Google Chrome Conceptual Architecture Presentation

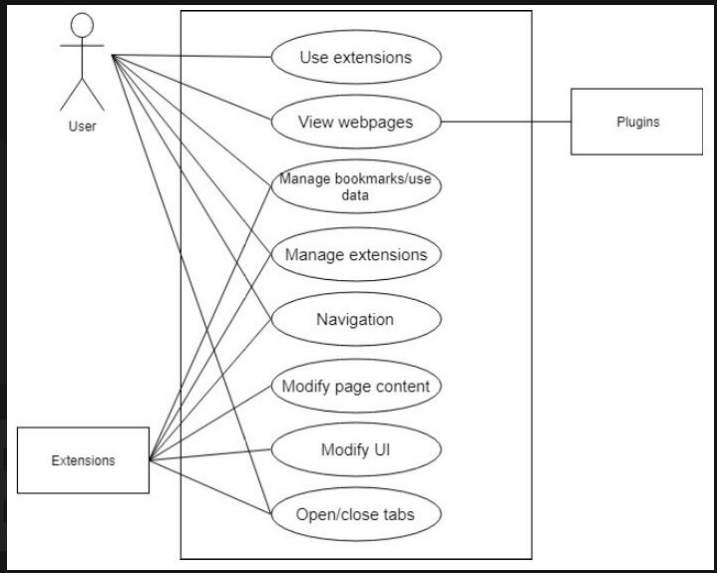
By: Brynnon Picard, Dongho Han, Sam Song, Bradley Kurtz, Alex Galbraith and Roy Griffiths

Derivation Process

1) Collect links relating to chrome architecture and organize.
2) Identify major functions of chrome.
3) Group major functions into components.
4) Block out architecture diagram and find dependencies between component intuitively.
5) Look through chrome documentation to find contradictions with our architecture and remedy these.
6) Create diagrams (Use case and sequence) of a general use case and check it’s consistency
7) Repeat 4 to 6 until our architecture diagram is consistent and complete.

