WEB PLATFORM.EXE

HOLYJS AUTUMN 2026

САМОЕ БОЛЬШОЕ ЛЕГАСИ, С КОТОРЫМ МЫ РАБОТАЕМ

Веб-платформа · Мария Кондаурова · BIOCAD



Status	still running	Uptime	36+ years
Users	billions	Rollback	unavailable

Start presentation

v2026

Обо мне

- ☑ FrontLead в Департаменте Вычислительной Биологии в BIOCAD
- ☑ Член программного комитета HolyJS
- ☑ Живу в Питере, крашу Warhammer и занимаюсь фехтованием



[0] LOADING MODULE

ПРОЛОГ

Продукт, который вырос

Или как мировой MVP превратился в проект, который нельзя переписать

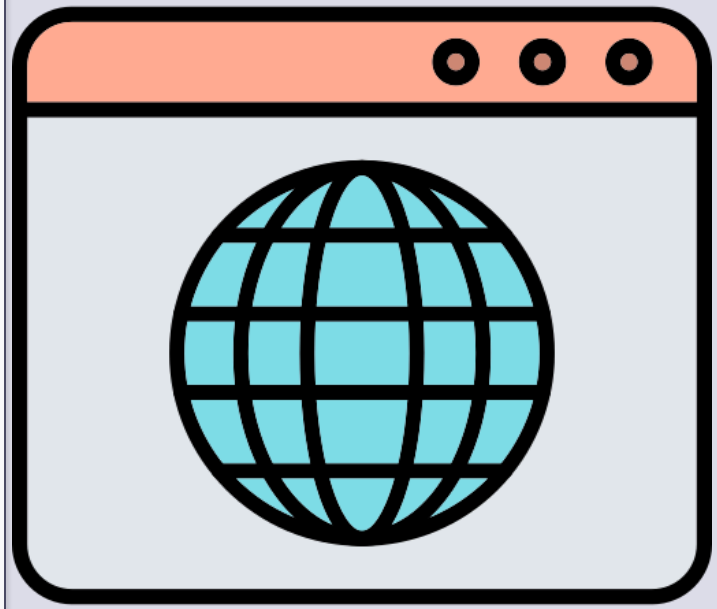
Как растёт любой продукт

- ☑ Сначала — маленький **scope**: простая полезная штука
- ☑ Потом приходят **пользователи**
- ☑ За ними — **новые требования** и **бизнес**
- ☑ Потом — **новые команды**, костыли, обратная совместимость
- ☑ И вот вы работаете с огромным **legacy**, которое нельзя просто переписать

Веб прошёл этот же путь — только в масштабе всего мира

Веб-платформа — самый большой живой **legacy-продукт** человечества

WEB.PNG — ПРОСМОТР



Из чего состоит продукт

Веб-платформа = Браузер + API + стандарты + тесты + комитеты

[1] LOADING MODULE

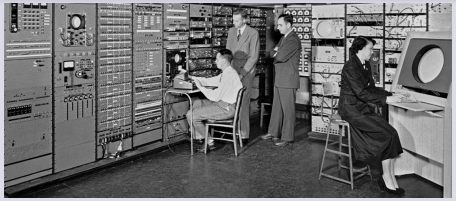
ГЛАВА 1

MVP, который оказался слишком успешным

Или как веб начинался с простого гипертекста

На секунду — про масштаб

ENIAC.PNG — ПРОС...



ENIAC · 1946

IPHONE17.PNG — П...



iPhone 17 Pro · 2025

Параметр	ENIAC (1946)	iPhone 17 Pro (2025)
Операций в секунду	≈5 000 сложений/сек	≥6 000 000 000 000 операций/сек
Память	**20 слов**	**6–8 ГБ** ОЗУ
Потребление	≈174 кВт	~10 Вт

Но железо — это половина истории...

Тим Бернерс-Ли

Создал первый браузер в 1990 году

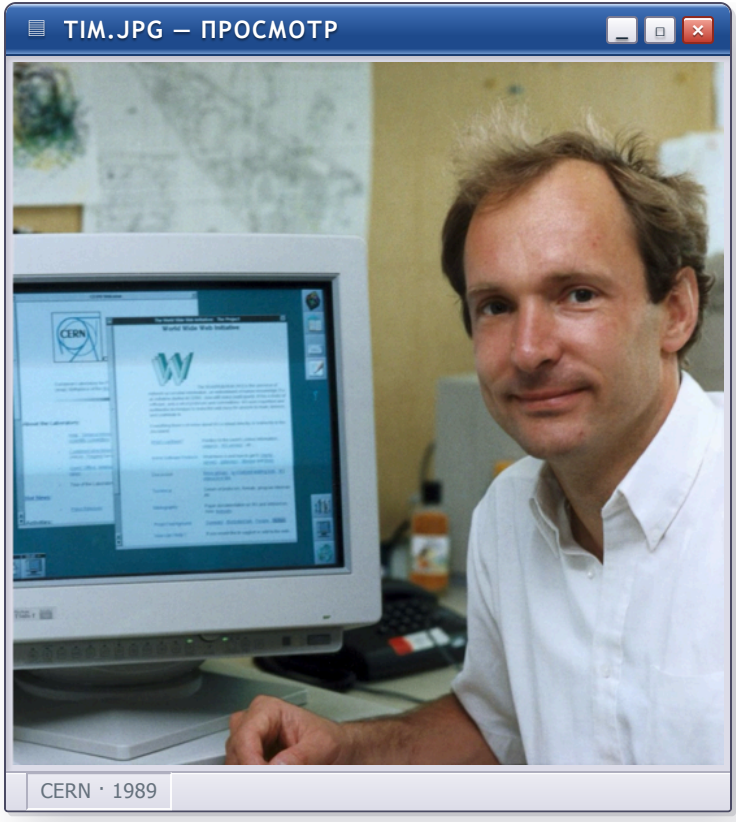


FIG. 1-1 TIM BERNERS-LEE — CREATOR OF THE WORLD WIDE W…

THE WORLD WIDE WEB PROJECT

http://info.cern.ch/hypertext/WWW/TheProject.html

1991

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Done

Это и был MVP: гипертекстовая система для обмена знаниями

Удачный минимальный scope

Эра «Невинного» веба

- ☑ Статичные HTML-страницы
- ☑ Документы, ссылки, немного картинок
- ☑ Табличная вёрстка

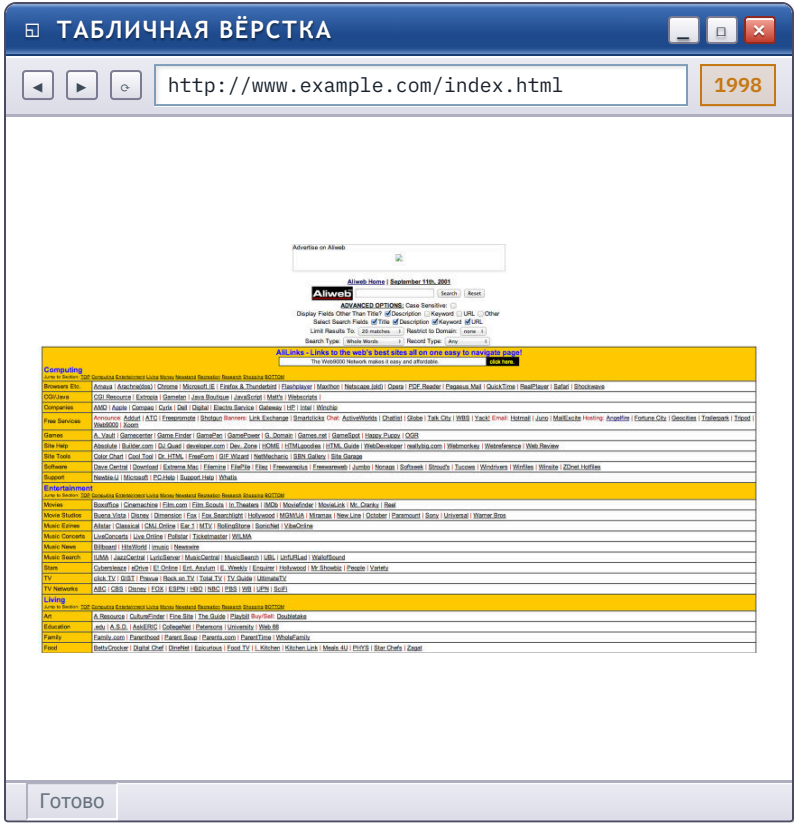


FIG. 1-2 EARLY WEB — TABLE-BASED LAYOUTS

APPLE-1981.PNG — ПРОСМОТР



Реклама домашнего компьютера · 1981

Люди стали генерировать контент и самовыражаться

АТМОСФЕРА РАННЕГО ПОЛЬЗОВАТЕЛЬСКОГО ВЕБА

EARLY-WEB.MOV — ПРОИГРЫВАТЕЛЬ

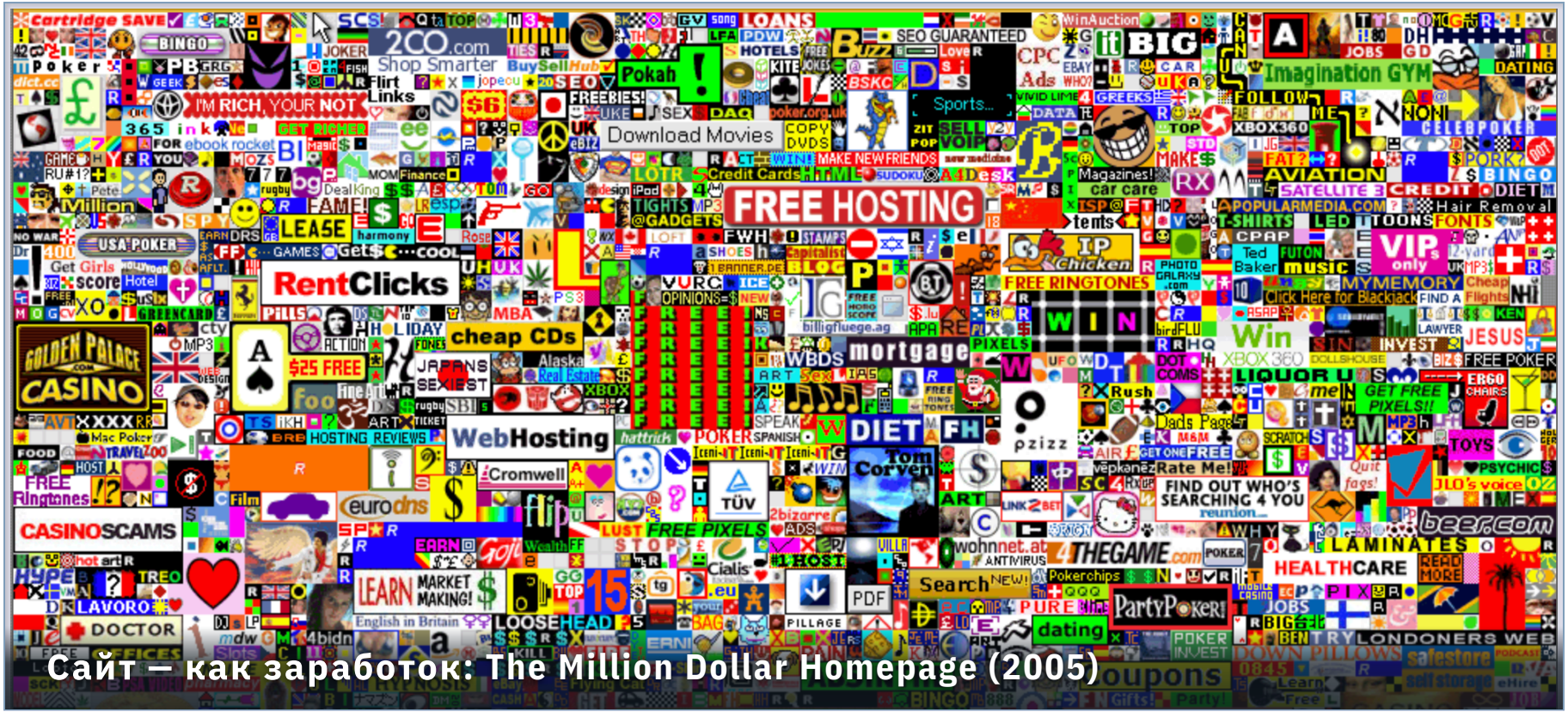


0:00 / 0:11

Geocities · гостевые книги · счётчики посещений



Сайт — как пиар-кампания фильма: Space Jam (1996)


