HTTP

- Текстовый
- Без состояния

Строка запроса – заголовки – тело

Статус ответа – заголовки – тело

Клиент - инициатор

HTTP/0.9 – HTTP/1.0 – HTTP/1.1 – HTTP/2 – HTTP/3

HTTP/2:

- Бинарный
- Server-Push
- Мультиплексирование

HTTP/3:

QUIC (UDP)

Клиент-сервер в Вебе

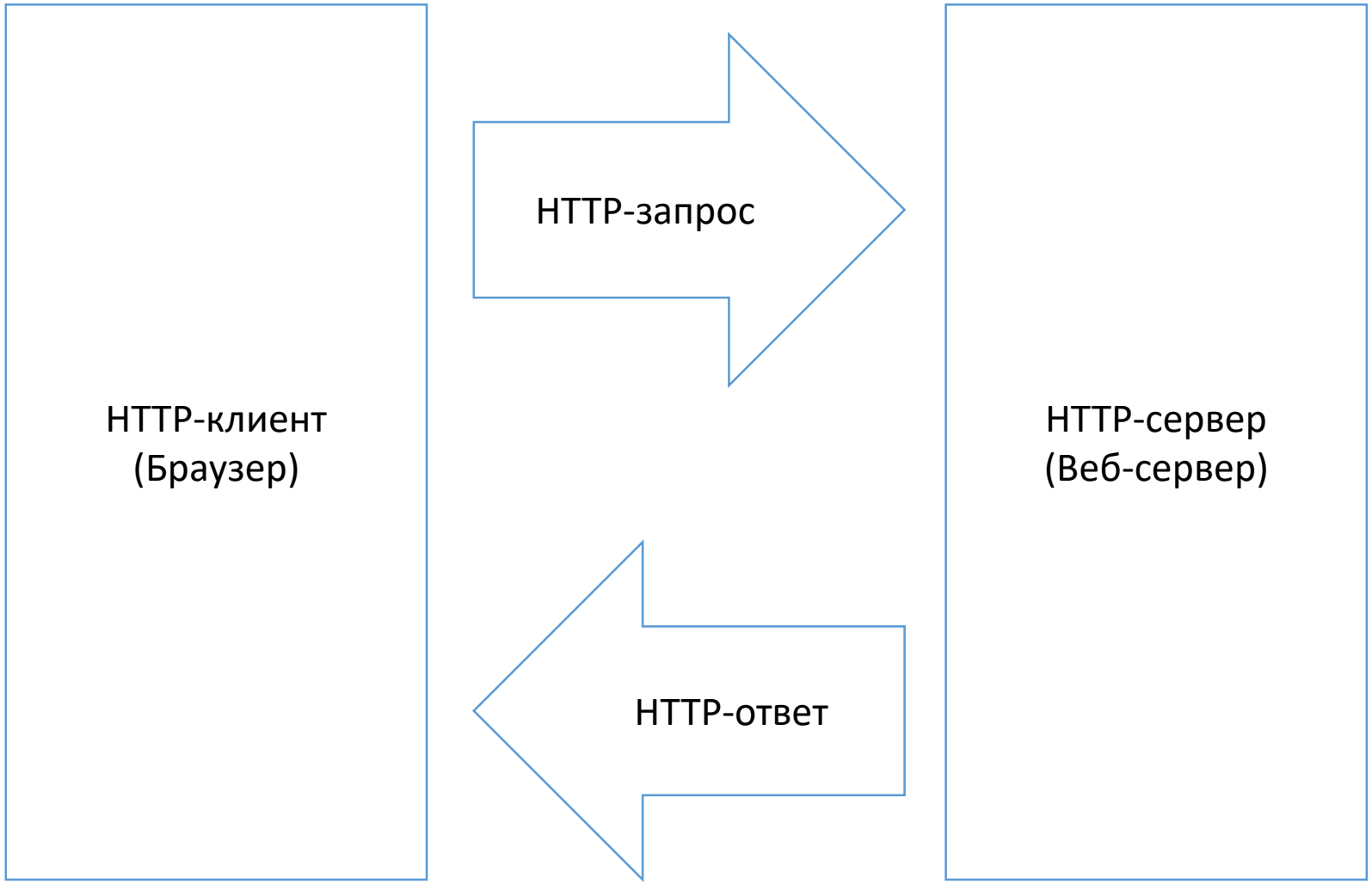
Клиент-сервер

Клиент-сервер —сетевая архитектура, в которой сетевая нагрузка распределена между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

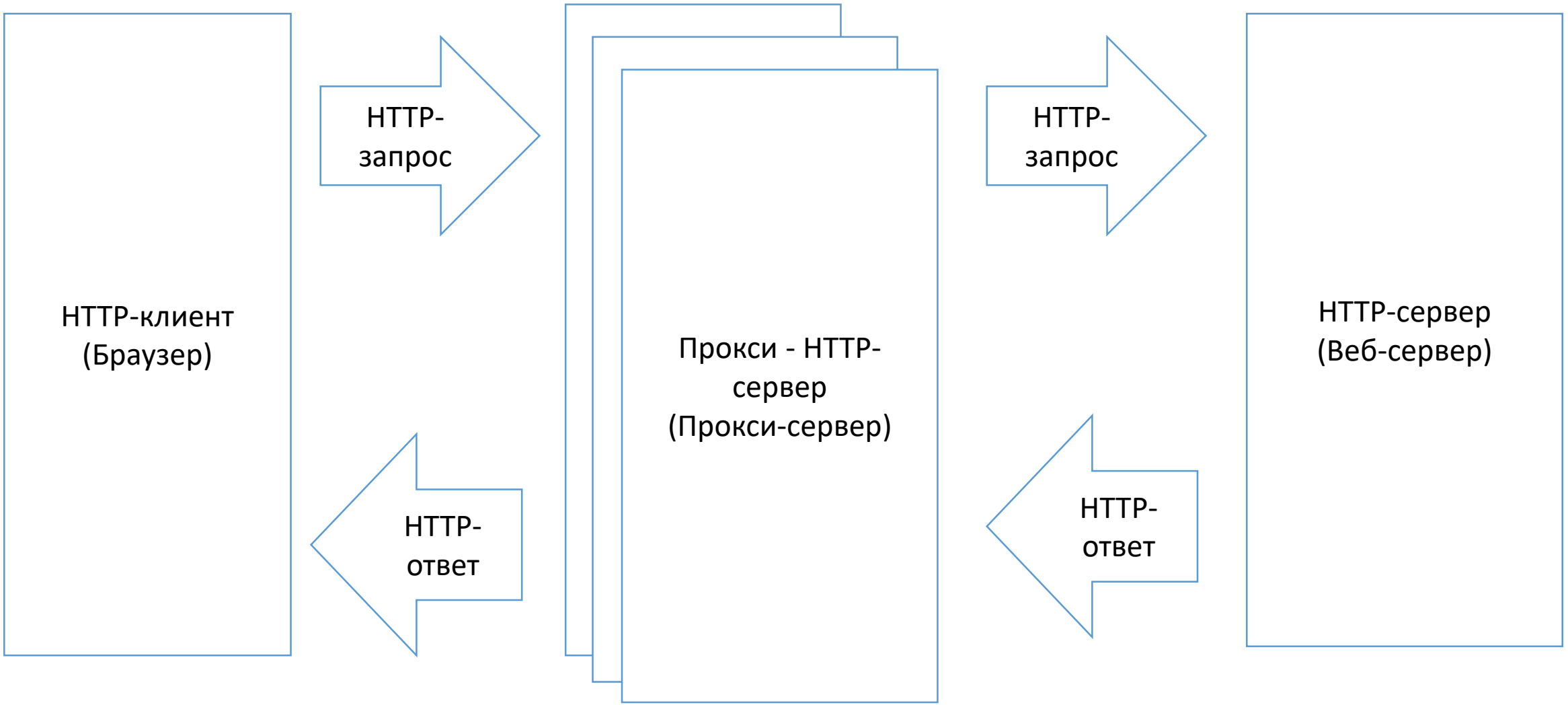
Классический «клиент-сервер» в Web

- Взаимодействие по протоколу HTTP/протоколам поверх HTTP
- HTTP работает поверх TCP
- Абстрагирование от физической реализации сети
- Адресация – на сокетах (ip-адрес + порт)
- Доменные имена преобразуются в ip-адрес
- Клиент – всегда инициатор сессии

