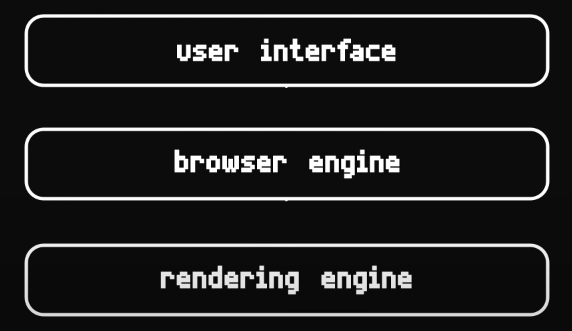
1. **Браузер**…………………………………………………………………………………….
   1. **Архитектура**…………………………………………………………………………..
   2. **Архитектура**

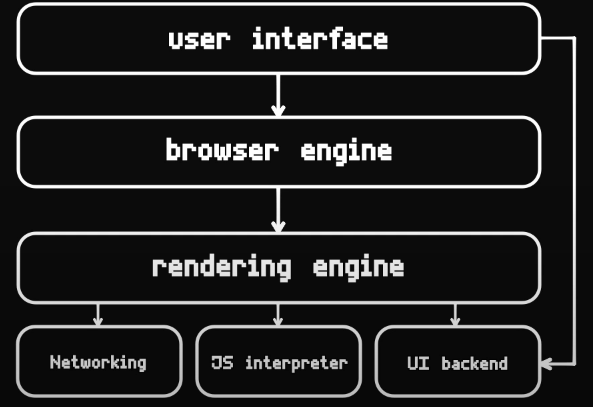
Структурная схема любого браузера



• user interface – вкладки, настройки, навбар, строка URL и т.д.  
• browser engine – соединительная часть между UI и механизмом рендеринга. На основе входных данных UI – browser engine взаимодейсмтвует с механизмом рендеринга и управляет им.

• rendering engine – благодаря ему получаем заветные страницы, приложения с которыми мы можем взаимодействовать. Например WebKit (Chrome) и Gecko (Firefox). Отвечает за обработку HTML, CSS и JS: строится DOM-дерево, строится CSSOM-дерева, определяет расположение элементов, там проходят в общем различные стадии рендера.

В свою очередь rendering engine состоит из компонентов



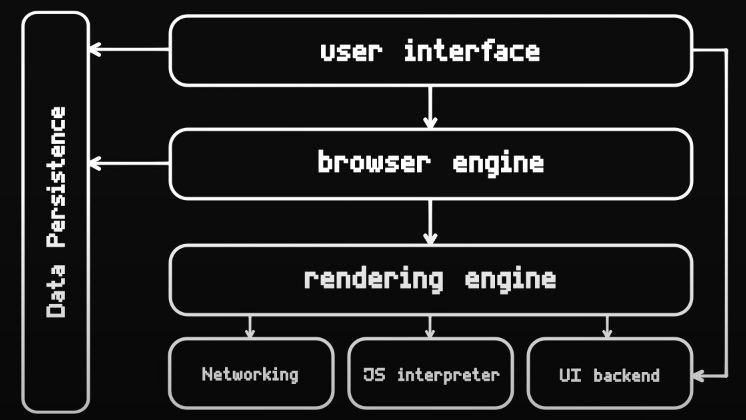
• networking – модуль предназначен для работы с сетью: хранит историю посещаемых сайтов, за доменные имена, за взаимодействие с DNS, работа с HTTP / HTTPS, открытие соединений по протоклам транспортного уровня (UDP / TCP) и т.д.

• js-interpreter – движок обработки JS. Например V8 у Chrome. Поверх этого движка построен Node JS, Google Chrome. Это абстрагированная, не зависимая от платформы технология которая выполняет задачу парсинга JS-кода и его компиляцию в машинный код.

Он также предоставляет heap, call stack, работа с памятью, выделение памяти, сборка мусора.

• ui-backend – логика UI самого браузера.

Есть также хранилище данных. Таким образом выглядит структурная **схема любого браузера**.



Local Storage, session storage, IDB, web SQL, файловая система, например для загрузки картинки в браузер. Также обрабатывает Cookie, хранение вкладок. В общем – локальная база данных.

**Пример**

Имплементация структурной схемы браузеров на примере Google Chrome.



Создать собственный браузер зада вполне реализуемая, поскольку по большей части все браузеры состоят из опенсорсных технологий, важно лишь понимать устройство, структуру браузера и как эти технологии скомпоновать.

**Архитектура движка рендера на примере webkit**

На вход подаётся HTML, CSS. Затем это парсится соответствующими парсерами. На выходе HTML парсера получается DOM дерево, с которым можно взаимождействовать через JS, а на выходе CSS-парсера получается набор стилей. К сгенерированному DOM-дереву применяются получивашееся CSS правила и получается дерево рендера. Это дерево проходит некоторые другие стадии – на выходе получается отрендеренная страница.

