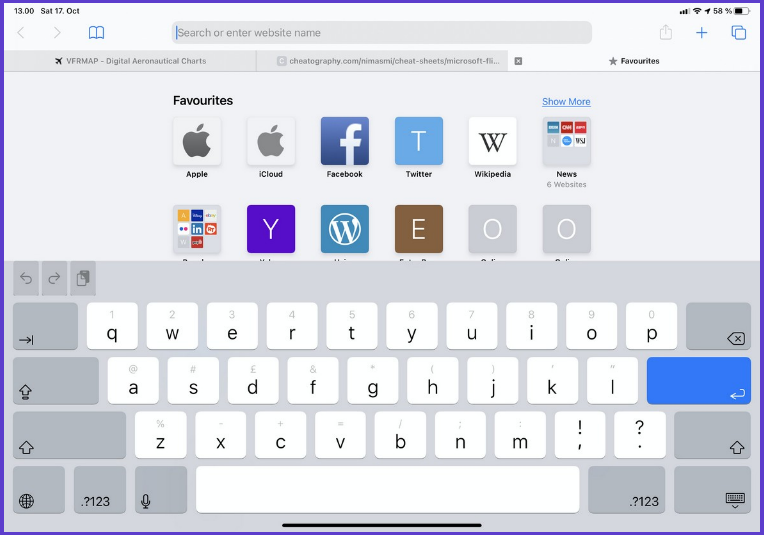
Multiple kinds of keyboards in iOS

- On-screen keyboard on iPhone. Very small (obviously).



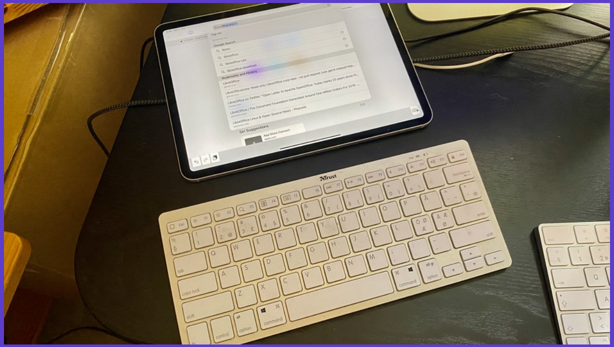
Multiple kinds of keyboards in iOS

- On-screen keyboard on iPad. Much larger, with some "function" keys.



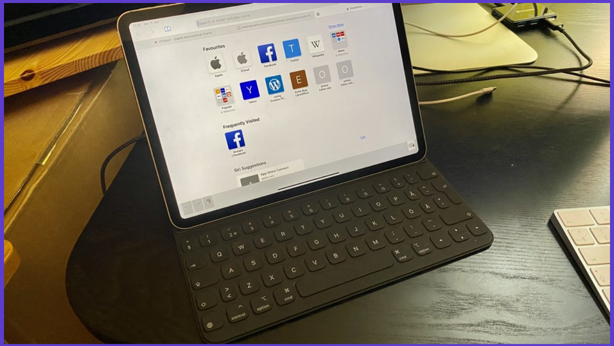
Multiple kinds of keyboards in iOS

- Hardware keyboard that connects with Bluetooth (battery-powered). Large number of manufacturers and models.



Multiple kinds of keyboards in iOS

- Hardware keyboard attached to the *Smart Connector* on iPad. The keyboard is powered from the iPad. No Bluetooth. Apple's own *Smart Keyboard Folio* and *Magic Keyboard for iPad* plus 3rd-party ones.



On-screen keyboard in “normal” cases

- In “normal” apps, and when viewing “normal” web pages in Safari, the on-screen keyboard pops up when you tap into a text field.
- But Collabora Online is not a very normal web page, and the Collabora Office app with Collabora Online in a WebView is not a very normal app.

The on-screen keyboard problem

- Often, either the on-screen keyboard was showing even if you were not doing any text input. On iPad there is a handy button on the on-screen keyboard to dismiss it. Not on iPhone.
- Equally often, the keyboard was not showing even if you needed it. This clearly is more serious. By tapping around a bit you usually got it eventually, though.