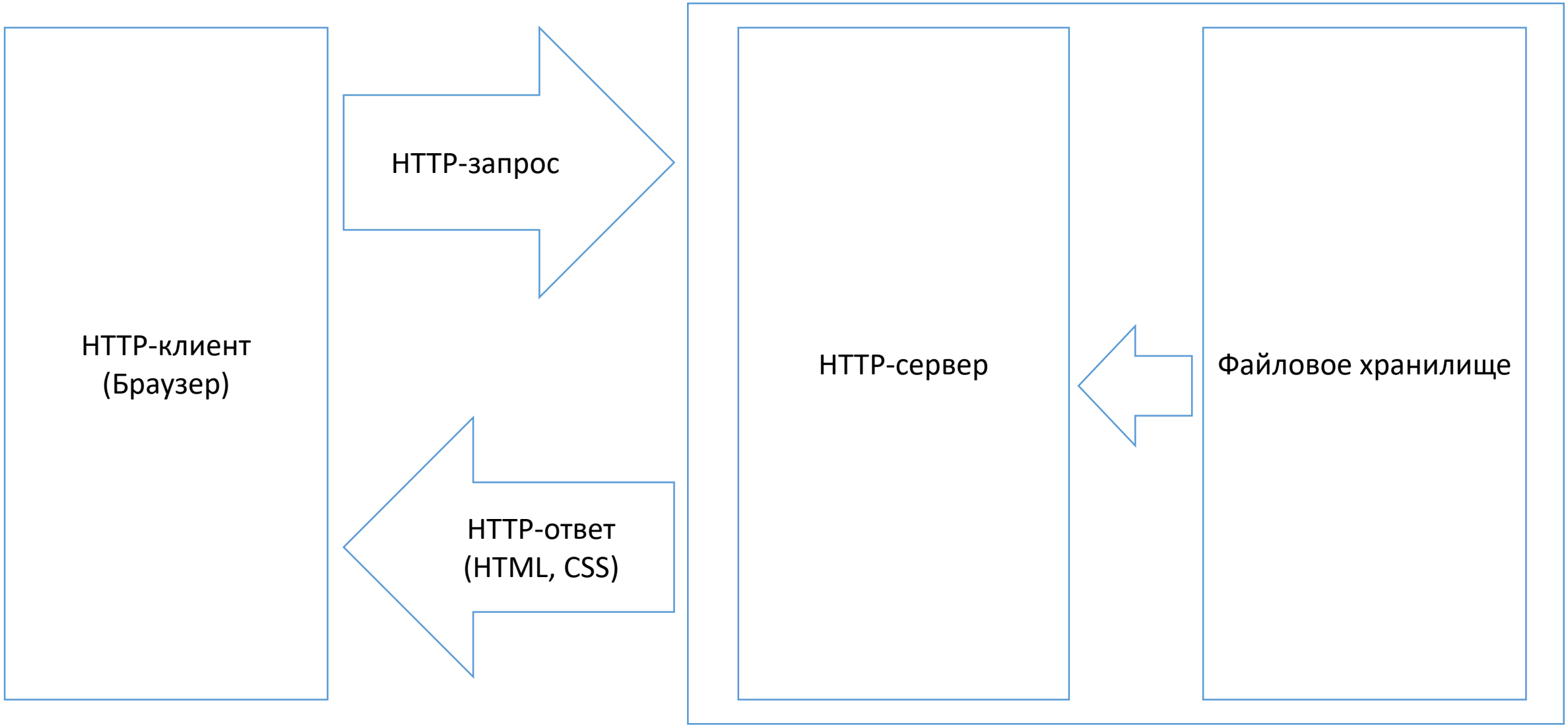
Клиент-серверное взаимодействие в HTTP



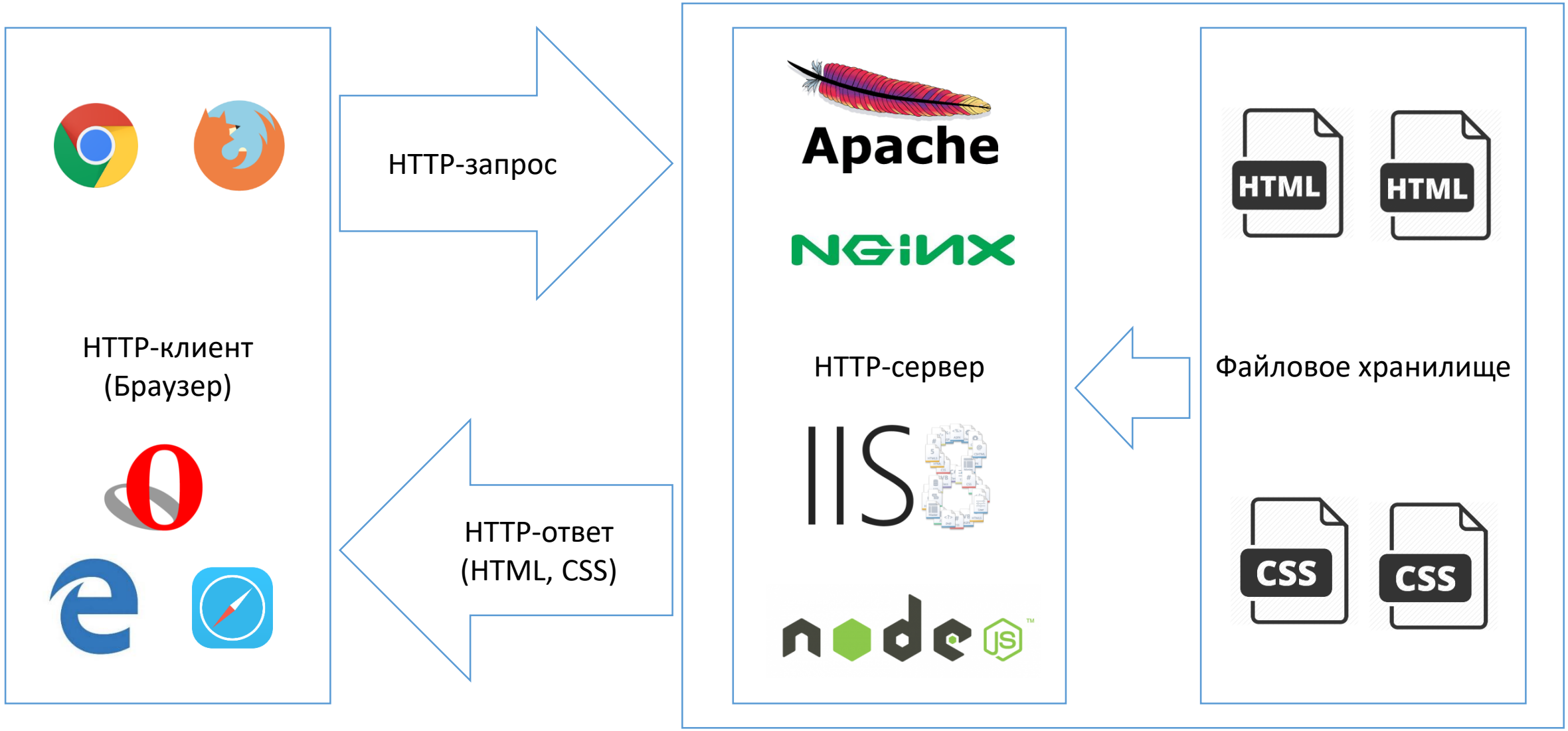
Клиент-серверное взаимодействие в HTTP



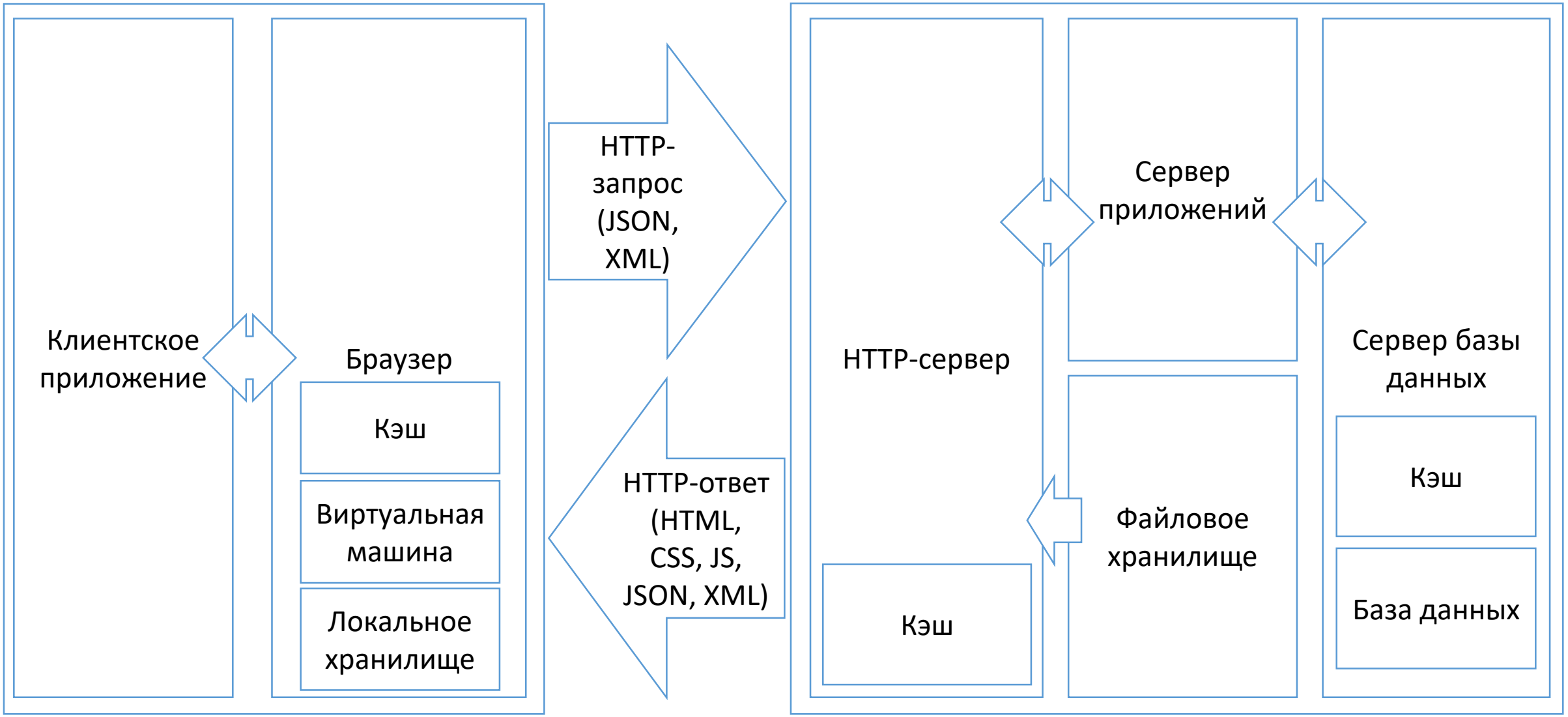
Клиент-серверная архитектура (статика)



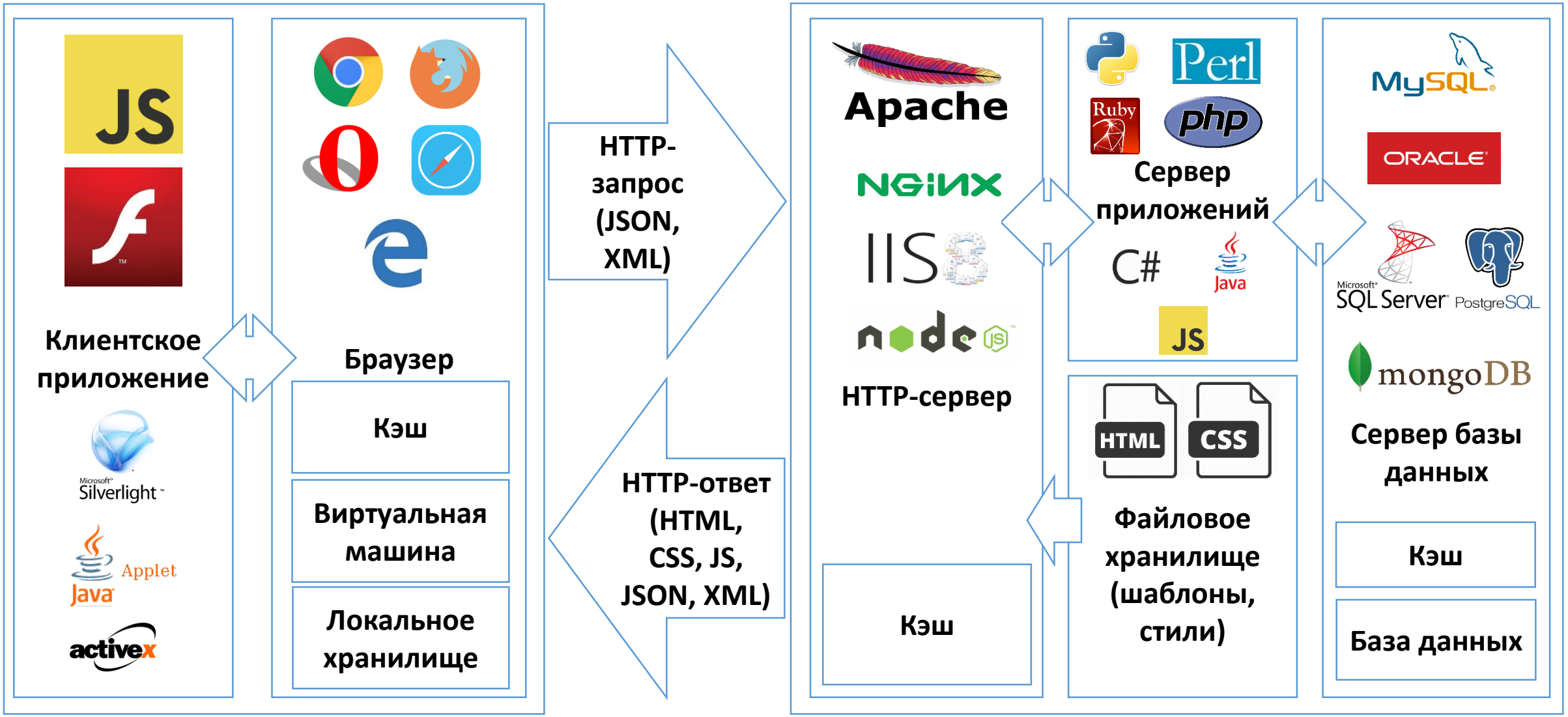
Клиент-серверная архитектура (статика)



Клиент-серверная архитектура (динамика)



Клиент-серверная архитектура (динамика)



