Project Plan
Asynchronize All The (Localization) Things!

The Capstone Experience

Team Mozilla
Collin Wing
Jack Smith
Jack Song
Jim Lennon
Raza Haider

Department of Computer Science and Engineering
Michigan State University
Fall 2018
Functional Specifications

• Mozilla Firefox is an English (US) based web browser looking to expand in the global market.
  ▪ Translate pages to the host countries native language not using DTD and .properties files.
• This current system requires developers and localizers (translators) to edit both the DTD and .properties files.
  ▪ Longer process for volunteer localizers.
  ▪ Formatting a language can also cause issues with the markup language.
• The Fluent system will streamline this process.
  ▪ Fluent is a localization framework.
  ▪ Translations with genders and grammatical cases, date, time, and number formatting, plural categories, full bidirectionality support, custom formatters and easy-to-read syntax.
  ▪ Stronger API between developers and localizers.
  ▪ Platform detection, increased expressive power, and Runtime translations on client and server side.
Design Specifications

- **Localizers**
  - Easier to read syntax
  - Easier logic construction

- **Developers**
  - Remove need for platform/plural checks at runtime
  - Easier to program
    - In XUL
    - In JS

- **End Users**
  - End goal remove series of steps and/or different build of Firefox to change languages
    - Step towards that goal
Screen Mockup: Converting to Fluent

```plaintext
openDir = Open Directory
# LOCALIZATION NOTE (macOpenDir): This is the Mac-specific variant of openDir.
# This allows us to use the preferred "Finder" terminology on Mac.
macOpenDir = Show in Finder
# LOCALIZATION NOTE (winOpenDir2): This is the Windows-specific variant of
# openDir.
winOpenDir2 = Open Folder

profiles-opendir =
    { PLATFORM() ->
        [macos] Show in Finder
        [windows] Open Folder
        *[other] Open Directory
    }
```
Screen Mockup: Cleaning up JS

```javascript
let string = "openDir";
if (AppConstants.platform == "win") {
    string = "winOpenDir2";
} else if (AppConstants.platform == "macosx") {
    string = "macOpenDir";
}
document.getElementById("button").setAttribute("class", "profiles-opendir");
```
Screen Mockup: Cleaning up XUL

```xml
<menuitem id="utils-installFromFile"
    label="&installAddonFromFileName.label;"
    accesskey="&installAddonFromFileName.accesskey;"
    command="cmd_installFromFile"/>

<menuitem id="utils-installFromFile"
    data-l10n-id="install-addon-from-file"
    command="cmd_installFromFile"/>
```
Screen Mockup: End User Experience

About Profiles
This page helps you to manage your profiles. Each profile is a separate world which contains separate history, bookmarks, settings and add-ons.

Create a New Profile

Profile: Collin

Abōoūūt Přoůfílēéż
This faad ēe žaana tēa raanaa gēe žaanaa priaarīlēéż. Each priaarīlēé is aa şečaarmāa aadēng which ciaaantaas şečaarmāa ĥistray, bāadākmaaak, jēťentš āaňd aądd-aąns.

Cřēéaart āa Nēew Přoůfílēéż

Přoůfílēéż: Collin
Technical Specifications

• Fluent Conversion
  ▪ Strings are moved from old-fashion files to FTL files manually.
  ▪ Localization of the browser achieved via modifying markup and Js files.

• String Migration
  ▪ Strings are moved from old-fashion files to FTL files by migration recipes.
  ▪ Migration test for comparison between manually and recipes-created FTL file.

• Integration
  ▪ Mercurial, Bugzilla and Phabricator
System Architecture

User Interface
- User Interface
  - UI Toolkit (XPFE)

Data Persistence
- User, Secure, And Engine Persistence
  - Networking
    - Necko
    - Security (NSS/PSM)
  - JavaScript Interpreter
    - Spider - Monkey
  - XML Parser
    - Expat
  - Display Backend
    - GTK+ Adapter
      - GTK+ / X11 Libraries
  - Localization
    - L10nRegistry
    - LocaleService

JavaScript
- React
- Vue
- FluentDOM
- Fluent
- FluentReact
- FluentVue

DOM
- HTML
- XUL

Layout Engine
- Gecko
  - (Browser Engine)
  - (Rendering Engine)
System Components

• Software Platforms / Technologies
  ▪ Mercurial
  ▪ Phabricator
  ▪ Bugzilla
Risks

• Localization Anti-Patterns
  ▪ Description: Translations may be violating design principles that we prefer for working with Fluent. Some of these strings will prove difficult to convert and migrate.
  ▪ Mitigation: We will identify potentially difficult strings ahead of time and determine if they will require more extensive work.

• Limited functionality of Fluent
  ▪ Description: Fluent may not have enough expressive power to support converting some strings we encounter.
  ▪ Mitigation: We will work with the Fluent designers to add features which are on the critical path to full support. We will land patches directly if we can.

• Migrating complex strings manually
  ▪ Description: Complex strings require laborious manual AST manipulation in the migration scripts. These prove difficult and time consuming.
  ▪ Mitigation: We will delegate migration issues to a particular team member who can gain a deep understanding of the system and efficiently solve problems.
Questions?