FREE HOSTING
RentClicks
cheap CDs
mortgage
WebHosting
CASINOSCAMS
Download Movies
Sports...
Imagination GYM
FREE RINGTONES
FREE POKER
FREE PIXELS
HEALTHCARE
PartyPoker
Search
dating
Pokah
2CO.com
Shop Smarter
I'M RICH, YOUR NOT
FREEBIES!
SEO GUARANTEED
Magazines!
car care
AVIATION
IP Chicken
GOLDEN PALACE CASINO
USA POKER
$25 FREE
FIND OUT WHO'S SEARCHING 4 YOU
THEGAME.com
Tom Corven
DOCTOR
LAVORO
LOOSEHEAD
Cialis
eurodns
Cromwell
TÜV
beer.com
LAMINATES
Rate Me!
Pokerchips
PURE
Sexy Singles
Dads Page
Quit fags!
JESUS
Hair Removal
Click Here for Blackjack
FREE Ringtones
Get FREE PIXELS!!
FIND A FLIGHT
LIQUOR
DIET
Win
WIN
Rush
Ted Baker
music
MYMEMORY
VIP only
CREDIT DIET
CELEBPOKER
BINGO
DATING
LONDONERS WEB
Сайт — как заработок: The Million Dollar Homepage (2005)

Интернет взрослеет

Самовыражение → сервис

Веб перестаёт быть просто страницами и становится средой для жизни

Браузер уже не тянет «старым» способом

[2] LOADING MODULE

ГЛАВА 2

Требования росли — веб отвечал

Или как пользователи хотели X, а веб отвечал Y

«Фронтенд развивается скачкообразно»

Home

History Runtimes GitHub

It's been

80

day(s) since the latest JavaScript framework.

Last JS framework: vinext released on February 24, 2026

You think the data is wrong? File an issue here!

Каждый скачок — это накопившееся давление требований

Захотели взаимодействовать, а не читать (2004)

! NEW REQUIREMENT DETECTED	2004
USER DEMAND	**Взаимодействовать, а не читать**
PLATFORM RESPONSE	**AJAX / XMLHttpRequest**
SIDE EFFECT	Состояние — вручную, спагетти-код

ДО 2004

- ☑ Каждое действие — новая страница
- ☑ Обновить статус — рефреш

GMAIL + AJAX

- ☑ Частичное обновление страницы
- ☑ Мгновенные ответы

AJAX дал скорость, но создал свои проблемы

- ☑ Управление состоянием вручную
- ☑ «Спагетти-код» повсюду **(привет, jQuery!)**
- ☑ Каждый разработчик делает свой велосипед

Нужна реиспользуемость

Захотели богатый UI и скорость разработки (2010)

! NEW REQUIREMENT DETECTED	2010
USER DEMAND	**Богатый UI и скорость разработки**
PLATFORM RESPONSE	**SPA: Angular, React, Vue**
SIDE EFFECT	**Тяжёлый клиент, SEO ломается**

БЫЛО (MPA)

- ☑ Каждый экран — отдельный HTML
- ☑ Сервер рендерит всю страницу
- ☑ Ограниченная интерактивность
- ☑ Много кода

СТАЛО (SPA)

- ☑ Angular/React/Vue
- ☑ Клиент — UI-машина
- ☑ Сервер — только API

Но телефоны не стояли на месте

КНОПОЧНЫЕ (2000–2007)









СЕНСОРНЫЕ (2007+)









С мобильностью пришли новые вызовы

- ☑ Тяжёлый JS тормозит на слабых устройствах
- ☑ Touch UI вместо hover/click
- ☑ 3G/4G вместо оптоволокна
- ☑ Экраны от 320px до 4K

Мир стал мобильным (2012)

! NEW REQUIREMENT DETECTED — 2012

USER DEMAND	**Мир стал мобильным**
PLATFORM RESPONSE	**Mobile-first, адаптивность 320px–4K**
SIDE EFFECT	**Хочется нативности**

ДЕСКТОП-FIRST (2010)

- ☑ Фиксированная ширина 1024px
- ☑ Hover и курсор мыши
- ☑ Быстрый интернет (DSL)
- ☑ Мощные ПК

MOBILE-FIRST (2012+)

- ☑ Адаптивность 320px–4K
- ☑ Touch-интерфейсы (Material UI)
- ☑ Производительность (lazy load)
- ☑ Сети 3G/4G

Но захотели нативности

- ☑ **Натив** умеет офлайн, пуши и иконку на экране
- ☑ ...но релиз неделями ждёт модерации, а обновляется — когда захочет пользователь
- ☑ **Веб** обновляется мгновенно и открывается по ссылке
- ☑ ...но без сети он пустой

Захотелось возможностей приложений — но без стора

Захотели возможности приложений (2015)

! NEW REQUIREMENT DETECTED	2015
USER DEMAND	То же, что умеют нативные приложения
PLATFORM RESPONSE	PWA: Service Workers, offline, push
SIDE EFFECT	Рендерные боли — опять

ОБЫЧНЫЙ ВЕБ

- ☑ Только онлайн
- ☑ Не устанавливается
- ☑ Нет push

PWA

- ☑ Offline-first
- ☑ Установка без стора
- ☑ Push-уведомления

Service Worker = прокси между сетью и кэшем

...но вылезли рендерные боли (опять)

- ☑ Тяжёлый JS на клиенте
- ☑ SEO страдает (SPA)
- ☑ TTFB (Time to First Byte) медленный
- ☑ Размытая ответственность между сервером и клиентом

Нужны интерактив + скорость + SEO (2020)

! NEW REQUIREMENT DETECTED	2020
USER DEMAND	**Интерактив + скорость + SEO сразу**
PLATFORM RESPONSE	**React Server Components**
SIDE EFFECT	**Границы сервер/клиент размыты**

КЛАССИКА

- ☑ Всё на клиенте (SPA)
- ☑ Или всё на сервере (MPA)
- ☑ Два кода
- ☑ SEO или скорость

RSC (REACT SERVER COMPONENTS)

- ☑ Серверный рендер статичного
- ☑ Клиентский только интерактив
- ☑ Один код (async/await)
- ☑ SEO + скорость + PWA

Сервер рендерит → Streaming → Клиент «оживляет»

1990
Статичный веб

2004
Ajax
Gmail заложил тренд веба как сервиса

2010
SPA-бум
AngularJS (2010), React (2013), Vue (2014)

2010
First-mobile
Развитие смартфонов

2015+
PWA
Офлайн, установка, пуш-уведомления

2020–2023
Server Components
Гибрид сервер/клиент

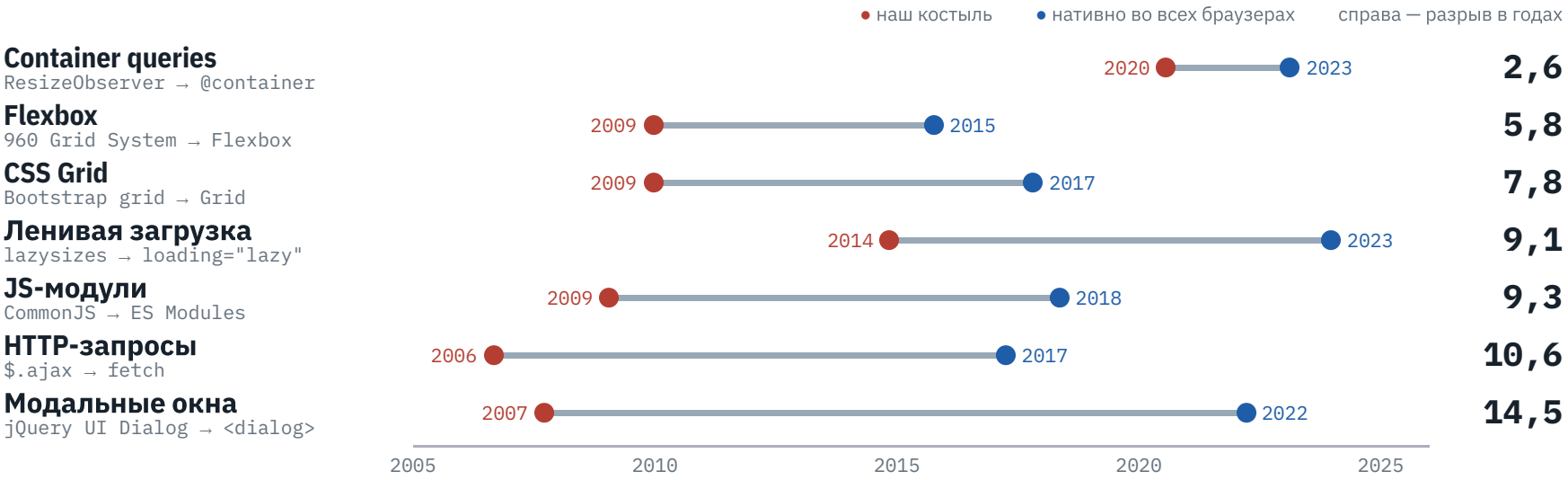
Всего за 20 лет

Темп колоссальный

Но у каждого из этих скачков был один и тот же автор

И это были не браузеры

Платформа догоняет. Всегда.



Медиана по 11 измеренным кейсам — **9,1 года**

Даты костылей — первая публикация в npm или репозитории, то есть нижняя граница; нативная возможность — дата Baseline newly available (работает во всех основных браузерах). A moment.js → Temporal в этот график не попал: разрыв ещё **открыт**.

Разработчики всегда впереди платформы

Мы выжимаем максимум из того, что есть сейчас — и делаем хак

Платформа **легитимизирует** этот хак. Через десять лет.