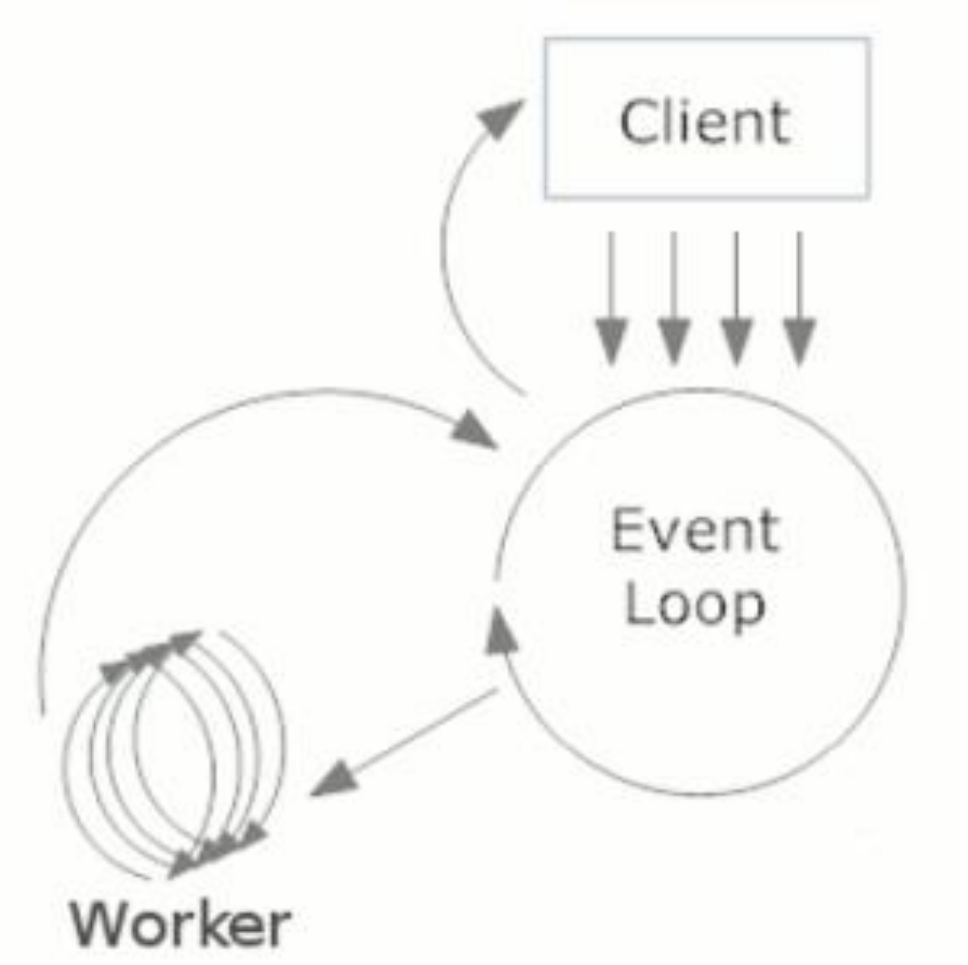
Клиент

Браузер

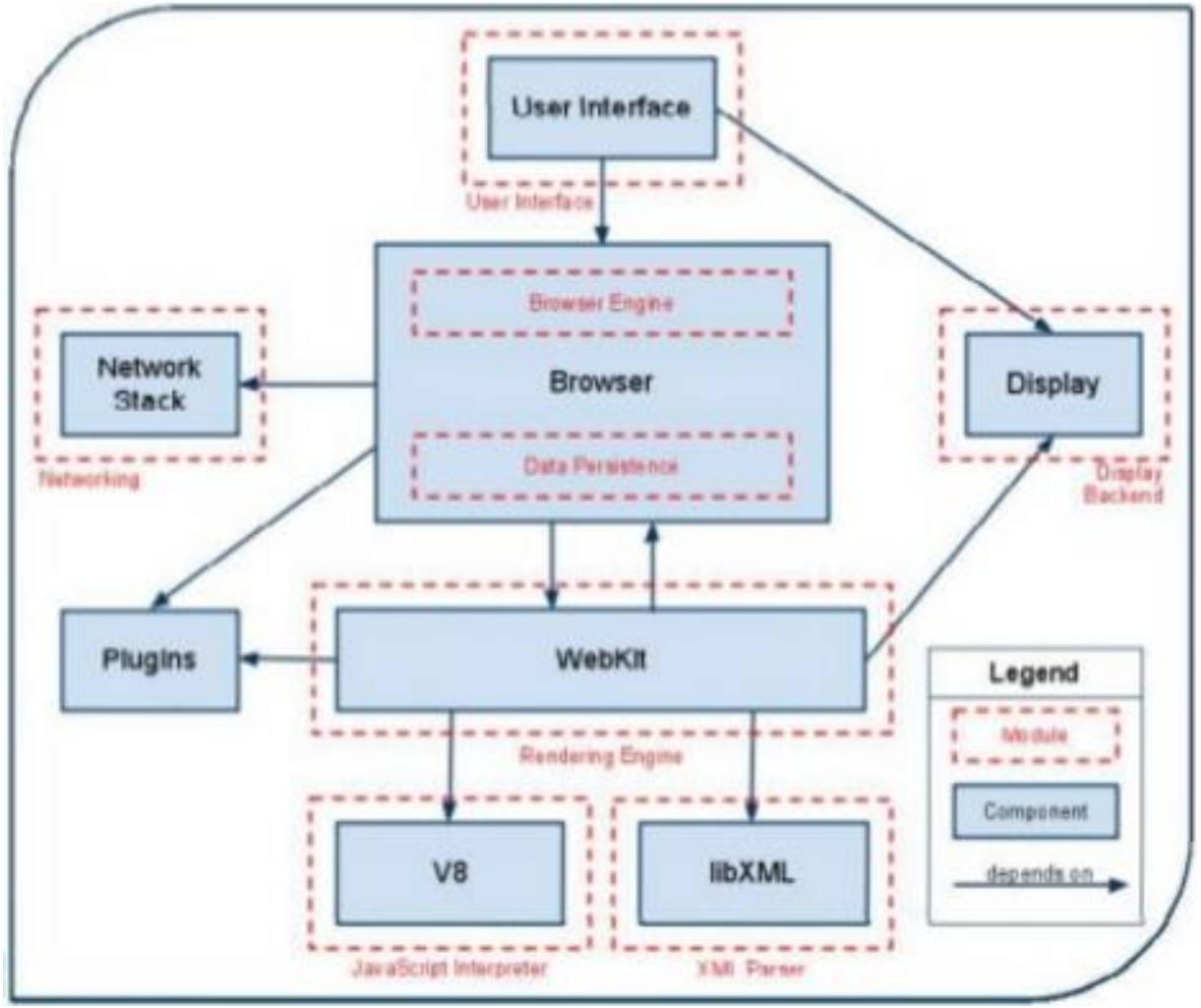
- **Реализация HTTP, DNS, TCP**
- **Интерпретаторы JS,HTML,CSS**
- **Визуализация (рендеринг)**
- Поддержка состояния
- Кеширование
- Авторизация
- Работа с окружением
- Система плагинов

Браузер

- Цикл событий (event loop)
- Асинхронность

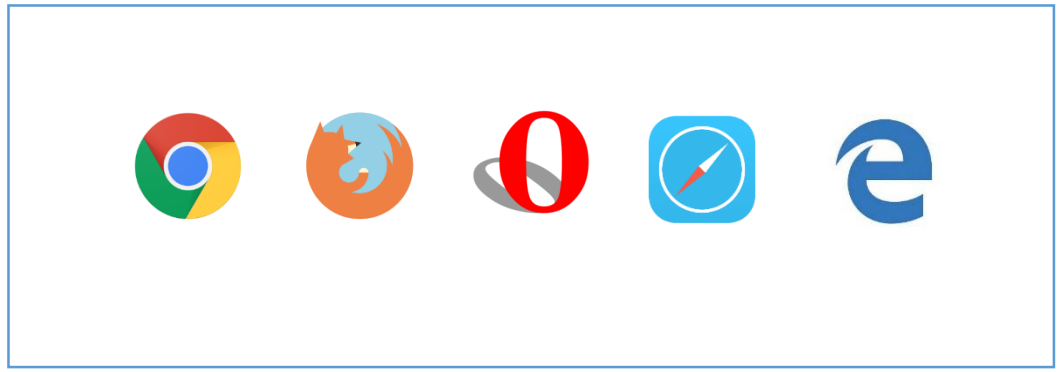


Архитектура браузера Chrome

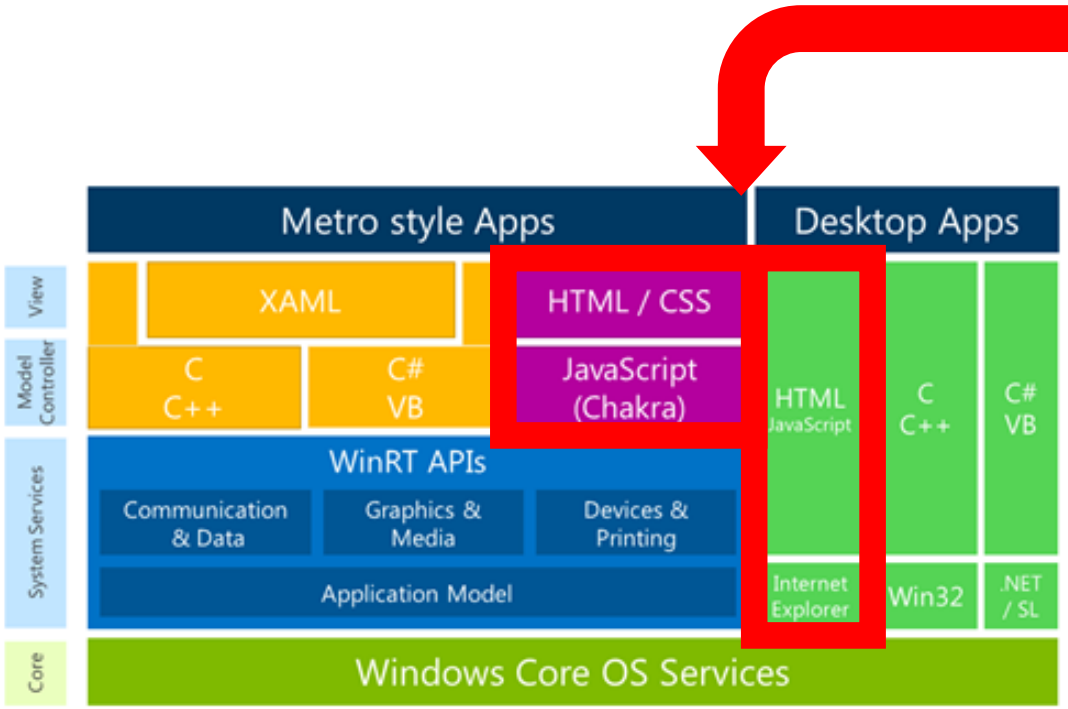


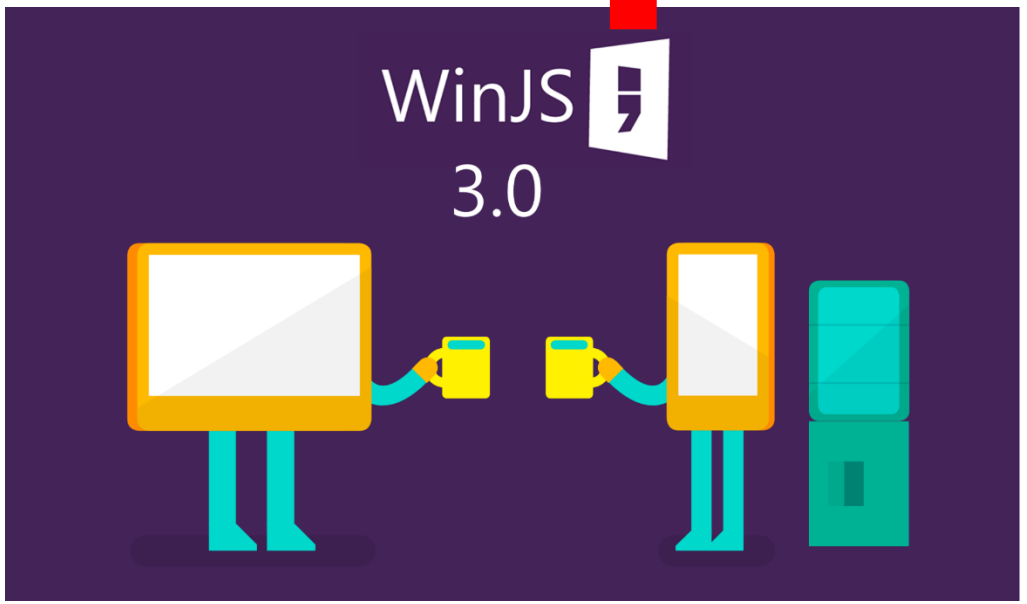
Клиент: браузер?

- Браузер
- Chrome app / Chrome OS
- Firefox app
- WinJS
- PhoneGap / Apache Cordova
- Smart TV
- Flash
- Silverlight

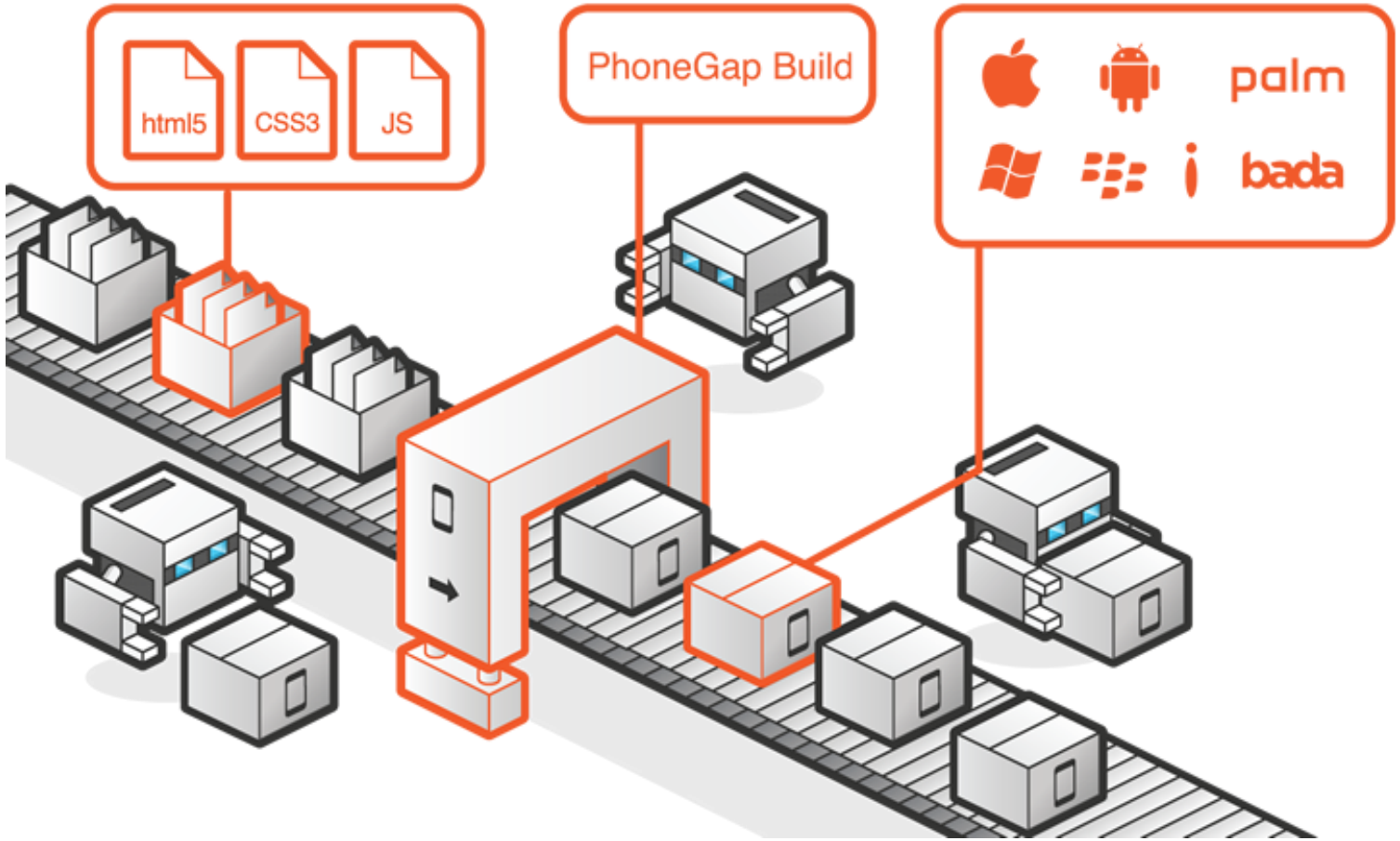


WinJS


Metro style Apps
Desktop Apps
View
Model Controller
System Services
Core
XAML
HTML / CSS
C
C++
C#
VB
JavaScript
(Chakra)
HTML
JavaScript
C
C++
C#
VB
WinRT APIs
Communication & Data
Graphics & Media
Devices & Printing
Application Model
Internet Explorer
Win32
.NET / SL
Windows Core OS Services


