1 вводит url

2 проверка сайт есть в кэше (есть ли у тебя инфа где находится адрес сервера)

2.1 идет в файлик hosts (xamp, openserver)

2.2 обращается к днс серверу (скажи ип адрес Яндекса, рекурсивный поиск от днс к днсу) tcp соединения (рукопожатие) оператор предоставляет адрес dns сервева. Top level domain (.ru, .com, .net) root-ru-yandex.ru-app.yande.ru

2.3 root-domain, top-level domain, second-level domains third-level domains (. -> edu, gov, org, com -> ya.ru, vk.com, dc.ru -> tips.dc.com, dev.vk.com

3. если нет делает http запрос сначала узнает ip сайта и потом совершает запрос

4. опрашивает сервер и сохраняет в кэшу

0. проверка hosts файла

1. кэш браузера (1мин)

2. кэш ос

3. кэш маршрутизатора

4. кэш провайдера

5. ip получен, теперь подключаемся к серверу (уст. Рукопожатие syn/ack)

Соединение безопасное

1. Браузер говорит сайту вот держи мой список шифровании, сервер отвечает ок я знаю как ты умеешь и сейчас я запущу алгоритм асимметричного шифрования. Сервер отдает (public key + certificate) public/private key
2. Приходит ответ с заголовком content-type: text/html
3. Dom + cssom = render tree
4. Layout operation => paint operation => composition operation

рендеринг в браузере

1. После парсинга html он превращается в dom + ccsom
2. Следующим шагом он объединяется в render tree (display: none, head)
3. Геометрия (высота, ширина) Flow/Layout но пикселя пока еще не нарисован
4. Отрисовка Paint
5. Фиолетовый – что-то считает
6. Зеленый – что-то рисует
7. Оранжевый – что-то вычисляет (js)
8. Will-change

HTML parsing

1. Браузер вынужден много прощать (слабая валидация) забыли закрыть тэг, вложили тэг туда куда не нужно. (div > li, ul/ol/menu) Он все равно постарается отобразить элемент. Соот-но медленный парсинг
2. Когда встречаем тэг script парсинг останавливается, парсится, выполняется и поэтому тэг script ставят в конце страницы. Defer – гарантирует порядок выполнения. Async – не соблюдает порядок

Css parsing

1. Селекторы имеют производительность

Microtask – promise, mutation observer, queue microtask ResizeObserver IntersectionObserver

Микрозадачи выполняются только тогда, когда очередь задач пуста.

Если вы хотите поместить задачу в микрозадачу, но ваш код не является Promise, вы можете использовать queueMicrotask. Этот метод добавляет вашу функцию в очередь микрозадач.

requestAnimationFrame – используется для того чтобы в обязательном порядке вызвать какой-то код пред следующей отрисовкой.

requestIdleCallback – можно выполнять задачи во время простоя страницы

Promise — это микрозадача. Микрозадачи выполняются только тогда, когда очередь задач полностью пуста

Макрозадачи - Их можно выполнять после полного выполнения всех задач и микрозадач.

100 – информационные ответы (101 switch)

200 – успех

300 – редирект (307, 308, 304-cache not-modified)

400 – ошибки клиента

500 – ошибки сервера

Заголовки:

General (date, connection)

Request

Response (server)

Entity (content-type, expires – ответ считается устаревшим)

Date, Location, Origin, Referer, Content-Type, user-agent, set-cookie(res), Cookie(req), Server

Set-Cookie используется для отправки cookies с сервера на агент пользователя.

Cookie – проблема http друг про друга не знают (авторизация, слежка, аналитика, реклама)

Те сервер говорит установи эту куки и при каждом запросе мы передаем, те постоянно гоняется между сервером и клиентом. Те прокси может увидеть

События

1. Bubbling - по умолчанию равно **true**
2. Capturing - по умолчанию равно **false** (очень редко используется в реальном коде)
3. Target Phase

Капчеринг(true) и баблинг(false)

itBubbles = event.bubbles

isCanPrevent = event.cancelable (resize, scroll, load)

stopPropagation – прерывает события по дереву

stopImmediatePropagation – отменяет прочие события + stopPropagation

как запретить http-кеширование - cache-control: no-store

закешировать на время – expires, max-age=3600

валидация if-modified-since or etag

proxy находится между браузером и сервером. Разрешить кэширование на стороне прокси – cache-control: public, private – разрешить только на стороне браузера.

Управлять кэшем из js нельзя.

Link rel:

dns-prefetch - разрешить DNS как можно быстрее

preconnect – dns + tcp + https

prefetch - загружает ресурс и сохраняет его в кеше браузера для последующего использования.

Prerender - Он сообщает браузеру запросить URL-адрес и загрузить все ресурсы.

Cookies – сессионные и персистентные.

Чтобы изм куку нужно учитывать key=value, domain, path иначе создаётся новая кука.

Чтобы удалить нужно дату в минус установить или в 0



Graceful degradation - Практика создания приложения для современных браузеров с сохранением его функциональности в старых браузерах.

Progressive enhancement - Практика создания приложения для базового уровня пользовательского опыта, но добавление функциональных улучшений, когда браузер поддерживает их.

CSS:

Избегать те стили, которые вызывают reflow, repaint, and compositing

**What causes reflow?**

1. DOM manipulation ( adding, removing, changing )
2. Changing the content of elements
3. Recalculation or changing CSS properties
4. Browser window size manipulation

### How to avoid reflow?

Try to avoid doing things in the above list and some more in the below

* avoid setting multiple inline style
* apply animation to the elements that are positioned fixed or absolute
* avoid tables for layout