Chrome:

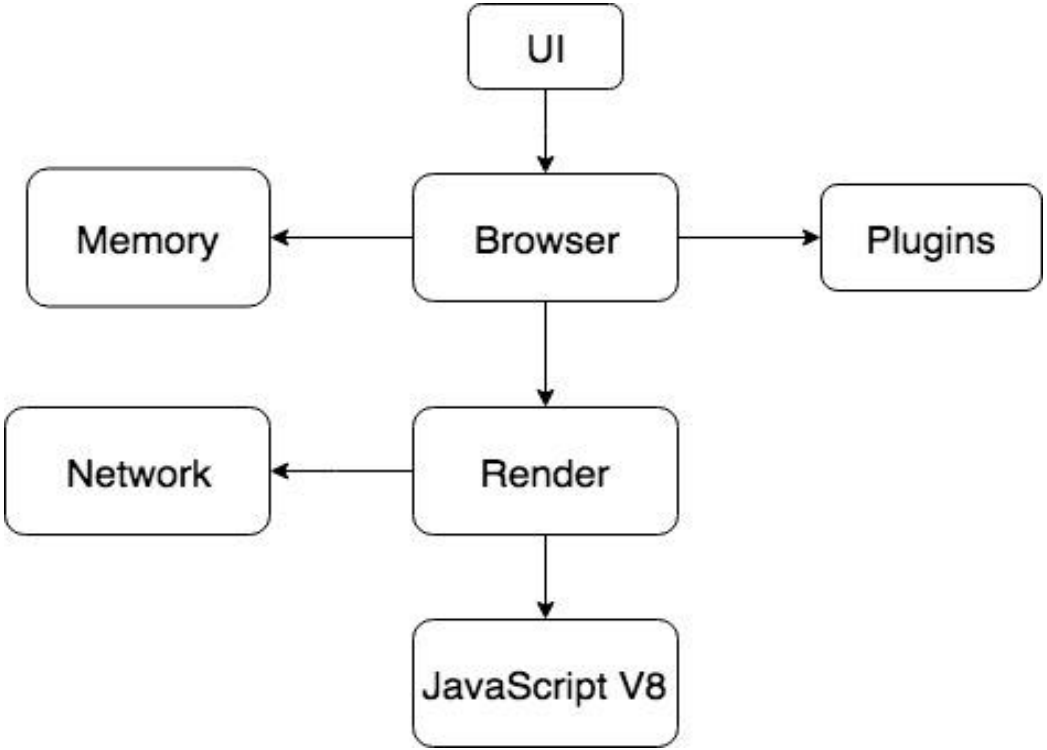
Concrete Architecture

Bits... Please!

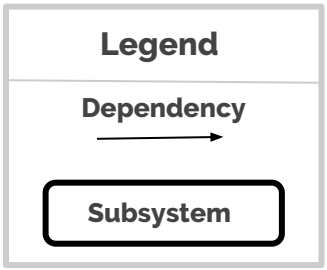
Agenda:

1. Derivation
2. Alternatives
3. Concrete Architecture
4. New Subsystems / Dependencies
5. Subsystem: Render
6. Use Case
7. Current Limitations/ Lessons Learned

