

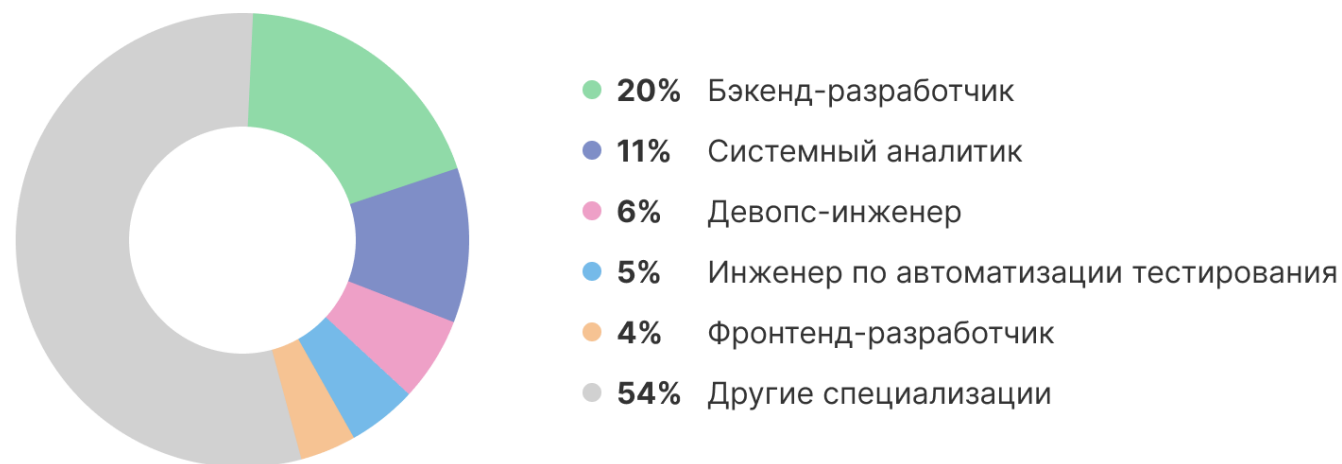
Веб-программирование

Вводное занятие

О современной веб-разработке

Веб-разработка в 2024

Топ 5 специализаций, которые искали во 2кв 2024



Активность найма на IT-рынке во 2кв 2024

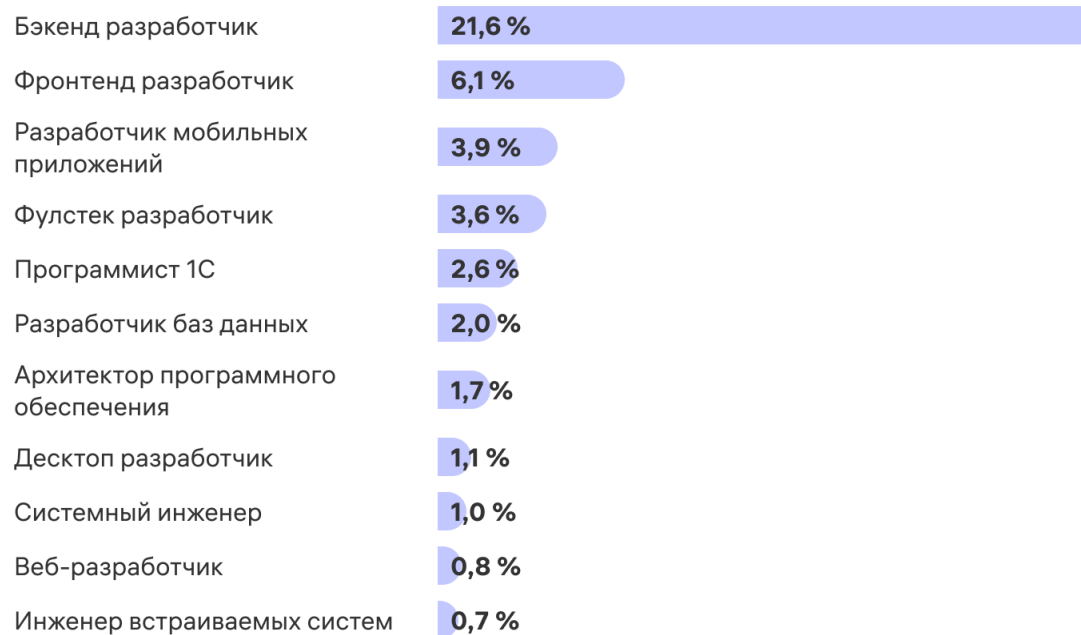
Хабр Карьера

https://habr.com/ru/companies/habr_career/articles/829182/

Веб-разработка в 2024

Самые востребованные разработчики

данные из вакансий на Хабр Карьере весной 2023 года











Зарплатные вилки весной 2023:
языки программирования и фреймворки

Хабр Карьера

https://habr.com/ru/companies/habr_career/articles/746038/

Веб-разработка в 2024

Sep 2024	Sep 2023	Change	Programming Language		Ratings	Change
1	1			Python	20.17%	+6.01%
2	3	↑		C++	10.75%	+0.09%
3	4	↑		Java	9.45%	-0.04%
4	2	↓		C	8.89%	-2.38%
5	5			C#	6.08%	-1.22%
6	6			JavaScript	3.92%	+0.62%
7	7			Visual Basic	2.70%	+0.48%
8	12	↑↑		Go	2.35%	+1.16%
9	10	↑		SQL	1.94%	+0.50%
10	11	↑		Fortran	1.78%	+0.49%