Сначала — откуда мы шли

ПАВЕЛ ДУРОВ | ВКОНТАКТЕ

◄ ► ⟳ https://vk.com/id1 2006



Готово

УЛИЧНЫЕ ГОНКИ — ГАРАЖ

http://gonki.ru/garage.php

2007

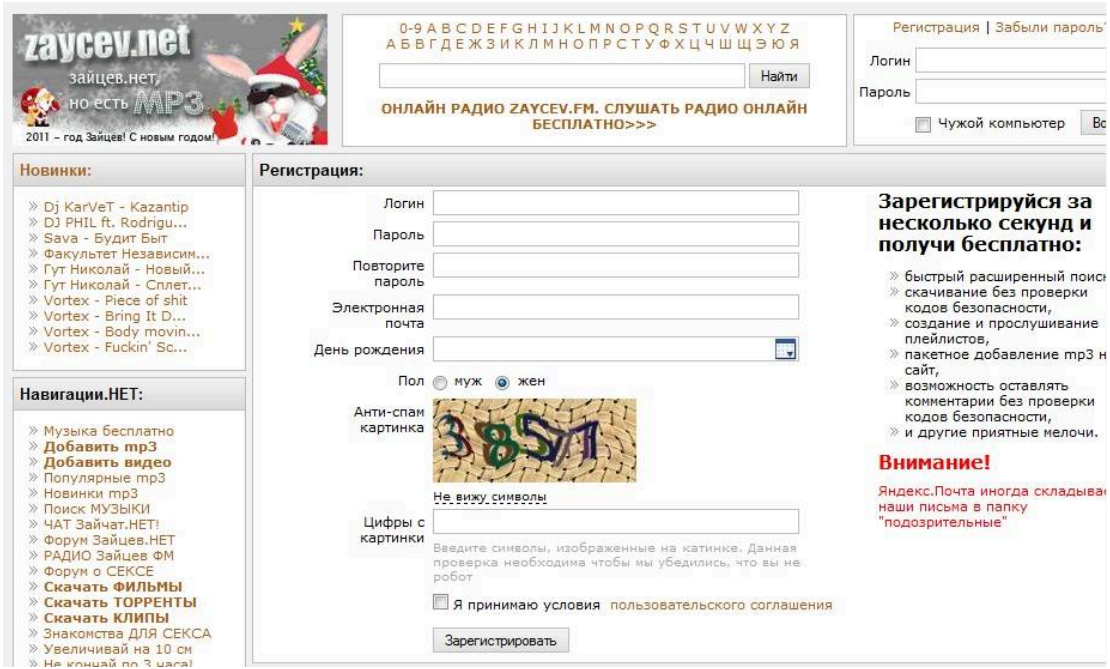


Готово

ZAYCEV.NET — МУЗЫКА MP3

http://zaycev.net/

2011



Готово

МУЗЕЙ САМОКАТА

https://museum.samokat.ru

2026



3D музей в браузере · React и Three.js

3D-музей в браузере (React и Three.js) https://museum.samokat.ru

MESSENGER — ABETO

https://messenger.abeto.co

2026



Многопользовательская игра · WebGL

Многопользовательская игра в браузере (WebGL) https://messenger.abeto.co

CSS DOOM

https://cssdoom.wtf/

2026

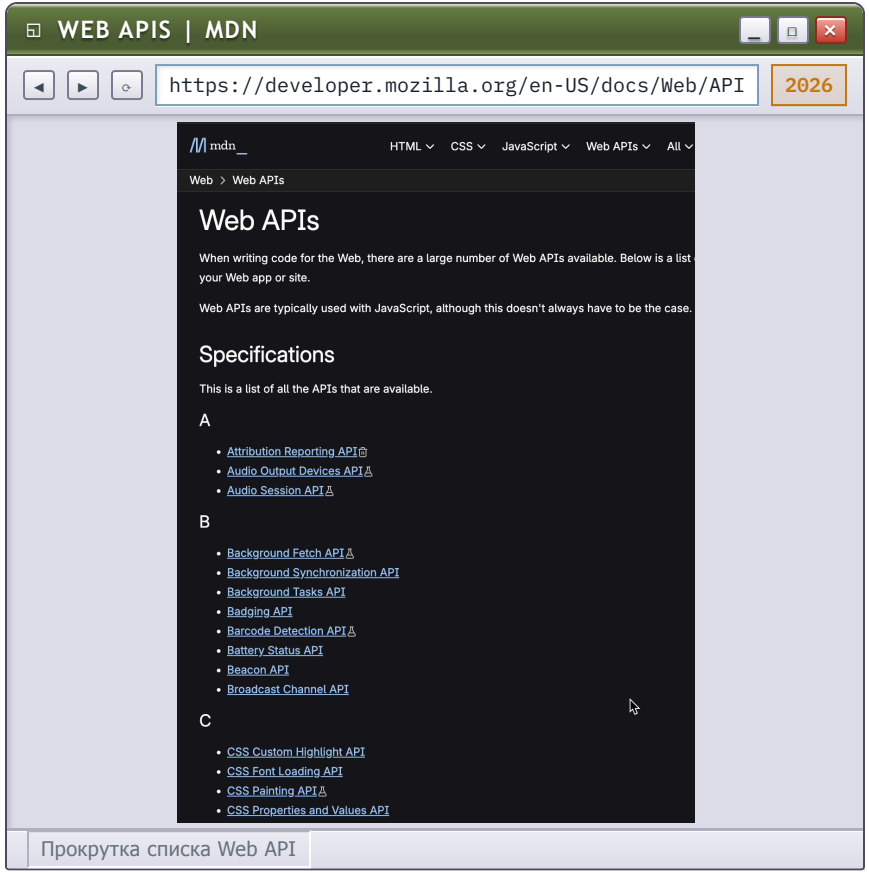


DOOM, отрисованный CSS и JS

DOOM на CSS и JS https://cssdoom.wtf/

150+ браузерных Web API с доступом к

- ☑ телефону (контакты, вибрация, bluetooth, уведомления)
- ☑ железу (GPU)
- ☑ AI (конец 2025-го)
- ☑ ...и много чему ещё



Прокрутка списка Web API

Это уже полноценная платформа

[3] LOADING MODULE

ГЛАВА 3

Веб переписал не только фронтенд

Или та же история глазами дизайна и бэкенда

Когда продукт растёт, меняется не только его код

Меняются интерфейсы, инфраструктура, процессы —
и даже то, **кто считается пользователем**

Сейчас мы пройдём этот же путь ещё дважды

Те же требования, те же годы — но не глазами фронтендера

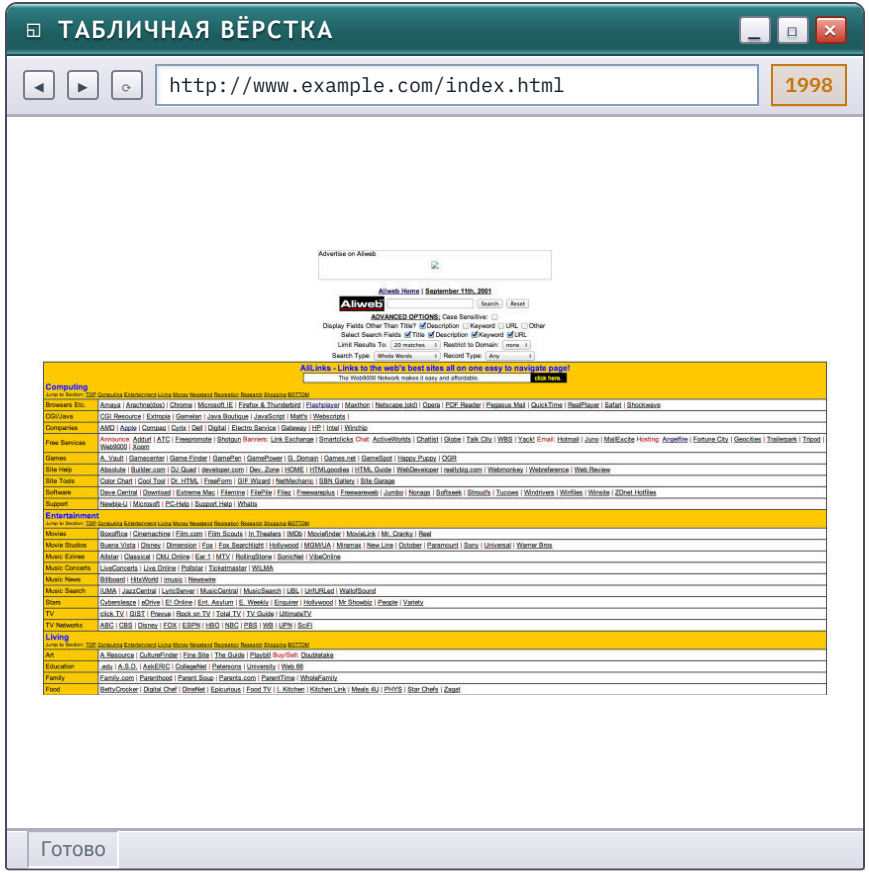
Арка 1 — **дизайн**. Арка 2 — **бэкенд**.

Арка 1 — Дизайн

От картинки к системе правил

Шаг 1. Дизайн = документ

- ☑ Макет — **картинка фиксированной ширины**: 640 → 800 → 1024
- ☑ Типографика, таблицы, `<font>`, `<center>`
- ☑ «Отдали в вёрстку» → **pixel perfect**
- ☑ Дизайнер точно знает, как это будет выглядеть у пользователя



Табличная вёрстка не умерла

Она переехала в почту (и живее всех живых!)

Outlook с 2007-го рендерит письма движком **Microsoft Word** — поэтому email-вёрстка до сих пор на вложенных `<table>` и inline-стилях

DEPRECATED BUT SUPPORTED слой, который нельзя выкинуть

Шаг 2. Появилось поведение — и дизайн стал рисовать состояния

БЫЛО

- ☑ Один макет = один экран
- ☑ Страница либо есть, либо её нет

СТАЛО (ПОСЛЕ AJAX)

- ☑ loading / empty / error / partial
- ☑ Переходы и анимации
- ☑ Оптимистичный UI

Дизайнер впервые описывает не картинку, а **поведение во времени**

И тут же родилась боль, которая не решена до сих пор: **«макет счастливого пути»**

И параллельно дизайн симулировал то, чего в платформе не было

- ☑ Нативный `<select>` **нельзя было застилить** — и индустрия начала писать свои селекты
- ☑ Нет модального окна — свой оверлей, focus trap, блокировка скролла
- ☑ Нет тултипа — своя обёртка с позиционированием
- ☑ Нет кнопки нужной формы — `<div role="button">`
- ☑ А ещё дизайн симулировал **нативные интерфейсы ОС**: объёмные кнопки, кожа, полки, скевоморфизм

Дизайнер рисовал не то, что умеет веб. Он рисовал то, что умеет всё остальное.

Платформа догнала

`<dialog>` — модалку наконец не надо писать руками. Кроссбраузерно с **марта 2022**.

`popover` — с **января 2025**. Широко доступным станет только в **июле 2027**.

