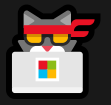
→ Accessing APIs

instructor
@alibek.cs

¡n!

nfactorial incubator
lecture 9

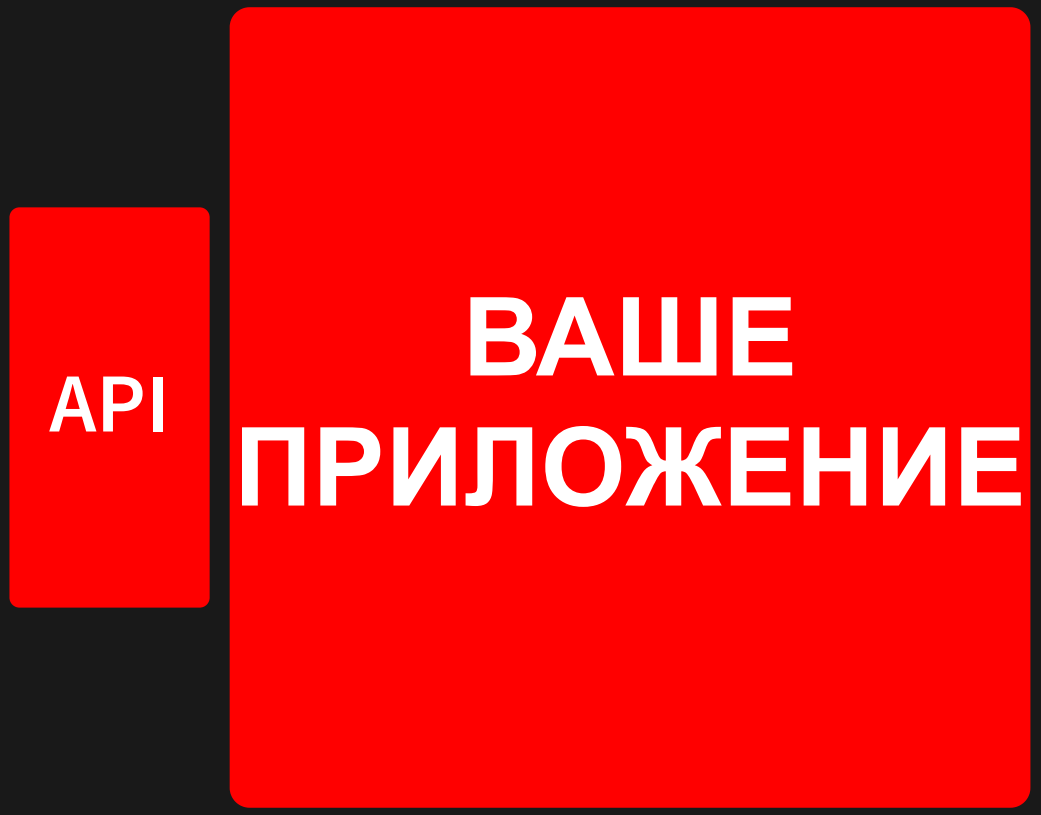
LESSON OBJECTIVES 

- API
- Axios
- HTTP requests and responses
- Real-life examples
- Homework
- Study resources

ЧТО ТАКОЕ API? 🤔

API (Application Programming Interface) — это набор правил и протоколов, позволяющих приложениям взаимодействовать друг с другом.