<https://www.tiobe.com/tiobe-index/>

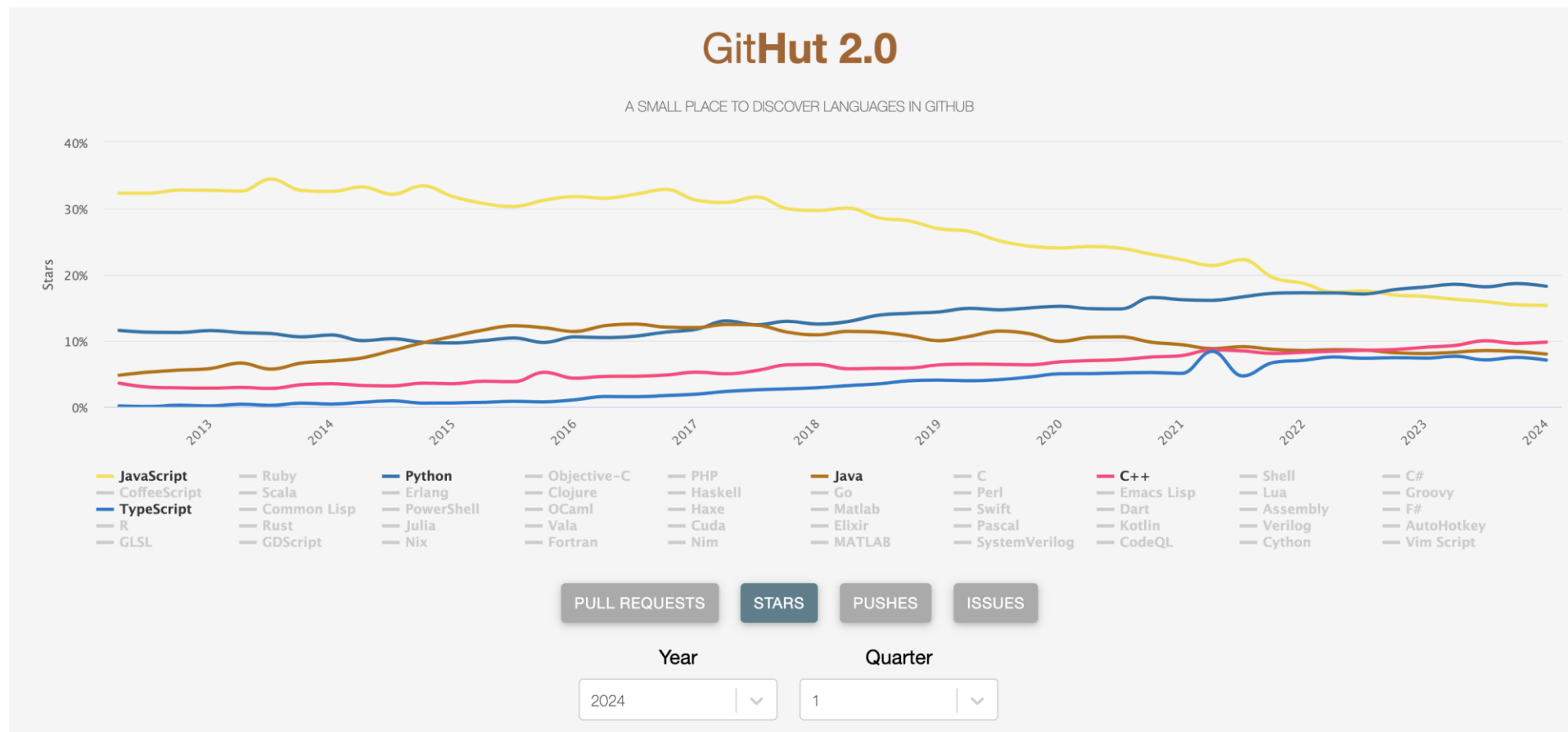
Веб-разработка в 2024

Worldwide, Sept 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	29.66 %	+1.6 %
2		Java	15.64 %	-0.2 %
3		JavaScript	8.3 %	-1.0 %
4		C#	6.64 %	-0.1 %
5		C/C++	6.46 %	-0.2 %
6	↑	R	4.66 %	+0.2 %
7	↓	PHP	4.35 %	-0.5 %
8		TypeScript	2.96 %	-0.0 %
9		Swift	2.69 %	+0.0 %
10	↑	Rust	2.65 %	+0.6 %

- PopularitY of Programming Language (PYPL):
<https://pypl.github.io/PYPL.html>
- Про рейтинги популярности языков программирования — на лето 2024 года:
<https://habr.com/ru/companies/ssp-soft/articles/821663/>

Веб-разработка в 2024

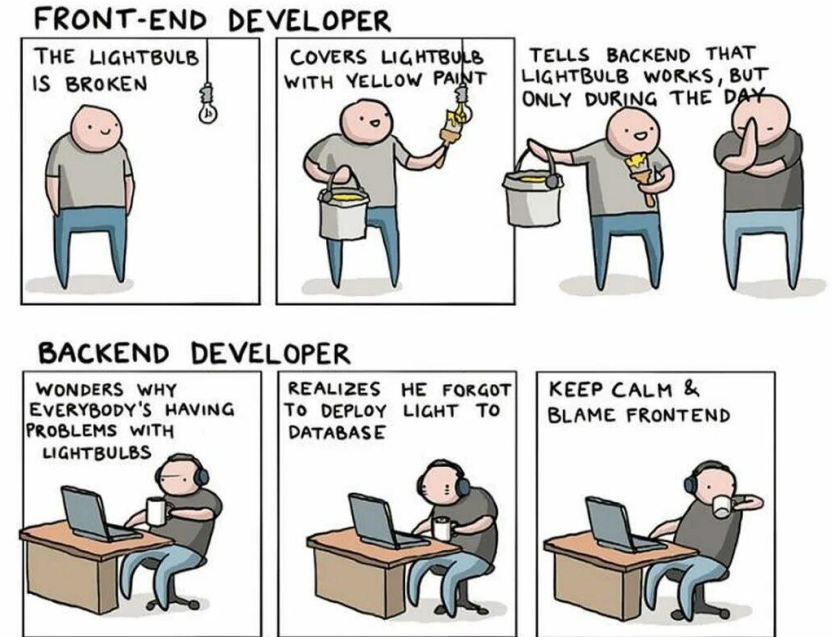


<https://madnight.github.io/github/#/stars/2024/1>

О курсе

Что даст курс?