WinJS
3.0

PhoneGap – инфраструктура над Apache Cordova



Flash и Silverlight

Сторонние плагины

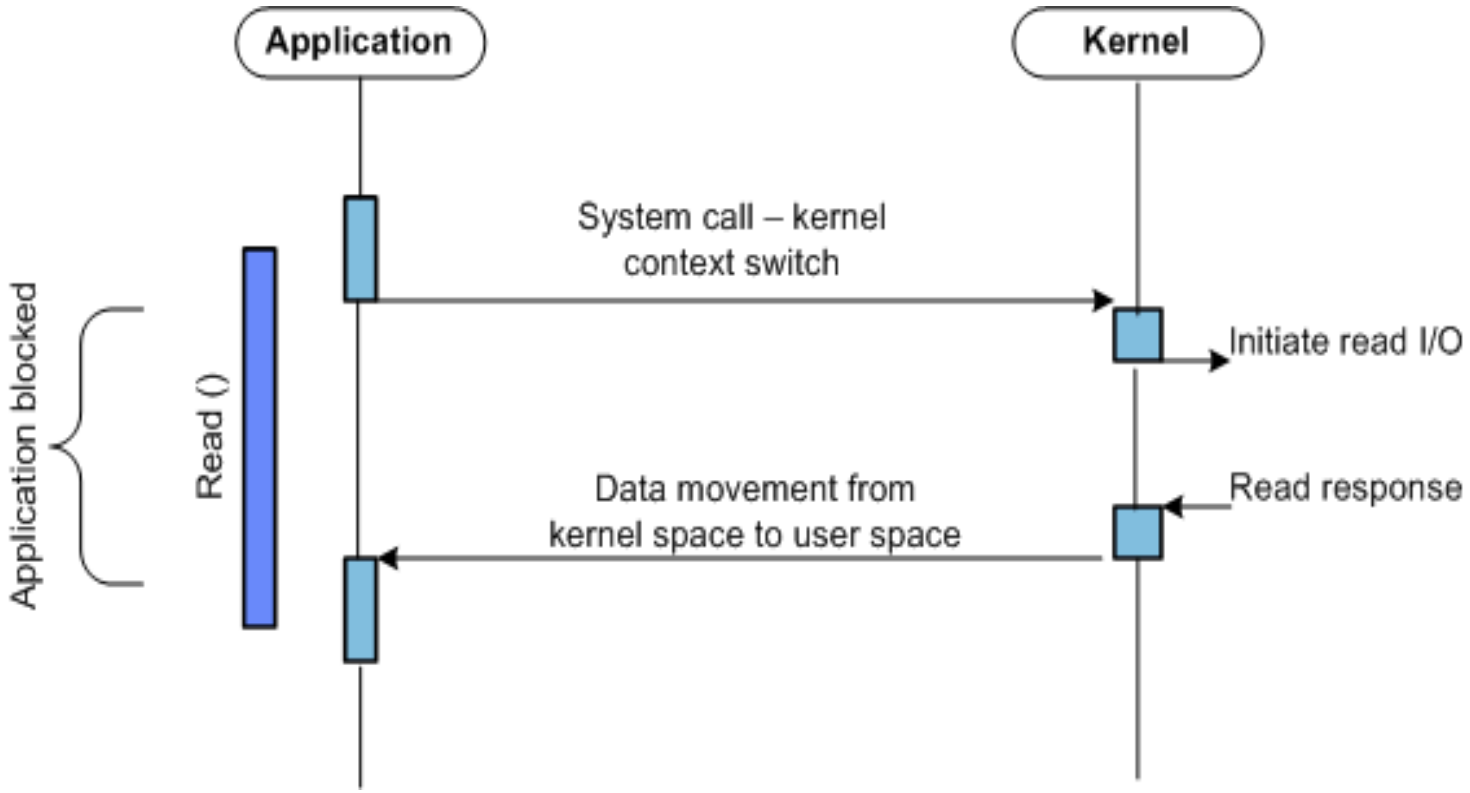
- Язык программирования (ActiveX, C#)
- Язык верстки (XAML)
- Среда выполнения

Сервер

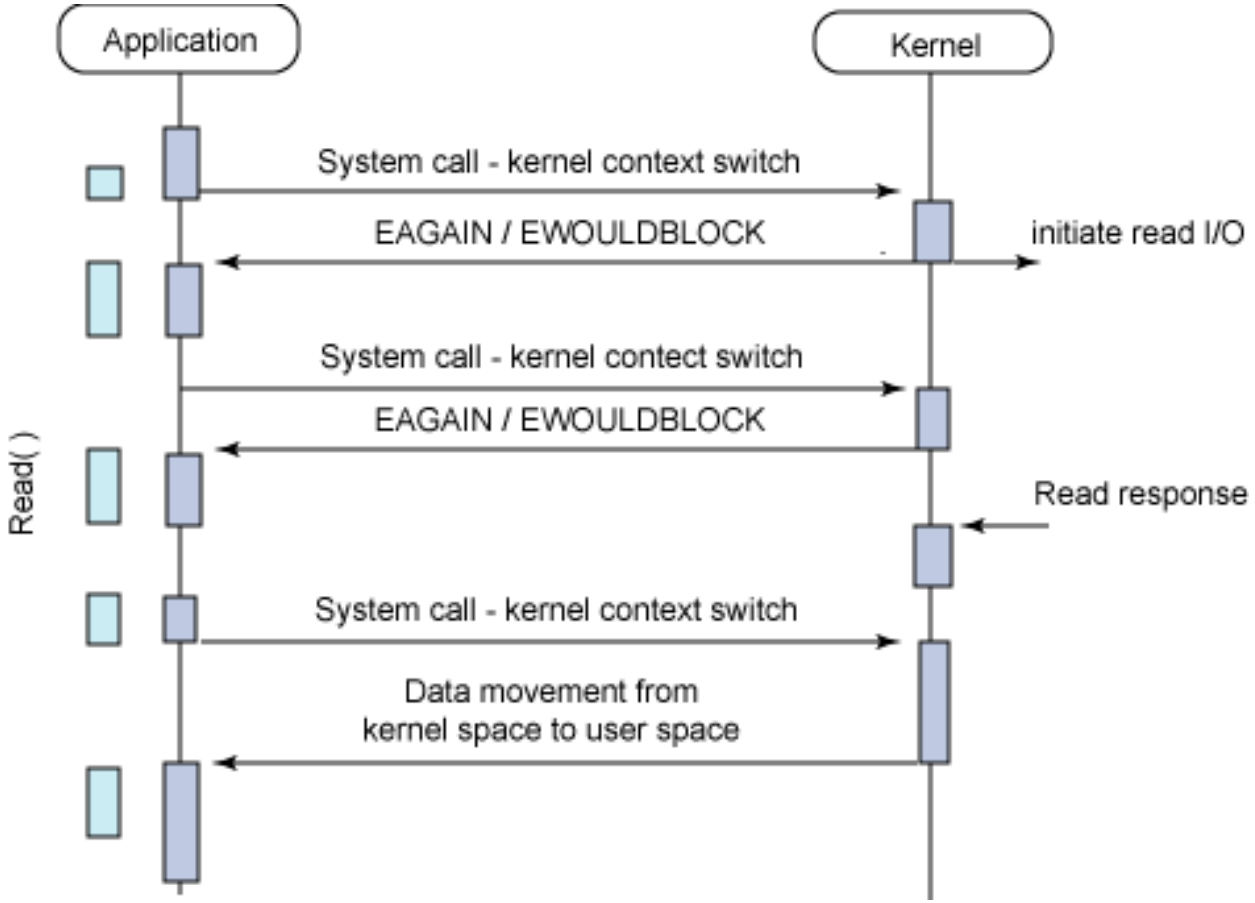
Обработка запросов

Обработка запросов – операция **ввода-вывода**

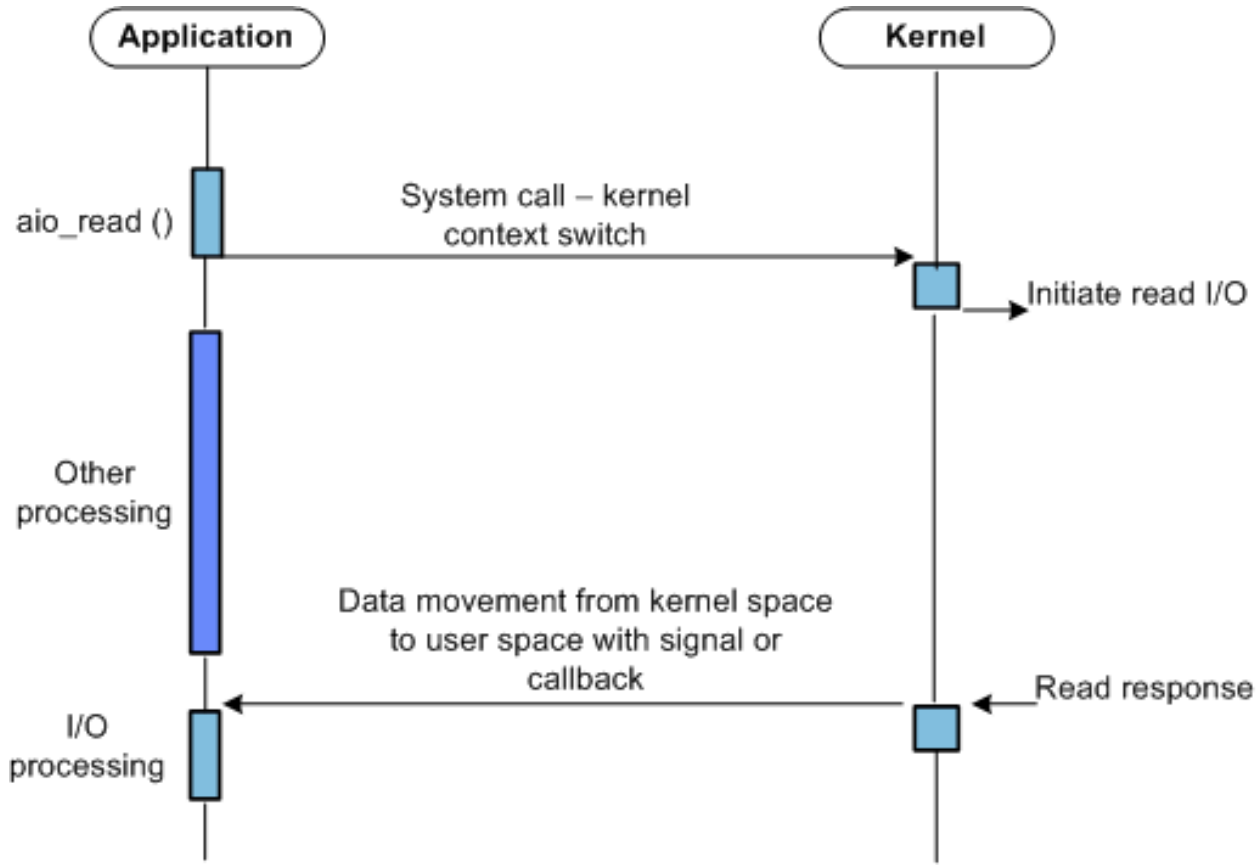
Обработка запросов. Блокирующий ввод-вывод