ВАШЕ
ПРИЛОЖЕНИЕ


API
ВАШЕ
ПРИЛОЖЕНИЕ

ЧТО МЫ МОЖЕМ ДЕЛАТЬ С ИНФОРМАЦИЕЙ? 🤔

CREATE

POST request

READ

GET request

UPDATE

PUT/PATCH request

DELETE

DELETE request

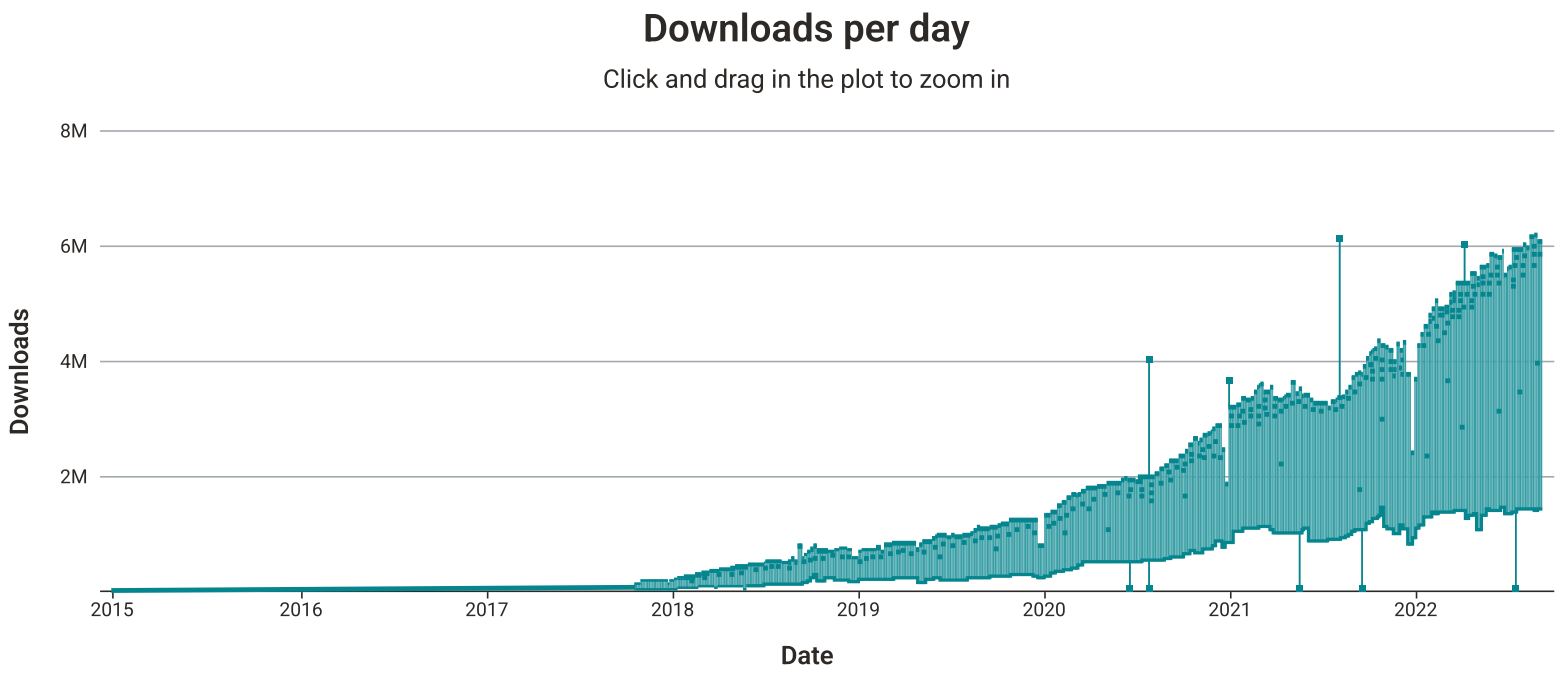
Frontend

API

Backend


JSON
Backend
Frontend

✨AXIOS✨ И НАСКОЛЬКО ОН ПОПУЛЯРНЫЙ



codilime

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1 import axios from 'axios' //import request library
2 const url = 'https://jsonplaceholder.cypress.io/posts/'
3 const response = await axios.get(url, {
4   params: {
5     _limit: limit, //выбираем сколько постов нам нужно
6     _page: page //выбираем конкретную страницу
7   }
8 })
```

Получаем список всех постов с Free Fake REST API

GET REQUEST EXAMPLE

```
1  import axios from 'axios'
2  const API_KEY = process.env.SECRET_API_KEY
3  const url = 'https://jsonplaceholder.cypress.io/posts/'
4  const response = await axios.get(url, {
5    params: {
6      _limit: limit,
7      _page: page
8    },
9    headers: {
10     Authorization: API_KEY
11   }
12 })
```

Если сервис не бесплатный нужно указывать headers

GET REQUEST REAL-WORLD EXAMPLE

```
1 export const maximumNumberOfSulpakPages = async (url) => {
2   const html = (await axios.get(url)).data
3   const $ = cheerio.load(html)
4   const pageNumber = parseInt($('.pages-list', html).childre
5   if (isNaN(pageNumber)) {
6     return 1;
7   }
8   return pageNumber
9 }
```

Получаем HTML страницы и парсим данные через cheerio.js

POST REQUEST EXAMPLE

```
1  const url = 'https://api.openai.com/v1/engines/davinci-code
2  const openAiKey = process.env.OPENAI_API_KEY
3  const postData = {
4    prompt: 'Write me a song about how cool Dalida is',
5    max_tokens: 50
6  }
7  const response = await axios.post(url, postData, {
8      headers: {
9        Authorization: openAiKey
10     }
11   }
12 )
13 console.log(response.data.choices[0].text)
```

При GET нельзя указывать body запроса, поэтому используется POST

PUT REQUEST EXAMPLE

```
1 const url = 'https://api.example.com/users/1';
2 const newData = {
3   name: 'John Doe',
4   email: 'john.doe@example.com'
5 }
6 const response = await axios.put(url, newData)
7
8 console.log(response.data); // HTTP status 204 with empty da
```

PUT request to update a user's data

DELETE REQUEST EXAMPLE