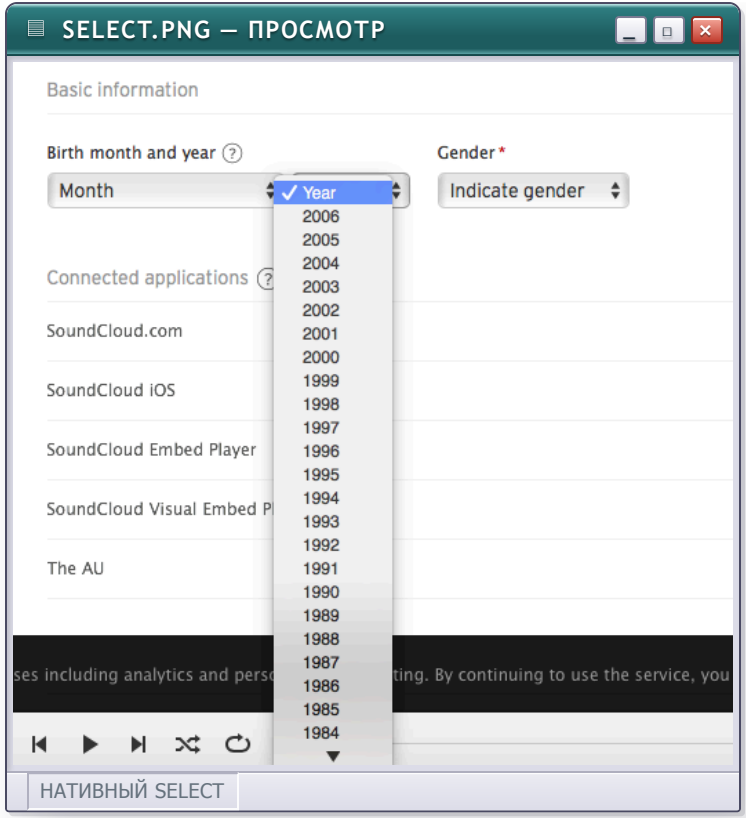


FIG. 3-1 НАТИВНЫЙ SELECT — НЕ СТИЛИЗУЕТСЯ

Двадцать лет индустрия писала свои селекты

Платформа не давала стилизовать нативный — и мы прятали его и рисовали свой заново

Стилизовать его стало можно в **апреле 2025** — `appearance: base-select`, Chrome 135. И до сих пор только в Chromium.

И до сих пор пишем

Шаг 3. Макет перестал быть картинкой

- ☑ **2010**, Ethan Marcotte, A List Apart — «Responsive Web Design»
- ☑ Число экранов больше **не ограничено**
- ☑ Комбинаторный взрыв: состояния × экраны × плотности × ориентации
- ☑ Нарисовать все макеты стало **физически невозможно**

5 состояний × 4 брейкпоинта × 2 темы

= 40 макетов на один экран

Раньше дизайн мог нарисовать результат — теперь нет.

Шаг 4. Ответ индустрии: тулинг дизайнера поменялся

- ☑ **2011** — Bootstrap: готовая сетка и компоненты
- ☑ **2013** — Atomic Design (Brad Frost): интерфейс как набор атомов
- ☑ **2014** — Material Design: дизайн как **спецификация с правилами**
- ☑ **2016** — Figma: макет как общий редактируемый исходник
- ☑ **Design tokens** — дизайн-решения становятся данными

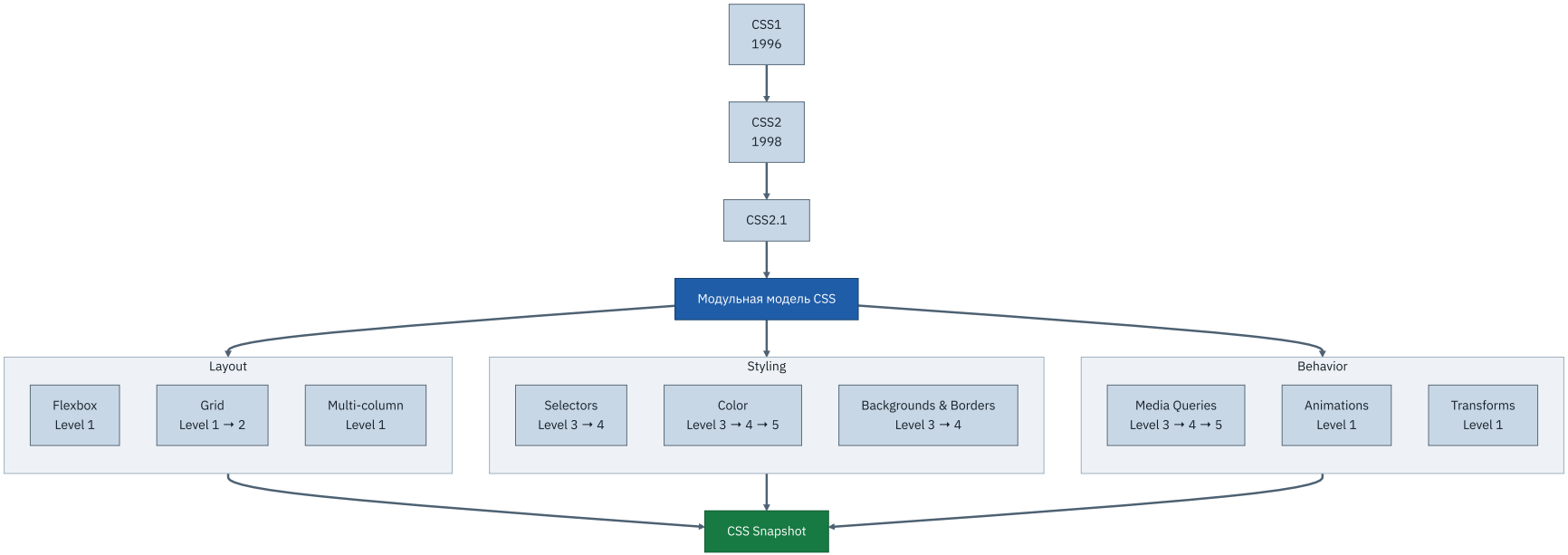
Дизайн стал кодом. И у него появились **версии, релизы и обратная совместимость**

Дизайн-системы выросли не от лени. Они выросли из арифметики.

И даже дизайн-токены уехали в комитет

W3C **Design Tokens Community Group** — чтобы токены работали одинаково в Figma, в вебе и в мобилке

Тот же процесс — но в CSS



Layout / Styling / Behavior — это буквально те три вещи, которые дизайнер перестал держать в одной картинке

Шаг 5. Дизайн контролирует не всё

- ☑ `prefers-color-scheme` — тёмная тема — это **не макет**, а согласие с настройкой ОС
- ☑ `prefers-reduced-motion` — анимацию может отменить пользователь
- ☑ `prefers-contrast`, `forced-colors` — цвета может переопределить система
- ☑ Accessibility: экранный ридер строит **свой** интерфейс из вашей разметки

Дизайн описывает **намерение**. Решение принимают пользователь и устройство.

Дизайнер перестал рисовать результат — он описывает правила

И это было не решение дизайнеров. Это давление веба.

Арка 2 — Бэкенд

От файлового сервера до edge

Шаг 1. Сервер отдаёт файл → сервер собирает страницу

- ☑ **1991** — httpd отдаёт файл с диска. И всё.
- ☑ **1993** — CGI: первая динамика, процесс на каждый запрос
- ☑ **1995+** — PHP, Perl, JSP: шаблоны на сервере, состояние в сессии
- ☑ HTML — это **результат работы бэкенда**

В этой точке бэкенд владеет интерфейсом целиком

Шаг 2. Сервер начал отдавать данные, а не страницы

- ☑ **1999** — XMLHTTP. Написан Microsoft **для веб-версии Outlook**
- ☑ **2000** — REST, диссертация Роя Филдинга
- ☑ **2005** — появляется само слово «AJAX»

Сервер больше не собирает страницу — он отдаёт данные

Корпоративная почта случайно родила весь современный фронтенд

А заодно **CORS** — защиту, которая нужна только потому, что фронтенд уехал на другой origin

Шаг 3. Один бэкенд — много клиентов

- ☑ Веб + iOS + Android + партнёрское API — из одного бэкенда
- ☑ **2015** — Facebook открывает GraphQL, придуманный под мобильную ленту
- ☑ **BFF**: у каждого клиента свой backend-for-frontend
- ☑ Версионирование API, потому что старое приложение у пользователя **не обновится**

Шаг 4. Мобильность — change request для бэкенда тоже

- ☑ Вес ответа стал **продуктовой метрикой**
- ☑ `srcset` / `<picture>`: картинки по размерам, WebP, AVIF
- ☑ CDN превращается из оптимизации в обязательный слой
- ☑ **HTTP/2** (2015) и **HTTP/3** — работа одного из комитетов (IETF)
- ☑ Brotli вместо gzip
- ☑ Погоня за метриками

Шаг 5. Сервер вернулся — но не в датацентр

- ☑ SSR и стриминг: рендер **снова на сервере**
- ☑ Edge и serverless: сервер в сотнях точек рядом с пользователем
- ☑ Гибридный рендеринг: часть страницы с сервера, часть с клиента

Круг замкнулся: рендер ушёл к клиенту и вернулся назад.
Только за это время веб-платформа **переписала обе стороны**.

Веб не просто изменил бэкенд. Он навязал ему свой API.

- ☑ В Node.js из коробки: `fetch`, `Request`, `Response`, `Headers`, `URL`, `AbortController`, Web Streams
- ☑ Это API, спроектированные **для браузера** комитетом WHATWG
- ☑ Deno, Bun, Cloudflare Workers построены на них **изначально**
- ☑ **WinterCG** (Ecma TC55) — комитет, который договаривается, какие браузерные API обязан иметь серверный рантайм

```
// Node.js 18+, Deno, Bun,
// Cloudflare Workers — один и тот же код

const res = await fetch(url)
const data = await res.json()

export default {
  // Request — из спеки WHATWG Fetch
  async fetch(request) {
    // Response — оттуда же
    return new Response('hello', {
      headers: {
        'content-type': 'text/plain',
      },
    })
  },
}
```

