Project Plan Presentation
Improve Firefox's Reader View

The Capstone Experience

Team Mozilla
Noel Lefevre
Jintao Hu
Chad Burnham
Tyler Kabaker
Emily Michaels
Steve Hagopian

Department of Computer Science and Engineering
Michigan State University
Spring 2022
Functional Specifications

• Guarantee simple and streamlined access to the internet
• Fix bugs and other issues with Reader Mode
• Implementation must be intuitive and easily accessible
• Manually fix frequently used websites
Design Specifications

• Features that will be improved upon or added:
  ▪ Quality of life improvements
    o Add shortcuts and buttons
    o Improve user experience
  ▪ Cross-browser compatibility
    o URL drag between browsers
  ▪ Document styling
    o Auto reset original formatting
  ▪ Page formatting
    o Formatting pictures
    o Adding border
Screen Mockup: User Experience

Before:

After:
Screen Mockup: User Interface

Before:

Main Page

Contributors to Wikimedia projects
6-8 minutes

From Wikipedia, the free encyclopedia

Jump to navigation Jump to search

From today's featured article

The Australian Air

After:

Main Page

Contributors to Wikimedia projects
6-8 minutes

From Wikipedia, the free encyclopedia

Jump to navigation Jump to search

From today's featured article

The Australian Air
Screen Mockup: User Interface

Before:

After:
Before:

```javascript
fetch('foo', {err, res} => {
  if (err) {
    // handle error
  }
  // handle response
})
```

The callback wouldn’t be invoked until the `foo` resource has been fetched, so its execution remains asynchronous and non-blocking. Note that in this model you could only specify a single callback, and that callback would be responsible for all functionality derived from the response.

After:

```javascript
fetch('foo', {err, res} => {
  if (err) {
    // handle error
  }
  // handle response
})
```

The callback wouldn’t be invoked until the `foo` resource has been fetched, so its execution remains asynchronous and non-blocking. Note that in this model you could only specify a single callback, and that callback would be responsible for all functionality derived from the response.
Technical Specifications

• Platform: Windows, Mac, and Linux
• Programming: HTML, CSS, JavaScript
• Codebase: Mercurial, Git
• Report and Track: Bugzilla
• Review: Phabricator
System Architecture

Resources
- JS
- HTML
- CSS

VS Code IDE

Applications
- Nightly
- Beta
- Full Release

Platforms
- Windows
- Linux
- MacOS

User
System Components

• Hardware Platforms
  ▪ Windows, Mac, Linux

• Software Platforms / Technologies
  ▪ HTML, CSS, JavaScript
  ▪ Visual Studio Code
  ▪ Bugzilla
  ▪ Mercurial, Git
  ▪ Search Fox, Browser Toolbox
  ▪ Phabricator
Risks

• Risk 1: Accidentally Implementing New Bugs
  ▪ While attempting to fix a bug, there is a possibility it may introduce a new bug
  ▪ Mitigation: Regression Testing & Sponsor Oversight

• Risk 2: Navigating Two Code Bases
  ▪ Git and Mercurial are used in different instances to push changes to the Reader View
  ▪ Mitigation: Understanding how to use both systems and confirming with clients.

• Risk 3: OS-Specific Bugs
  ▪ Bugs may arise that are only on an unfamiliar OS
  ▪ Mitigation: Familiarizing ourselves with all OSs so we are prepared for a bug
Testing Plan

• Local Testing
  ▪ Testing for correctness on our local patches
• Review
  ▪ Feedback and further requests from sponsors
• Automated Testing
  ▪ Test across entire codebase to ensure everything is functioning
Questions?