Обработка запросов. Неблокирующий ввод-вывод



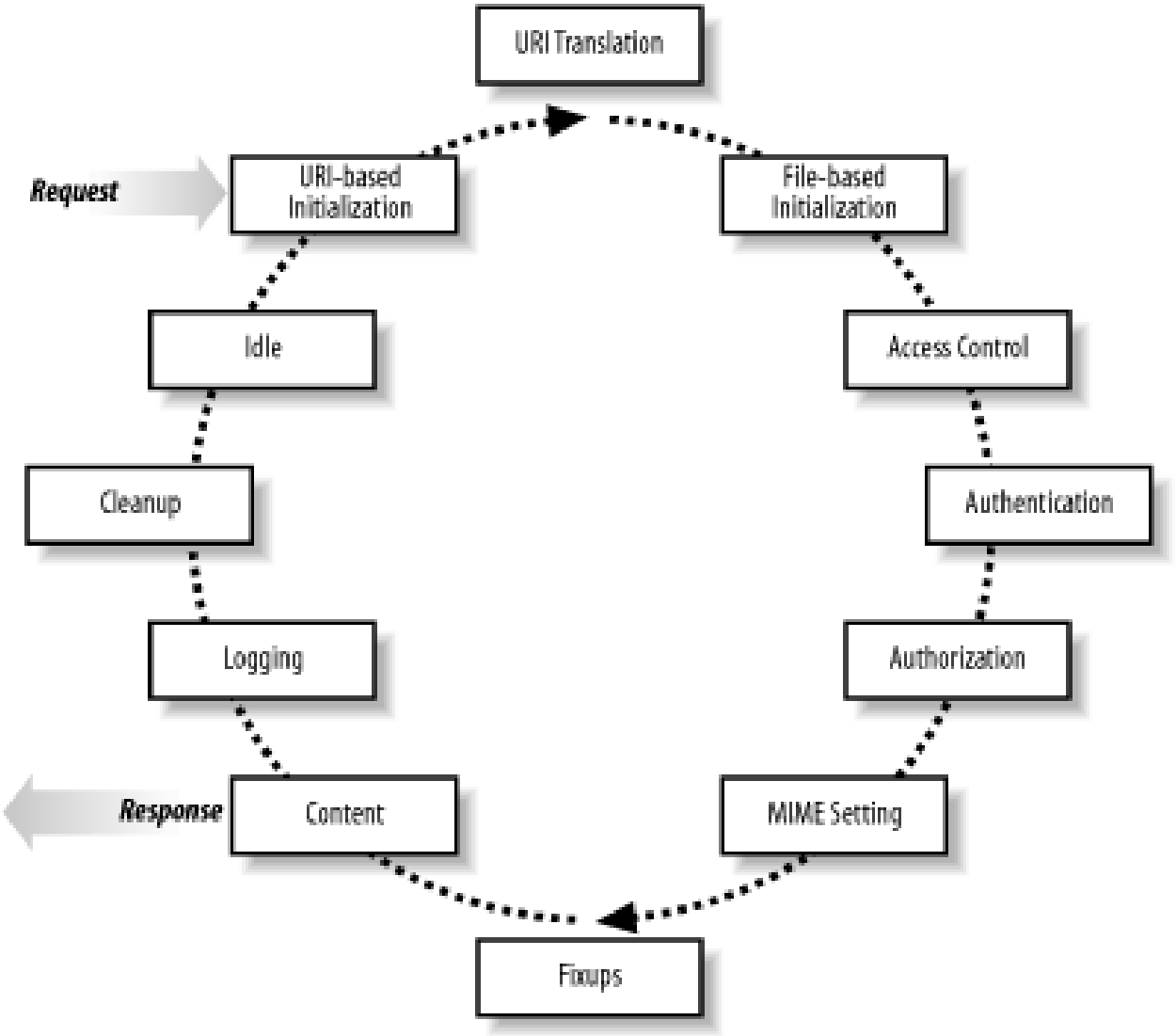
Обработка запросов. Асинхронный ввод-вывод



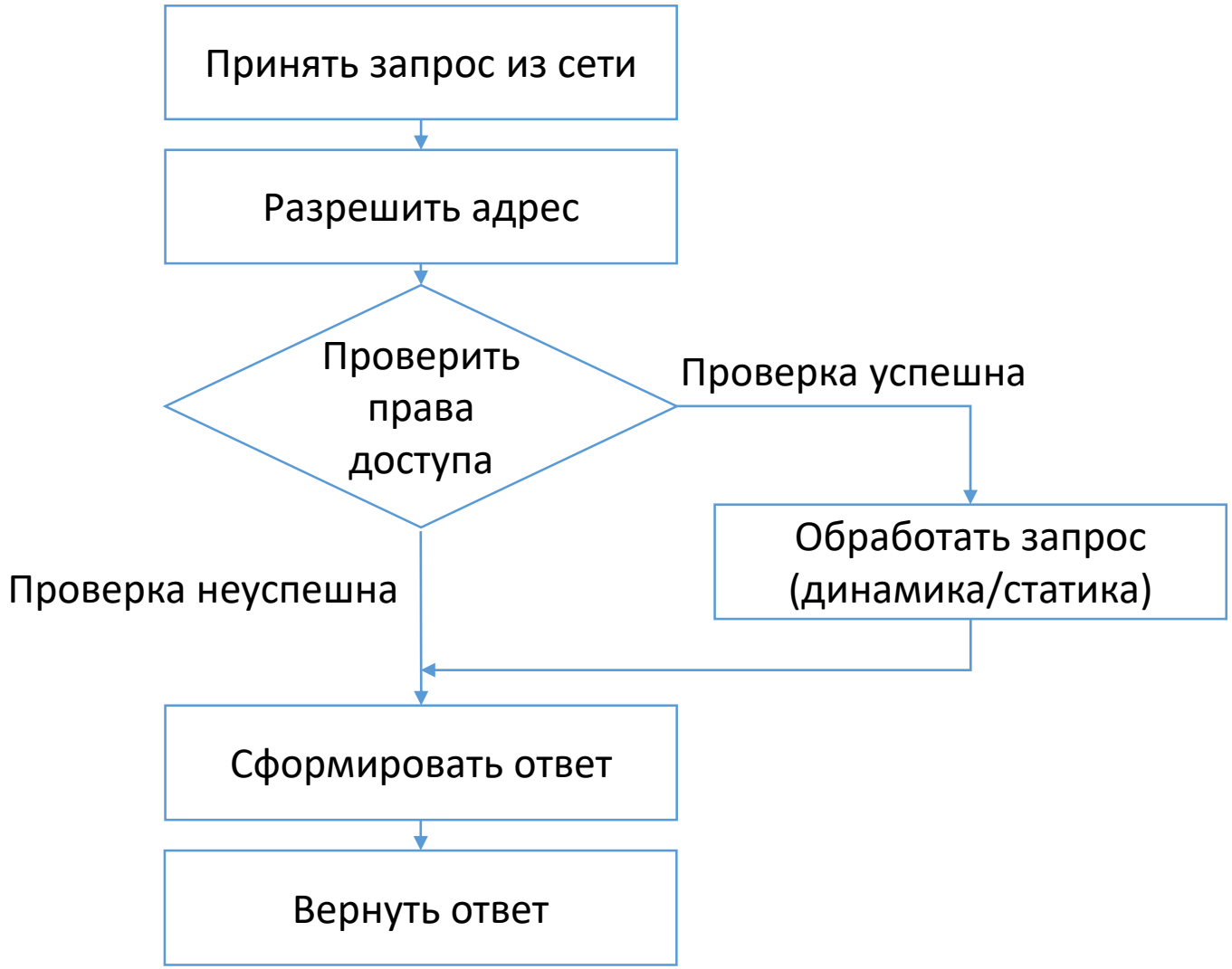
Популярные веб-сервера

Веб-сервера	Обработка запросов	Язык
Apache	prefork (процессы) worker (процессы+потоки)	C
nginx, lighttpd	Неблокирующие	C
IIS, Tomcat, Jetty	threads (потоки)	C#, Java
Node.JS, Tornado, POE	Асинхронные, несколько рабочих процессов	языки высокого уровня
Starman, Hypnotoad	prefork (процессы)	языки высокого уровня
Erlang!	Легковесные потоки Erlang	Erlang
Для разработки (Django, webrick..)	Один поток	языки высокого уровня

Алгоритм веб-сервера



Алгоритм веб-сервера



Задачи веб-сервера

- Поддержка HTTP (HTTP-сервер)
- Хостинг файлов
- Распознавание URL
- Проверка пользовательских данных
- Управлением сессиями пользователей
- Логирование
- Взаимодействие с сервером приложений

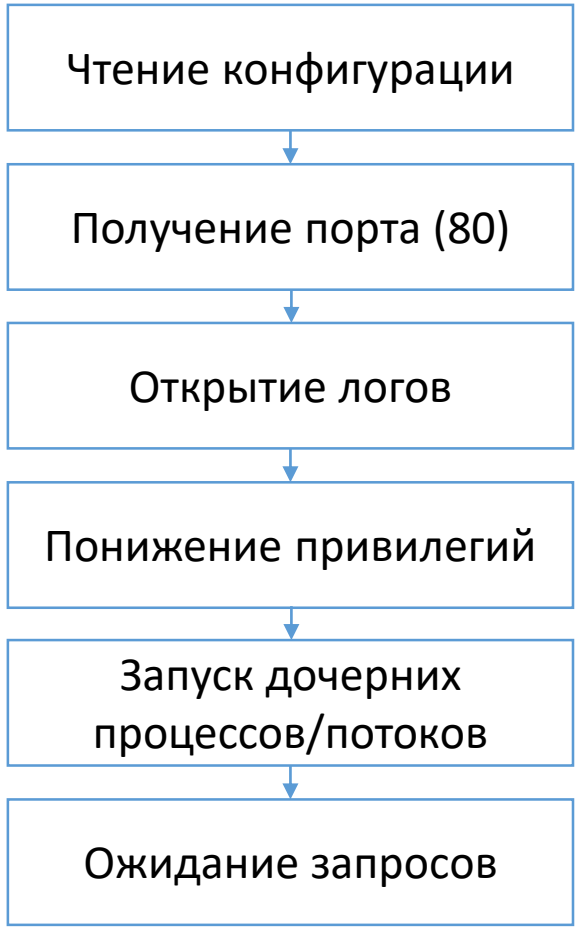
Терминология веб-серверов

- Порт (port)
- Соединение (connection)
- Сокет (socket)
- Запрос (request)
- Путь (location)
- Файлы и директории (directory)
- Файл настроек (httpd.conf, nginx.conf)
- Лог доступа (access.log)
- Лог ошибок (error.log)

Типичная организация веб-сервера

- Процессы
- Master (root, 1 процесс)
  - Чтение и валидация конфигурации
  - Открытие сокетов
  - Открытие файлов логов
  - Запуск и управление дочерними процессами (worker)
  - Graceful restart, Binary updates
- Worker (www-data, 1+ процессов)
  - Обработка входящих запросов

Типовой запуск веб-сервера

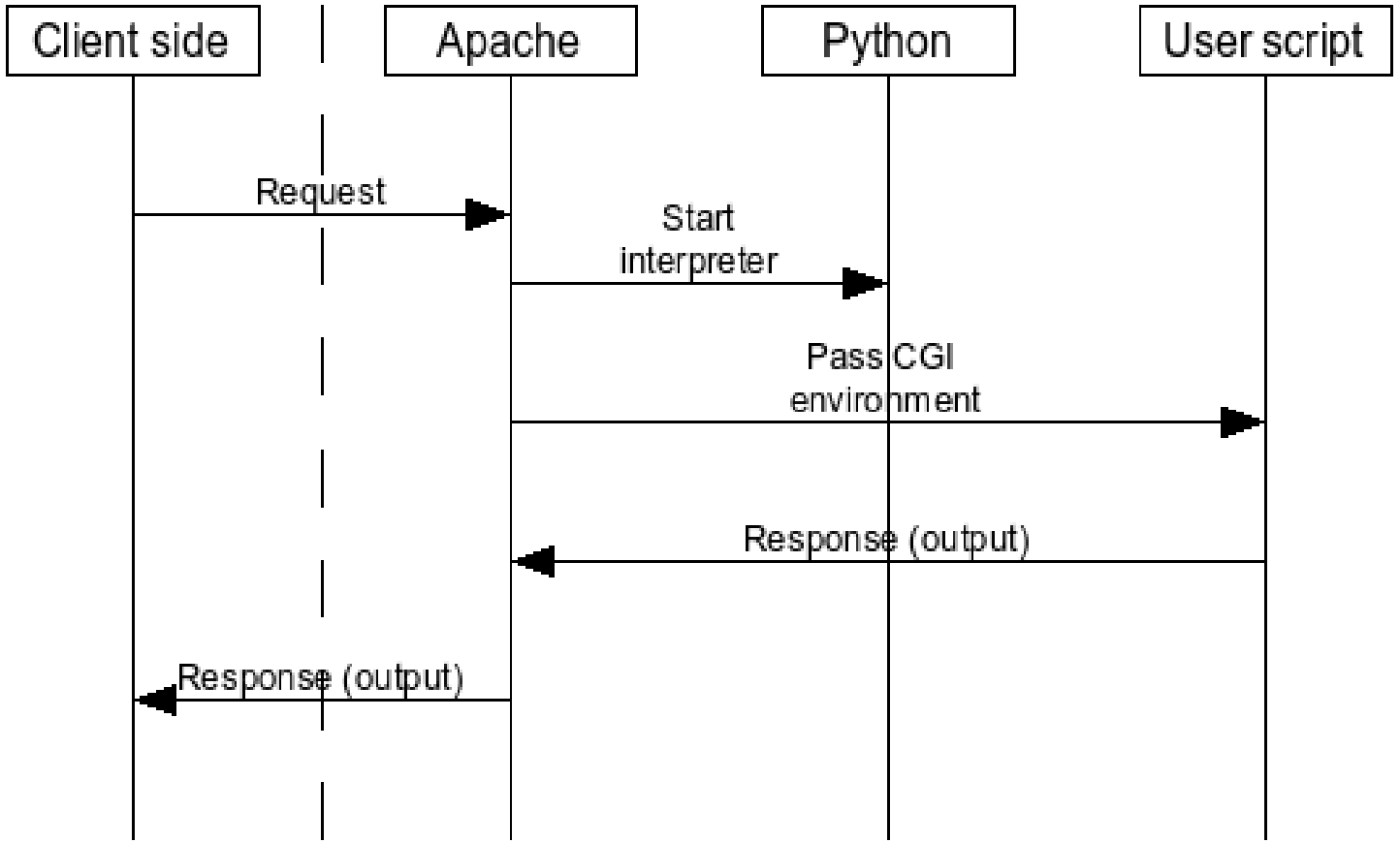


Статический веб-сервер

Выдача по запросу:

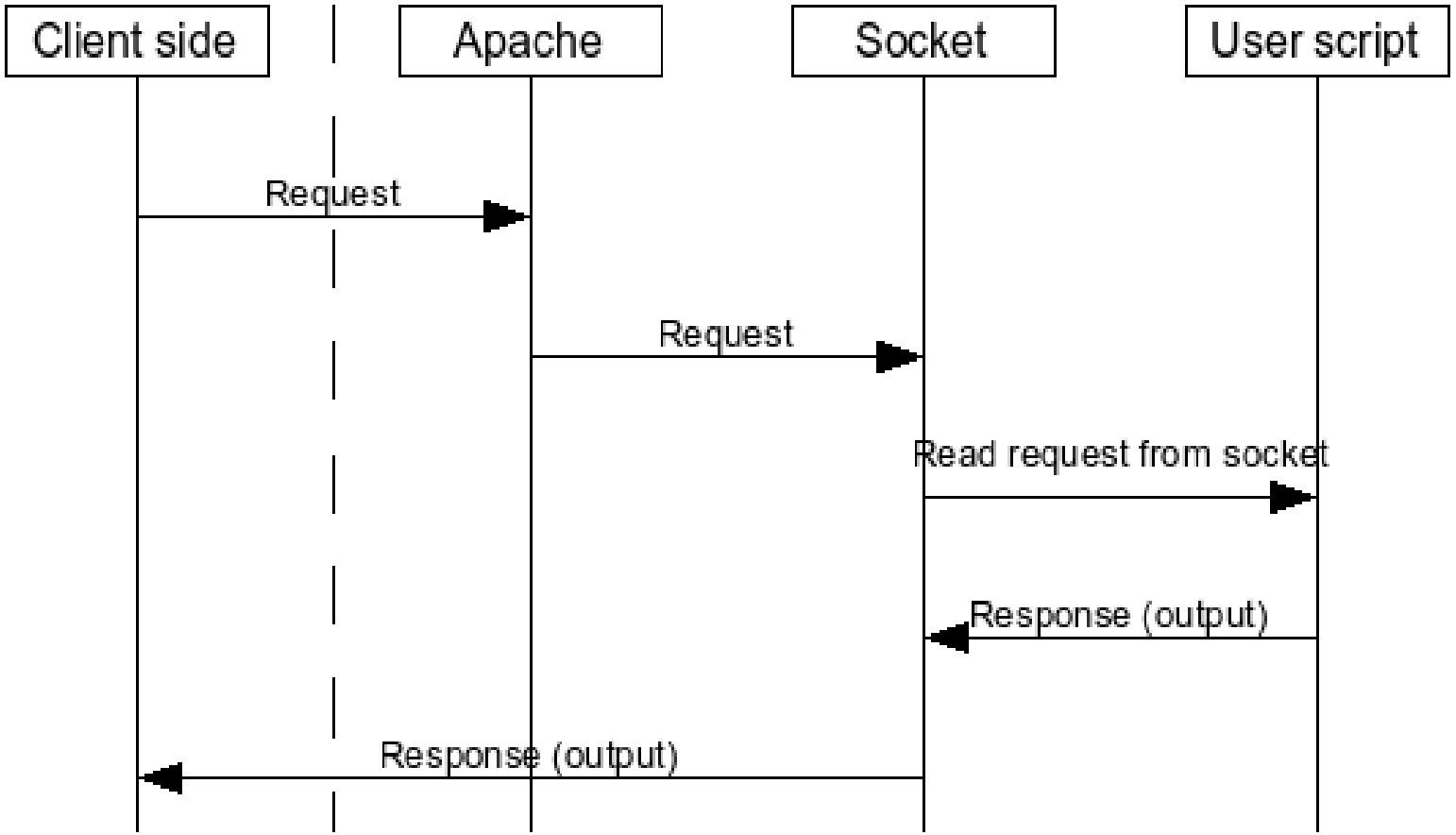
- Статический контент (html, css, jpeg, png, txt, etc.)
- Устаревшее: «страница как есть» (http-ответ)

Динамический веб-сервер. CGI



- Новый процесс на каждый запрос
- Передача данных через stdin, stdout, переменные окружения

Динамический веб-сервер. FastCGI



- Пулл процессов
- Передача данных через сокеты

- SCGI – похож на FastCGI, но проще
- PSGI, WSGI, Rack, JSGI – для Perl, Python, JS, Ruby

Динамический веб-сервер. SSI

Server Side Includes - первый язык шаблонизации на сервере с помощью макрокоманд

- Выполнение скриптов CGI
- Условные операторы
- Включения других файлов

Динамический веб-сервер. Собственные программные интерфейсы

Модульная архитектура веб-серверов

- Internet Server Api (ISAPI) для IIS
- Apache Server Api (модули)

- Индеец или заплатка?
- Апрель 1995 – первая официальная версия Apache 0.6.2, на основе NCSA httpd 1.3
- Июнь 1999 года – сформирована Apache Software Foundation

- Модульность (> 500)
- Высокая надежность
- Открытые исходные коды
- Свободная лицензия
- Виртуальные хосты

Архитектура Apache

Состав	Подробнее
Ядро	Конфигурация протокол http загрузка модулей
Система конфигурации	Конфигурация сервера (httpd.conf) Конфигурация виртуального хоста (httpd.conf) Конфигурация уровня директории (.htaccess)
Механизм виртуальных хостов	
Система модулей	

Модули Apache

Тип	Примеры
Поддержка языков программирования	mod_cgi, mod_fastcgi, mod_perl, mod_php, mod_python, mod_wsgi, apache-ruby, apache-asp
Добавление функций	mod_include
Исправление ошибок или модификация основных функций	mod_fcgid
Усиление безопасности	mod_ssl, mod_auth_basic, mod_security

Мультипроцессорные модули Apache

Имя	Описание	Назначение
worker	Гибридная мультипроцессорно-мультипоточная модель	Среднезагруженные веб-серверы
pre-fork	Предварительное создание отдельных процессов, без потоков	Безопасность и стабильность за счёт изоляции процессов друг от друга. Совместмость.
pre-child	Гибридная модель, с фиксированным количеством процессов.	Высоконагруженные серверы
netware	Мультипоточная модель	Серверы Novell NetWare
winnt	Мультипоточная модель	Серверы Windows Server

IIS 

- Многопоточный, по процессу на каждое приложение
- CGI, FastCGI, ISAPI, SSI
- ASP, ASP.NET через ISAPI и FastCGI

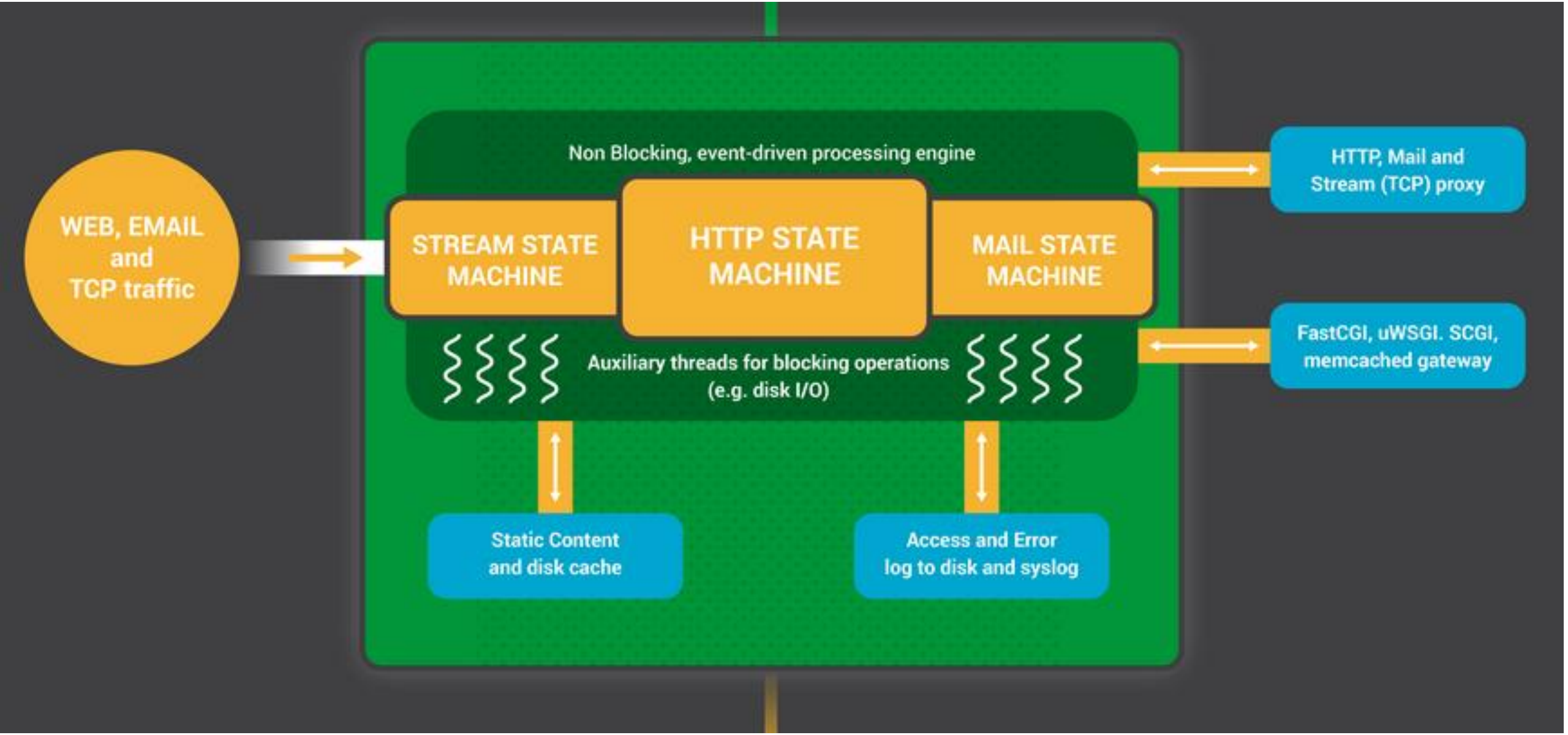
- 2002 год
- Игорь Сысоев, выпускник МГТУ им.Н.Э.Баумана