Бэкенд перестал владеть интерфейсом — и начал обслуживать платформу

Со своей обратной совместимостью, своими комитетами и своим legacy

Одно требование к вебу — влияет на три отрасли сразу

Legacy веба — это legacy всей индустрии

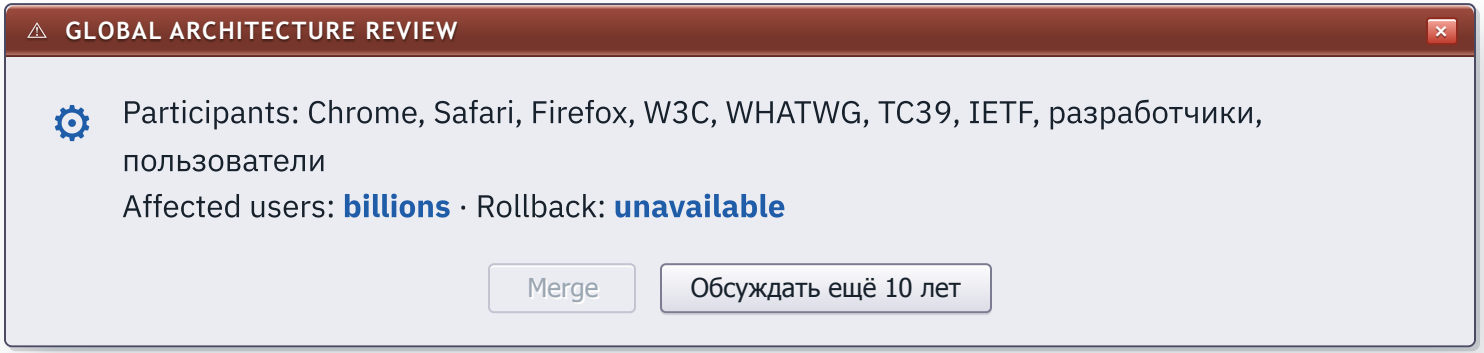
[4] LOADING MODULE

ГЛАВА 4

У мирового продукта нет одного владельца

Или зоопарк комитетов как закономерность масштаба

В обычном продукте вы собираете architecture review



Ни у кого нет права нажать **Merge**

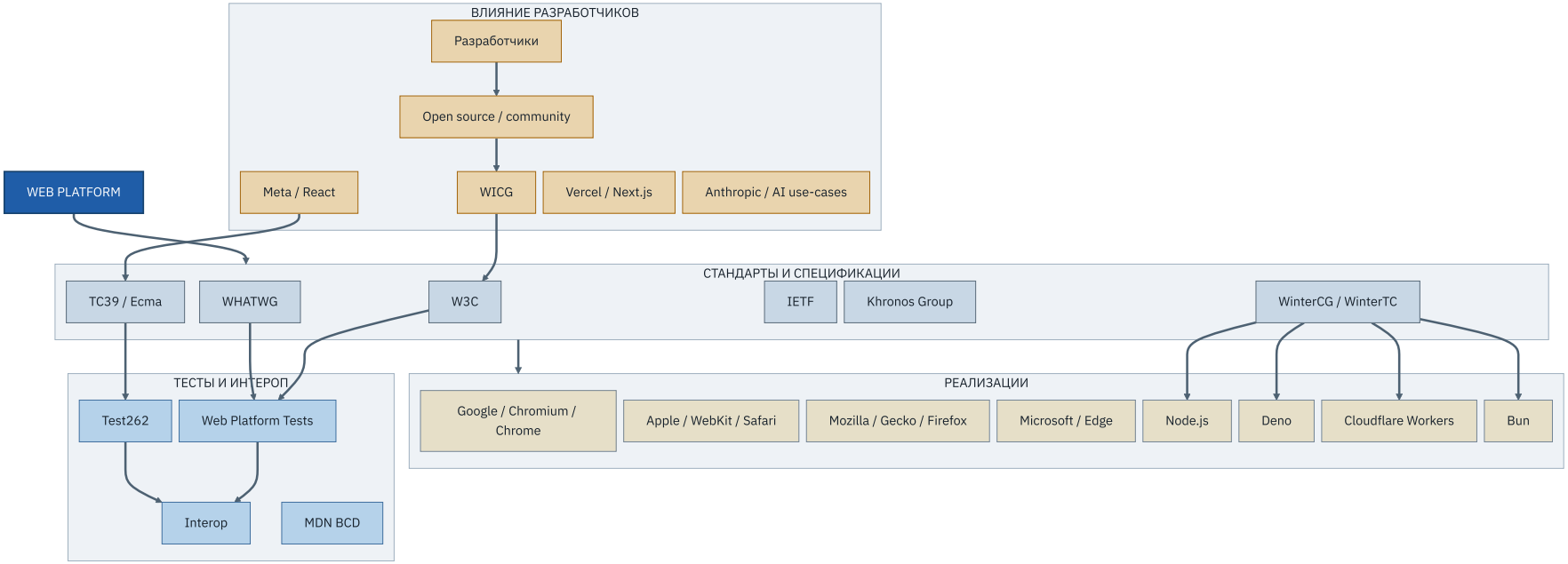


FIG. 4-1 КТО КОНТРИБЬЮТИТ В WEB PLATFORM

ОБЗОР: стандарты, реализации, тесты и экосистема на одной карте.

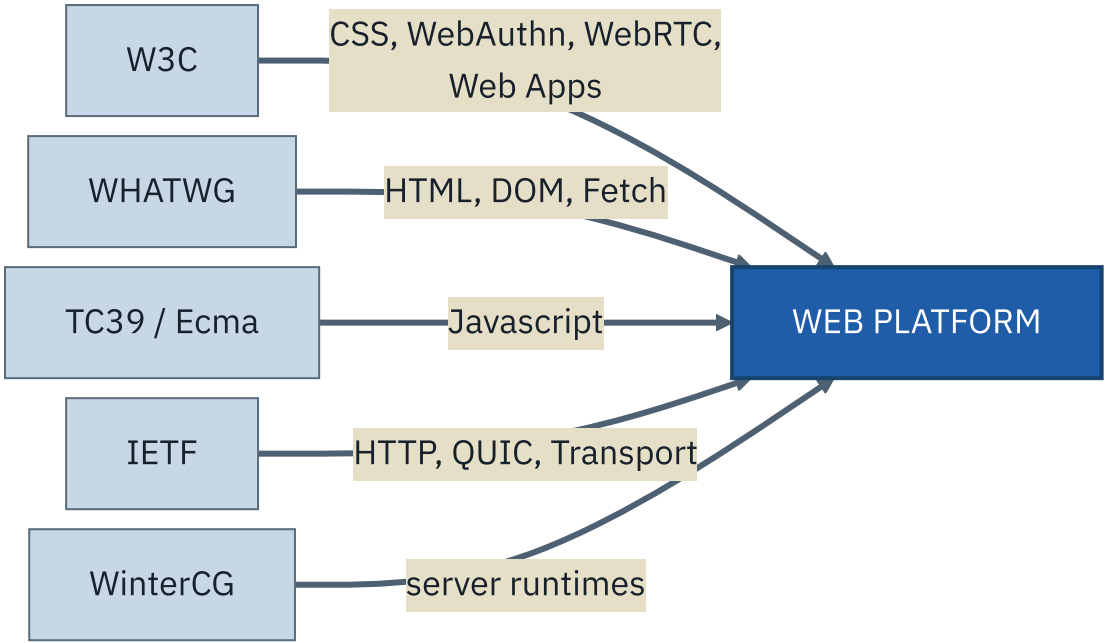


FIG. 4-2 ОСНОВНЫЕ ПЛОЩАДКИ СТАНДАРТИЗАЦИИ

WHATWG ведёт HTML Living Standard; TC39 отвечает за JavaScript; W3C и IETF покрывают значительную часть веб-платформы и сетевого стека.

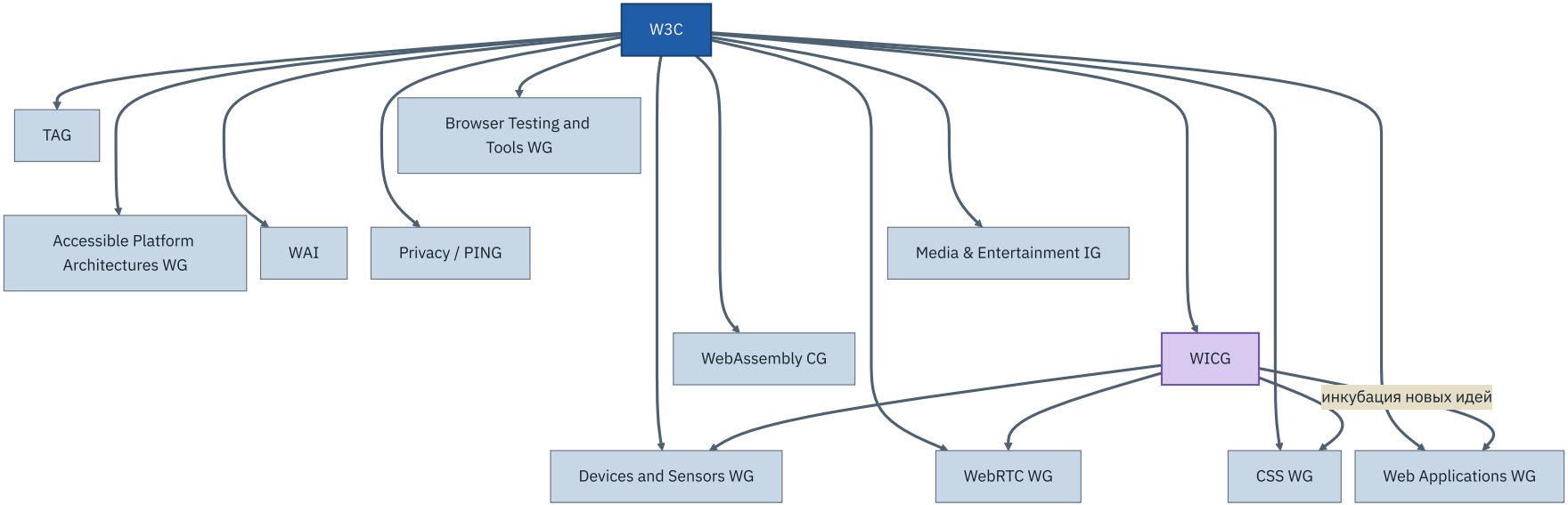
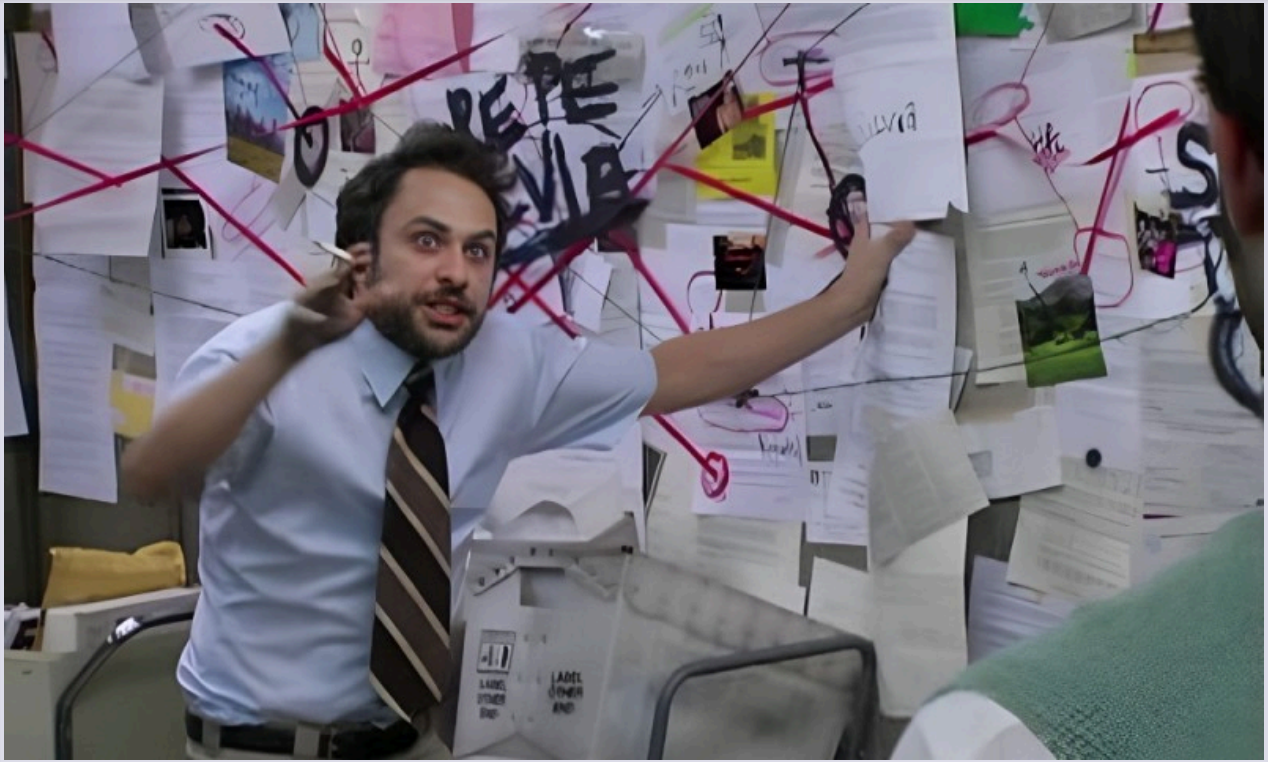


FIG. 4-3 ВНУТРЕННЯЯ СТРУКТУРА W3C

WICG — инкубатор новых веб-идей; внутри W3C множество рабочих и community groups с разной ответственностью.