```
1 const url = 'https://api.example.com/users/1';
2 const response = await axios.delete(url);
```

Sends a DELETE request to delete a user.

USESTATE + USEEFFECT + AXIOS = TEMA

```
1  const [data, setData] = useState(null);
2
3  const fetchData = async () => {
4    try {
5      const response = await axios.get(API_URL);
6      setData(response.data);
7    } catch (error) {
8      alert(error.message)
9    }
10 }
11 useEffect(() => {
12   fetchData();
13 }, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
1  const [data, setData] = useState(null);
2
3  const fetchData = async () => {
4    try {
5      const response = await axios.get(API_URL);
6      setData(response.data);
7    } catch (error) {
8      alert(error.message)
9    }
10 }
11 useEffect(() => {
12   fetchData();
13 }, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
const [data, setData] = useState(null);

const fetchData = async () => {
  try {
    const response = await axios.get(API_URL);
    setData(response.data);
  } catch (error) {
    alert(error.message)
  }
}
useEffect(() => {
  fetchData();
}, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
const [data, setData] = useState(null);

const fetchData = async () => {
  try {
    const response = await axios.get(API_URL);
    setData(response.data);
  } catch (error) {
    alert(error.message)
  }
}
useEffect(() => {
  fetchData();
}, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
1  const [data, setData] = useState(null);
2
3  const fetchData = async () => {
4    try {
5      const response = await axios.get(API_URL);
6      setData(response.data);
7    } catch (error) {
8      alert(error.message)
9    }
10 }
11 useEffect(() => {
12   fetchData();
13 }, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
const [data, setData] = useState(null);

const fetchData = async () => {
  try {
    const response = await axios.get(API_URL);
    setData(response.data);
  } catch (error) {
    alert(error.message)
  }
}
useEffect(() => {
  fetchData();
}, []);
```

Отправляем request, сохраняем response, радуемся жизни

USESTATE + USEEFFECT + AXIOS = TEMA

```
1  const [data, setData] = useState(null);
2
3  const fetchData = async () => {
4    try {
5      const response = await axios.get(API_URL);
6      setData(response.data);
7    } catch (error) {
8      alert(error.message)
9    }
10 }
11 useEffect(() => {
12   fetchData();
13 }, []);
```

Отправляем request, сохраняем response, радуемся жизни

REACT REAL-LIFE EXAMPLE

```
1  const [action, setAction] = useState("Idle");
2  const [songs, setSongs] = useState([])
3  const [playing, setPlaying, setAudio] = useAudio(audioUrl);
4  const fetchSongs = async () => {
5    const response = await SongService.getSongs()
6    setSongs(response.data)
7  }
8  useEffect(()=>{
9    fetchSongs()
10   if(playing === true) setAction('twerkingdance')
11 }, [])
```

Получаем музыку, сохраняем в стейт, наслаждаемся флексом Кайрата Нуртаса

REACT REAL-LIFE EXAMPLE

```
1  const [action, setAction] = useState("Idle");
2  const [songs, setSongs] = useState([])
3  const [playing, setPlaying, setAudio] = useAudio(audioUrl);
4  const fetchSongs = async () => {
5    const response = await SongService.getSongs()
6    setSongs(response.data)
7  }
8  useEffect(()=>{
9    fetchSongs()
10   if(playing === true) setAction('twerkingdance')
11 }, [])
```

Получаем музыку, сохраняем в стейт, наслаждаемся флексом Кайрата Нуртаса

REACT REAL-LIFE EXAMPLE

```
1  const [action, setAction] = useState("Idle");
2  const [songs, setSongs] = useState([])
3  const [playing, setPlaying, setAudio] = useAudio(audioUrl);
4  const fetchSongs = async () => {
5    const response = await SongService.getSongs()
6    setSongs(response.data)
7  }
8  useEffect(()=>{
9    fetchSongs()
10   if(playing === true) setAction('twerkingdance')
11 }, [])
```

Получаем музыку, сохраняем в стейт, наслаждаемся флексом Кайрата Нуртаса

REACT REAL-LIFE EXAMPLE

```
1  const [action, setAction] = useState("Idle");
2  const [songs, setSongs] = useState([])
3  const [playing, setPlaying, setAudio] = useAudio(audioUrl);
4  const fetchSongs = async () => {
5    const response = await SongService.getSongs()
6    setSongs(response.data)
7  }
8  useEffect(()=>{
9    fetchSongs()
10   if(playing === true) setAction('twerkingdance')
11 }, [])
```

Получаем музыку, сохраняем в стейт,
наслаждаемся флексом Кайрата Нуртаса

REACT REAL-LIFE EXAMPLE