- Представление о том, как разрабатываются современные веб-приложения
- Понимание того, чем и как занимаются фронтенд-разработчики
- Осознание того, что инструменты фронтенда используются ещё и в:
 - Бэкенде
 - Desktopных/мобильных приложениях
 - Машинном обучении
- Взгляд на дизайн с точки зрения разработчика
- Желание верстать и писать код на JavaScript (надеюсь)



Структура курса

Занятие 1

- Современная веб-разработка
- Базовые концепции HTML и CSS
- ДЗ: курс по HTML/CSS

Занятие 2

- Верстаем по макетам из Figma
- ДЗ: сверстать своё приложение

Структура курса

Занятие 3

- Базовые концепции JavaScript
- Добавляем JavaScript в проект
- ДЗ: задачи на JavaScript

Занятие 4

- API и его подключение на стороне фронтенда
- ДЗ: подключение API к проекту

Структура курса

Занятие 5

- Разворачиваем проекта в облаке
- ДЗ: развернуть проект в облаке

Занятие 6

- Введение в React
- ДЗ: переписать проект на React

Занятие 7 (и далее)

- Введение в TypeScript
- ML на JavaScript
- Yet another lecture

FAQ

A. Где найти домашние задания?

Q. В репозитории курса на GitHub: <https://github.com/lyaplyap/frontend-development-course> (а также они упоминаются после соответствующих лекций).

A. Как сдавать задания?

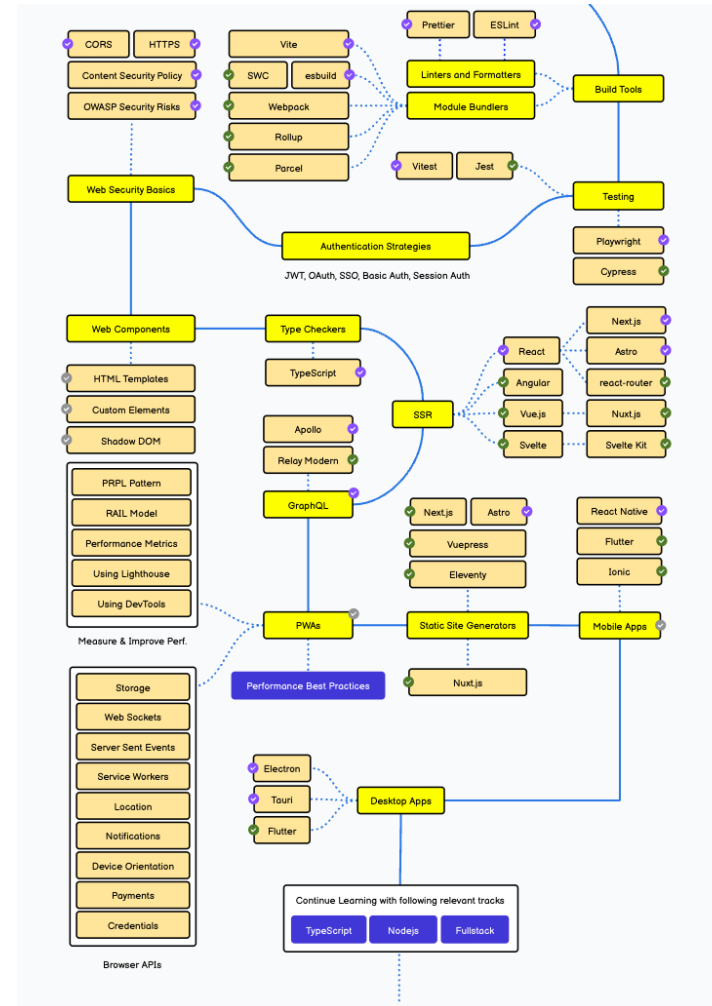
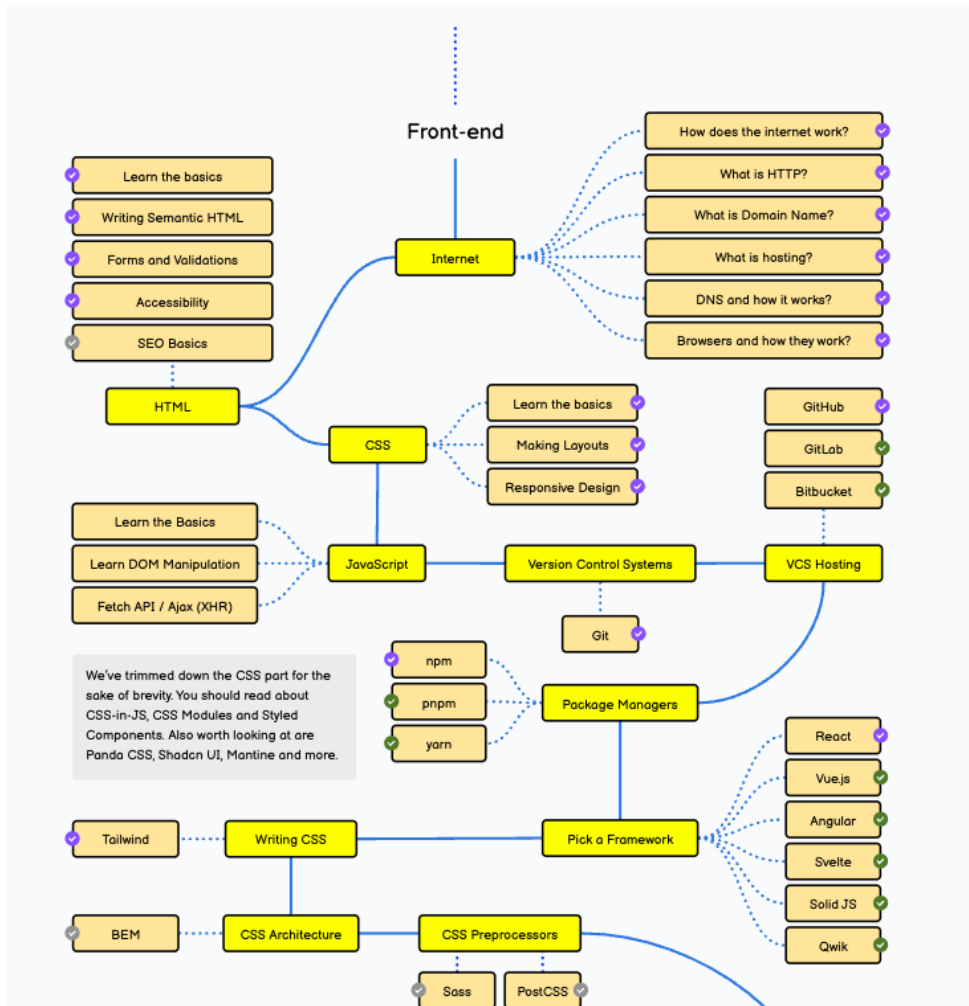
Q. Если хотите, чтобы по вашей работе оставили комментарии, то — в виде ссылки на pull-request в GitHub, если не нуждаетесь в ревью, то — просто в виде ссылки на GitHub.