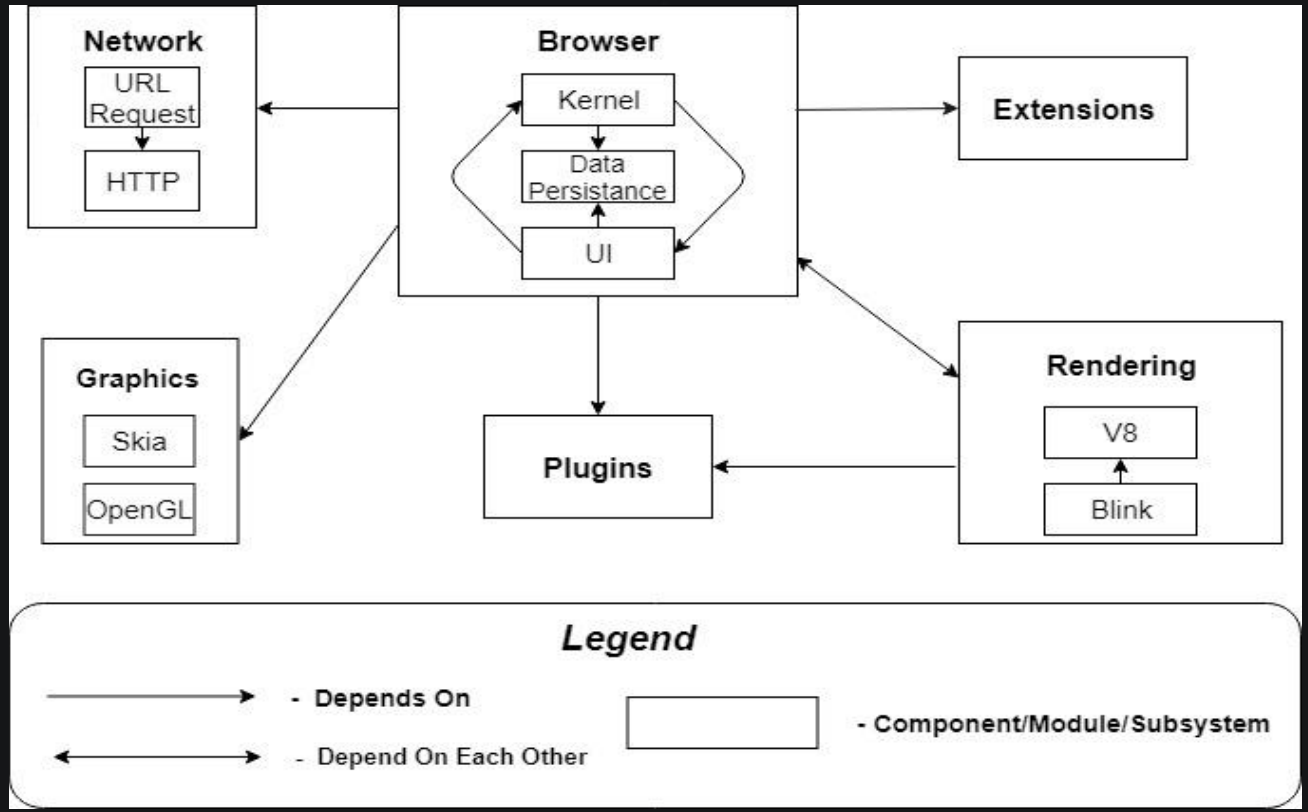
Functionality of Google Chrome

- Web Browser functionality
  - Parse and display HTML, CSS
  - Run Javascript
  - Integrate plugins
- Chrome functionality
  - Bookmark synchronization via google cloud system
  - User data management
  - External extension support
  - Tab support



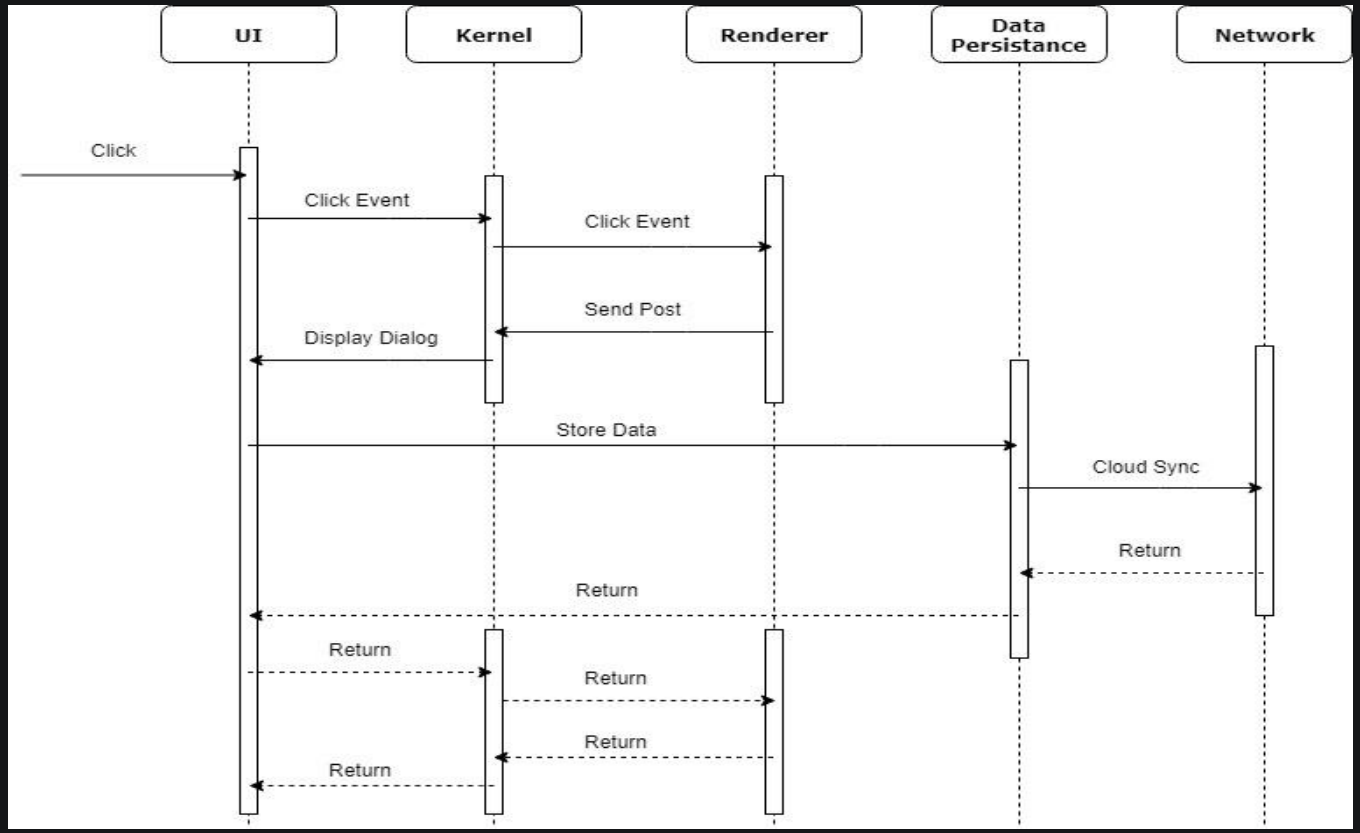
SKYNET

Conceptual Architecture



SKYNET

Sequence Diagram



SKYNET

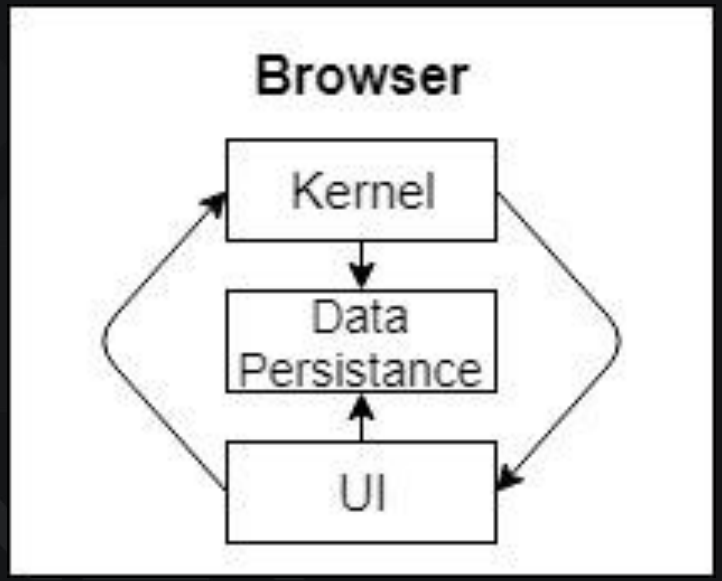
Architecture and Subsystems: Browser

Subsystems:

- Kernel is the “brain” of the browser.
- Data persistence stores and syncs user data.
- UI runs the user interface and receives user input.

Relationships:

- UI relies on data persistence and kernel.
- Kernel relies on data persistence and UI.
- Data Persistence does not depend of either.



SKYNET

Concurrency

How Architecture Supports Concurrency

- Separation of browser and rendering module allow for multiple cohesive rendering instances to be spawned as separate processes.
- The browser acting as a central control module allows the browser to catch crashed processes.

Browser Side

- IO Thread
- UI Thread

Renderer Side

- IO thread
- Render thread

Renderer Process Models

- Single Process: Renderer and Browser run together in a single process. For testing only.
- Per Tab: Each tab and those connected via JS run in their own process.
- Per Site: All sites of same domain run under one process.
- Per Site Instance: One process per open site unless connected via JS. Default model.

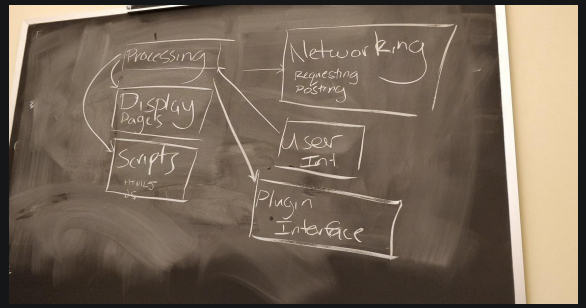
Team issues

- Google spent 5 years building its Chrome team through acquisitions and hiring experienced browser developers
  - Acquired GreenBorder, a web security company, which was working on what would become Chrome's "Sandbox" architecture model
  - Hired Lars Bak, who spent many years working on virtual machines, such as Sun's Java VM, who created the high-performance V8 JavaScript engine
- Team struggled with needs of balancing speed and security
  - Over time, users have become more conscious of security issues and Chrome has become less focused on speed and more focused on security
  - Security
    - Plugins
    - HTTP vs HTTPS

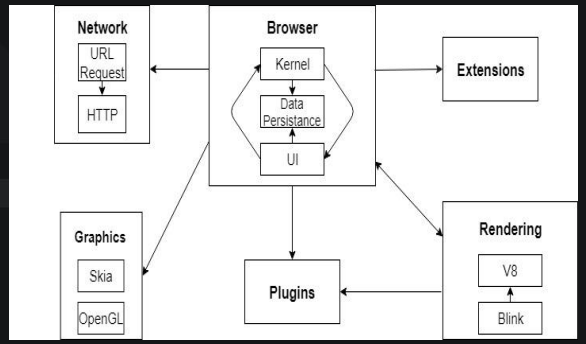
Lessons learned/limitations

- First assumed plugins and extensions are the same.
- Initially found it hard to find what the browser component does, as there were many conflicting definitions.
- Learned why many struggle during specification phase.
- Dependencies and data control is hard to verify.
- Finding information on the team structure of Chrome is difficult since it is not open source. Even chromium has little information on the team structure.

First Draft



Final Draft



SKYNET

Conclusion

- Hybrid of object oriented and implicit invocation architecture.
- The browser component is central to Chrome’s architecture and marshals data between all other components.
- Chrome is constantly evolving and improvements are constantly made through acquisitions and hiring experienced software developers.
- Multi process for concurrency.
- Chrome’s architecture is constantly subject to change; plugins could disappear from the architecture diagram any moment.

References

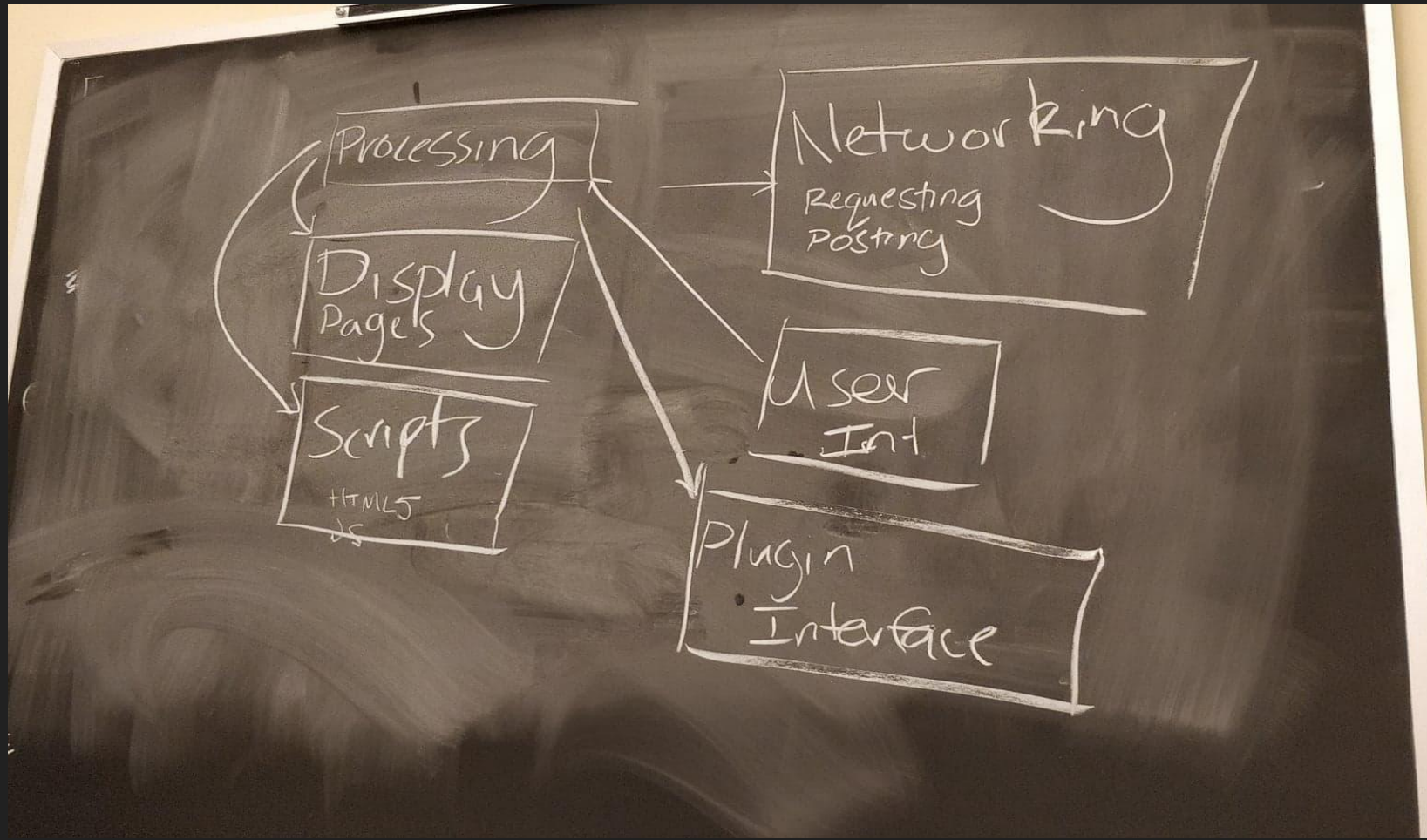
-http://szeged.github.io/sprocket/architecture_overview.html
-https://chromium.googlesource.com/chromium/src/+/master/docs/threading_and_tasks.md
-https://medium.com/@zicodeng/explore-the-magic-behind-google-chrome-c3563dbd2739
-https://www.howtogeek.com/124218/why-does-chrome-have-so-many-open-processes/
-https://www.google.com/googlebooks/chrome/
-https://www.chromium.org/developers/design-documents/plugin-architecture
-https://docs.google.com/document/d/1aitSOucL0VHZa9Z2vbRJSyAIsAz24kX8LFByQ5xQnUg/edit
-https://www.chromium.org/developers/design-documents/displaying-a-web-page-in-chrome
-https://www.niallkennedy.com/blog/2008/09/google-chrome.html
-https://www.chromium.org/developers/design-documents/graphics-and-skia

How Chrome Evolved

- Development is lead by 4 core principles
  - Speed - make fastest browser
  - Security - principle of least privilege
  - Stability - process isolation
  - Simplicity - Optimization of UI
- Chrome’s development is divided into four channel
  - Canary
  - Developer
  - Beta
  - Stable
- Major updates
  - Version 3.8.5 first release of Blink Engine, fork of existing WebKit browser engine
  - Version 55 first introduction of V8 Javascript Engine

Interaction between subsystems

- Browser sends url request to the network subsystem, then network module receives HTTPS data and send it back to the browser
- Browser will talk to extensions before displaying web page to the user. For example, ADblock extensions will selectively remove all source codes related to advertisement
- After receiving data of webpage, browser will ask rendering engine to compile all HTML, CSS and JS
- After all rendering is finished, browser will send rendered web page to the graphics in order to display visual illustration of web page on user’s display


Processing
Networking
Requesting
Posting
Display
Pages
Scripts
HTML5
User
Int
Plugin
Interface