Proposal to start a new implementation of Thunderbird based on web technologies

Ben Bucksch ben.bucksch at beonex.com
Fri Mar 24 17:04:39 UTC 2017

- Previous message: New Council Roles and Hiring Update
- Next message: Proposal to start a new implementation of Thunderbird based on web technologies
- **Messages sorted by:** [date] [thread] [subject] [author]

Summary:

  * Our base goes away. Gecko will change dramatically in the future,
    and dropping many features that Firefox does not need anymore, but
    that Thunderbird relies on.
  * Our codebase is now roughly 20 years old. It heavily and
    intrinsically relies on those very Gecko technologies that are now
    being faded out, more or less aggressively.
  * JavaScript and HTML5 have evolved dramatically. Entire applications
    in JS are now realistic, and have been done. There are several
    existing JS libraries we might leverage. JavaScript is an efficient
    language, which allows fast development. A rewrite in JavaScript
    makes sense now.
  * We will learn from shortcomings of existing Thunderbird, and solve
    them, for example a more flexible address book, and cleanly
    supporting virtual folders without overhead.
  * The goal of the rewrite is to be close to the existing Thunderbird,
    in UI and features, as a drop-in replacement for end users, without
    baffling them. They should immediately recognize the replacement as
    the Thunderbird they love. It will install and run as normal desktop
    application, like Thunderbird does today. It keeps user data local
    and private.
  * We can also make a new, fresh desktop UI, as alternative to the
    traditional one, for new users. The technology also gives us the
    option to run it as mobile app.
  * While we implement the new version of Thunderbird, the old codebase
    based on Gecko will be maintained until the rewrite is ready to
    replace the old one.
  * I expect this effort to take roughly 3 years: 1 year until some
    dogfood (usable by some developers and enthusiasts). 2 years until a
    basic feature set is there. 3 years until we can replace Thunderbird.

I will describe each point in more detail below.

  0.1. Proposal

I would like to put this up for discussion. I have not discussed this
with the rest of the council yet, but rather I start the discussion
here, to make it public, so that everyone can follow the discussion.

Ultimately, this will be a team effort. The TB Council, even with
employees, will not be able to do this alone. I hope that Thunderbird
can fund a number of developers on this, but I would like this to be an
open-source project in the sense that everybody can participate. Please
see this also as a call for help. If you are an experienced JavaScript
developer who already wrote applications and libraries in JavaScript,
please step forward and write us that you want to help.

  0.2. Importance

I've been working on Thunderbird since 18 years, and I care about it a
lot. More importantly, I care about communicating with my friends in a
way that's private and not controlled by any single entity, but
decentralized and running and stored on my own computer. In other words,
we need a desktop (or smartphone) email client. Not just today, but also
30 years from now, if the world didn't collapse until then.

This is important for the freedom of the individual, of communication,
and of computing. There are not many contenders left that defend the
freedom of communication, because most have an interest to bind users to
their own services. If I want to communicate with teenagers today, or
even my mother, the best way is WhatsApp or SnapChat, closed systems. In
order to make a different in this world, Thunderbird needs to be a
viable option, not just next year, and not only for the existing
userbase, but for the whole world, all age groups and all usage
patterns. The current Thunderbird codebase has served well - 25 million
users are amazing - but will not be able to defend freedom for all of
us. Given the current technical challenges we face, this is the perfect
time for a fresh start.

  0.3. Our base goes away

Firefox has discontinued the Firefox XUL extensions, killing tens of
thousands of small projects this way. This is a massive cut that goes to
the base of Firefox. They did that, because these extensions use XUL and
XPCOM, and this is preventing Firefox developer from making more drastic
changes in the internal architecture. They are gradually moving to using
some Rust components, which means the XPCOM base is no longer viable. At
the same time, they also want to gradually decrease reliance on XUL in
favor of HTML, and remove parts of the XUL implementation that they no
longer need to run Firefox. That's why they remove XUL extensions.
Thunderbird heavily relies on XUL, all its UI is written in XUL. Some
parts of XUL - esp. some features of the tree widget - were back then
written only for Thunderbird. We can expect breakage in this area, as
Mozilla has clearly stated that they will no longer care about breaking
Thunderbird. In fact, I think this problem is part of the motivating
force behind Mozilla asking us to be independent.