A. Есть ли штрафы за сдачу заданий после дедлайна?

Q. Да. Максимальная оценка за выполненное задание — 10 баллов, минимальная — 3 балла. В первые три недели опоздания снимается по 1 баллу (по баллу за каждую неделю), в четвёртую и пятую по 2 балла.

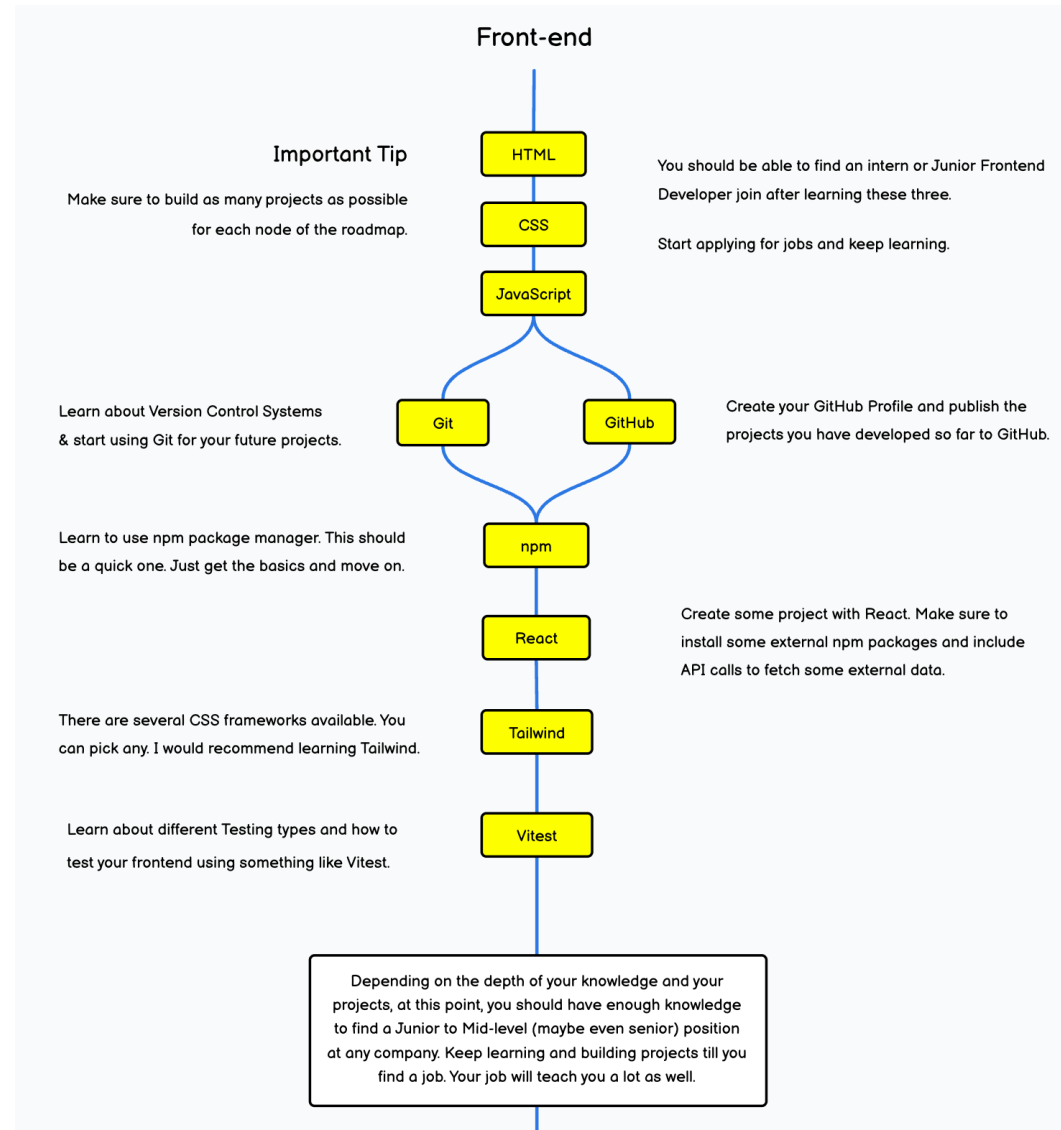
Кто такой фронтенд-разработчик?

