

Серверные компоненты в React

Как мы пишем фронт на бэке и зачем это нужно

Обо мне

Андрей Охотников

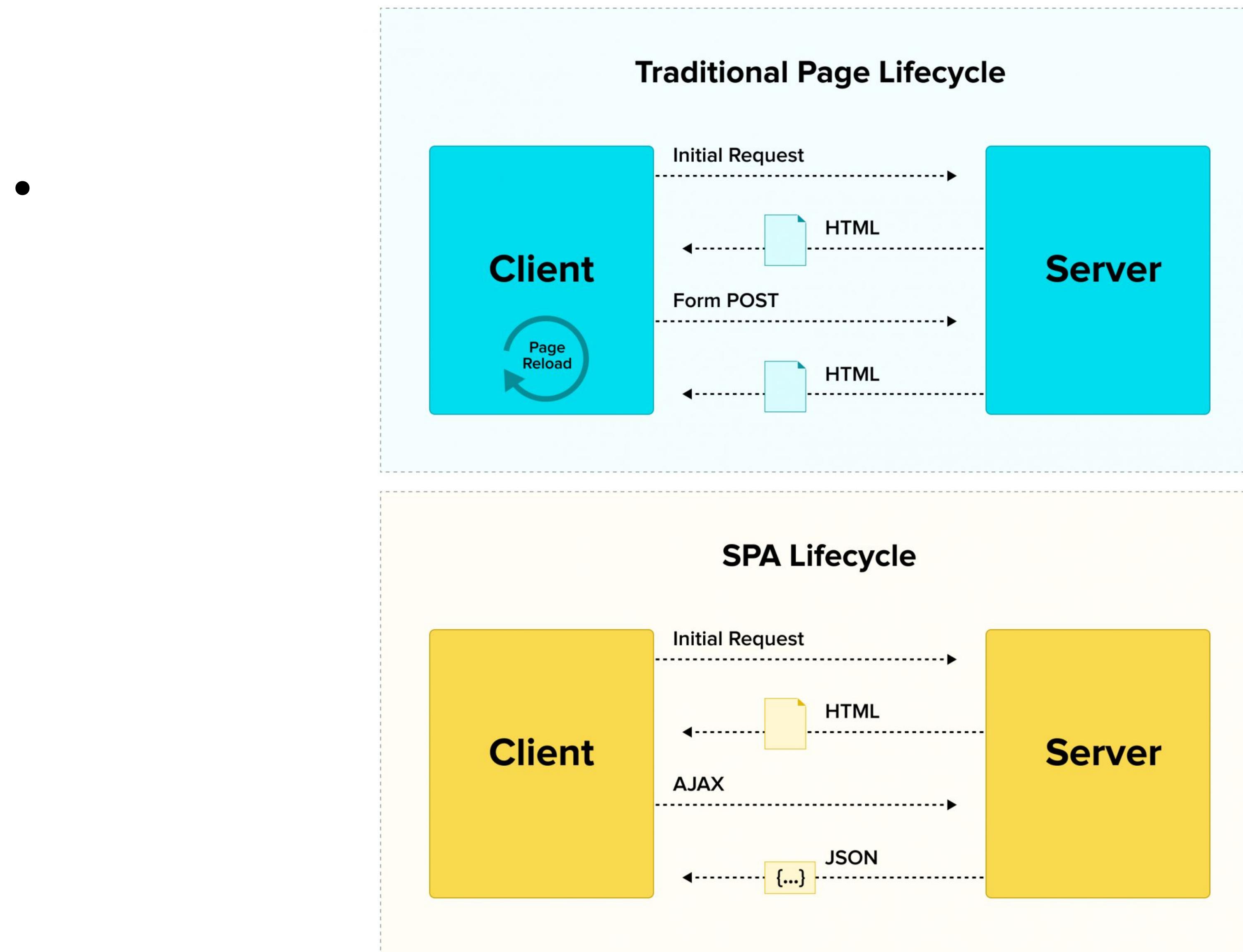
Lead frontend developer

Ведущий фронтэнд разработчик
в подразделении It Befree
Melon Fashion Group

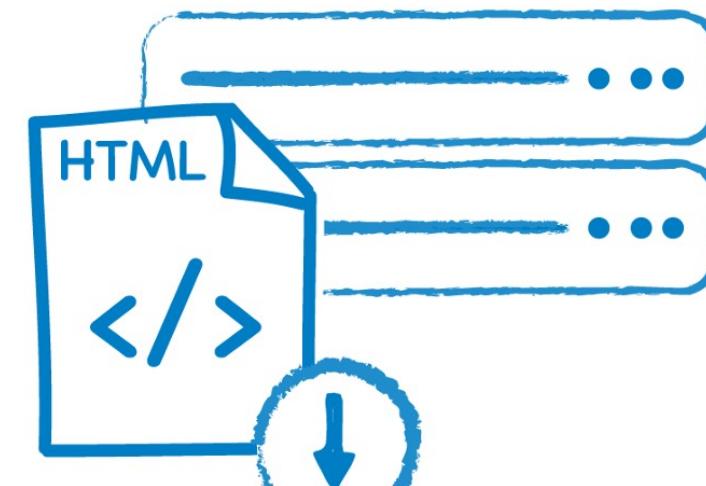
Befree



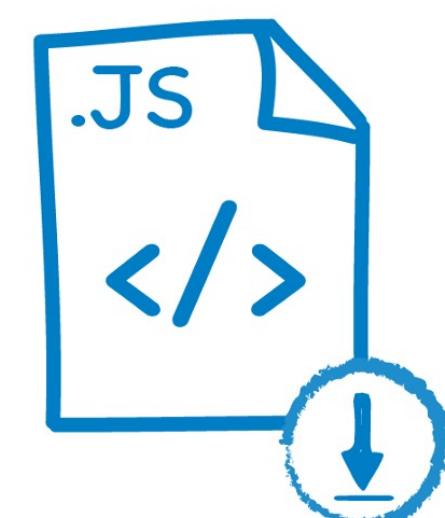