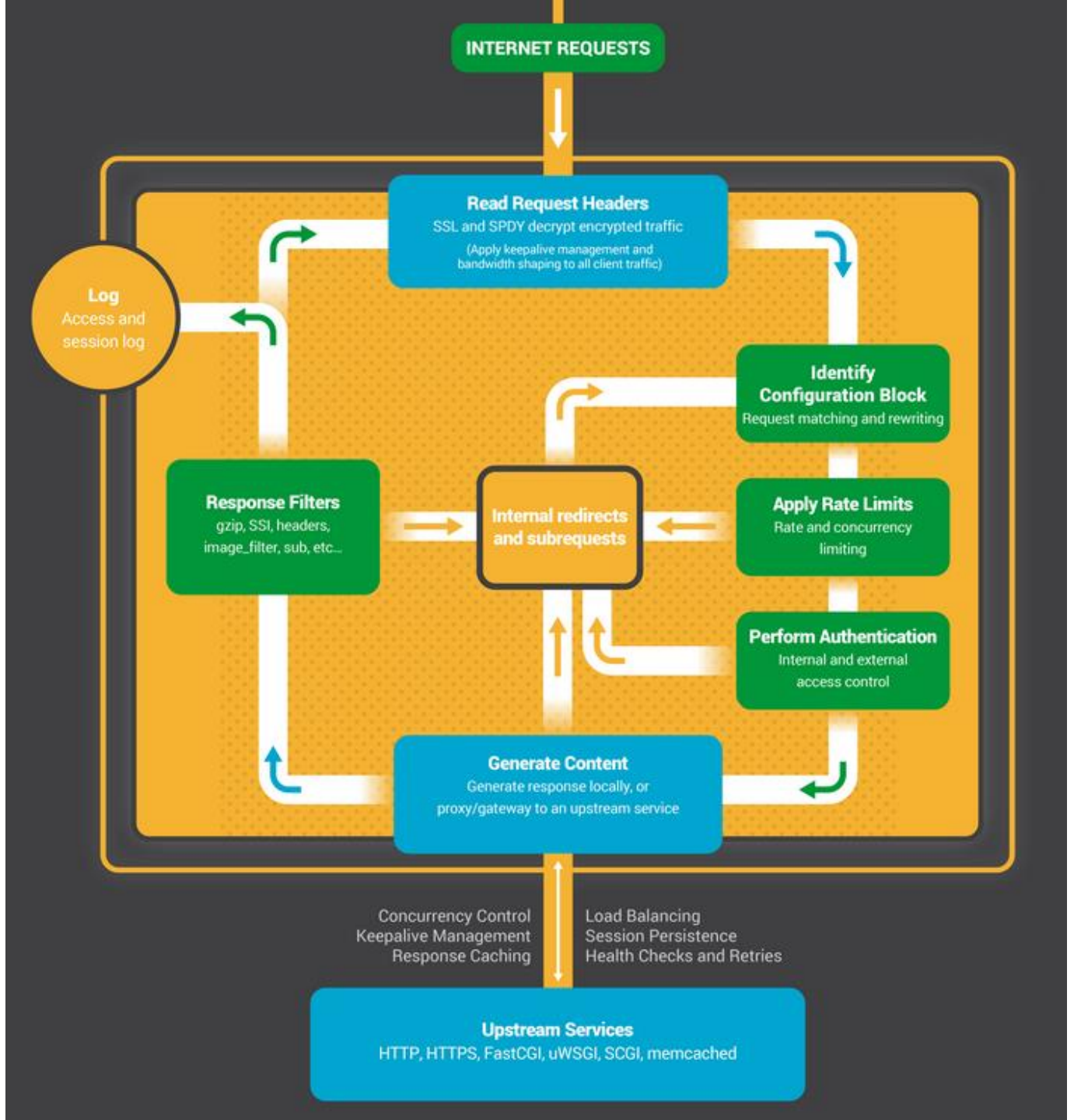
- Простой
- Быстрый
- Надежный
- Идеален для статики и как прокси для других веб-серверов

NGINX

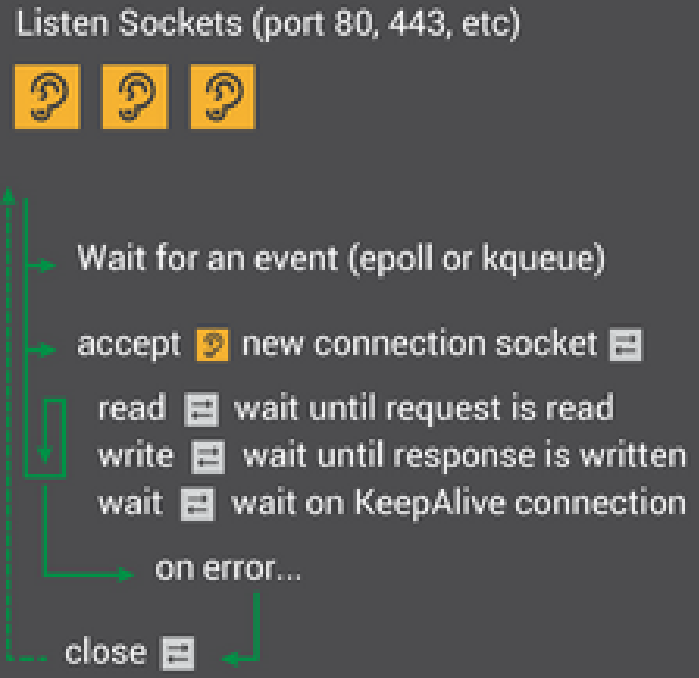
Состав	Подробнее
Ядро	базовый функционал web-сервера обратное проксирование web и электронной почты
Промежуточные модули http и mail	обработка последовательностей событий, связанная с протоколами прикладного уровня HTTP, SMTP или IMAP.
Функциональные модули	модули обработки событий, модули обработки фазы, выходные фильтры, модули обработки именованных переменных, модули реализации протоколов, модули взаимодействия с вышестоящими серверами и балансировщики нагрузки


MASTER PROCESS
Child Processes
Shared memory is used for cache, session persistence, rate limits, session log
CM
CL
W
W
W
W
Cache Manager
Cache Loader
Worker processes handle HTTP and other network traffic


WEB, EMAIL
and
TCP traffic
Non Blocking, event-driven processing engine
STREAM STATE MACHINE
HTTP STATE MACHINE
MAIL STATE MACHINE
Auxiliary threads for blocking operations
(e.g. disk I/O)
HTTP, Mail and
Stream (TCP) proxy
FastCGI, uWSGI. SCGI,
memcached gateway
Static Content
and disk cache
Access and Error
log to disk and syslog

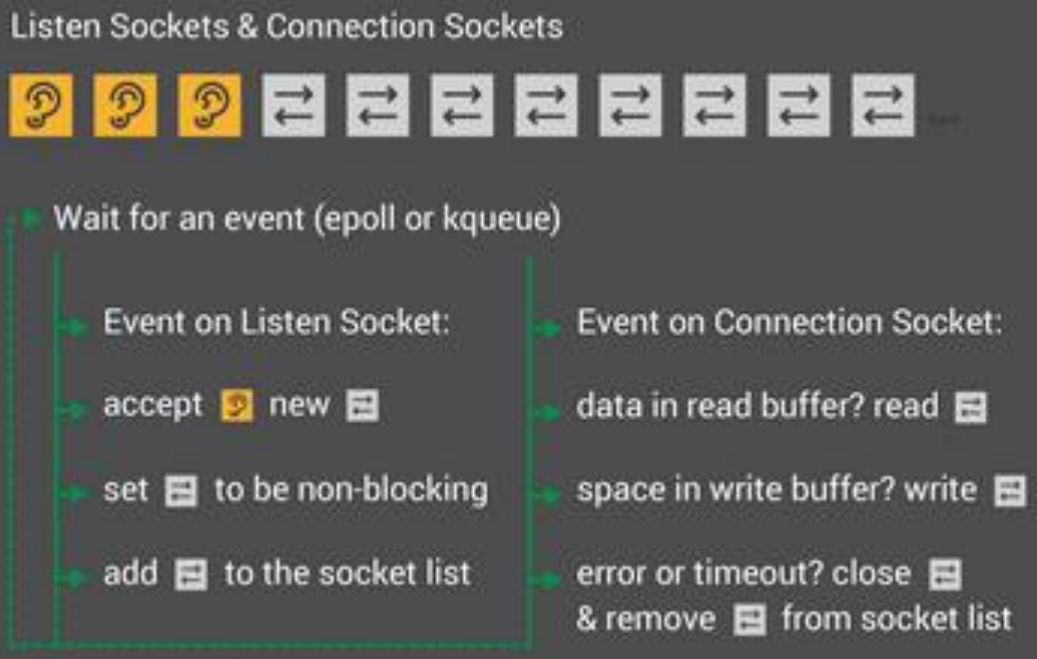

INTERNET REQUESTS
Read Request Headers
SSL and SPDY decrypt encrypted traffic
(Apply keepalive management and bandwidth shaping to all client traffic)
Log
Access and session log
Identify Configuration Block
Request matching and rewriting
Response Filters
gzip, SSI, headers, image_filter, sub, etc...
Internal redirects and subrequests
Apply Rate Limits
Rate and concurrency limiting
Perform Authentication
Internal and external access control
Generate Content
Generate response locally, or proxy/gateway to an upstream service
Concurrency Control
Keepalive Management
Response Caching
Load Balancing
Session Persistence
Health Checks and Retries
Upstream Services
HTTP, HTTPS, FastCGI, uWSGI, SCGI, memcached

Most web application platforms use blocking (waiting) I/O

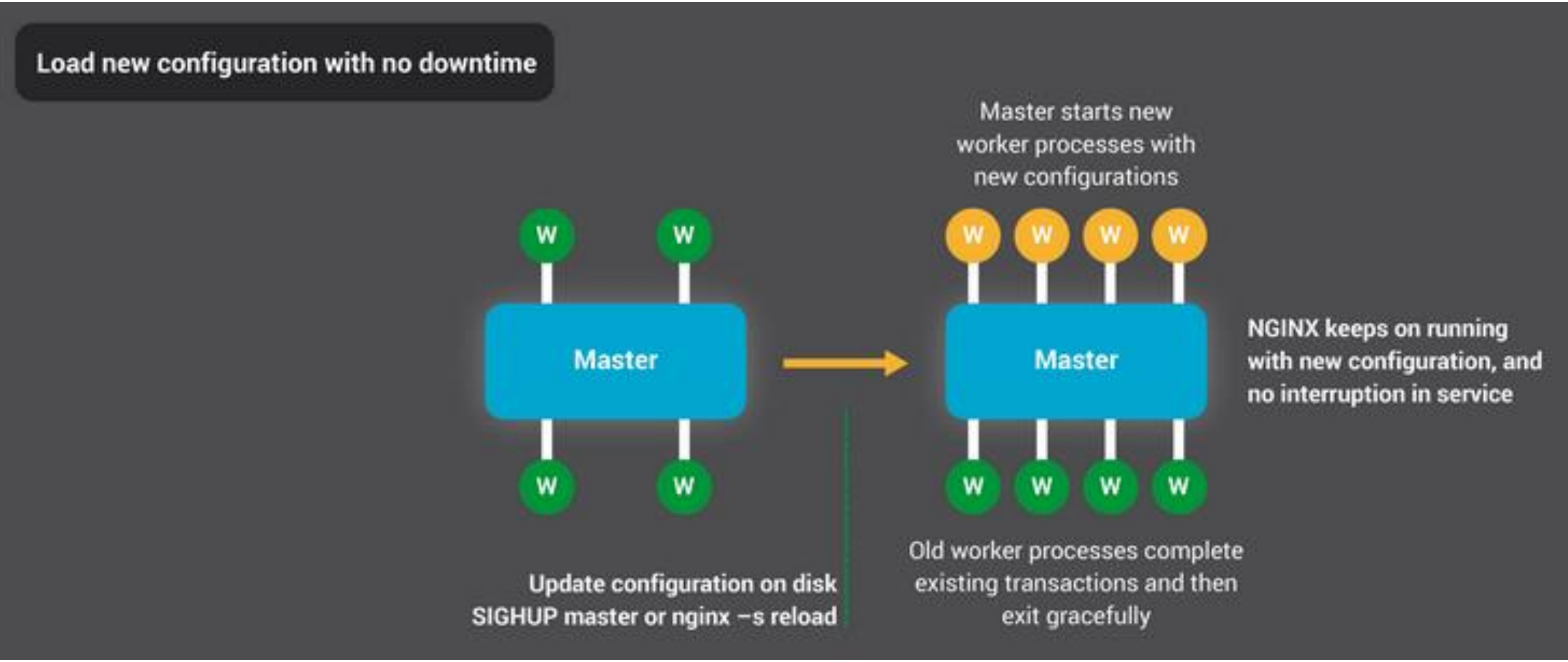


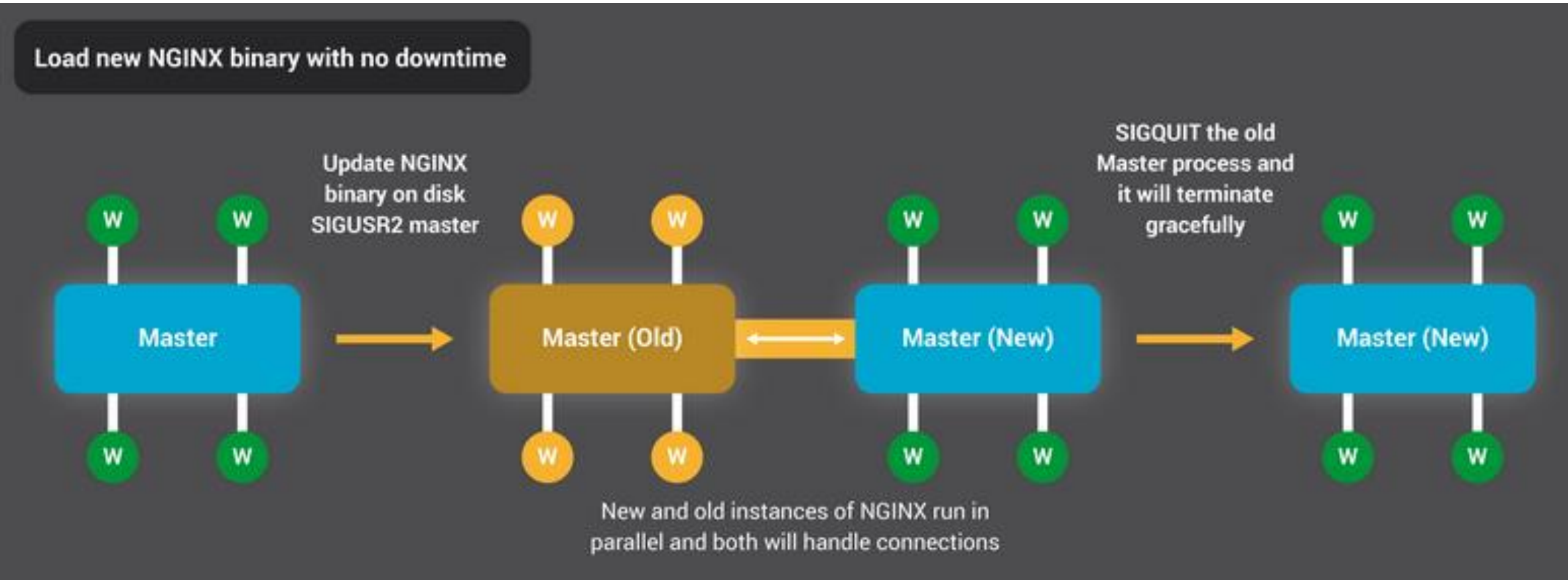
Each worker can only process one active connection at a time