Дорожная карта фронтенд-разработчика



<https://roadmap.sh/frontend>

Дорожная карта фронтенд-разработчика



Дорожная карта фронтенд-разработчика

- **База:**

- HTML, CSS
- JavaScript (+TypeScript)

- **Инфраструктура:**

- Node.js + npm (пакетные менеджеры)
- Сборщики
- Линтеры

- **UI-библиотеки/фреймворки:**

- React
- Vue / Angular / Svelte

О современных веб-приложениях

Ретроспектива

How it started

```
RETRO
├── howitsgoing
├── howitstarted
│   ├── index.html
│   ├── script.js
│   └── styles.css
└── ...

howitstarted > < index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>My web app</title>
6     <link rel="stylesheet" href="./styles.css">
7     <script src="./script.js"></script>
8   </head>
9   <body>
10    <div>Hello, world!</div>
11  </body>
12 </html>
```

How it's going

```
RETRO
├── howitsgoing
│   ├── node_modules
│   ├── public
│   │   ├── vite.svg
│   │   └── src
│   │       ├── assets
│   │       ├── App.css
│   │       ├── App.tsx
│   │       ├── index.css
│   │       ├── main.tsx
│   │       ├── vite-env.d.ts
│   │       ├── .gitignore
│   │       ├── eslint.config.js
│   │       ├── index.html
│   │       ├── package-lock.json
│   │       ├── package.json
│   │       ├── README.md
│   │       ├── tsconfig.app.json
│   │       ├── tsconfig.json
│   │       ├── tsconfig.node.json
│   │       └── vite.config.ts
└── ...

howitsgoing > src > App.tsx > ...
1 import { useState } from 'react'
2 import reactLogo from './assets/react.svg'
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7   const [count, setCount] = useState(0)
8
9   return (
10     <div>
11       <a href="https://vitejs.dev" target="_blank">
12         <img src={viteLogo} className="logo" alt="Vite logo" />
13       </a>
14       <a href="https://react.dev" target="_blank">
15         <img src={reactLogo} className="logo react" alt="React logo" />
16       </a>
17     </div>
18     <h1>Vite + React</h1>
19     <div className="card">
20       <button onClick={() => setCount((count) => count + 1)}>
21         count is {count}
22       </button>
23     </div>
24     <p>
25       Edit <code>src/App.tsx</code> and save to test HMR
26     </p>
27   )
28 }
```

Как работают веб-приложения?

Фронтенд (клиентская часть / клиент)

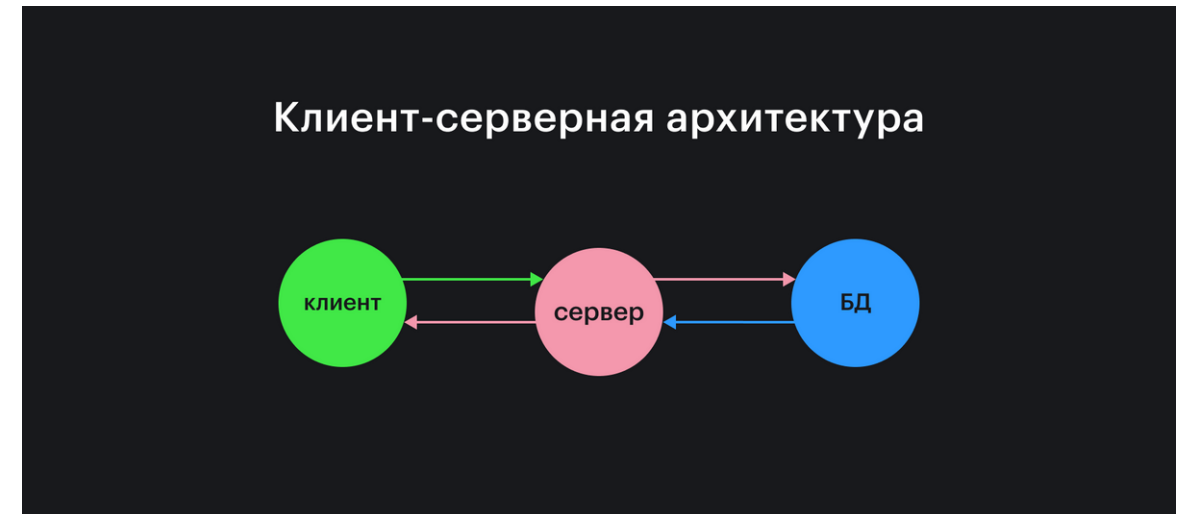
- Обменивается данными с бэкендом
- Отрисовывает данные в нужном виде

Бэкенд (серверная часть / сервер)

- Обменивается данными с фронтендом
- Обрабатывает данные

База данных (БД)

- Хранит данные



<https://doka.guide/tools/web-app-works/>

HTML, CSS и JS

- **HTML** — язык разметки для создания структуры веб-страницы и представления контента. Благодаря разметке браузер знает в каком порядке отображать элементы, и что они значат.
- **CSS** — язык каскадных стилей, который задаёт визуальное оформление для HTML.
- **JavaScript** — язык программирования широкого спектра. На нём можно как создавать динамические интерфейсы, так и работать с базами данных и операционными системами.