Single Page Application



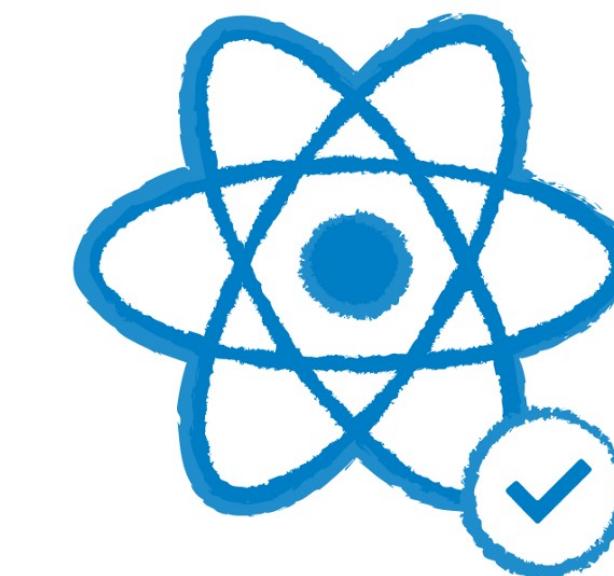
CSR



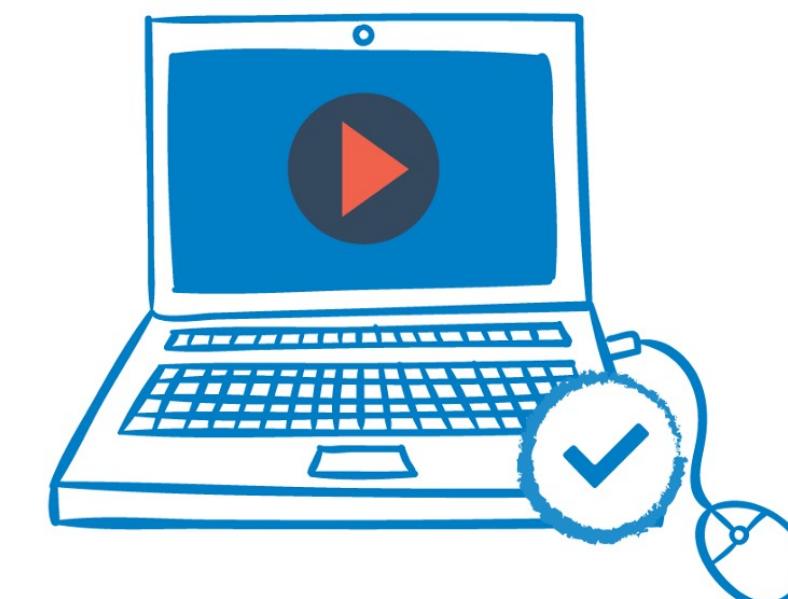
Сервер отправляет
пустой html в
браузер



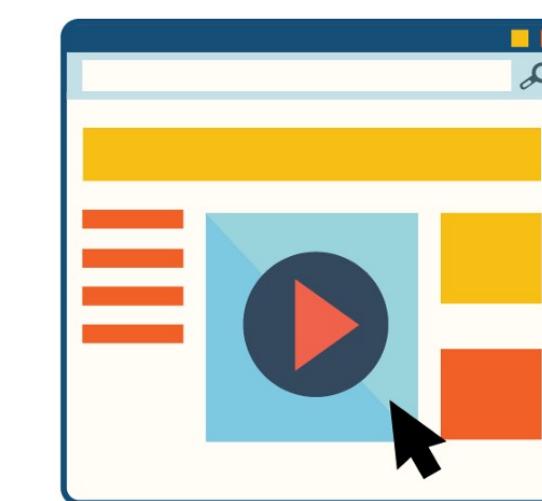
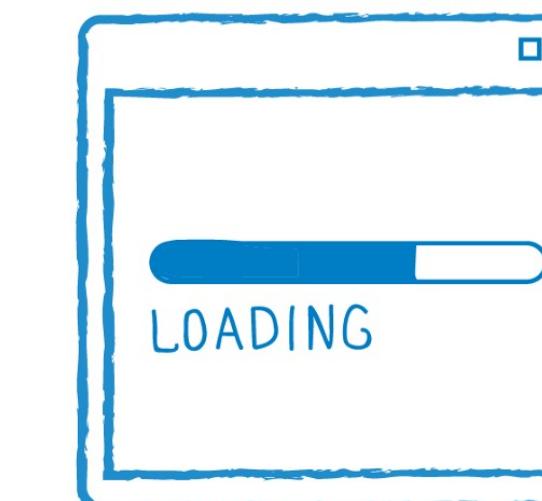
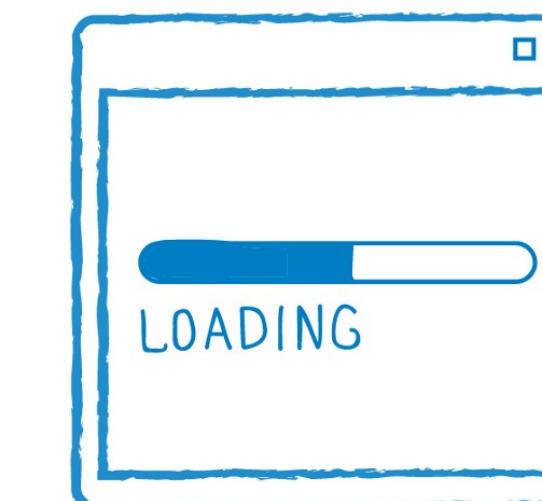
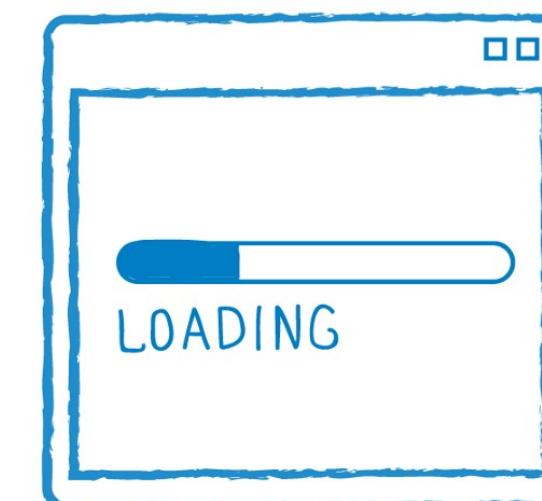
Загружаются js
файлы