```
1  const [action, setAction] = useState("Idle");
2  const [songs, setSongs] = useState([])
3  const [playing, setPlaying, setAudio] = useAudio(audioUrl);
4  const fetchSongs = async () => {
5    const response = await SongService.getSongs()
6    setSongs(response.data)
7  }
8  useEffect(()=>{
9    fetchSongs()
10   if(playing === true) setAction('twerkingdance')
11 }, [])
```

Получаем музыку, сохраняем в стейт, наслаждаемся флексом Кайрата Нуртаса

HOMEWORK :(

```
1  const TODO_API = 'https://api.todoist.com/rest/v1/tasks'
2  const TODOIST_TOKEN = process.env.TODOIST_TOKEN
3  const PROJECT_ID = 'your project id'
4  class TaskService {
5    static async getAllTasks(){}
6    static async getTaskById(id){}
7    static async createNewTask(task){}
8    static async updateNewTask(id, task){}
9    static async deleteNewTask(id){}
10 }
11 export default TaskService;
```

УРОВЕНЬ: КРАСАУЧИК

Реализовать CRUD-service по https://todoist.com/

УРОВЕНЬ: БЕСПРЕДЕЛЬЩИК

Переписать localStorage-based todo-app на TaskService

УРОВЕНЬ: ХОДЯЧАЯ МАШИНА

Написать ProjectService для создания коллекций тудушек

Ulbi TV

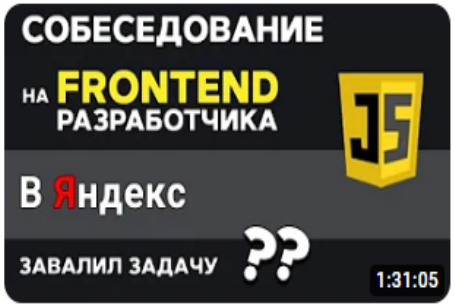
@UlbiTV 221 тыс. подписчиков 150 видео

Привет друзья!) Меня зовут Ульби Тимур. Я fullstack разработчик) >

Вы подписаны

ГЛАВНАЯ | ВИДЕО | SHORTS | ПЛЕЙЛИСТЫ | СООБЩЕСТВО | КАНАЛЫ | О КАНАЛЕ

Новые | Популярные

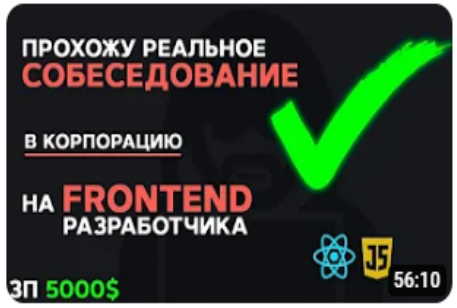


Прохожу собеседование на FRONTEND Разработчика в Яндекс. Как решать...

64 тыс. просмотров • 12 дней назад

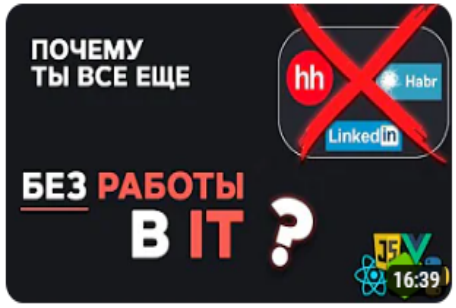
Продвинутый Frontend. В Production на React. Обновленный финальный курс

31 тыс. просмотров • 1 месяц назад



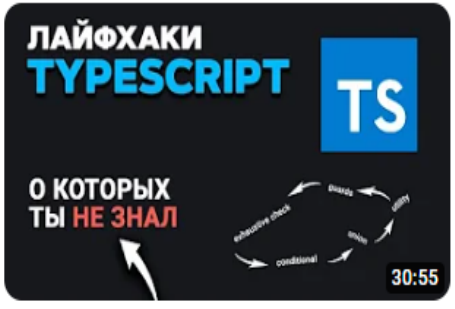
РЕАЛЬНОЕ СОБЕСЕДОВАНИЕ НА FRONTEND РАЗРАБОТЧИКА В...

106 тыс. просмотров • 1 месяц назад



Почему ты НЕ НАЙДЕШЬ работу в IT? Что ты делаешь не так?

127 тыс. просмотров • 2 месяца назад



Фишки TypeScript о которых ТЫ НЕ ЗНАЛ!

57 тыс. просмотров • 2 месяца назад

Что с IT в 2023? Про работу можно забыть? Анализ IT рынка

Event Loop от А до Я. Архитектура браузера и Node JS. Движки и рендер....

Zustand и React query. State management в React без боли

✨ СПАСИБО ЗА ВНИМАНИЕ ✨