STANDARDS.PNG — ПРОСМОТР



Где именно решается твой вопрос

Если кратко

- ✔ очень много комитетов и рабочих групп
- ✔ в том числе поставщики браузеров
- ✔ не всегда прозрачные обязанности
- ✔ внутренняя «кухня» и политика

GAME-OF.PNG — ПРОСМОТР



Кухня стандартизации

Спецификации

Кто из вас читал спецификации по JS/HTML/CSS?

…и не сбежал через минуту?

Краткая история спеки

- ☑ **1989–1991** — HTML рождается в CERN вместе с Web.
- ☑ **1994–1999** — W3C формализует HTML как набор версий и рекомендаций.
- ☑ **2004** — **реализация опередила стандарт** и появился комитет WHATWG — как реакция браузерных вендоров.
- ☑ **2012** — у индустрии **две версии** HTML: W3C и WHATWG.
- ☑ **2019** — W3C и WHATWG договариваются о единой версии HTML и DOM, HTML как **living standard**.



HTML STANDARD

https://html.spec.whatwg.org/multipage/

2026

9. *Convert*: This is the step that converts the path to a new path that represents its stroke.

Create a new path that describes the edge of the areas that would be covered if a straight line of length equal to *style*'s `lineWidth` was swept along each subpath in *path* while being kept at an angle such that the line is orthogonal to the path being swept, replacing each point with the end cap necessary to satisfy *style*'s `lineCap` attribute as described previously and elaborated below, and replacing each join with the join necessary to satisfy *style*'s `lineJoin` type, as defined below.

Caps: Each point has a flat edge perpendicular to the direction of the line coming out of it. This is then augmented according to the value of *style*'s `lineCap`. The `"butt"` value means that no additional line cap is added. The `"round"` value means that a semi-circle with the diameter equal to *style*'s `lineWidth` width must additionally be placed on to the line coming out of each point. The `"square"` value means that a rectangle with the length of *style*'s `lineWidth` width and the width of half *style*'s `lineWidth` width, placed flat against the edge perpendicular to the direction of the line coming out of the point, must be added at each point.

Points with no lines coming out of them must have two caps placed back-to-back as if it was really two points connected to each other by an infinitesimally short straight line in the direction of the point's *directionality* (as defined above).

Joins: In addition to the point where a join occurs, two additional points are relevant to each join, one for each line: the two corners found half the line width away from the join point, one perpendicular to each line, each on the side furthest from the other line.

A triangle connecting these two opposite corners with a straight line, with the third point of the triangle being the join point, must be added at all joins. The `lineJoin` attribute controls whether anything else is rendered. The three aforementioned values have the following meanings:

The `"bevel"` value means that this is all that is rendered at joins.

The `"round"` value means that an arc connecting the two aforementioned corners of the join, abutting (and not overlapping) the aforementioned triangle, with the diameter equal to the line width and the origin at the point of the join, must be added at joins.

The `"miter"` value means that a second triangle must (if it can given the miter length) be added at the join, with one line being the line between the two aforementioned corners, abutting the first triangle, and the other two being continuations of the outside edges of the two joining lines, as long as required to intersect without going over the miter length.

The miter length is the distance from the point where the join occurs to the intersection of the line edges on the outside of the join. The miter limit ratio is the maximum allowed ratio of the miter length to half the line width. If the miter length would cause the miter limit ratio (as set by *style*'s `miterLimit` attribute) to be exceeded, then this second triangle must not be added.

The subpaths in the newly created path must be oriented such that for any point, the number of times a half-infinite straight line drawn from that point crosses a subpath is even if and only if the number of times a half-infinite straight line drawn from that same point crosses a subpath going in one direction is equal to the number of times it crosses a subpath going in the other direction.

Готово

Note

Thus, the bitmap of such a context starts off as opaque black instead of transparent black; `clearRect()` always results in opaque black pixels, every fourth byte from `getImageData()` is always 255, the `putImageData()` method effectively ignores every fourth byte in its input, and so on. However, the alpha component of styles and images drawn onto the canvas are still honoured up to the point where they would impact the output bitmap's alpha component; for instance, drawing a 50% transparent white square on a freshly created output bitmap with its alpha set to false will result in a fully-opaque gray square.

The `CanvasSettings` object also has a **desynchronized** boolean. When a `CanvasSettings` object's desynchronized is true, then the user agent may optimize the rendering of the canvas to reduce the latency, as measured from input events to rasterization, by desynchronizing the canvas paint cycle from the event loop, bypassing the ordinary user agent rendering algorithm, or both. Insofar as this mode involves bypassing the usual paint mechanisms, rasterization, or both, it might introduce visible tearing artifacts.

Note

The user agent usually renders on a buffer which is not being displayed, quickly swapping it and the one being scanned out for presentation; the former buffer is called back buffer and the latter front buffer. *A popular technique for reducing latency is called front buffer rendering, also known as* single buffer *rendering, where rendering happens in parallel and racily with the scanning out process. This technique reduces the latency at the price of potentially introducing tearing artifacts and can be used to implement in total or part of the desynchronized boolean. [MULTIPLEBUFFERING]*

Note

The desynchronized boolean can be useful when implementing certain kinds of applications, such as drawing applications, where the latency between input and rasterization is critical.

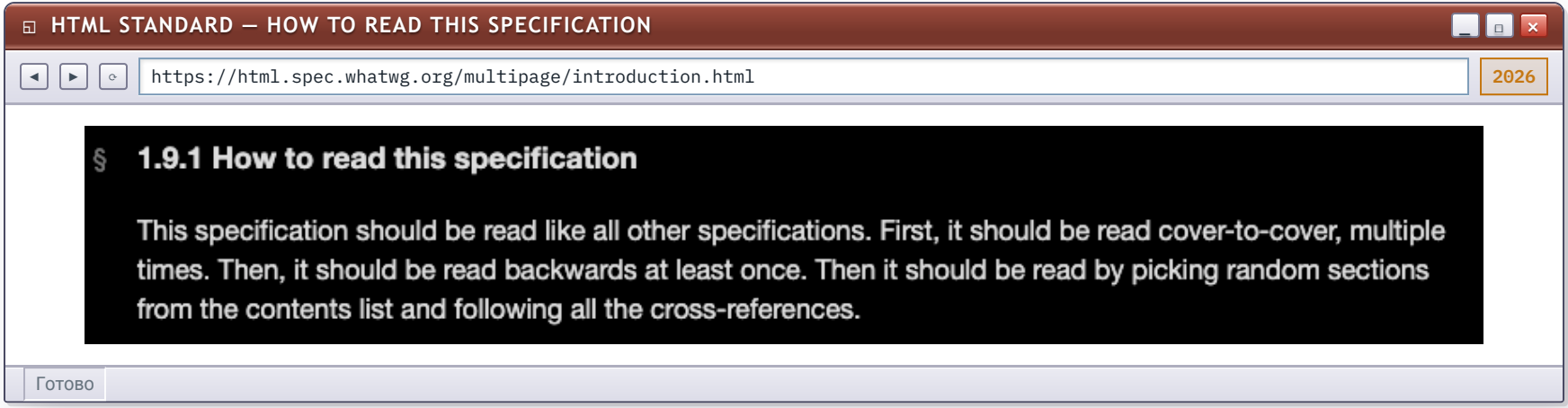
The `CanvasSettings` object also has a **will read frequently** boolean. When a `CanvasSettings` object's will read frequently is true, the user agent may optimize the canvas for readback operations.

Note

On most devices the user agent needs to decide whether to store the canvas's output bitmap on the GPU (this is also called "hardware accelerated"), or on the CPU (also called "software"). Most rendering operations are more performant for accelerated canvases, with the major exception being readback with `getImageData()`, `toDataURL()`, or `toBlob()`. `CanvasSettings` objects with will read frequently equal to true tell the user agent that the webpage is likely to perform many readback operations and that it is advantageous to use a software canvas.

Прокрутка...

HTML Living Standard (Last Updated 28 September 2026)



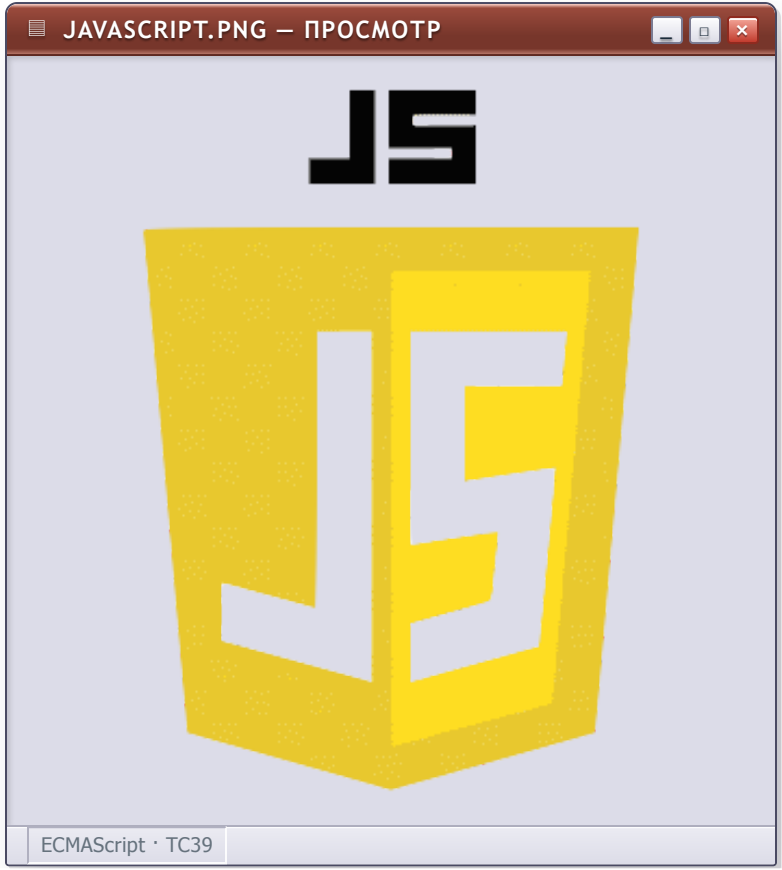
- ☑ **Эту спецификацию следует читать так же, как и все другие спецификации.**
- ☑ Сначала — **от корки до корки**, несколько раз.
- ☑ Потом — в **обратном** направлении, как минимум один раз.
- ☑ Затем — выбирая случайные разделы в оглавлении и переходя по всем перекрёстным ссылкам.