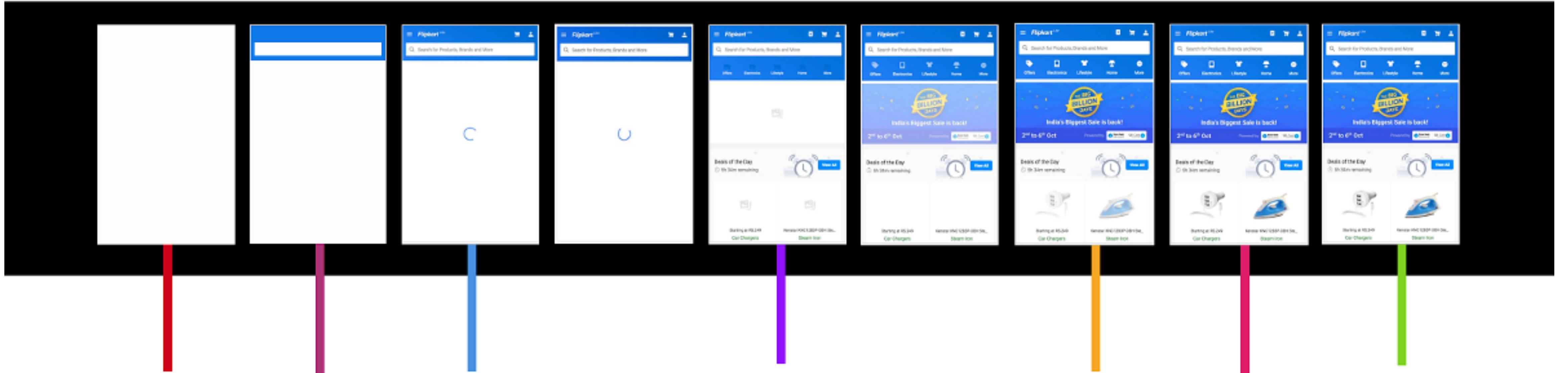


Реакт запускает
рендер страницы
Запросы к апи



Отрисовка
страницы с
данными





Navigation begins

Получение ответа сервера

First Contentful Paint

Отрисовка первого значимого элемента

First Meaningful Paint

Основной контент виден пользователю

First Paint

Рендер первого элемента

Visually ready

Визуальная часть готова

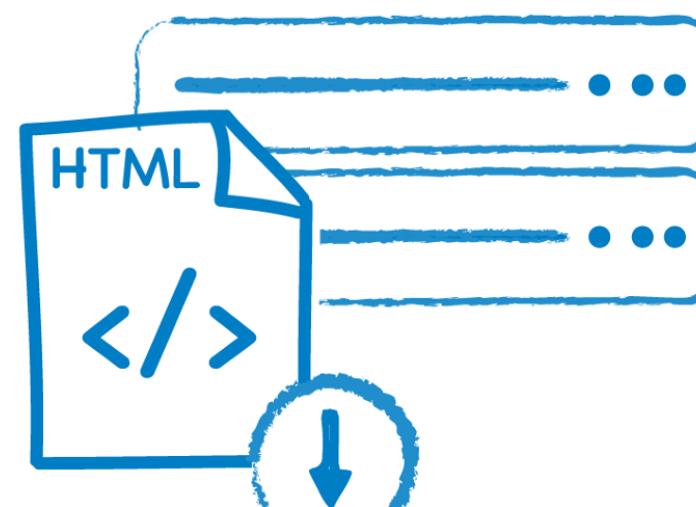
Fully Loaded

Готовность 100%

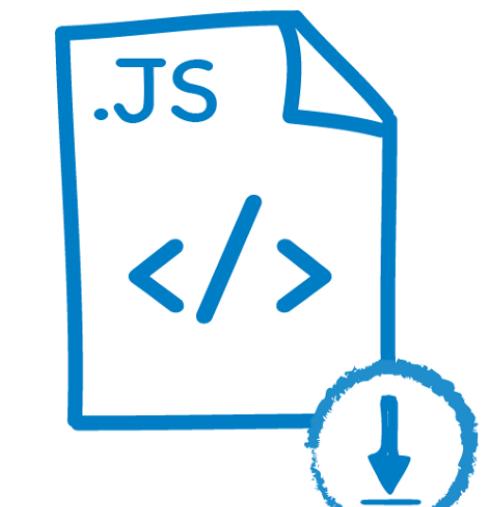
Time to Interactive

Возможность взаимодействовать со страницей

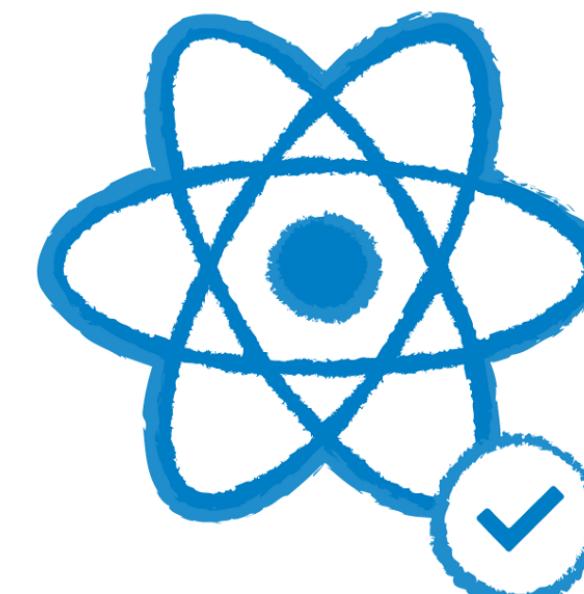
SSR



HTML
генерируется на
сервере



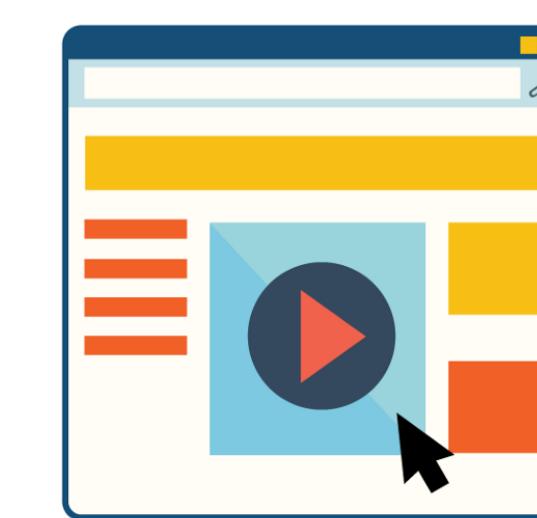
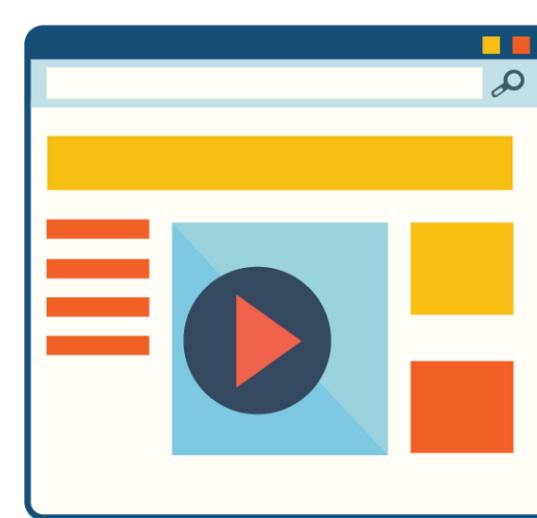
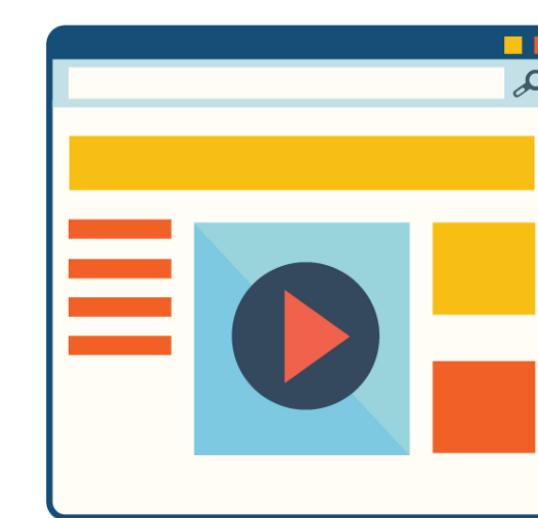
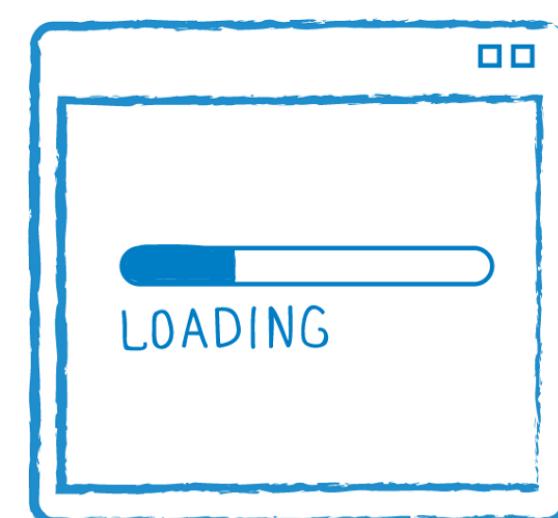
Страница
отрисовывается
Подгружается js



Гидратация
Реакт



Страница
становится
интерактивной



Преимущества SSR

- SEO оптимизация - браузер получает готовую разметку html, поисковые роботы могут проиндексировать содержимое
- Быстрая загрузка страницы - сгенерировать страницу на сервере быстрее, при получении ответа браузер сразурендерит страницу с данными

Варианты рендера

- CSR - Client-side Rendering
- SSR - Server-side Rendering
- SSG - Static Site Generation

События

Bicon 1

08.12.23

Первая конференция ИТ Befree

CSR

Рендеринг на стороне клиента

page.tsx

```
8  export default function EventsCsr() {
9    const [events, setEvents] = useState([])
10
11   const getEvents = async () => {
12     const res = await fetch('/api/events')
13
14     const data = await res.json()
15
16     setEvents(data)
17   }
18
19   useEffect(() => {
20     getEvents()
21   }, [])
22
23   if (!events.length) return <p>loading...</p>
24
25   return (
26     <main className="flex min-h-screen flex-col items-center p-5">
27       <h1 className={`text-4xl font-semibold mb-10`}>События</h1>
28
29       {events.map((item: EventCardType) => (
30         <EventCard key={item.id} event={item} />
31       ))}
32     </main>
33   )
34 }
```

Snipped

SSG

Статическая генерация

```
● ● ● page.tsx

5  export const revalidate = 30
6
7  export default async function EventsIsg() {
8      const events = await getEvents()
9
10     return (
11         <main className="flex min-h-screen flex-col items-center p-5">
12             <h1 className={`text-4xl font-semibold mb-10`}>События</h1>
13
14             {events.map((item: EventCardType) => (
15                 <EventCard key={item.id} event={item} />
16             )))
17         </main>
18     )
19 }
```

Snipped

Bicon 1

Серверные компоненты в React. Как мы пишем фронт на беке и зачем это нужно



Все новое - это хорошо забытое старое!
Фронтенд возвращается на бек, но в новом
обличии! Что это - откат к истокам или
эволюция?

Андрей Охотников
Lead frontend developer

Наши попытки сделать так, чтобы после каждого релиза фронт не разваливался и что из этого вышло



Как перестать беспокоиться и начать
релизить по пятницам? Достаточно всего
лишь каждый раз перед деплоем на прод...

Ольга Ковалева
Frontend developer

SSR

Динамический рендер на стороне сервера

page.tsx

```
29 const Event = async ({ params }: { params: { slug: string } }) => {
30   const event: EventCardType = await getEvent(params.slug)
31
32   if (!event) {
33     return notFound()
34   }
35
36   return (
37     <div>
38       <h1 className={`text-6xl font-semibold text-center py-10`}>
39         {event.title}
40       </h1>
41
42       <div className="grid-cols-12">
43         {event?.reports?.map((item: ReportCardType) => (
44           <ReportCard
45             key={item.id}
46             title={item.title}
47             description={item.description}
48             image={item.author?.photo}
49             author={item.author}
50           />
51         )))
52       </div>
53     </div>
54   )
55 }
56
57 export default Event
```

Snipped

Статический рендер

```
●●● page.tsx

7  export default async function Events() {
8    const events = await getEvents()
9
10   return (
11     <main className="flex min-h-screen flex-col items-center p-5">
12       <h1 className={`text-4xl font-semibold mb-10`}>События</h1>
13
14       {events.map((item: EventCardType) => (
15         <EventCard key={item.id} event={item} />
16       ))}
17     </main>
18   )
19 }
```

Snipped

Route (app)	Size	First Load JS
○ /	182 B	92.3 kB
○ /_not-found	882 B	86 kB
○ /about	157 B	85.3 kB
○ /api/events	0 B	0 B
λ /api/register	0 B	0 B
○ /contacts	157 B	85.3 kB
○ /events	183 B	92.3 kB
○ /events-csr	1.23 kB	93.3 kB
λ /events-csr/[slug]	157 B	85.3 kB
○ /events-isg	183 B	92.3 kB
λ /events-isg/[slug]	157 B	85.3 kB
○ /events-ssg	182 B	92.3 kB
λ /events-ssg/[slug]	156 B	85.3 kB
λ /events/[slug]	157 B	85.3 kB
○ /registration	1.53 kB	86.7 kB
+ First Load JS shared by all	85.1 kB	
- chunks/472-d4f778e96905bbab.js	29.9 kB	
- chunks/fd9d1056-fde9ac1f75f2f6fb.js	53.2 kB	
- chunks/main-app-f26ca3075c4d7493.js	232 B	
- chunks/webpack-d3d26b8cfb49a40d.js	1.76 kB	

Динамический рендер

```
●●● page.tsx

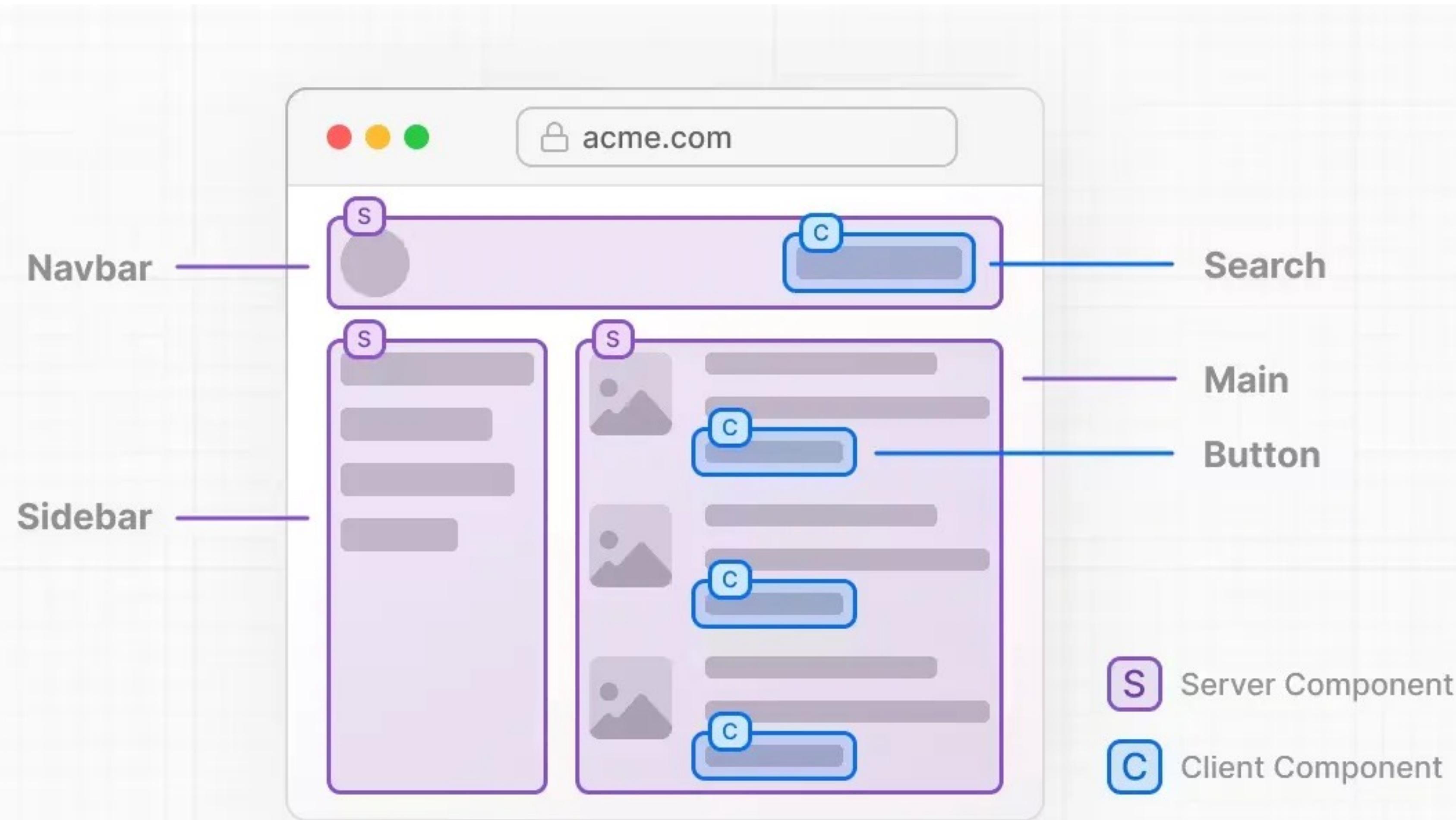
9  export default async function Events() {
10  const cookieStore = cookies()
11  const authToken = cookieStore.get('authToken')
12
13  const events = await getEvents()
14
15  return (
16    <main className="flex min-h-screen flex-col items-center p-5">
17      <h1 className={`text-4xl font-semibold mb-10`}>События</h1>
18
19      {events.map((item: EventCardType) => (
20        <EventCard key={item.id} event={item} />
21      ))}
22    </main>
23  )
24 }
```

Snipped

Route (app)	Size	First Load JS
o /	182 B	92.3 kB
o /_not-found	882 B	86 kB
o /about	157 B	85.3 kB
o /api/events	0 B	0 B
λ /api/register	0 B	0 B
o /contacts	157 B	85.3 kB
λ /events	183 B	92.3 kB
o /events-csr	1.23 kB	93.3 kB
λ /events-csr/[slug]	157 B	85.3 kB
o /events-isg	183 B	92.3 kB
λ /events-isg/[slug]	157 B	85.3 kB
o /events-ssg	182 B	92.3 kB
λ /events-ssg/[slug]	156 B	85.3 kB
λ /events/[slug]	157 B	85.3 kB
o /registration	1.53 kB	86.7 kB
+ First Load JS shared by all	85.1 kB	
chunks/472-d4f778e96905bbab.js	29.9 kB	
chunks/fd9d1056-fde9ac1f75f2f6fb.js	53.2 kB	
chunks/main-app-f26ca3075c4d7493.js	232 B	
chunks/webpack-d3d26b8cfb49a40d.js	1.76 kB	
Route (pages)	Size	First Load JS
[/_app	0 B	83.9 kB
λ /events-ssr-old	661 B	84.6 kB
+ First Load JS shared by all	86.8 kB	
chunks/framework-b7ba9a8e7304c68b.js	45.5 kB	
chunks/main-f6f5368e42766fb8.js	33.1 kB	
chunks/pages/_app-5c633383c22b6ee8.js	3.51 kB	
chunks/webpack-d3d26b8cfb49a40d.js	1.76 kB	
css/24b9094ad6e5e9ed.css	2.95 kB	
o (Static) prerendered as static HTML		
λ (Dynamic) server-rendered on demand using Node.js		

**А причем тут вообще
серверные компоненты?**

Клиентские и серверные компоненты



Серверные компоненты

- Получение данных
- Безопасность
- Кеширование
- Размер бандла
- Скорость загрузки страницы
- Сео оптимизация
- Стриминг

Клиентские компоненты

- Интерактивность
- Состояния
- Работа с браузерным апи

Регистрация на Bicon

Имя:



Email:

Зарегистрироваться

```
'use client'

import { ChangeEvent, useState } from 'react' 4.1k (gzipped: 1.8k)
import { registerUser } from '@/utils/register'

const RegistrationForm = () => {
  const [formData, setFormData] = useState({ ... })
}

const handleChange = (e: ChangeEvent<HTMLInputElement>) => { ... }

const responseHandler = (status: boolean) => { ... }

const handleSubmit = async () => {
  const { status } = await registerUser(formData)

  responseHandler(status === 'ok')
}

return (
  <div>
    <form action={handleSubmit}>
      <div className="mb-4">...
    </div>
  </form>
)
```

```
page.tsx

1 import RegistrationForm from './Form'
2
3 const Registration = async () => {
4   return (
5     <div className="bg-white p-8 rounded shadow-lg max-w-md m-auto mt-5">
6       <h1 className="text-2xl font-bold mb-6 text-center">
7         Регистрация на Bicon
8       </h1>
9       <RegistrationForm />
10    </div>
11  )
12}
13
14 export default Registration

Snipped
```

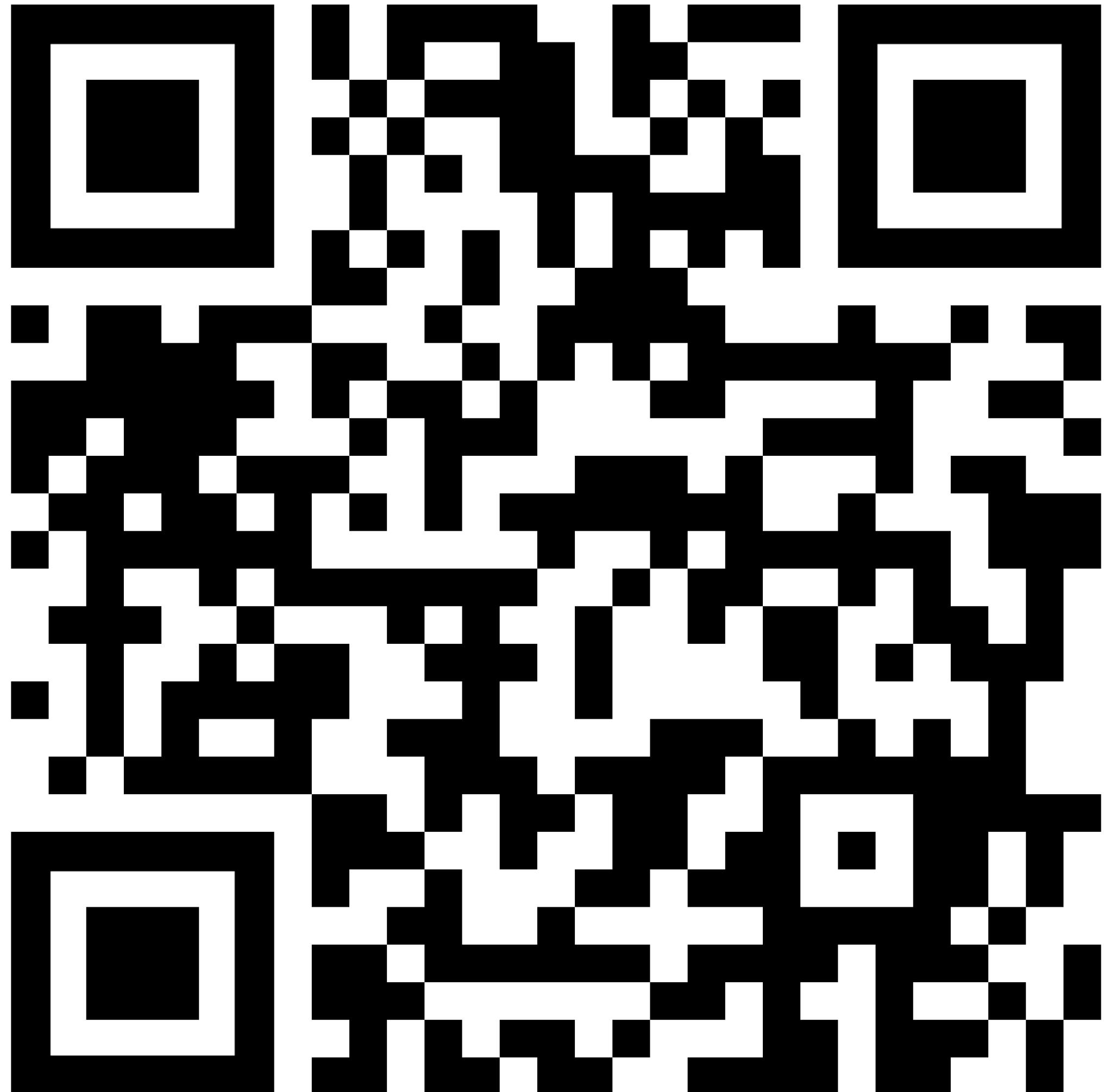
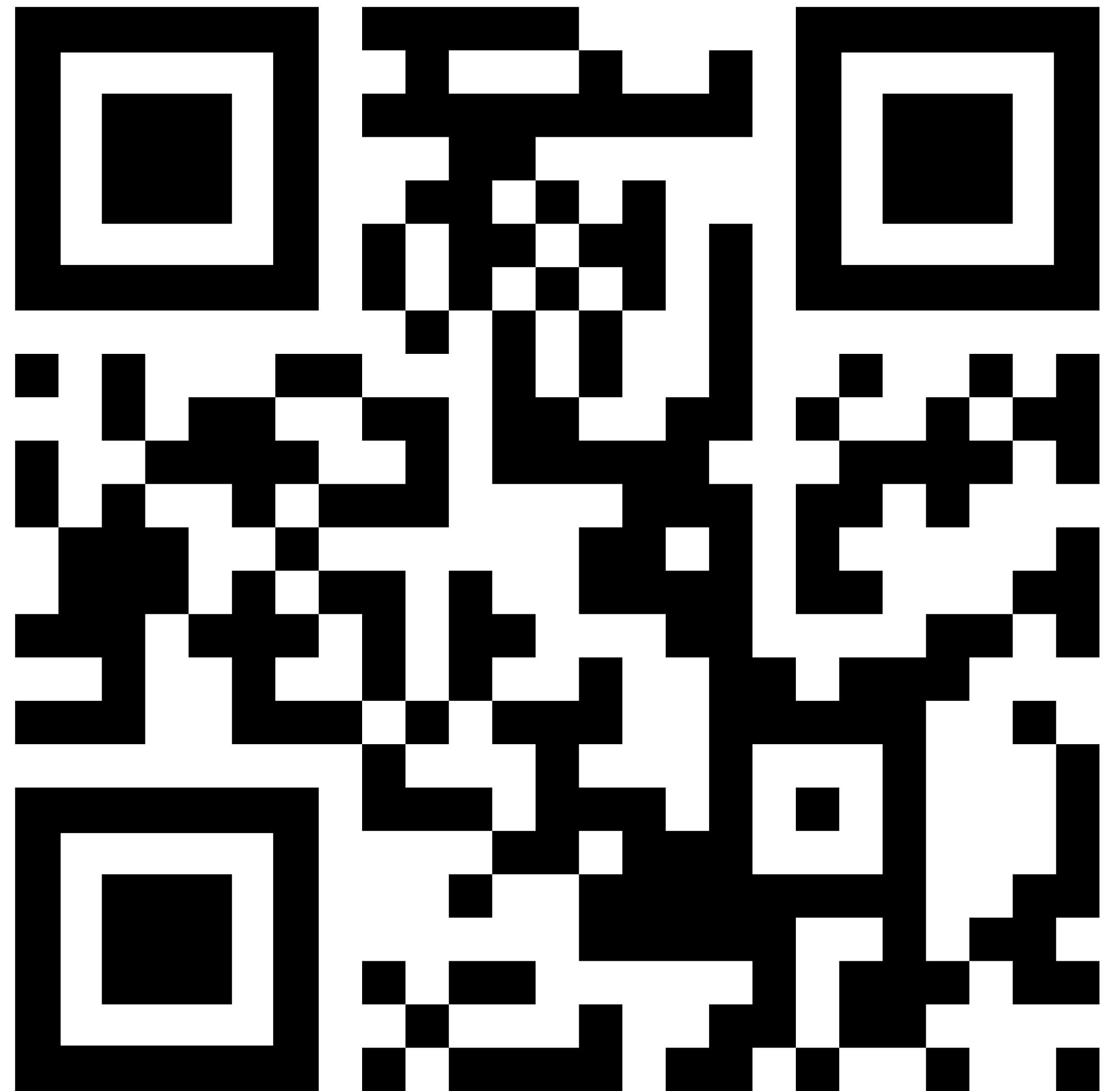
```
register.ts

1 'use server'
2
3 export const registerUser = async ({ ...
4   name,
5   email,
6 }: {
7   name: string
8   email: string
9 }) => {
10   // Здесь логика регистрации – запросы к апи / базе данных
11
12   return { status: 'ok' }
13 }

Snipped
```

Выводы и рекомендации

- Фронт возвращается на бэк, пользуйтесь бэком для ускорения фронта
- Не увлекайтесь бизнес логикой в серверных компонентах



Telegram