How the JavaScript code handles the keyboard

- The JavaScript UI code attempts to make the on-screen keyboard show up and hide by assigning focus and removing focus from a hidden text field in the page.
- This worked to some extent but not always. Also, the code has evolved for a long time and is quite complex and not easy to understand. Most development happens on desktop browsers and unintended side-effects in the iOS app are noticed only much later.

Solution: Do it in native iOS code

- In the iOS app case, don’t try to do the focus dance at all.
- Instead, call into native iOS code to either explicitly display the on-screen keyboard, or hide it.
- A separate so-called framework, CollaboraOnlineWebViewKeyboardManager, intended to be used by the Nextcloud iOS app, too, when it uses Collabora Online to edit a document.

Solution: Do it in native iOS code 2

- In theory, quite simple. In practice, various complications.
- It turns out that the JavaScript code loves to send a quick succession of display and hide requests. For things to work as intended, hide requests that quickly follow a display request need to be ignored. Etc.
- COKbdMgr uses a (hidden) UITextView and calls its becomeFirstResponder to display the on-screen keyboard.

Solution: Do it in native iOS code 3

- Extra benefit: word completion suggestions. Works only based on what you have typed that time through CollaboraOnlineWebViewKeyboardManager, though.
- To get suggestions based on existing text in front of the caret in the document would require some more work, in core too, but is definitely a possibility.

Solution: Do it in native iOS code 4

- Another benefit: Handling of shortcuts on a hardware keyboard is now much more straightforward.
- Handling ⌘-A, ⌘-C, ⌘-V, ⌘-X etc in the JavaScript often regressed (partially because on other platforms shortcuts use Ctrl, not Cmd) and was not noticed in time.
- Simple to explicitly catch them in COKbdMgr and call out to the JavaScript to call back to core to do the appropriate thing.

A few things I have worked personally on recently
2. Password-protected ODF documents

NSS needed

- Until recently, LibreOffice core for iOS (and Android) was built without the external nss (Netscape Security Services) module.
- The NSS build system is a nightmare and adapting it for cross-compilation was seen as too painful.
- Until a customer actually wanted to be able to open password-protected documents.

NSS needed 2

- LibreOffice core for the iOS app is built completely statically. No dynamic libraries. (Because originally such were not allowed in iOS apps. Nowadays they would need to be packaged as frameworks according to Apple’s rules, and that seems pointless for little gain.)
- Unmodified NSS loads a couple of libraries dynamically in every case, though. Fixing that required some patching.

A few things I have worked personally on recently
3. Multitasking

In iOS 13, Apple introduced “multitasking” to the iPad UI

- “Multitasking” is here not used in the OS Internals 101 sense; the iOS kernel and apps run fully multi-tasked since day one.
- It means that the same app process can have multiple instances, visible either in “Split View” at the same time, or in separate “workspaces” (think “virtual desktops” on Linux desktop).
- Implementing this would have been easy if it hadn’t been so hard.

Implementing multitasking in the app

- In the app’s iOS-specific UIKit code, not very hard.
- The problems were all in the C++ code shared with Collabora Online, and in the related LibreOfficeKit-specific code in core. It was not prepared at all to handle a LibreOfficeKit-using process having multiple documents open simultaneously.

Implementing multitasking in the app 2

- In the Collabora Online server there is one separate “Kit” process per document open. (That document can then have multiple “views”, one for each user collaborating on editing it.)
- In addition, the main “WSD” process and a “ForKit” process.
- But in the iOS app, there is one single app process with an unbounded lifetime.

Implementing multitasking in the app 3

- The iOS app “emulates” the set of processes in a Collabora Online server with multiple threads. This plumbing is quite fragile but worked.
- But multitasking required significant changes. There were global variables for “the” document and a concept of “current” this and that, here and there. Hopefully most of that is now fixed, at least well enough for the iOS app’s needs.

Collabora Office

Thank you for watching

Tor Lillqvist

@TorLillqvist

tml@collabora.com