NGINX uses a Non-Blocking "Event-Driven" architecture

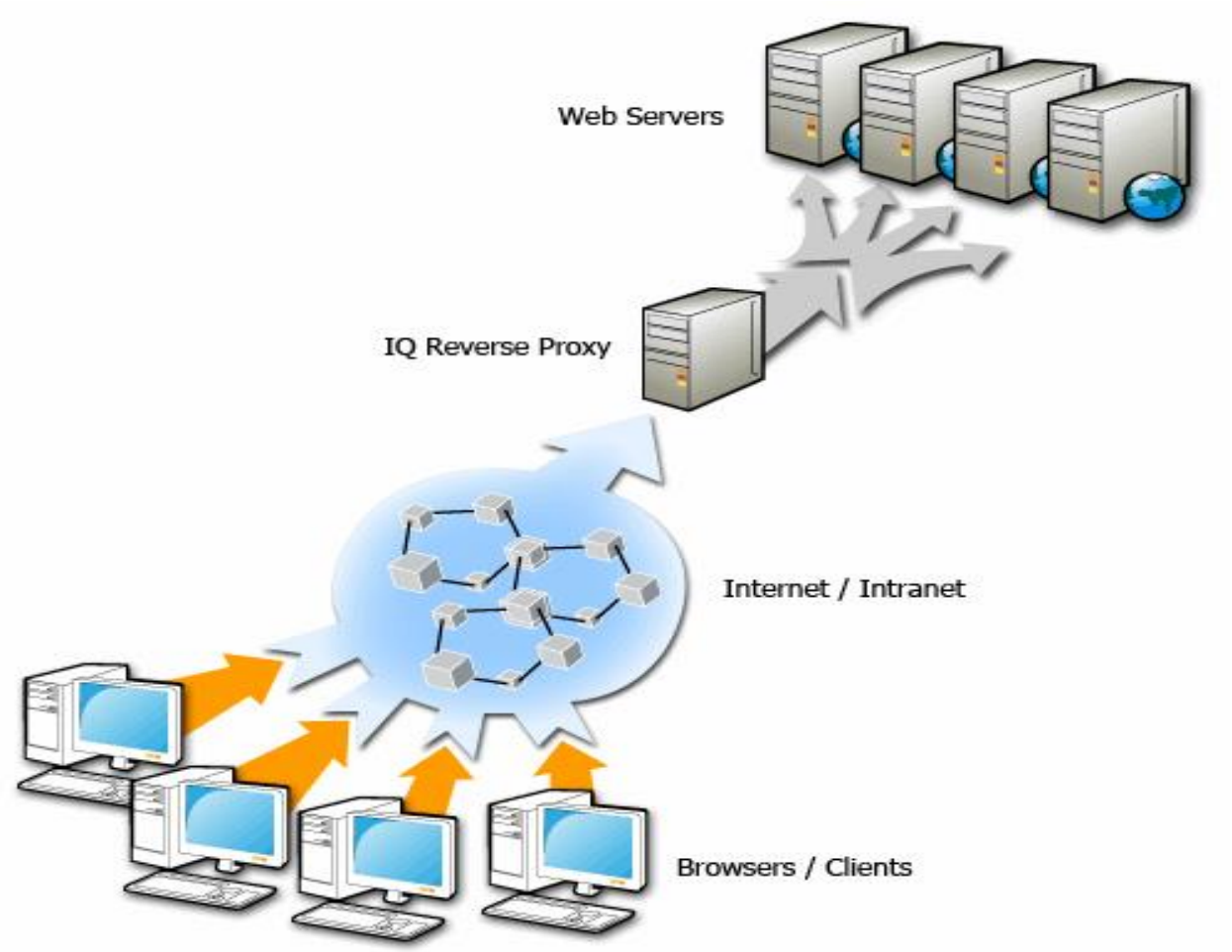


An NGINX worker can process hundreds of thousands of active connections at the same time


Load new configuration with no downtime
Master starts new worker processes with new configurations
W
W
W
W
Master
Master
W
W
W
W
W
W
W
W
NGINX keeps on running with new configuration, and no interruption in service
Update configuration on disk
SIGHUP master or nginx –s reload
Old worker processes complete existing transactions and then exit gracefully


Load new NGINX binary with no downtime
Update NGINX binary on disk SIGUSR2 master
SIGQUIT the old Master process and it will terminate gracefully
Master
Master (Old)
Master (New)
Master (New)
W
New and old instances of NGINX run in parallel and both will handle connections

Frontend-Backend

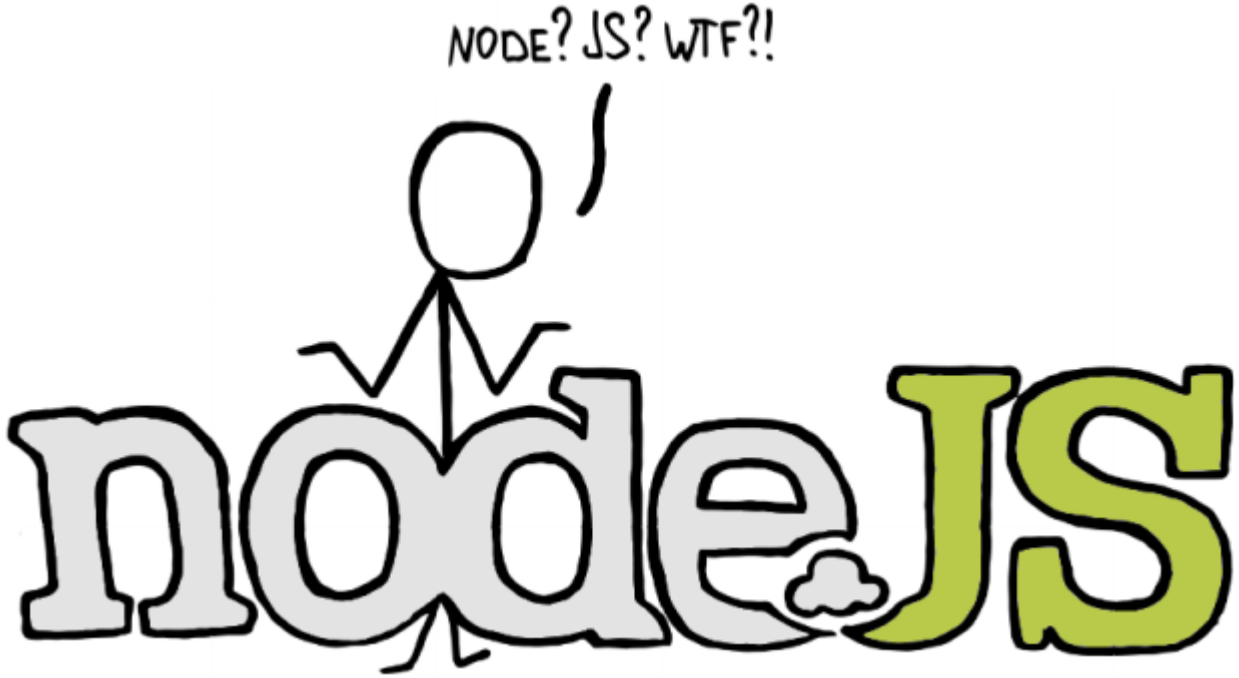


Frontend (NGINX)

- отдача статических файлов
- проксирование (reverse proxy)
- балансировка нагрузки
- кеширование
- сборка SSI
- авторизация, SSL, нарезка картинок, gzip

Backend (Apache)

- Запуск скриптов и обработка запросов
- CGI, FastCGI



Платформа Node.js была создана в 2009 году Райном Далом

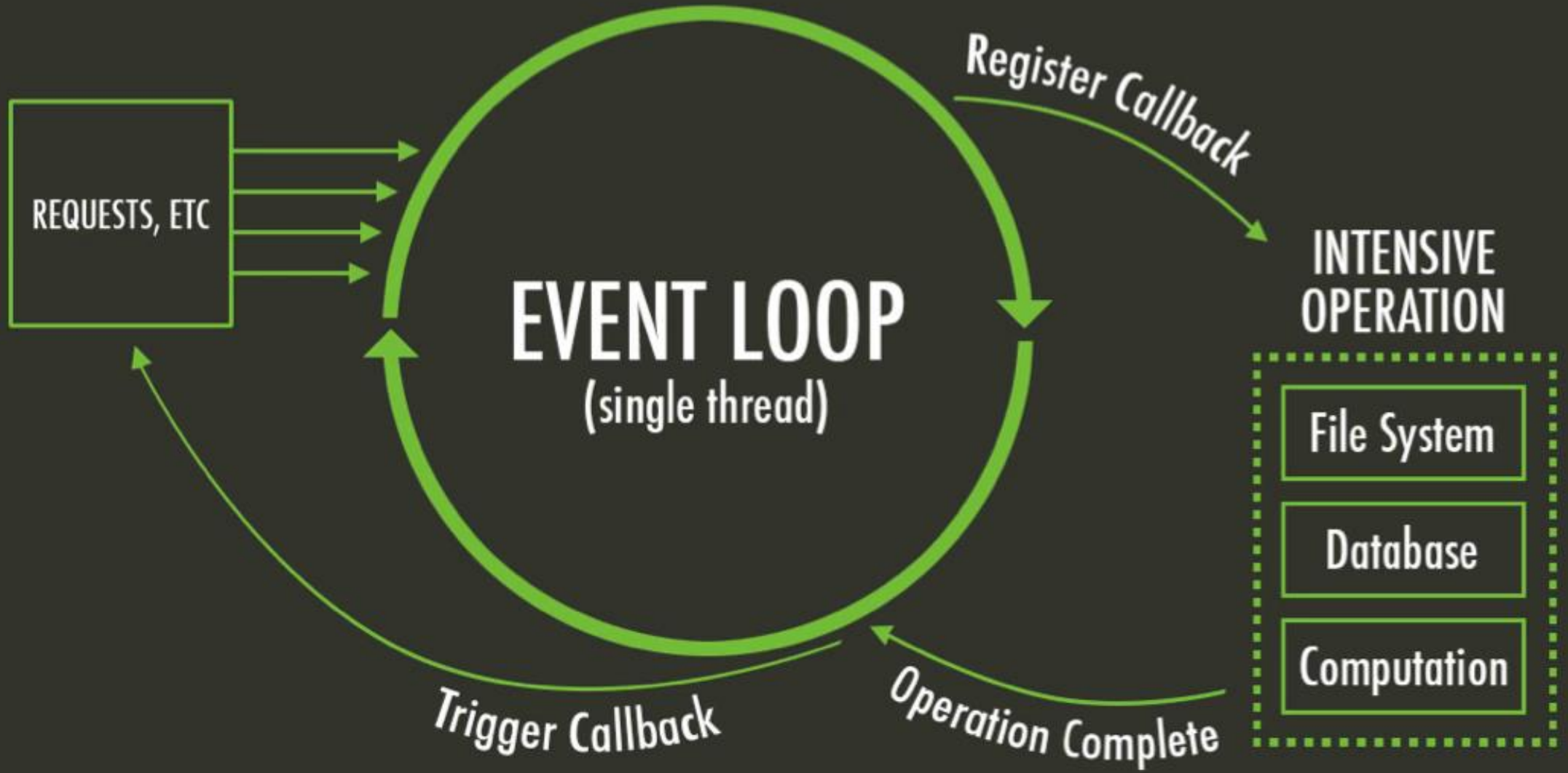
- Асинхронно-событийный веб-сервер на JS
- Сервер приложений на JS
- Виртуальная машина V8 от Chromium
- Модульная архитектура
- Возможность добавления модулей на си
- Работа с потоковым вводом-выводом, процессами, файлами
- Пакетный менеджер npm

Основы 

В основе лежит библиотека libev, реализующая цикл событий (event loop).

Libev – это написанная на C библиотека *событийно-ориентированной обработки данных, предназначенная для упрощения асинхронного неблокирующего ввода/вывода*

- выполняются функции, установленные на предыдущей итерации цикла с помощью особого метода – **process.nextTick()**, обрабатывающего события libev, в том числе таймеров;
- выполняется опрос libeio (библиотеки для создания пула потоков – thread pool) для завершения операций ввода/вывода и выполнения установленных для них кэллбеков.


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete