Практическая работа №14

Глубокое изучение Fetch API

Цель: Закрепить навыки работы с Fetch API через разнообразные практические задания.

Задание 1

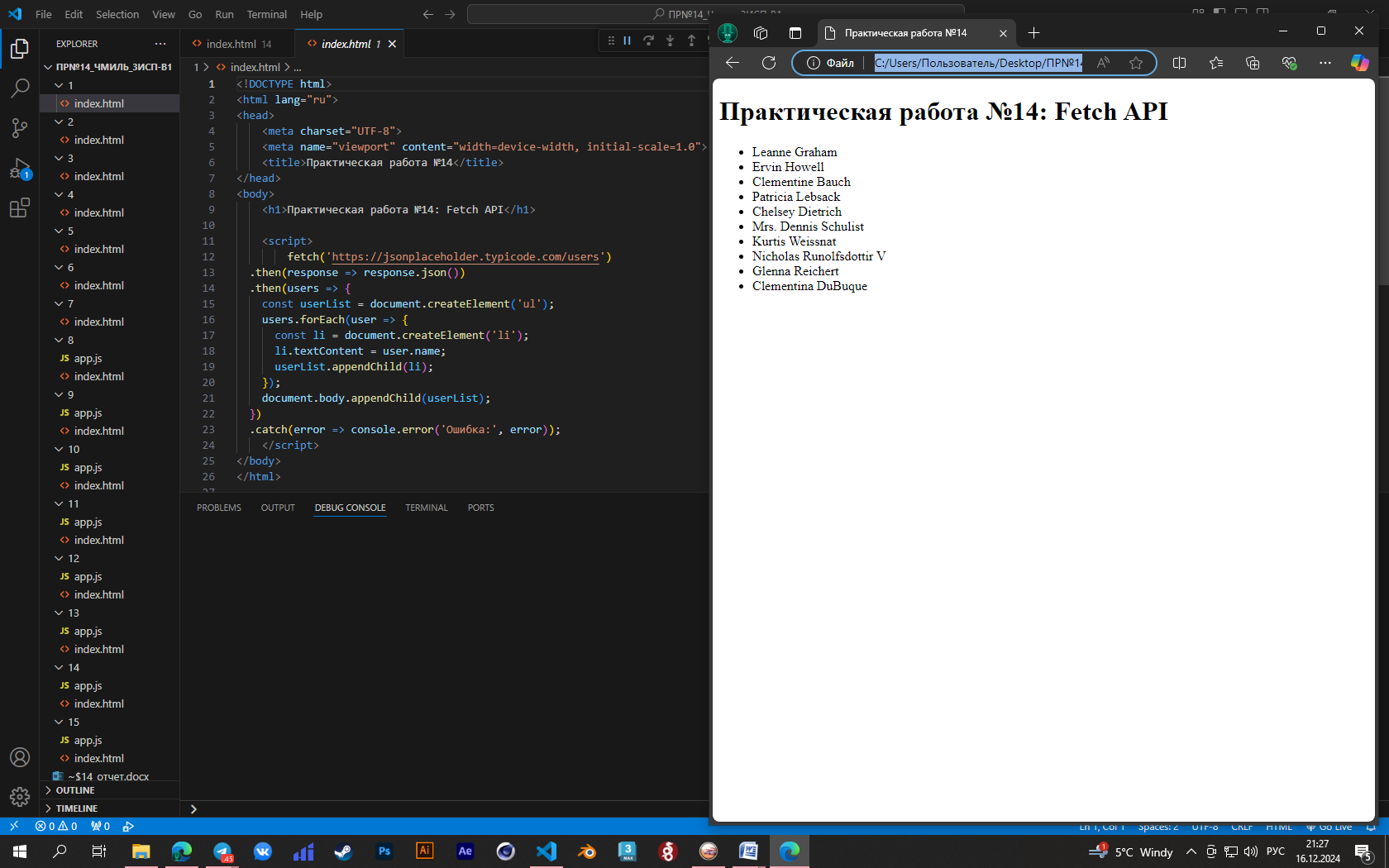
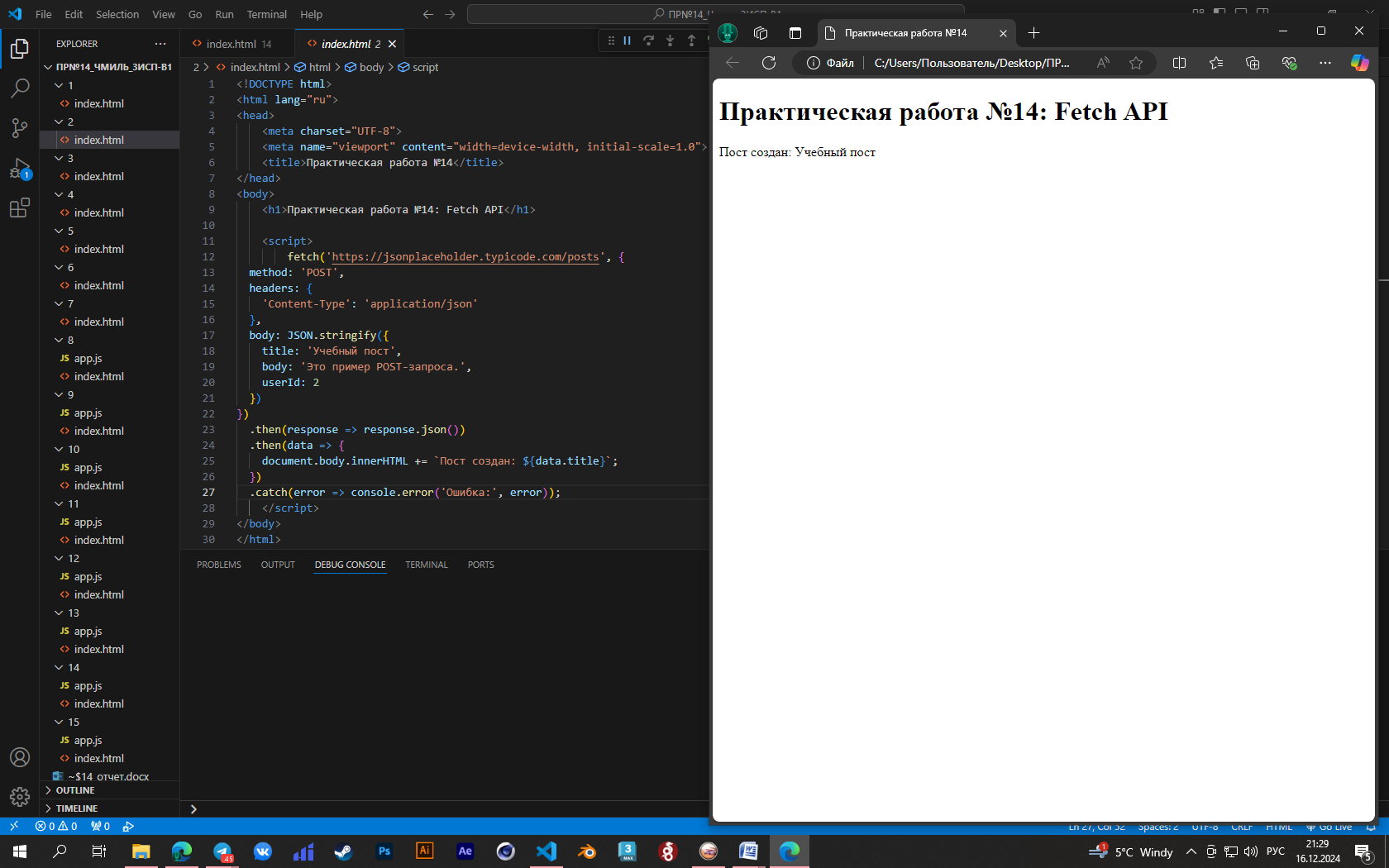
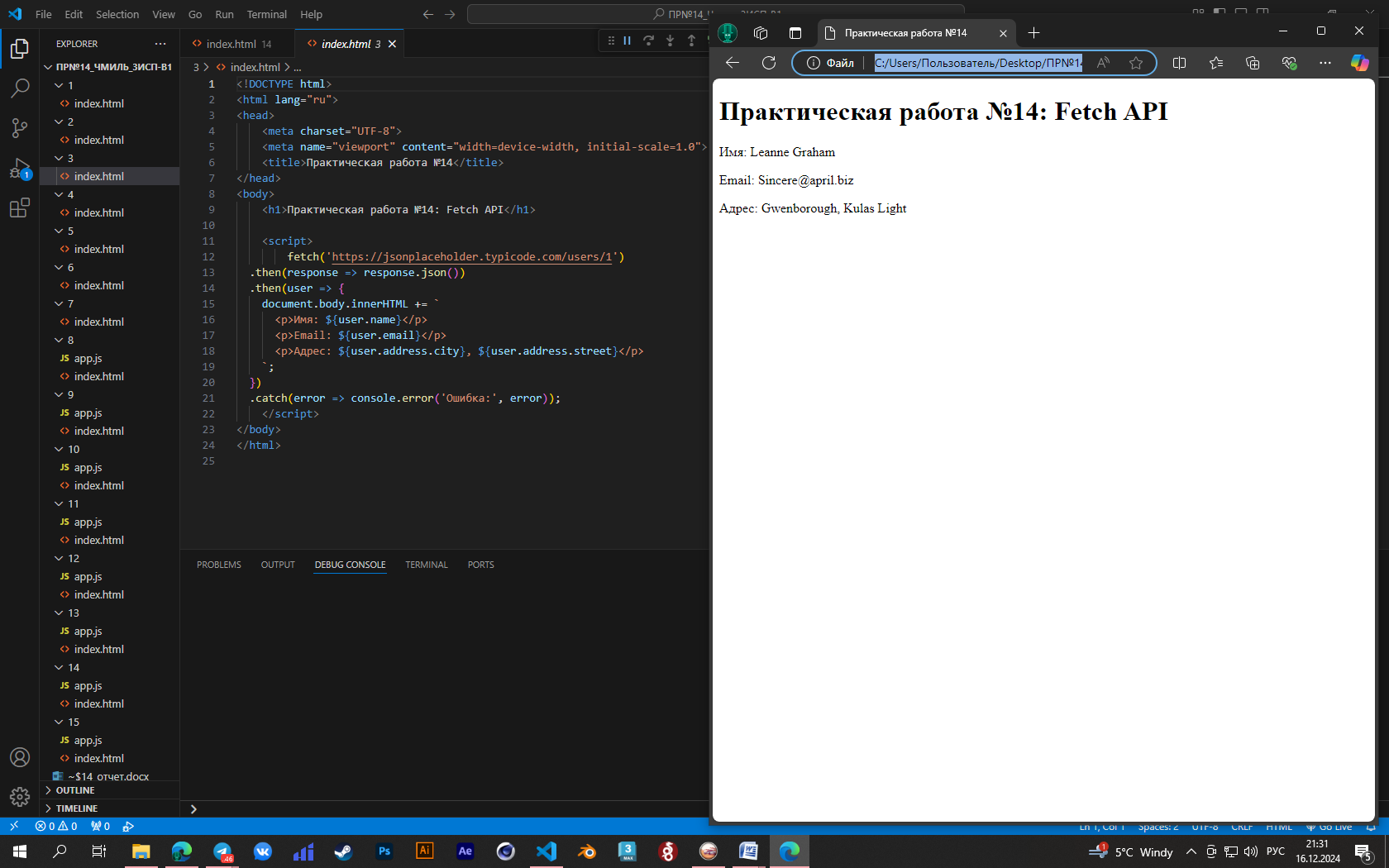


Рисунок 1 – Код

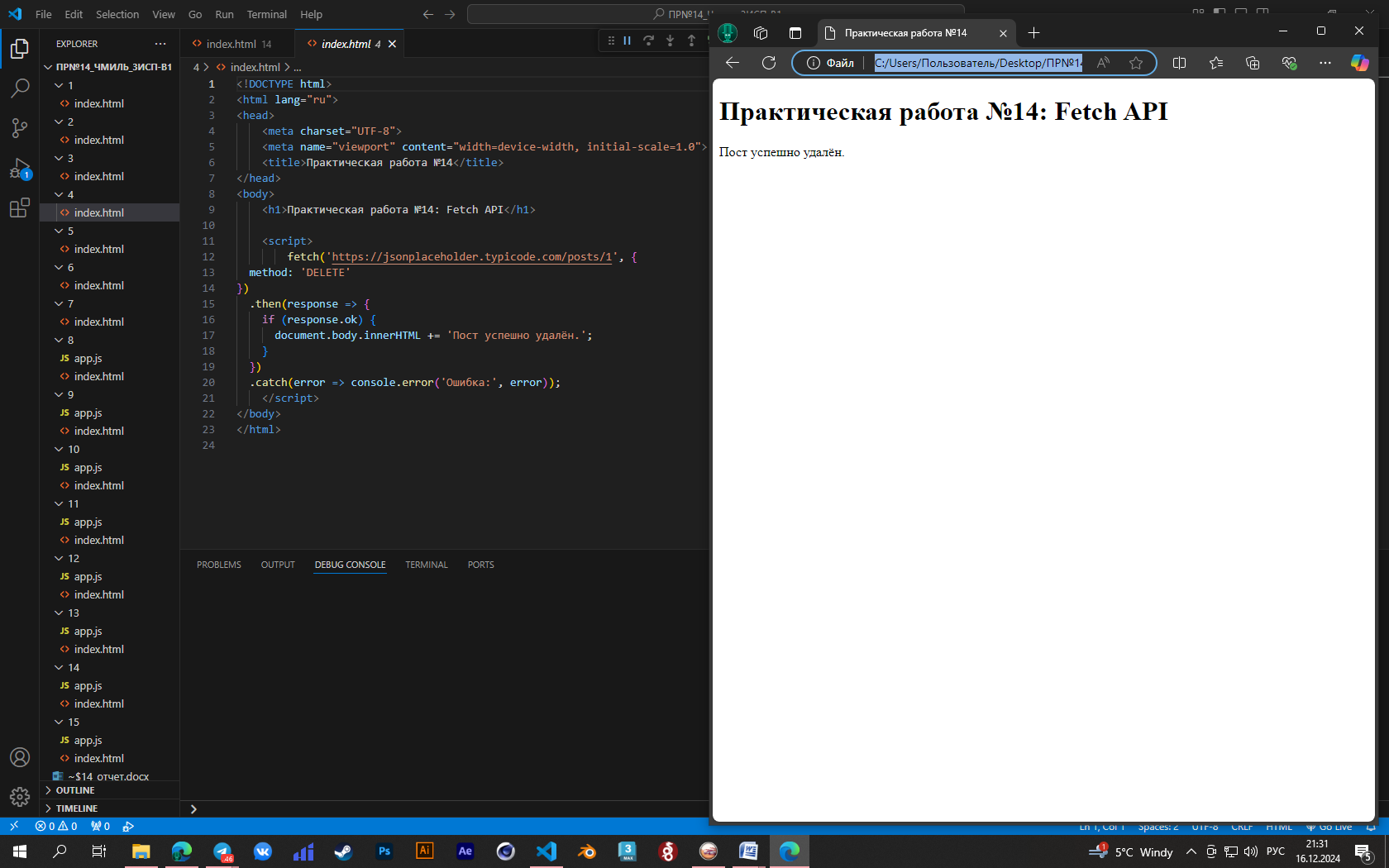
Задание 2



Задание 3



Задание 4



Задание 5

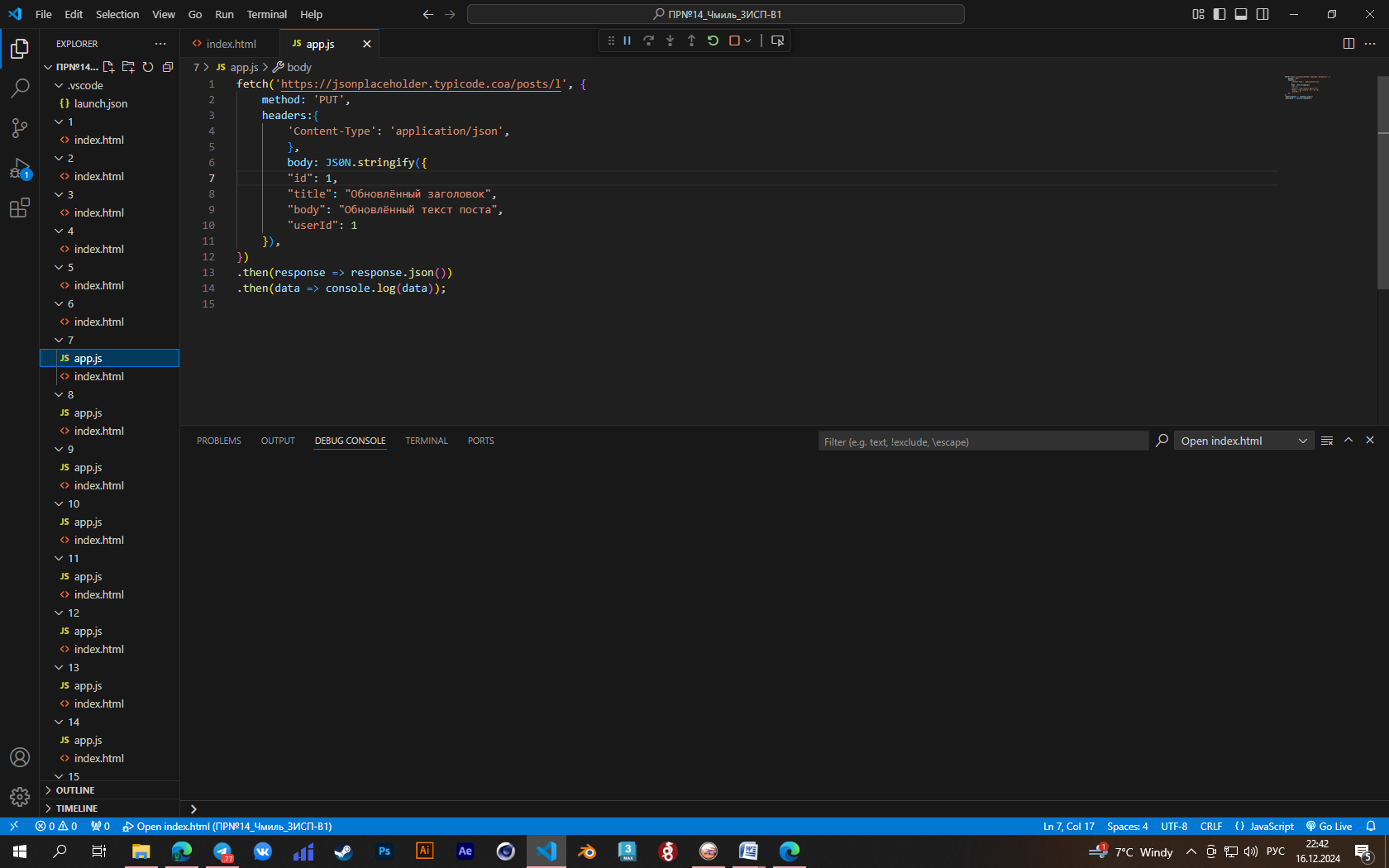
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

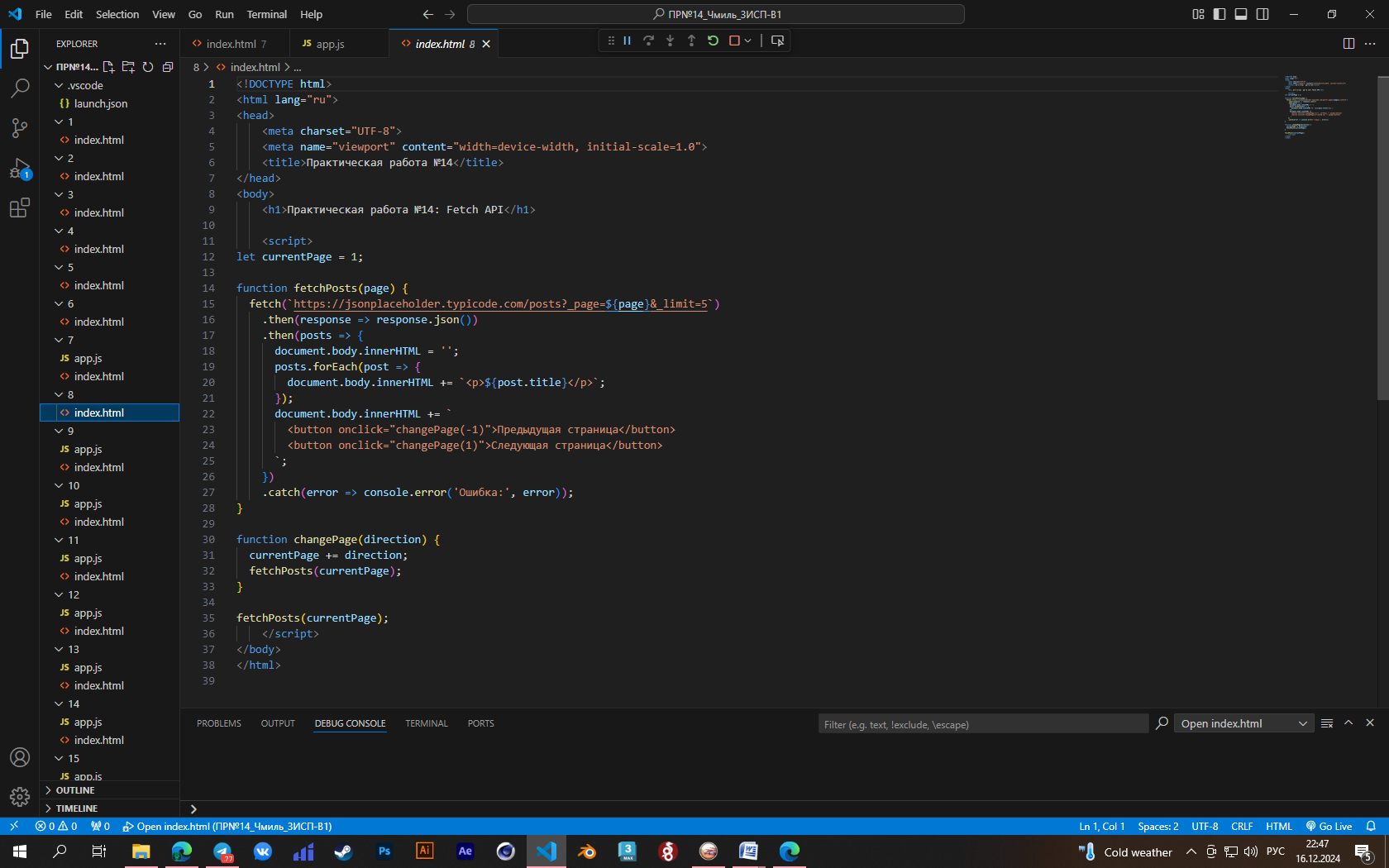
Задание 7



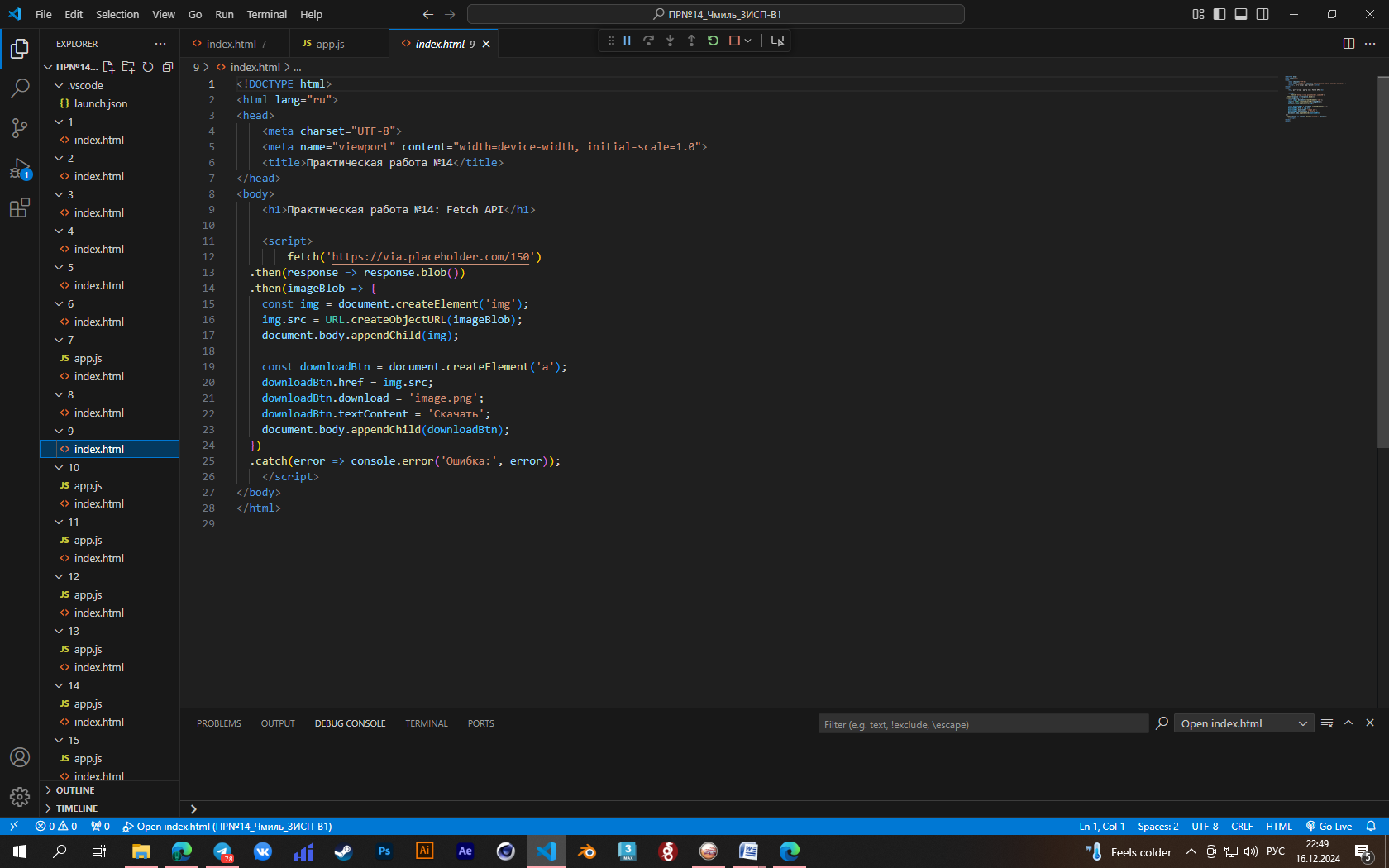
Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

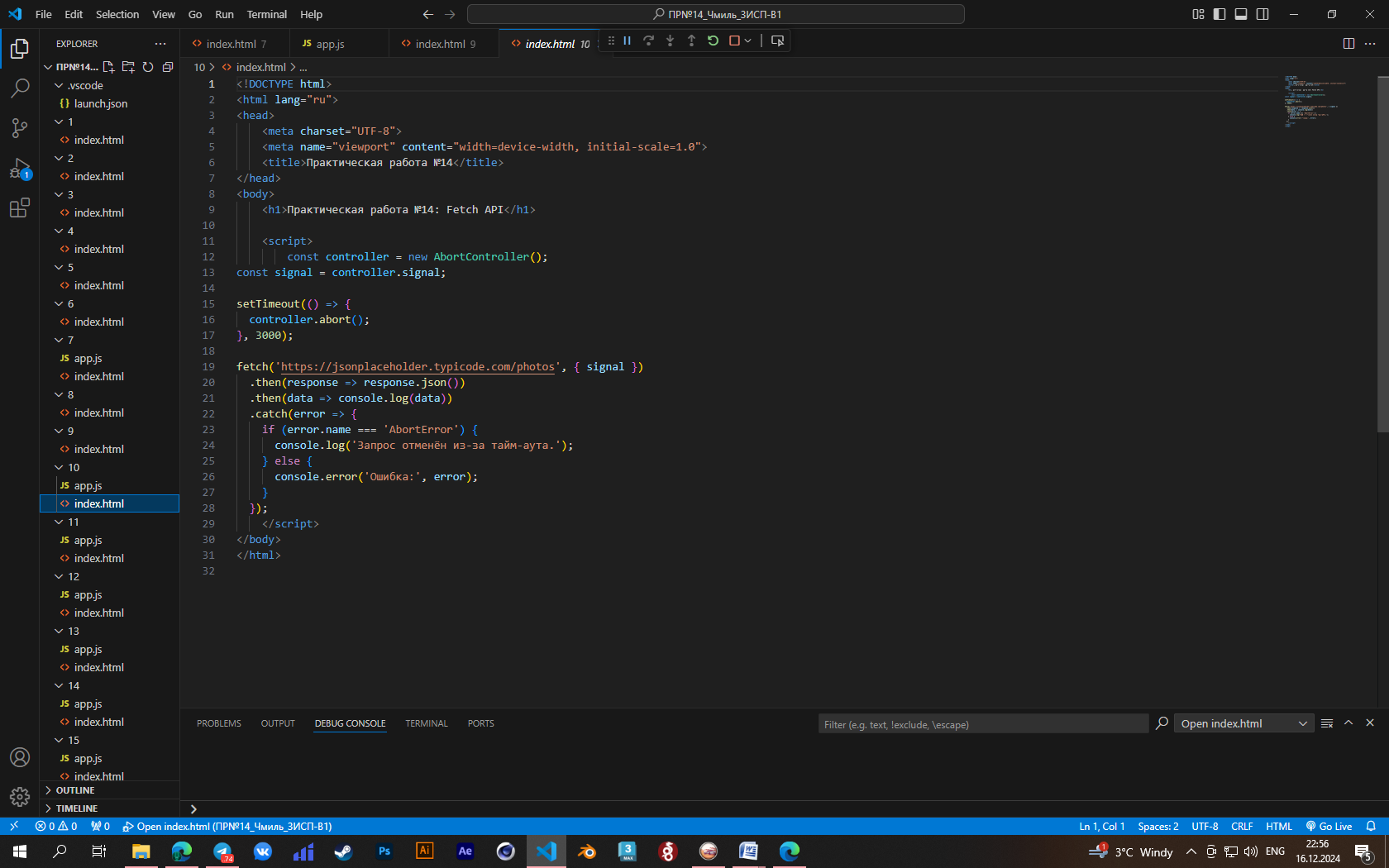
Задание 8



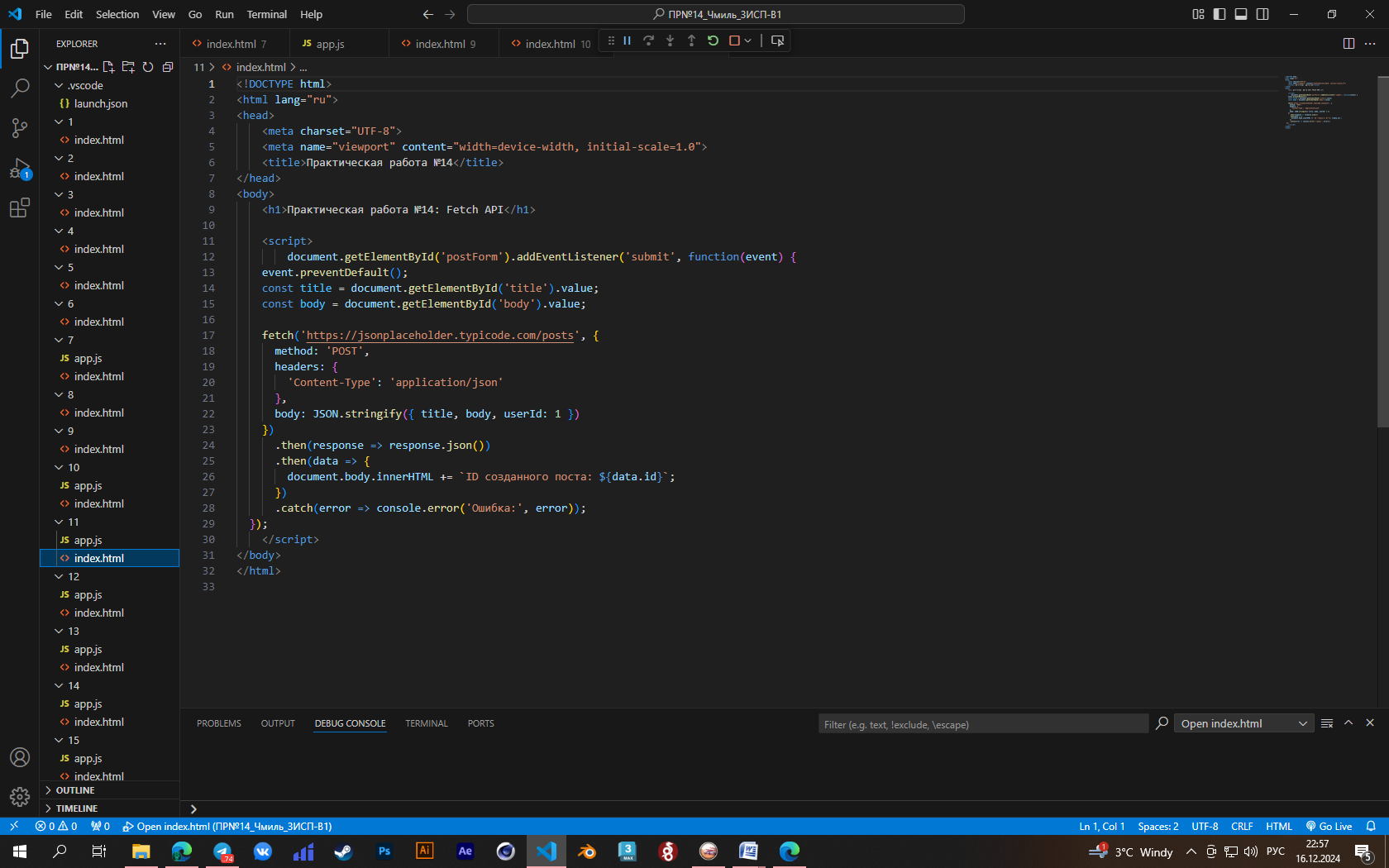
Задание 9



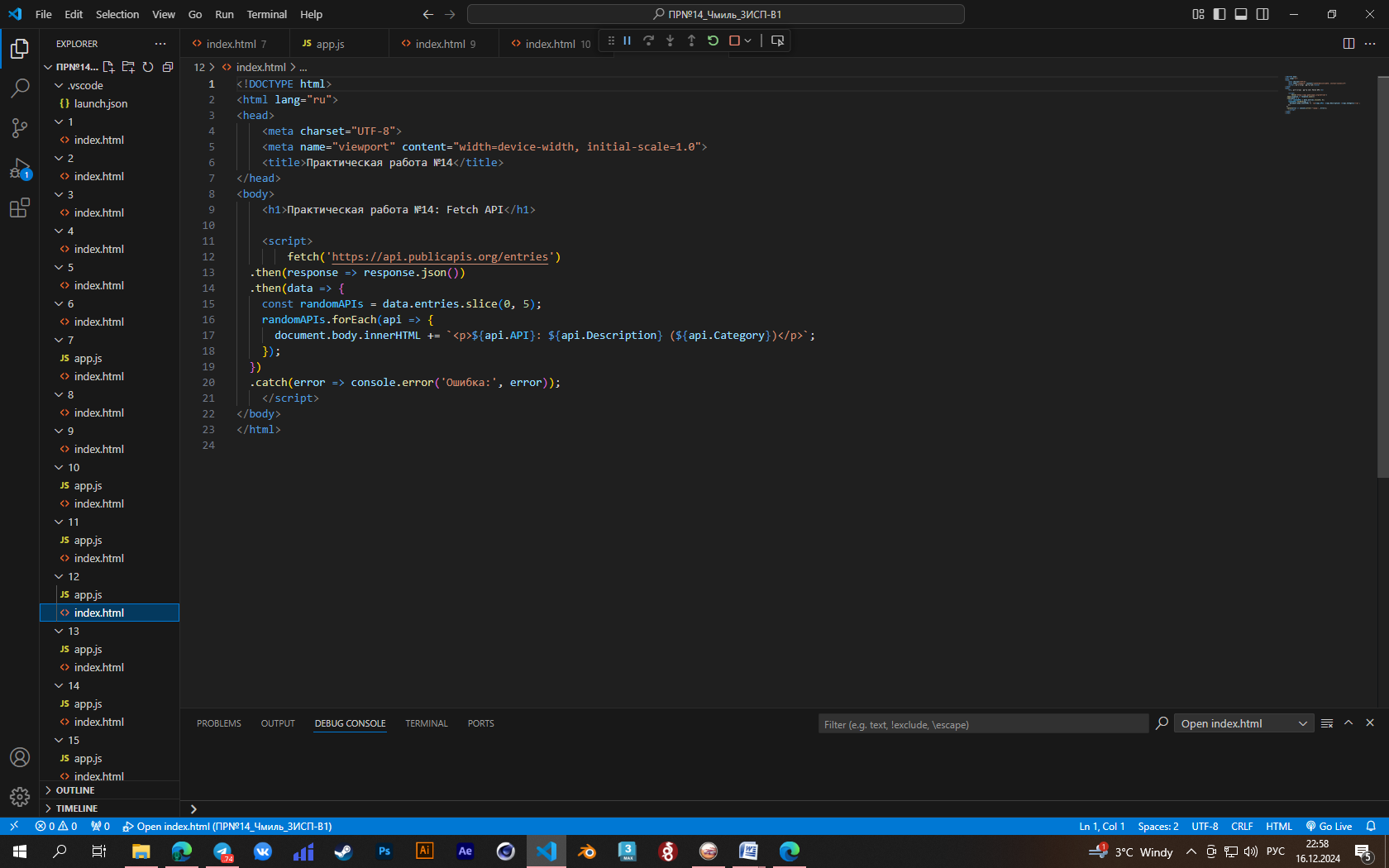
Задание 10



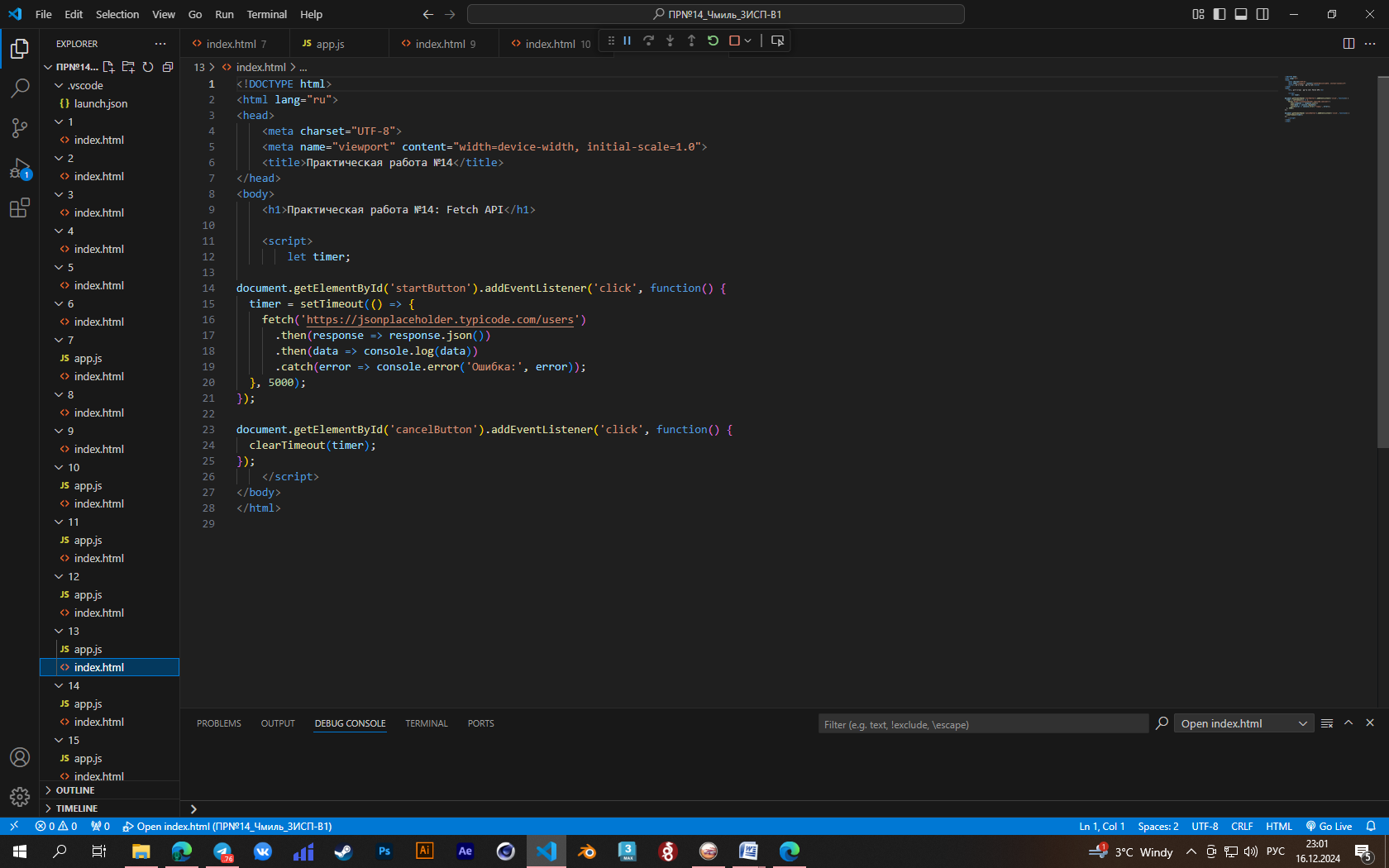
Задание 11



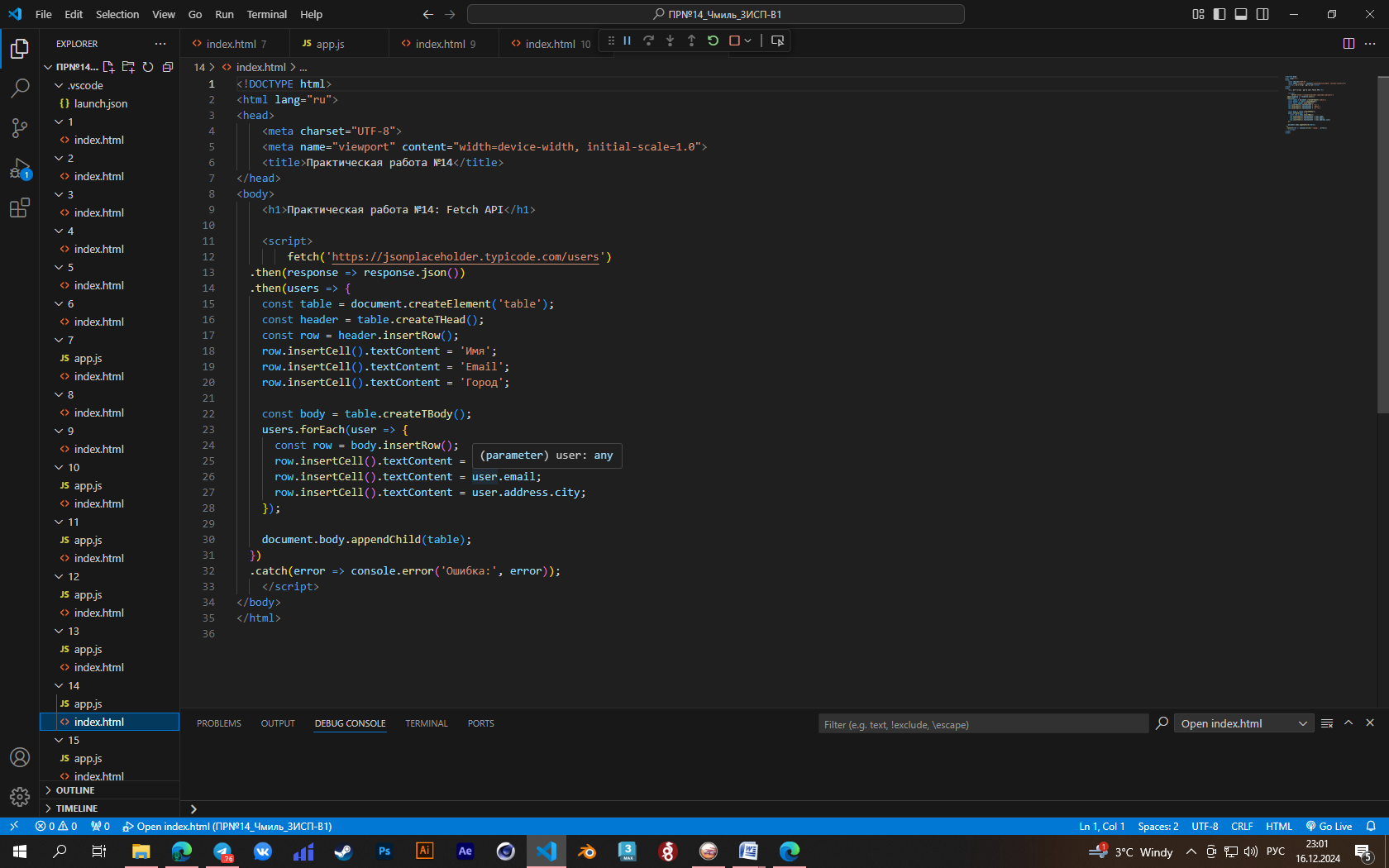
Задание 12



Задание 13



Задание 14



Задание 15

**Ответы на вопросы самоконтроля**

**1. Что такое Fetch API? Для чего он используется?**

Fetch API — это современный интерфейс для работы с HTTP-запросами в JavaScript. Он позволяет выполнять асинхронные запросы к серверу для получения или отправки данных. Fetch API упрощает работу с запросами, предоставляя более удобный и мощный способ по сравнению с устаревшим XMLHttpRequest.

**2. Как выполнить GET-запрос с помощью Fetch? Какие параметры нужно передать?**

Чтобы выполнить GET-запрос с помощью Fetch, нужно просто вызвать функцию fetch() с URL-адресом ресурса.Параметры, которые можно передать, включают URL и объект настроек (опционально), в котором можно указать метод, заголовки и другие параметры.

**3. Как обработать ответ от Fetch-запроса? В каком виде возвращается результат?**

Ответ от Fetch-запроса возвращается в виде объекта Response. Чтобы извлечь данные, нужно использовать методы, такие как json(), text(), blob(), и т.д.

**4. Что произойдёт, если сервер вернёт ошибку, например, 404? Как это обработать?**

Если сервер вернёт ошибку, например, 404, Fetch не выбрасывает исключение. Вместо этого объект Response будет содержать статус ошибки. Чтобы обработать это, нужно проверить response.ok

**5. Как использовать Fetch для отправки данных (POST-запрос)? Какие шаги нужно выполнить?**

Чтобы отправить данные с помощью POST-запроса, нужно указать метод и тело запроса в объекте настроек.

**6. Для чего нужен метод JSON.stringify() в Fetch-запросах?**

Метод JSON.stringify() используется для преобразования JavaScript-объекта в строку JSON. Это необходимо, когда вы отправляете данные на сервер в формате JSON, так как тело запроса должно быть строкой.

**7. Какие типы данных можно получить с помощью Fetch? Как их обработать?**

С помощью Fetch можно получить различные типы данных, такие как JSON, текст, бинарные данные (например, изображения) и т.д. Для обработки данных используются соответствующие методы:

* response.json() для JSON.
* response.text() для текстовых данных.
* response.blob() для бинарных данных.

**8. Что такое AbortController, и как он используется для отмены Fetch-запросов?**

AbortController — это интерфейс, который позволяет отменять Fetch-запросы. Он создаёт объект, который можно передать в Fetch-запрос, и при необходимости можно вызвать метод abort() для его отмены.

**9. Как работают заголовки в Fetch-запросах? Для чего используется параметр headers?**

Заголовки в Fetch-запросах используются для передачи дополнительной информации о запросе, такой как тип контента, авторизация и т.д.

**10. Как использовать async/await для работы с Fetch? Какие преимущества это даёт?**

async/await позволяет писать асинхронный код более читаемо и удобно. Вместо использования цепочки then, можно использовать await для ожидания завершения Fetch-запроса.

Преимущества: более чистый и понятный синтаксис.

**11. Как можно обрабатывать ошибки Fetch-запросов с помощью try...catch?**

Ошибки Fetch-запросов можно обрабатывать с помощью блока try...catch, что позволяет ловить как ошибки сети, так и ошибки, связанные с обработкой ответа.