  0.4. Thunderbird relies on those very Gecko technologies

Thunderbird is based entirely on XPCOM (backend module API) and XUL
(user interface). If you take out XPCOM and XUL from Thunderbird, you
have effectively rewritten it. This change is as dramatic as the change
from Netscape 4.5 to Mozilla was, when it was rewritten with XUL and
XPCOM. 90% of the code was new. We are faced with a similar problem.

  0.5. Gradual change would be very costly

Of course, the changes in Gecko are gradual, but eventually, the
technology will go away, so the overall work is there nonetheless.

Given that XPCOM is the very technology that creates the API between the
modules, is hard to replace step by step. I think we would spent as much
time integrating the new modules with the old code as we would take
writing the module in the first place, which means the overall effort
increases at least 2-fold by trying to do it step by step.

Having worked on Thunderbird since 18 years, I do not deem it feasible
to modernize it component by component. I estimate it would be 4 times
faster to start with a clean slate and start from 0.

  0.6. JavaScript is today the best choice

JavaScript, if used diligently and with good design, is a very efficient
language. Both in execution time, but more importantly for developers.
Personally, I wrote apps in many languages, including C++, Java and
JavaScript. Of those, JavaScript is by far the most productive - I am
personally 4-10 times as productive as with C++.

Today, we have node.js, Electrolysis, Cordova and other platforms that
are already ready-made to create desktop applications using HTML5 and
JavaScript. There are already other applications that have been written
this way, and there is existing knowledge and libraries on how to do this.

  0.7. Solve shortcomings of Thunderbird

There are some known shortcomings of Thunderbird, as it is implemented
and designed today. Some are on the feature level, for example the
address book is not flexible enough to capture even 3 email addresses
for one person, it cannot link 2 persons (e.g. a couple), and so on. A
modern re-write would consider today's user requirements.

Deeper goes the problem of virtual folders. Thunderbird has them
implemented as a bolt-on, which essentially keeps a copy of the mails,
as far as I understand, and they are slow and inflexible. A new
Thunderbird implementation could take an approach of where folders are a
view that is computed in real time (like views in databases). This
matches what GMail does with the AllMails folder and tags as properties,
and then tags also form a virtual folder. There are important
implications that this has: I can have a unified inbox, one inbox per
folder, and I can have the emails sorted into specific project folders
on arrival, all without overhead. The same incoming email appears in all
3 folders in the same second, and is marked read at the same time. I no
longer have to move emails, but they are filtered.

Even deeper are design shortcomings of Thunderbird. It establishes a
rough border between user interface and backend modules for IMAP etc.,
but is not very consistent in enforcing the independence of the 2 layers.

Right now, large parts of the logic are written as part of the frontend
code, which is one of the big reasons why Thunderbird code is so
difficult to understand and modify. It also makes it practically
impossible to write alternative frontend. Instead, the architecture
should be flexible enough to allow this, by strictly separating logic
from UI using design patterns.

Some backend modules call the frontend, to update changes. Instead, we
should use a classic observer/subscriber design pattern, where the
frontend subscribes to changes in the backend, and updates itself. That
allows more flexibility for the frontend. All lists should be
observable, which is a simple but very powerful way to decouple logic
from UI. See Ling in C#, and
https://wiki.mozilla.org/Jetpack/Collections , which takes it a step
further. This could be the technical basis for very flexible virtual
folders that update immediately and are fast. But it would be used
whenever there is a list.

  0.8. Be close to the existing Thunderbird

Even though we write almost all code from scratch, we will save a lot of
time by having a clear goal: We want to replicate the current
Thunderbird, from an end user perspective. That means, the user will
find the same 3-pane window layout, the same way how folders and message
lists and the thread pane operate. The theme will be similar. Existing
Thunderbird users should feel right at home.

We retain the overall UI and most features and qualities like
performance, even if we do not copy all little details.

Disclaimer: Given that the technological basis - particularly HTML - is
completely different, there will be some things that work differently in
some ways. Hopefully, many will be better. We will have some technical
limitations. Some will be just different, because the underlying
implementation is completely different. The goal is not to copy bug for
bug, but to make existing users immediately recognize this as
Thunderbird, and feel at home, even if some details are different.

We must pay attention to also keep technical qualities that many of our
users rely on. An obvious one is that the new implementation must be
able to quickly scroll through a list of up to 100000 messages. There is
no such HTML widget that allows that, we would have to create one, but I
think it's feasible, I already have ideas how to do that. We also need
to preserve privacy and security qualities that Thunderbird has, or even
improve on it.

  0.9. Fresh UI, more platforms

In addition to replicating the current Thunderbird UI, we should also
experiment with new forms of UI, in parallel. For example, we should
create a UI that's suitable for the new generation of users that never
used a desktop email client before. These people do not feel at home
with Thunderbird today, and we should create something for them.

A lot of the new userbase is on mobile platforms and on tablets. That
new UI should be "responsive" (automatically adapting to different
screen sized) so that it runs well on tablets and smart phones as well.
With Cordova and similar toolkits, we have a technological basis to
quickly make a mobile app out it so that it installs like any other app.
It would be a replacement for the system "email" app.

The goal for the new UI are 1 billion users.

  0.10. Maintain old Thunderbird code base

During the time while the new rewrite is implemented, a smaller part of
the staff will maintain Thunderbird, as they have done in the last 2
years. We keep up with Gecko changes, and fix smaller bugs. But we will
not do major refactoring or big new features on the old codebase.

This gives users a continuously updates Thunderbird. Particularly
important are security updates for security holes found in Gecko, which
Thunderbird inherits. They might be exposed in HTML emails, RSS feeds or
other places where Thunderbird shows HTML and renders images.

  0.11. Effort

The majority of the team should concentrate on the rewrite. This is also
why this decision is important to make now, because Thunderbird has now
resources to hire a number of developers, and we need to decide on which
efforts we spend the precious little money we have through the generous
donations of our existing users.

I would estimate that we need a team of 10 full time developers, for 3
years. We need 2 persons for the framework, 3 for backend modules, 4 for
frontend UI, and 1 for theming.

    0.11.0.1. Year 1

In the first year, we are laying the foundation, getting the framework
sorted out, building infrastructure etc..

After 1 year, I would expect a first demo that can read and write email,
but with a very minimal feature set and many rough edges. A few
enthusiastic first alpha users might be able to use it for their email
needs, and some developers can use it for their daily needs.

    0.11.0.2. Year 2

In the second year, we would concentrate on building the important
features that are needed for most users.

After 2 years, we should have an email client that's appealing to most
normal users. Even some power users might like it, because we have
advanced features that no other client has, but they will miss some
features.

    0.11.0.3. Year 3

In the third year, we concentrate on feature parity to the old
Thunderbird. We add any features that Thunderbird currently has and are
appreciated by the existing userbase. We also add functionality that
allows larger deployments, as a significant part of userbase are
enterprises.

Any features that are used by significantly less than 1% of the userbase
will not be implemented, in favor of other features that are desperately
needed today.

After 3 years, 95% of the existing Thunderbird userbase should find the
features they need in the new implementation. There will be some changes
and adoptions necessary, but there should be a way to do what they need.
That's what I call "feature parity".

    0.11.0.4. Year 4-20

Keep improving

- Previous message: New Council Roles and Hiring Update
- Next message: Proposal to start a new implementation of Thunderbird based on web technologies
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the tb-planning mailing list