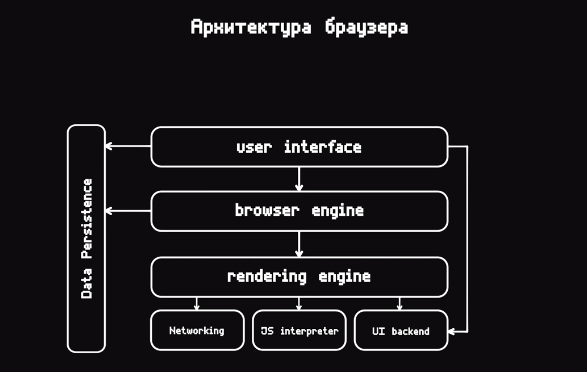
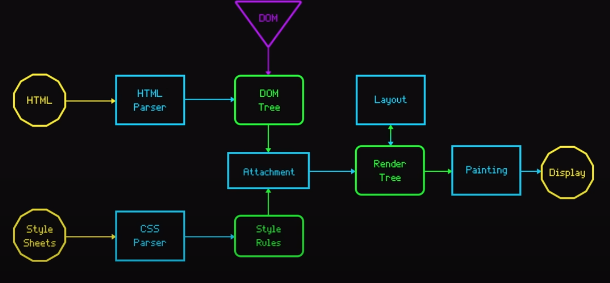
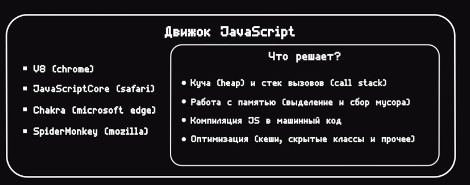
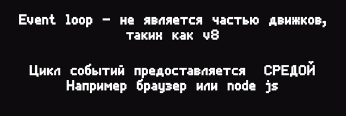
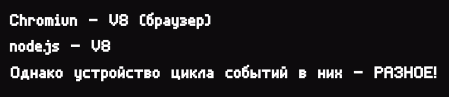
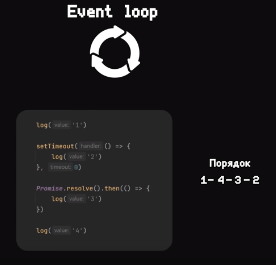
EventLoop.  
  
Архитектура браузера.

- user interface

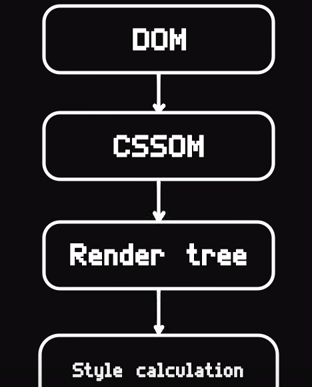
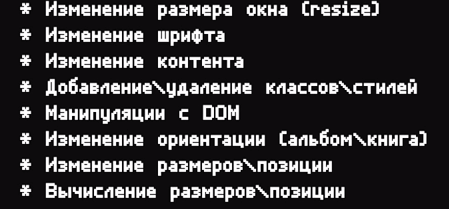
- движок браузера

- движок рендера (благодаря которому всё работает): webKit, geko – обработка JS, постройка ДОМ дерева

--- Networking(история сайтов, доменные имена, правила обработки различных соединений)  
--- Движок JS (часть, которая обрабатывает js code) : V8 движок. Он предоставляет heap, stack, компилирует JS в машинный код.  
--- UI backend – подкопотная логика  
  
- Определенное хранилище данных: localstorage, файловая система. Крч хранилище, которое использует браузер.   
  
  
  
  
  
  
  
  
  
  
  
  
На примере Google Chrome.  
  
  
Архитектура движка рендера (webkit).  
  
  
  
  
  
  
  
  
  
  
  
**Браузерный Event Loop.**  
Event Loop это не часть JS. Event Loop в node.js и в браузере это разные вещи которые решают одну задачу.  
  
Event loop – отдельный механизм, который позволяет использовать неблокирующую модель ввода и вывода (не ждать конца запроса например).  
  
**CallStack. Стэк вызовов.**  
За него отвечает движок JS (например V8).  
Он складывает функции.  
  
CallStack не бесконечный! Его можно крашнуть.  
  
**Очередь задач. Асинхронный код.**  
Event Loop внутри себя содержит очередь задач (task queue).  
Туда попадают асинхронные задачи.  
  
Задачи из очереди выполняются только после вызова всех функций из стека.  
  
**Как задачи попадают в очередь? Задачи JS движка.**  
За очередь отвечает Event Loop, а за CallStack движок JS.  
  
  
!!!!!!!!!!!!!!!  
  
  
  
  
У каждого браузера есть Web Api (addEventListener, fetch, и т.д.).  
В web api происходит регистрация таймера, по истечению времени задача попадает в очередь и после выполнения всех синхронных задач, задача попадает в stack.  
  
Так же работает с addEventListener, после выполнения их из стэка они регистрируются в Web Api (и будут там пока мы их явно не удалим), и при клике генерируется задача и попадает в очередь задач.   
  
**Промисы и очереди микро и макро задач.**  
  
  
Как event loop понимает какие задачи брать ему быстрее?  
На самом деле есть очередь задач это 2 очереди: очередь событий(macrotask queue) и очередь задач(microtask queue).  
У этих очередей есть приоритет.  
  
Промисы всегда попадают в очередь MicroTask.  
  
Специфика работы Event Loop – в приоритете всегда выполнять сначала все microtasks, потом ОДНА задача из macrotasks(может генерировать новые microtasks).  
  
Итоги: сначала выполняются все синхронные задачи, потом все микротаски, потом ОДНА макротаска, если порождает микротаски, то сначала выполняются снова они.  
  
  
  
  
  
 **Что порождает микротаски и макротаски?**Микротаски 99% - промисы.  
Но также микротаск можно создать явно.  
Их также порождает mutationObserver.

Макротаски – таймеры, события, браузерные нюансы(подкопотное).

**Что такое mutataionObserver?**Если кратко, то специальный механизм, который позволяет следить за изменениями в DOM.

**Стадии рендера. Как браузер отрисовывает страницу.**1) калькуляция стилей.  
  
2) layout, на выход получаем layout tree.  
  
3) Paint разукраска полученного layout tree в соответствии со стилями.  
  
  
4) Compositing.  
Создаёт дерево слоёв.  
  
  
Когда же строится DOM дерево?  
При получении html и css, раньше первой стадии.   
Так при первичной отрисовке.  
  
  
  
Рендер – дорогостоящая операция.  
Что его вызывает?  
  
  
  
  
**Что такое node.js? Из чего состоит?**  
Задача Node.js: преобразовать исходный js код в машинный код.  
Движок V8.  
Libuv – библиотека, она знает как работать на MAC, WINDOWS, etc.  
Libuv отвечает кроссплатформенный ввод и вывод и Event loop.  
  
Многопоточное программирование.  
  
Event Loop преимущества и недостатки:  
+ обработка больших операций ввода вывода.  
- много асинхронного кода  
 **Блокирующий и неблокирующий ввод и вывод.**  
Классическая модель – блокирующий ввод и вывод.  
Пока команда не выполнилась, к следующей не переходим.  
  
Сам по себе JS однопоточный и вся асинхронщина достигается из-за event loop.  
  
Что же про Node.js? Сам по себе однопоточный.  
Но в основе лежит Libuv, которая может управлять потоками.  
  
**Демультиплексор событий.**  
**Event Loop в NODE.JS.**