**12. Что такое CORS, и как оно может влиять на Fetch-запросы?**

CORS (Cross-Origin Resource Sharing) — это механизм, который позволяет ограничивать доступ к ресурсам на одном домене из другого домена. Если вы пытаетесь выполнить Fetch-запрос к ресурсу на другом домене, сервер должен явно разрешить это, добавив соответствующие заголовки. Если CORS не настроен правильно, запрос может быть заблокирован.

**13. Что происходит, если Fetch-запрос превышает время ожидания? Как это можно исправить?**

Fetch API не имеет встроенной поддержки таймаутов. Чтобы установить таймаут, можно использовать AbortController и вызывать abort() через определённое время

**14. Какие ограничения имеет Fetch API по сравнению с другими методами работы с запросами, например, XMLHttpRequest?**

* **Совместимость**: Fetch API поддерживается в современных браузерах, но не во всех старых версиях. XMLHttpRequest более широко поддерживается.
* **Синтаксис**: Fetch API имеет более чистый и современный синтаксис, но может быть сложнее в использовании для некоторых сценариев (например, для загрузки прогресса).
* **Обработка ошибок**: Fetch не выбрасывает исключение при HTTP-ошибках (например, 404 или 500), в отличие от XMLHttpRequest, который может обрабатывать такие ошибки более явно.