CSS: модульная модель = философия микрорелизов

Каждый модуль — свои уровни, состояние фиксирует CSS Snapshot

Javascript

- ☑ **ECMAScript 6** — последняя большая версия (2015)
- ☑ Новые фичи теперь проходят через **proposal stages**
- ☑ Современный JavaScript — **непрерывный поток улучшений** с ежегодными мини-релизами
- ☑ Почти **10 лет** понадобилось, чтобы «починить время» в вебе



Все спецификации пришли к философии микрорелизов

Браузерные войны

Браузерные войны

Анархия и отсутствие
стандартов между браузерами

WARS.PNG — ПРОСМОТР

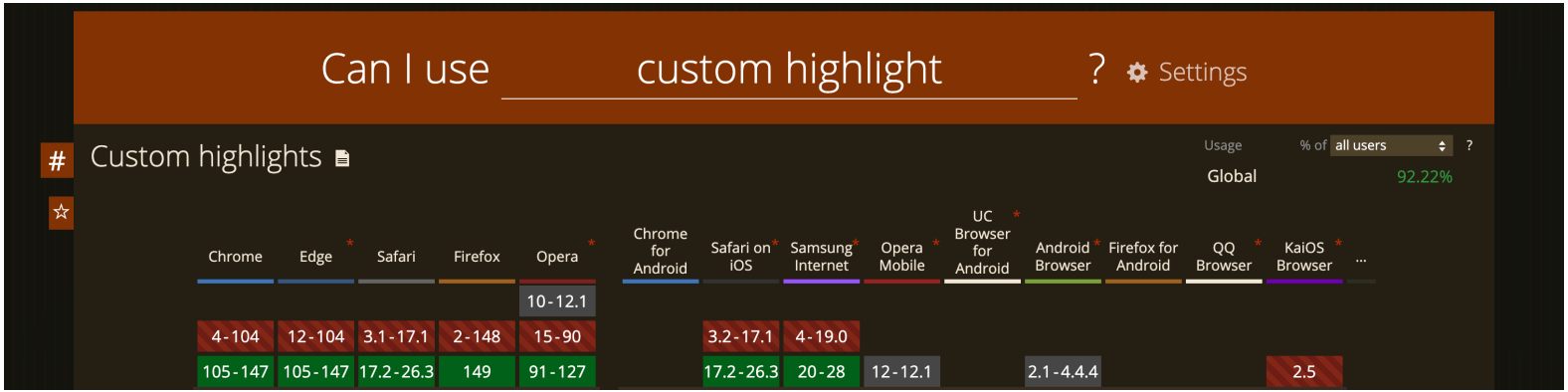


Браузерные войны · 1995–2001

CAN I USE… — SUPPORT TABLES

https://caniuse.com/

2026



Готово

- ☑ Узнаешь из статьи/доклада про новую крутую фичу
- ☑ Пробуешь внедрить — не работает
- ☑ Плачешь в подушку / пишешь полифилл / загоняешь юзеров в один браузер
- ☑ …Хорошо, если года через 3 станет доступно везде

Поэтому мы не ждали

Работа программиста — это борьба со сложностью.
А борются со сложностью **дроблением**.

JavaScript: модулей не было 20 лет

- ☑ **1995** — язык появился. Модулей нет. Всё в глобальной области.
- ☑ IIFE, пространства имён, `window.MyApp = {}`
- ☑ **2009** — CommonJS и AMD/RequireJS. Модули придумало **сообщество**, не платформа.
- ☑ **2011+** — Browserify, webpack, Rollup
- ☑ **2015** — ES-модули в стандарте. **2017–2018** — наконец в браузерах.

Бандлеры существуют **только** потому, что модули опоздали на двадцать лет

CSS: область видимости появилась недавно

- ☑ Один глобальный неймспейс на весь документ
- ☑ Каскад, специфичность, `!important` — и `.button` из чужого файла
- ☑ **2005–2009** — BEM в Яндексе: инкапсуляция **через соглашение об именах**
- ☑ **2015** — CSS Modules. **2016** — styled-components. **2017** — Tailwind.
- ☑ **2023** — `@scope` в Chrome. Во всех браузерах — только с **декабря 2025**.

BEM — это **протез** отсутствующей в языке инкапсуляции.

HTML: компонентов не было

- ☑ Разметку нельзя переиспользовать. Копипаста или шаблонизатор на сервере.
- ☑ **2011** — идея Web Components
- ☑ **2013** — выходит React. Ждать перестали.
- ☑ **2016–2018** — Custom Elements и Shadow DOM доезжают до браузеров
- ☑ Инкапсуляция приехала, но стилизовать сквозь границу больно до сих пор

Фреймворки победили не потому, что были лучше стандарта.
Они приехали **на пять лет раньше**.

Всё, что мы называем современным фронтендом —

это протезы отсутствующей в платформе модульности

Webpack, Babel, BEM, CSS-in-JS, компонентные фреймворки — это не изобретения. Это компенсации.

А почему модульность опоздала?

Потому что договориться о ней должны были **все сразу**

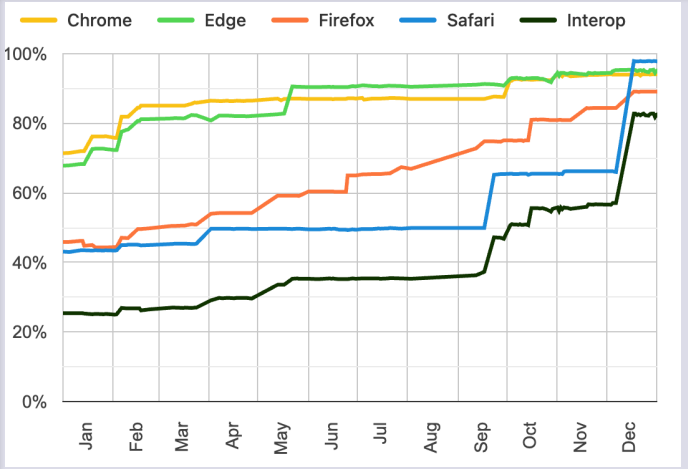
Модули в языке — это не фича. Это решение, которое ломает всё, что написано до него.

Прошло 32 года

Interop

Google, Mozilla, Microsoft, Apple, Bocoup и Igalia — объединились для борьбы с проблемами несовместимости браузеров и стандартизации

PROGRESS.PNG — ПРОСМОТР



Прогресс Interop по годам

INTEROP 2026 DASHBOARD

https://wpt.fyi/interop-2026

2026



Готово

1995–2001
Браузерные войны
Совместимость — через UA-sniffing

2004
WHATWG
Living standard вместо снапшотов

2011
Test262
Тесты соответствия для JS

2012–2017
Web Platform Tests
Один репозиторий тестов на всех

2022
Interop
Общие focus areas всех вендоров

2023
Baseline
Единый сигнал «этим можно пользоваться»

2025
WinterTC
Рантаймы — в Ecma TC55

[5] LOADING MODULE

ГЛАВА 5

Цена успеха — legacy

Или почему нельзя просто взять и переписать веб

но почему так сложно?



Хотите увидеть, как выглядит legacy, которое нельзя удалить?

Оно прямо сейчас уезжает с каждым вашим запросом

User-Agent: тридцать лет наслоений

```
1993  NCSA_Mosaic/2.0
1994  Mozilla/1.0                                      Netscape: кодовое имя «Mosaic killer»
1996  Mozilla/2.0 (compatible; MSIE 3.0; Windows 95) IE притворяется Netscape,
                                                       чтобы ему отдавали фреймы
2004  Mozilla/5.0 (...) Gecko/20100101                 Firefox: дата в строке — фальшивая
                                                       и намертво замороженная
2003  Mozilla/5.0 (...) AppleWebKit/85.7
      (KHTML, like Gecko) Safari/85.5                  Safari притворяется Gecko
2008  Mozilla/5.0 (...) AppleWebKit/525.13
      (KHTML, like Gecko) Chrome/0.2 Safari/525.13     Chrome притворяется всеми сразу
```