(взято из <https://doka.guide>)



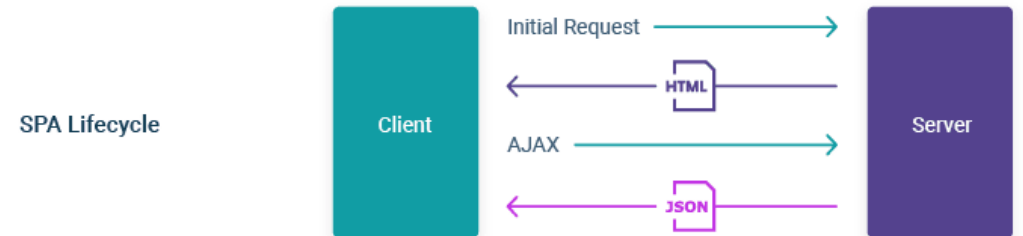
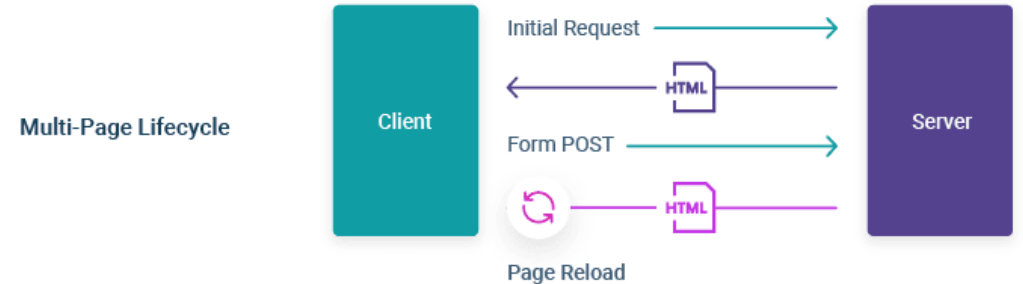
Виды веб-приложений

Статические многостраничные приложения (MPA)

- Набор готовых свёрстанных страниц
- Динамическая генерация HTML на сервере

Одностраничные приложения (SPA)

- Client Side Rendering (CSR)
- Server Side Rendering (SSR)



<https://dev.to/adnanbabakan/deploy-a-single-page-application-with-200-html-2p0f>

Серверный JavaScript

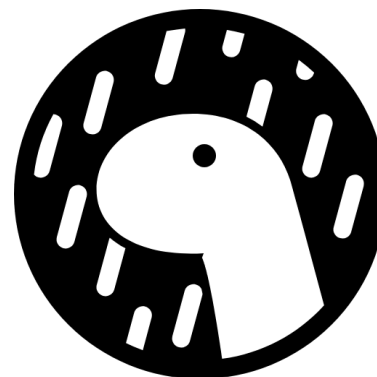
Node.js — среда выполнения кода JavaScript вне браузера, позволяющая писать серверный код для веб-страниц и веб-приложений, а также для программ командной строки.

Альтернативы:

- Deno
- Bun



Node.js



Deno



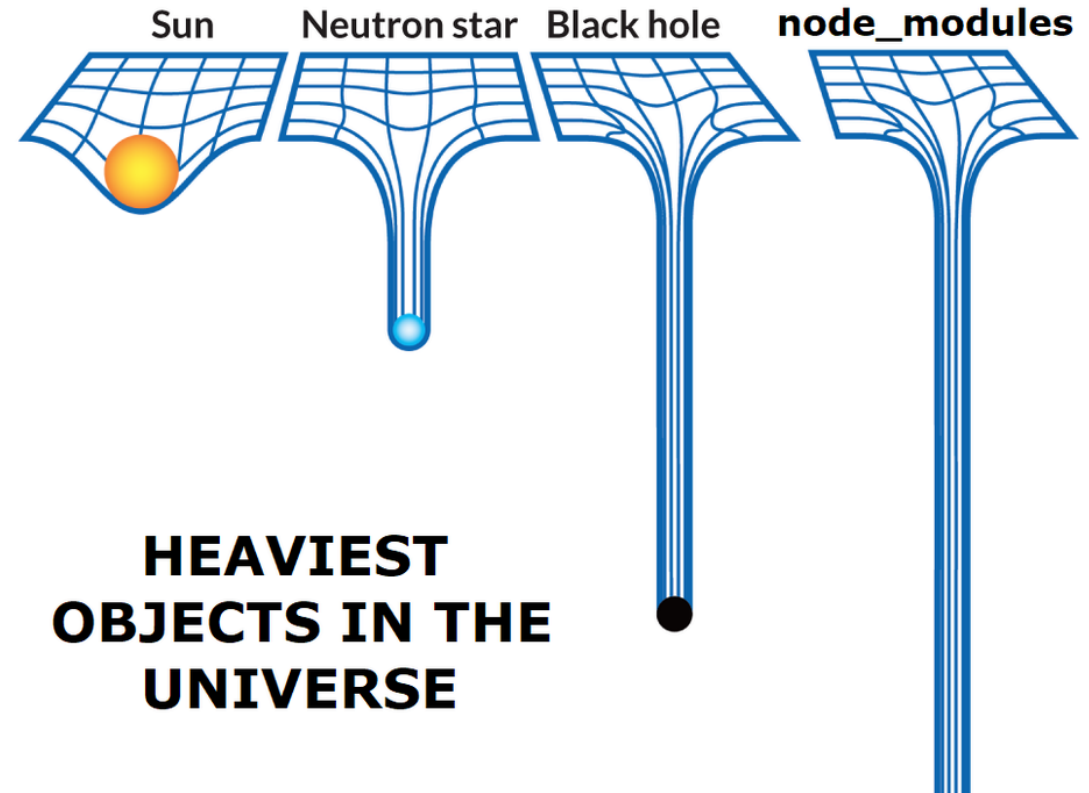
Bun

Пакетные менеджеры

Для удобной работы с ними существуют пакетные менеджеры (менеджеры зависимостей). Они помогают выбрать версию библиотеки и гарантируют безопасность.

Популярные менеджеры:

- npm – стандартный (идёт сразу с Node.js)
- yarn
- pnpm



UI-библиотеки и фреймворки

В мире фронтенда много различных фреймворков для разработки SPA.

Большинство из них различаются подходами к разработке и инструментами, которыми они обладают.

Но в основе всех этих фреймворков часто лежат одни и те же концепции.

Основные концепции:

- Реактивность
- Virtual DOM



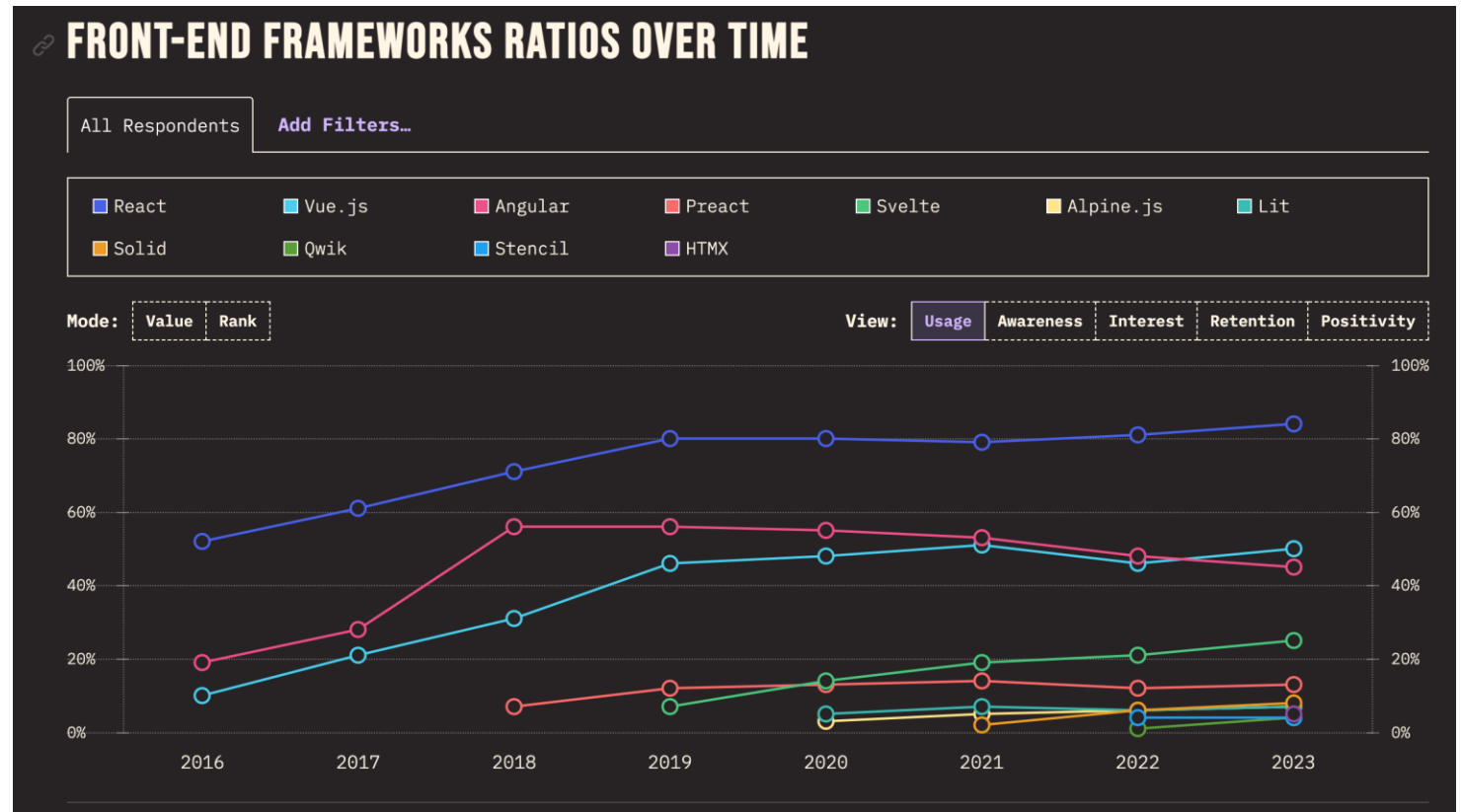
JavaScript Developers



UI-библиотеки и фреймворки

Популярные инструменты:

- React
- Vue
- Angular
- Svelte



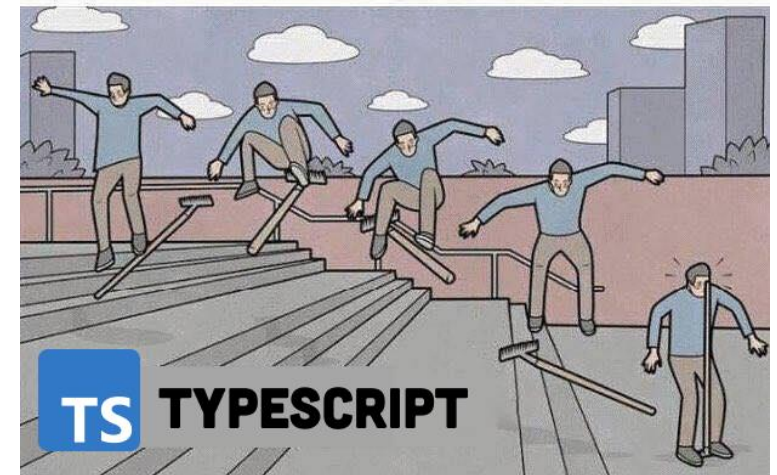
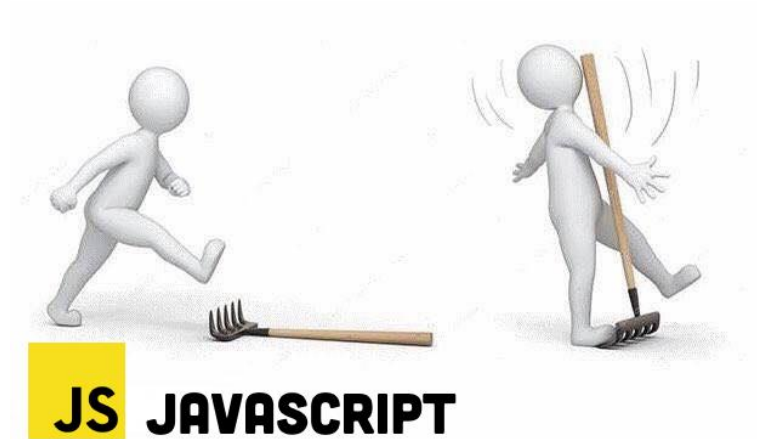
<https://2023.stateofjs.com/en-US/libraries/front-end-frameworks/>

TypeScript

В JavaScript слабая динамическая типизация

- При операции с переменными разных типов они будут автоматически приведены к одному
- Любая переменная может произвольно менять свой тип во время выполнения программы

TypeScript = JavaScript + строгая типизация



(Хотя скорее наоборот)

Системы контроля версий (VCS)

VCS (version control systems) — были придуманы, чтобы контролировать изменения исходного кода и комфортно работать над ним большой командой.

Примеры VCS:

- Git:
 - GitHub
 - GitLab
 - BitBucket
- SVN
- Mercurial
- Другие решения



ЛИНТИНГ

Процесс исследования исходного кода без запуска называют статическим анализом или линтингом, а программу, которая это делает — статическим анализатором или линтером.

Самый популярный линтер для JavaScript — **ESLint**

Самый популярный форматтер для JavaScript — **prettier**

Плюсы линтинга:

- Уменьшается разноречивость в коде
- Уменьшается количество ошибок по неосторожности
- Ревью проходит быстрее
- Код во всей кодовой базе становится одинаковым



Сборка

Бандлер (сборщик) — программа, которая упаковывает сложный проект со многими файлами и внешними зависимостями в один или несколько файлов, которые будут отправлены браузеру.

Популярные сборщики:

- webpack
- esbuild
- parcel

Популярные бойлерплейты:

- create-react-app (cra)
- vite



Тестирование

Тесты делают код более прочным и живучим. Одновременно с этим тесты — это отличная документация, которая не врёт и не устаревает. Также тесты можно использовать как инструмент разработки программы.

Виды тестов:

- Юнит-тесты
- Интеграционные тесты
- e2e-тесты
- Скриншотные тесты



Тестирование

Популярные инструменты для тестирования:

- Jest (юнит, интеграционные и снапшоты)
- MochaJS (юнит)
- Karma (юнит)
- Cypress (интеграционные и e2e)
- Hermione (скриншотные и e2e)
- Playwright (скриншотные и e2e)



No-code решения

No-code-инструменты позволяют обычным пользователям создавать веб-приложения без необходимости написания программного кода (или с минимальной необходимостью – **Low-Code**)

Примеры:

- WordPress/Wix/Webflow/etc
- Notion
- [И некоторые другие](#)

Особенности:

- Быстрый запуск при меньшей гибкости
- Ограниченность рамками доступных шаблонов

No Code

No code is the best way to write secure and reliable applications. Write nothing; deploy nowhere.

Getting Started

Start by not writing any code.

This is just an example application, but imagine it doing anything you want. Adding new features is easy too:

The possibilities are endless.

Полезные материалы

<https://github.com/lyaplyap/frontend-development-course> — данный курс на GitHub

<https://doka.guide/> — про веб-разработку понятным языком от профессиональных разработчиков

<https://www.youtube.com/watch?v=aoK2axqgR7k&list=PLXtiZNKlobF6U1V0yV248MT8ubBMg8iH2> — лекции от школы разработки интерфейсов Яндекса (ШПИ 2024)

<https://htmlbook.ru/> — один из самых толковых ресурсов HTML и CSS

<https://learn.javascript.ru/> — самый подробный учебник по JavaScript с примерами и задачами

<https://developer.mozilla.org/ru/> — документация HTML, CSS, JS и WebAPI от Mozilla