Conceptual Architecture



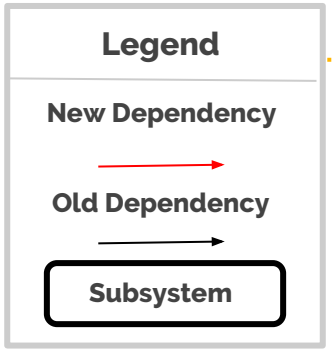
Layered

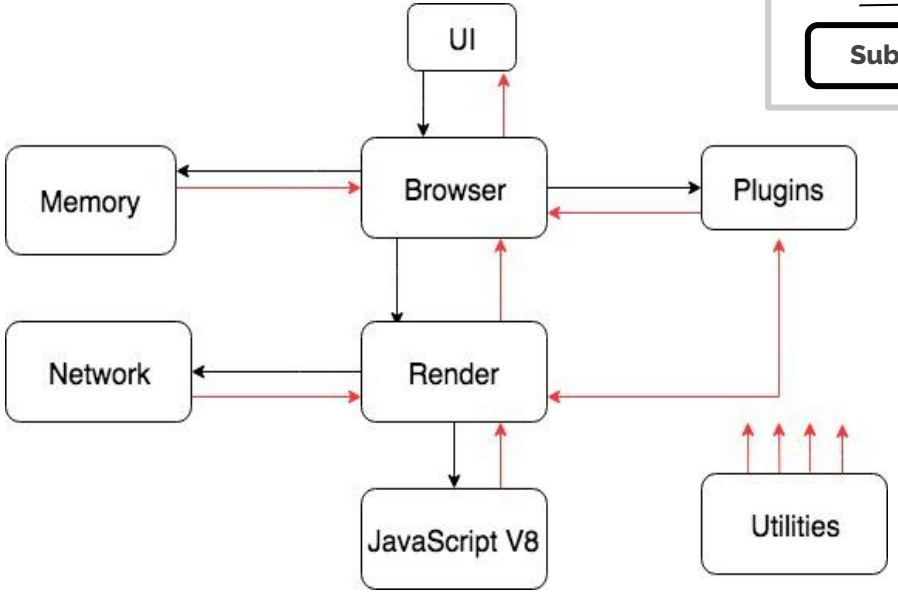


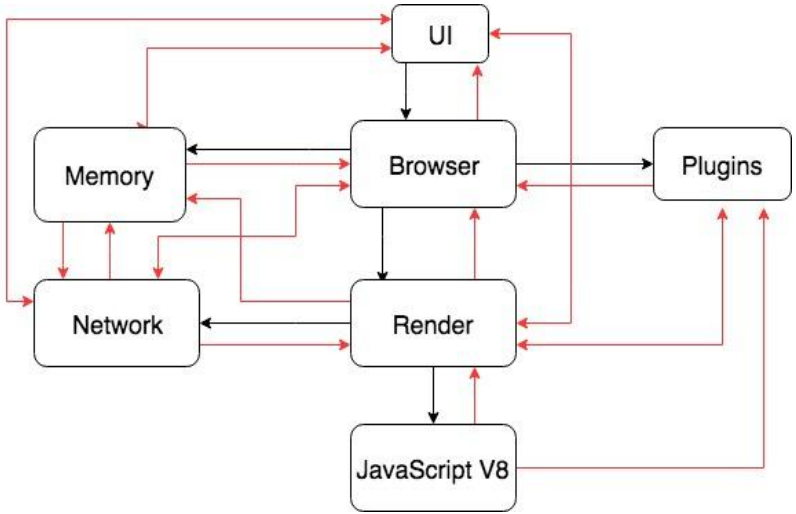
Derivation Process

- Use code analysis tool called Understand
- Create a new architecture with the components of conceptual architecture
- Addition of Utilities and Communication component
- Map source code to the correct subsystem based on functionality

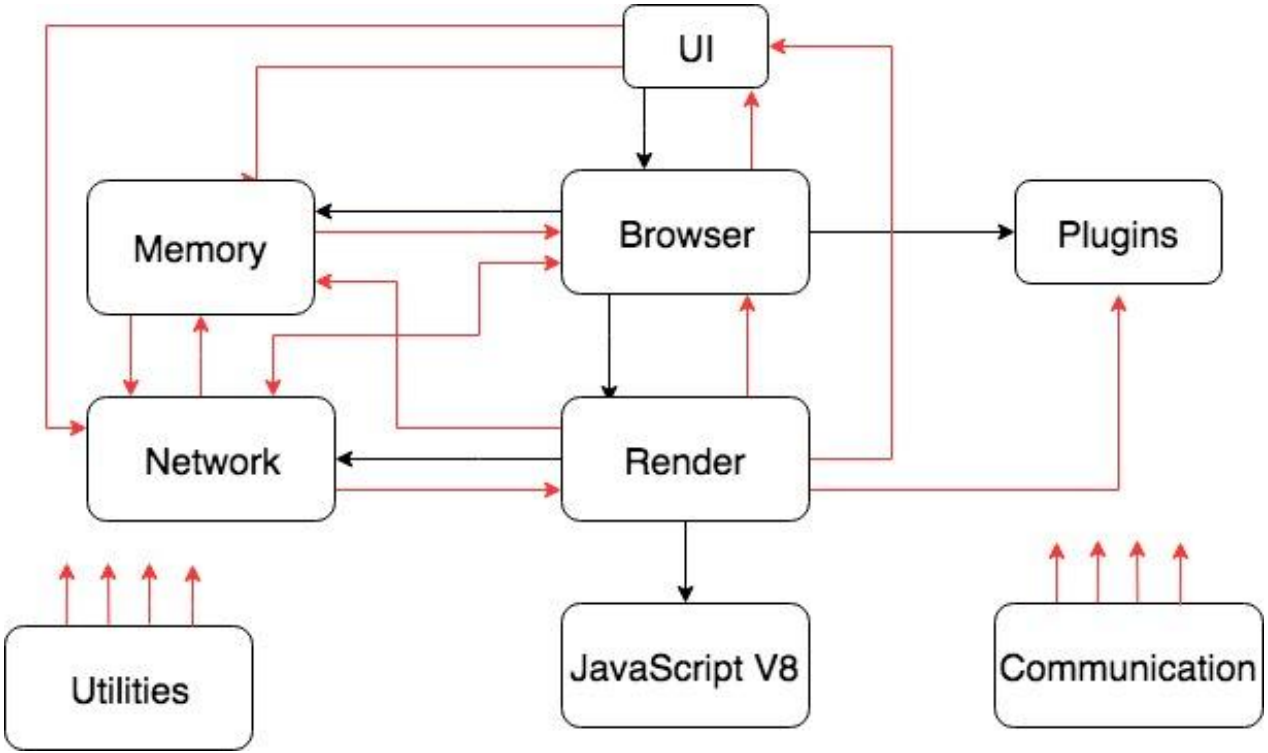
Alternatives



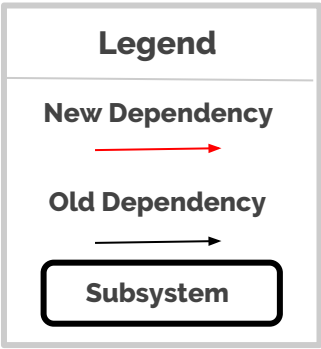




Concrete Architecture



Object-Oriented



New Subsystems and Dependencies

Communication

- Encapsulates Mojo and threading
- Allows all aspects of the architecture to communicate via message passing and IPCs

Utilities

- Contains the “Base” code which is the shared code between all modules
- Contains string manipulation and general utilities
- Contains specific code for each operating system

Render -> Memory

- The Render uses a security filter
- Displays a message to the user that is stored in memory
- Example: “phishing” or “malware” detection

Render -> UI

- Renderer extension uses the UI layout

Render -> Browser

- The Render allows safe browsing based on the type of browser and message passing

Render -> Network

- Is the network connection is up and running?

UI -> Memory

- Makes window resizing appear smooth

UI -> Network

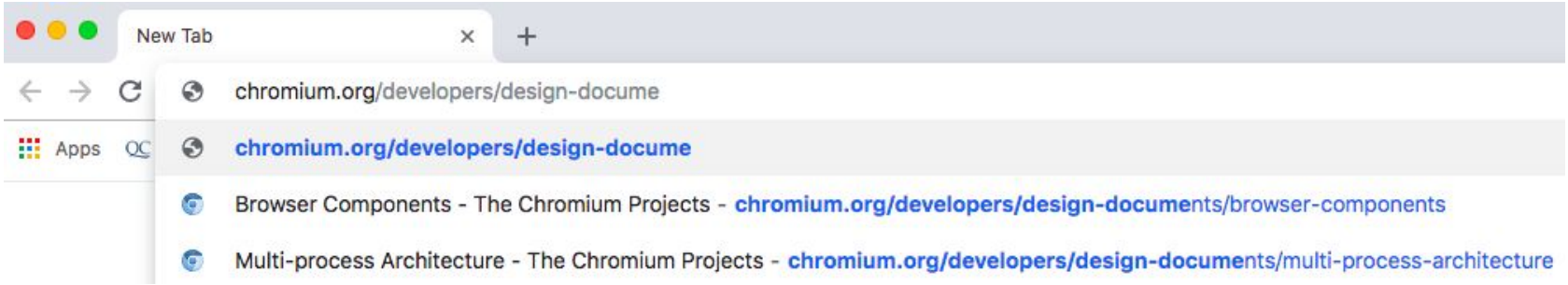
- The UI connects to the network to fetch the URL and format to display it

UI -> Render

- Play cc animations

Memory -> UI

- Memory stores previously accessed URLs
- As typing begins in the UI, Memory can display what the possible URL may be



Memory -> Browser

- Saves the snapshot file of each process in memory and can return it to the browser if necessary

Network -> Memory

- From network, IO Buffer streamed to memory for optimal read operations of data

Network -> Browser

- Handles some of the testing testing and determining network crashes

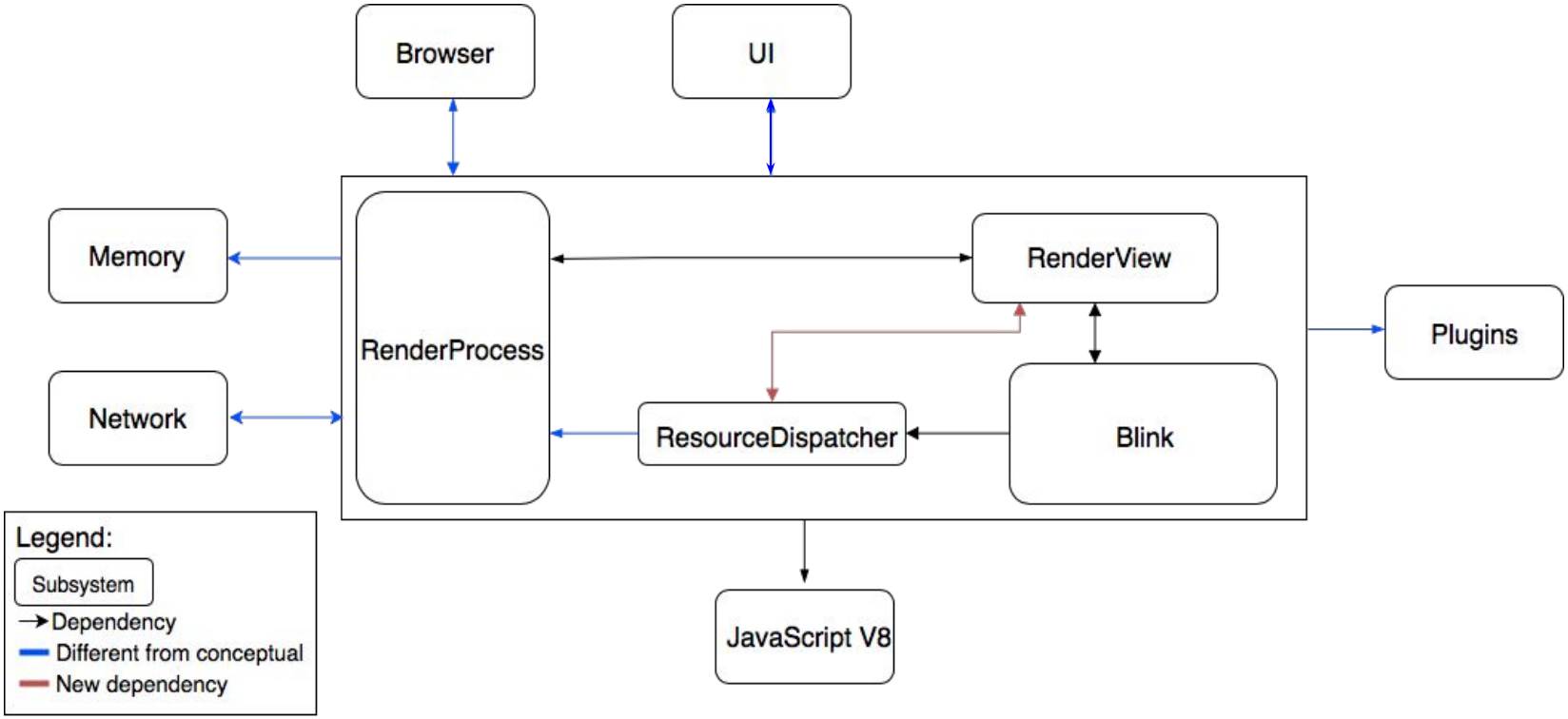
Network -> Render

- Renderer has a Network Interface List that the Network must be able to access

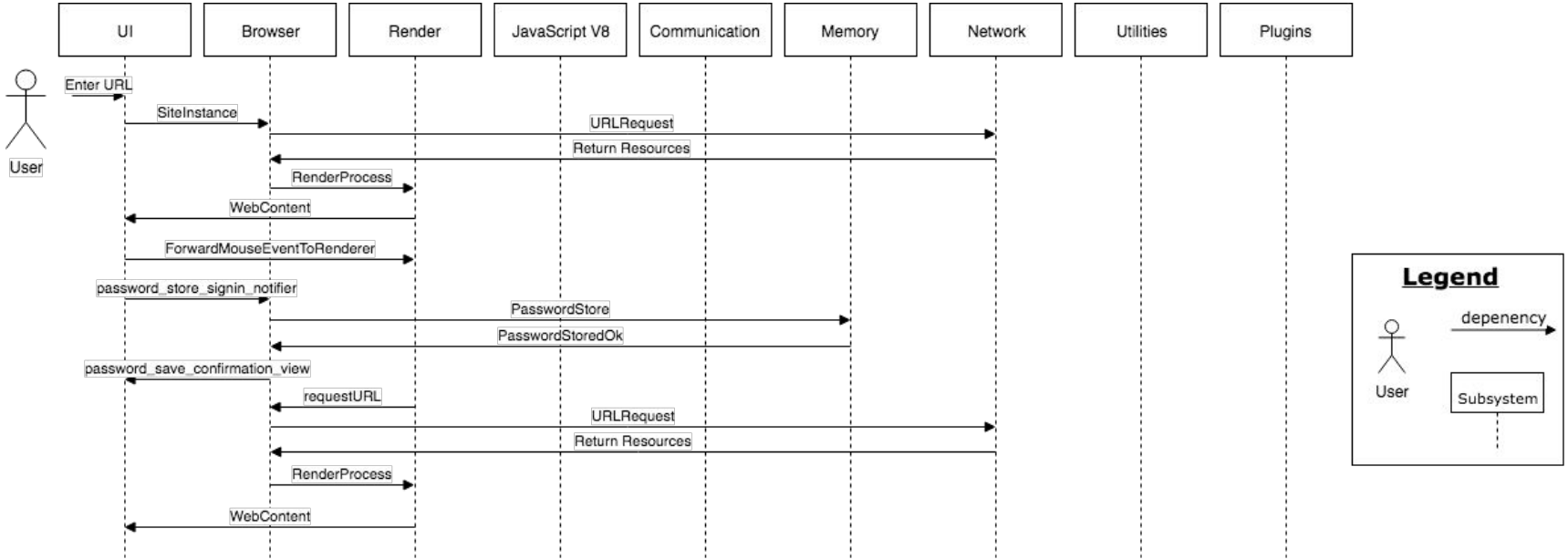
Browser -> UI

- If there is a conflict/error in the Browser, it communicates with UI to display an error message to the user

Subsystem: Render



Use Case 1


UI
Browser
Render
JavaScript V8
Communication
Memory
Network
Utilities
Plugins
User
Enter URL
SiteInstance
URLRequest
Return Resources
RenderProcess
WebContent
ForwardMouseEventToRenderer
password_store_signin_notifier
PasswordStore
PasswordStoredOk
password_save_confirmation_view
requestURL
URLRequest
Return Resources
RenderProcess
WebContent
Legend
depenency
User
Subsystem

Current Limitations and Lessons Learned

Current Limitations

- Lack of commenting and code readability
- This lead to difficulties in tracing the dependencies between subsystems
- Lacking JavaScript V8 and Plugins source code

Lessons Learned

- Communication
- Set deadlines
- How to use Understand

Team Issues within Chrome

- Independent teams working on different subsystems made it challenging to track who worked on what subsystem when
- Code updating with high coupling

Feature - Facial Authentication

Problem

- Users have many passwords associated with numerous websites
- Difficult to remember all passwords

Solution:

- Add your face authentication to Chrome
- Passwords are saved to the user account
- Use facial authentication to use saved passwords

Conclusion

- Object-Oriented concrete architecture
- Added dependencies
- Utilities and Communication subsystems added

Thanks!

Any questions?