А вот что ваш Chrome отправляет сегодня

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
```

Он представляется как **Mozilla**, как **AppleWebKit**, как **KHTML**, как **Gecko** и как **Safari**

Ни одно из этих утверждений не является правдой

А `537.36` — это не версия. Это константа, **замороженная с 2013 года**, потому что на неё где-то что-то смотрит.

Починить UA-строку не смогли. Её заморозили.

- ☑ **User-Agent Reduction** в Chrome, 2022–2023
- ☑ Минорная версия теперь всегда `0.0.0`
- ☑ Версия ОС и модель устройства — огрублены
- ☑ Детали переехали в отдельный механизм — Client Hints

Менять её оказалось **опаснее**, чем оставить как есть

Браузеры возят с собой патчи под конкретные сайты

- ☑ **Opera** — файл `browser.js`: список исправлений под отдельные сайты, обновлялся вместе с браузером
- ☑ **Firefox** — webcompat interventions: подмена UA и патчи для сайтов, которые иначе не открываются
- ☑ **WebKit** — файл `Quirks.cpp` с обходными путями, где прямо перечислены домены

Это не костыли отдельных команд. Это **штатный механизм** веб-платформы.

Legacy, записанное в стандарт

- ✔ `document.all` — единственный объект в JavaScript, который **врёт про свой** `typeof`

  Он фальсив по спецификации — в ECMAScript для этого есть отдельный внутренний слот. Зачем? Чтобы код детекта Internet Explorer, написанный в 1999 году, уходил в правильную ветку.

- ✔ `<marquee>` не удалён — он **специфицирован** в разделе устаревших возможностей

DO NOT DELETE Устаревание в вебе не удаляют. Его **документируют и поддерживают вечно**.

Но Space Jam 1996 года всё ещё открывается

SPACE JAM

https://www.spacejam.com/1996/

1996



Done — без единой правки

Тот же HTML, тот же дизайн — работает в сегодняшнем Chrome. Страница, свёрстанная тридцать лет назад под браузер, которого больше нет.

В этом одновременно сила и приговор веб-платформы.

Для чего создавался веб?



Legacy — это не всегда ругательство

Не умерло, значит нужно пользователям

Принципы развития веб-платформы

- ☑ **Не ломай веб.**
- ☑ **Интероперабельность** — должно работать везде и одинаково.
- ☑ **Расширяй**, а не заменяй.
- ☑ **Безопасность по умолчанию.**
- ☑ **Доступность по умолчанию.**
- ☑ Патентная политика без роялти.

Именно поэтому всё так медленно и осторожно — **цена ошибки слишком велика**

У этих принципов есть цена

И платим её мы

Мы не можем писать на голой тройке

- ☑ Нет типов
- ☑ Нет области видимости в стилях
- ☑ Компоненты приехали на пять лет позже React
- ☑ Модули опоздали на двадцать лет
- ☑ Реактивность, роутинг, формы — собирайте сами

Всё, что хайпует — это сахар вокруг HTML, CSS и JS

JSX компилируется в вызовы функций, которые дёргают DOM

Tailwind компилируется в классы. TypeScript — стирается. Svelte и Vue SFC — в тот же JS.

Ни один фреймворк не заменил платформу. Все они её **обвязка**.

Мы приезжаем на Порше к даче с трактором

Тулинг оброс мощностью. Дорога под ним не менялась.

Но это не чья-то вина

- ☑ Каждый раз выбор был не «удобно или неудобно»
- ☑ Выбор был: **сделать удобно** или **не сломать миллиард страниц**
- ☑ Платформа каждый раз выбирала второе
- ☑ А ещё договориться должны были все вендоры сразу — и ни у кого нет права нажать Merge

Плохой Developer Experience — это счёт за обратную совместимость.

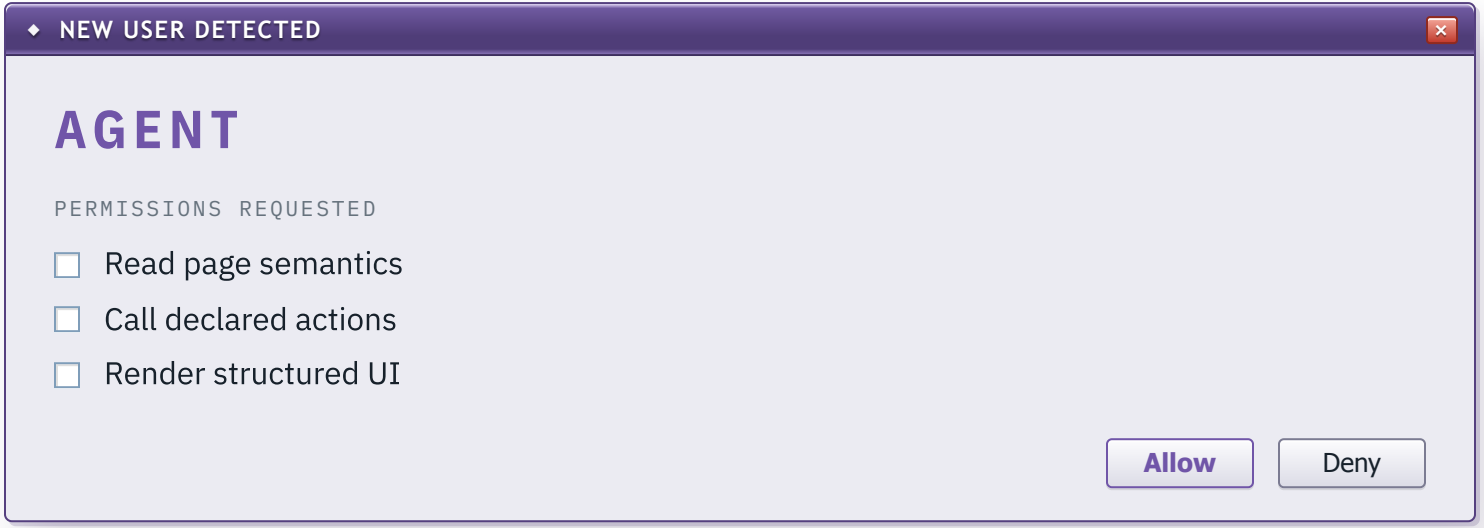
Но страдаем мы.

[6] LOADING MODULE

ГЛАВА 6

Новый пользователь: агент

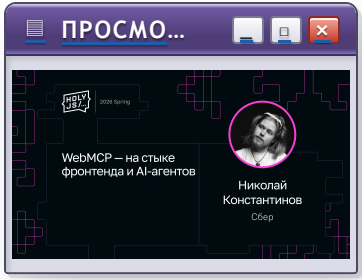
~~Заменит ли нас ИИ~~ · или новый change request для legacy-продукта

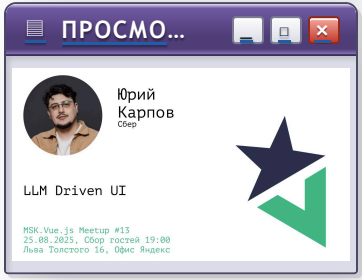

NEW USER DETECTED
AGENT
PERMISSIONS REQUESTED
Read page semantics
Call declared actions
Render structured UI
Allow
Deny

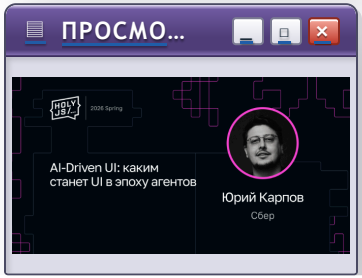
Платформа уже отвечает

- ☑ **WebMCP** — сайт сам объявляет агенту свои действия, вместо протыкивания кнопок вслепую
- ☑ **A2UI** — агент описывает интерфейс JSON-ом, приложение собирает его из своих компонентов
- ☑ **Prompt API** — «страница» обращается к модели через браузер

Но ничто ещё не стандарт





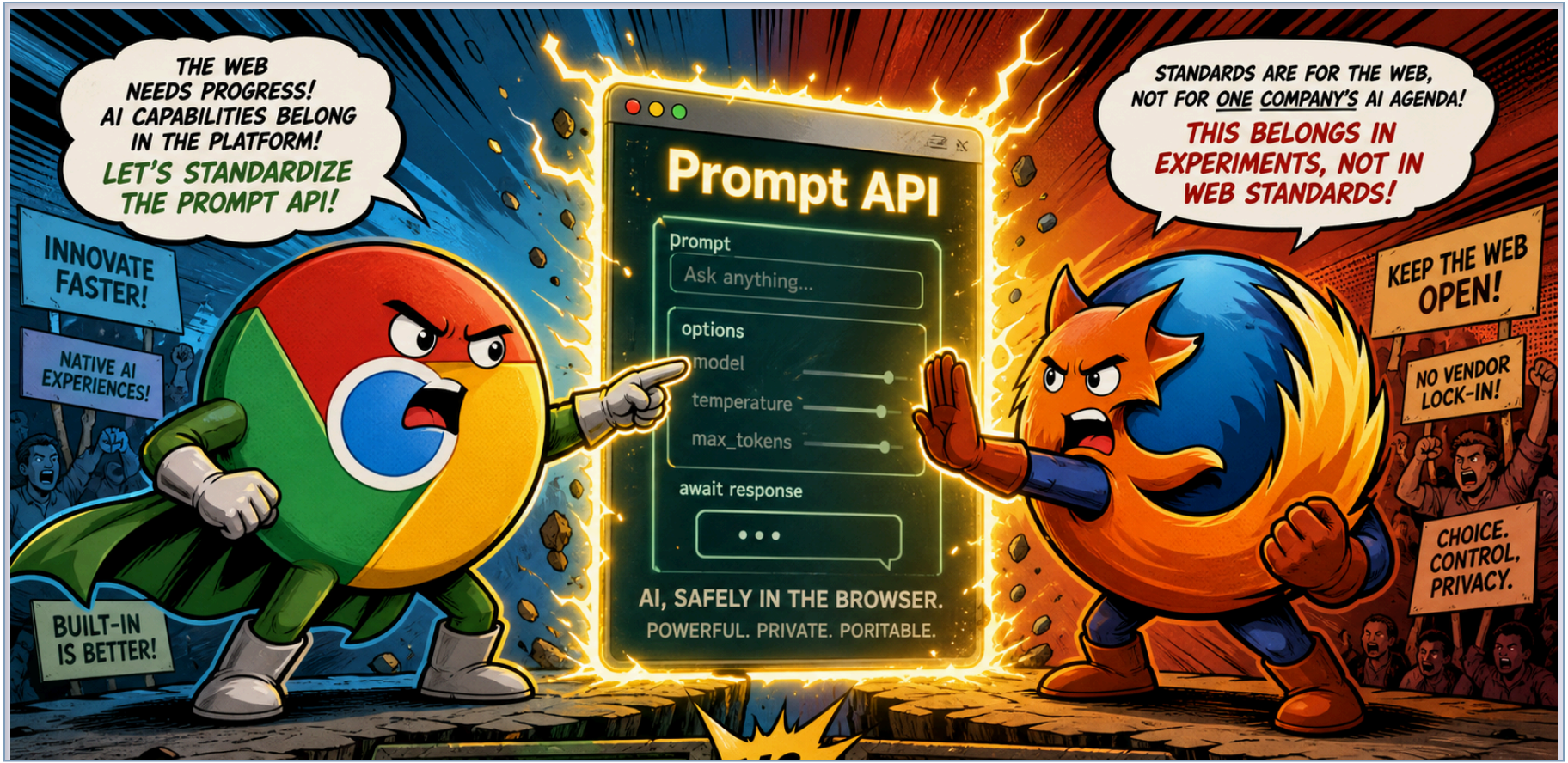


Prompt API

- ☑ предложен Google весной, с мая 2026-го — в стабильном Chrome
- ☑ вызов модели прямо на сайте при помощи js
- ☑ быстродействие, одно «окно входа»
- ☑ ...
- ☑ можно использовать **только** Gemini Nano

position: negative

concerns: interoperability


THE WEB NEEDS PROGRESS! AI CAPABILITIES BELONG IN THE PLATFORM!
LET'S STANDARDIZE THE PROMPT API!
INNOVATE FASTER!
NATIVE AI EXPERIENCES!
BUILT-IN IS BETTER!
Prompt API
Prompt
Ask anything...
options
model
temperature
max_tokens
await response
AI, SAFELY IN THE BROWSER.
POWERFUL. PRIVATE. PORITABLE.
STANDARDS ARE FOR THE WEB, NOT FOR ONE COMPANY'S AI AGENDA!
THIS BELONGS IN EXPERIMENTS, NOT IN WEB STANDARDS!
KEEP THE WEB OPEN!
NO VENDOR LOCK-IN!
CHOICE. CONTROL. PRIVACY.

И сообщество продавило: не только Gemini

Другие вендоры и разработчики —
защита платформы от монополии

Мы не хотели семантически верстать и писать документацию для кожаных мешков — а для агентов хотим.

Best practices никуда не делись.

Мы не знаем, каким будет следующий веб

Но мы точно знаем, что старый никуда не денется

Выводы

- ☑ веб — самый большой живой **legacy-продукт** в мире
- ☑ менялся не только фронтенд, но и **дизайн** и **бэкенд**
- ☑ **комитеты — следствие** масштаба, а не причина
- ☑ наш тулинг — **протезы и костыли**
- ☑ плохой DX — **счёт за обратную совместимость**
- ☑ legacy не удаляют — его **документируют и поддерживают вечно**
- ☑ система живёт **36+ лет** (и развивается)

Обычный продукт делает одна компания. Веб — весь мир.

Разница только в масштабе.

Послесловие

Всё не так плохо 😊

Классной вам конференции!

Спасибо

HOLYJS AUTUMN 2026

HQ, DEPT OF THE PRESENTATION

