CPL: Project Assignment Elm

November 17, 2015

Assignment

In this assignment you will implement a To-do List/E-mail manager. Its functionality can be compared to Google Inbox.

An example implementation of the base application can be found on this page. Your solution should mimic the behavior of this implementation. If something is not clear in this assignment, you can consider the example implementation as part of the instructions.

Expected working time

We expect you to spend 40 hours max. on this assignment. This document will describe various phases of the assignment, starting with the base application and ending with difficult extensions. You are not required to do all the extensions.

A perfect implementation of the base application receives at least a passing grade on this assignment. To get a higher grade we expect extensions to be implemented. Keep in mind that when you have a bug and submit just the base application you will not get a passing grade on this assignment, therefore we recommend that you implement at least one extensions.

Requirements

A short description of the application features is given here. If something is not clear play around with the example implementation.

Items

Each item (both reminders and e-mails) can be pinned and marked as done with buttons. Depending on the length of an e-mail's body a third button should appear to disable truncation. Only 200 characters should be shown of the email's body.

The following features and information should be there for each item:

- Reminder: pin button, mark as done button, body and creation date
- E-mail: pin button, mark as done button, truncate button, from, title, body and date

Item Feed

The main component in the application is the 'Item Feed'. It is a split list of items that is always sorted. The split is done based on the 'done' and 'to-do' status of items. Items marked as done are split off the main list and appear on the bottom. Headers in the user interface should make the distinction obvious (example implementation simply uses headers 'To-do' and 'Done').

Both sections of the list are sorted. The basic sort primarily prefers 'pinned' items, the secondary sorting property should be the date. Older items go to the bottom and newer items are on top.

Hotkeys

The entire 'Item Feed' should be controllable from the keyboard:

- Alt + J : focus the next item on the feed (jumps from the 'to-do' to the 'done' splits when necessary)
- Alt + K : focus the previous item on the feed (jumps from the 'done' to the 'to-do' splits when necessary)
- Alt + O : toggle the truncation of the currently selected item
- Alt + P : toggle the 'pinned' status of the currently selected item
- Alt + X : toggle the 'done' status of the currently selected item
- Alt + S : as long as this key is held down, the sorting function of the feed should change to just 'old items on top' ignoring the pinned status and reversing the date priority

'Next' and 'previous' item is always the first item under and above the currently selected item (selection wraps when you reach the top or bottom). This constraint should be held at all times even in the following scenarios.

There are 3 items {1, [2], 3}, 2 is focused:

- Action: Pin current item
- 2 becomes pinned and the order of the feed changes to {2, [1], 3}, 1 is focused instead
- Action: Previous item
- Focus changes to the previous item: {[2], 1, 3}

There are 3 items {1, [2], 3}, 2 is focused and 3 has the oldest date:

- Action: Switch sorting (hold Alt + S)
- order of the feed changes to {3, [2], 1}
- Action: Previous item
- Focus changes to the previous item: {[3], 2, 1}

Reminders

It should be possible to add reminders to the list, they are marked 'to do' and are not pinned by default.

User interface

There are no user interface requirements, you can make it as ugly or pretty as you want as long as there is a visual indication for all the features that would otherwise be unnoticed (pin, selection, ...).

Technical requirements

There are no real technical requirements for the project. Different elm libraries can be used and some are even recommended/required for extensions, the elm-html library was used to make the example.

Extensions

You can choose which extensions you implement, e.g. you do not have to implement the first 4 extensions to implement the 5th.

Some of these extensions will require new elm concepts to execute effects or to interface with Javascript APIs using the FFI.

Some useful links:

- http://elm-lang.org/guide/reactivity#tasks
- http://elm-lang.org/guide/interop#ports

From easy to difficult:

- Add a hotkey to toggle the visibility of 'done' items.
- Hide the 'add reminder' functionality and add a hotkey to toggle its visibility.
- Add a deadline property to reminders and mark all reminders that are past their deadline.
- Put the current date as the default in the date picker when adding reminders.
- Add a 'snooze' feature to items, to 'snooze' an item you must provide a date on which the item has to 'un-snooze'. 'snoozed' items are not visible.
- On startup, read e-mails from a Json document at http://people.cs.kuleuven.be/~bob.reynders/2015-2016/emails.json
- Periodically check for e-mails from Json (same url).
- Add persistence to your application by using Html local storage so that newly added reminders are still there after a reload.
- Come up with your own extension!

Tips & Practical Requirements

Before you start writing the application, read the Elm documentation! They provide helpful tutorials to build real-world applications and touch on subjects such as modeling the problem and program architecture.

Do not spend too much time on the user interface. We do not grade the appearance of your application. We also do not test on dated browsers, it just has to work on the latest version of Chrome or Firefox. This means you can make use of as much Html 5 features as you want. Html 5's input type date for example will save you a lot of effort when creating the 'add reminder' functionality.

Documentation

Here are some links with documentation material for Elm:

- Documentation for the Elm standard library (2.1.0 for 0.15.1).
- A document containing an overview of Elm syntax.

Turning in the assignment

We expect you all to turn in a zip with a valid Elm project (the assignment comes with a zip to start you off). It should be possible to unzip your project and perform the following command to produce a standalone Html document that can be opened in a browser. Test this!

```
elm-make Main.elm --output=main.html
```

The zip that we provide starts you off with a default elm-package.json file and two modules: Main and Static. Main is a simple skeleton file that should contain your main value, it also contains some comments that you should complete with your personal information such as a student ID, your name, which extensions you attempted and a small summary of your progress in the extensions. Do not forget this!

Elm is a young project and versions are released frequently. The Elm platform that is installed on the PCs in the labs is 0.15.1, it is possible that a newer version will be released before the deadline. We do not mind versions newer than 0.15.1 as long as your elm-package.js configuration correctly reflects this and the elm-make command still provides a proper Html file!

The deadline is Thursday December, 31th at 23h59. A Toledo assignment is available to submit your zip file. If you have any questions, there is a discussion forum on Toledo that will be actively monitored by the assistant.