Индивидуальная работа

Предмет: Java Script

Тема: Асинхронный JavaScript, модули и обработка ошибок.

Подготовила: Доцен Анна

Группа: IA2303

Проверил: преподаватель Nartea Nichita

Кишинев, 2024

Оглавление

**Теоретическая часть 3**

**Формулировка задачи 3**

**Описание целей и основных этапов работы3**

**Практическая часть4**

**Краткое описание реализации4**

**Вывод и ссылка на репозитории Git4**

**Ответы на контрольные вопросы5**

**Список использованных источников5**

**Теоретическая часть**

**Цель**

Ознакомиться с продвинутыми функциями JavaScript, включая асинхронный JavaScript, модули и обработку ошибок.

**Описание целей и основных этапов работы**

- Создать файл index.html с основной структурой веб-страницы.

<!doctype html>

<html lang="ru">

<head>

   <meta charset="UTF-8">

   <meta name="viewport"

         content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

   <meta http-equiv="X-UA-Compatible" content="ie=edge">

   <title>Activity for Captain Smith</title>

</head>

<body>

<h1>Hey, Captain Smith, you can:</h1>

<i id="activity"></i>

<script type="module" src="src/index.js"></script>

</body>

</html>

- Создать файл index.css для определения стилей страницы.

- Создать директорию /src, где будут размещены файлы JavaScript.

- В директории /src создать файл index.js (или main.js) с основным кодом JavaScript.

- В директории /src создать файл activity.js, в котором будет содержаться логика для получения данных со сторонних ресурсов.

- Написать функцию getRandomActivity(), которая будет делать запрос и получать данные со стороннего ресурса: http://www.boredapi.com/api/activity/.

- Отобразить полученную активность (текст активности) на странице index.html.

function getRandomActivity() {

    fetch('https://www.boredapi.com/api/activity/')

     .then(response => response.json())

     .then(data => {

         // отображение случайного действия на HTML странице

     })

}

- Добавить обработку ошибок в функцию getRandomActivity(). В случае ошибки добавьте следующий текст в файл index.html: "К сожалению, произошла ошибка".

- Изменить функцию getRandomActivity() так, чтобы она использовала ключевые слова async / await. Каковы основные различия между использованием async / await и Promise?

- Добавить функционал обновления данных каждую минуту. Используйте функцию setTimeout().

- Изменить функцию getRandomActivity() так, чтобы она возвращала данные, и добавьте функцию updateActivity(), которая будет отображать полученные данные.

**Практическая часть**

**Краткое описание реализации**

Данный проект является компактным приложением для капитана Смита.

Структура проекта включает в себя файлы HTML, CSS и JavaScript, а также каталог для JavaScript файлов.

Функция getrandomactivity() использует fetch для получения случайной активности с внешнего ресурса и отображает ее на веб-странице.

Добавлена обработка ошибок при запросе данных и уведомление пользователя о возникшей проблеме.

Функция getrandomactivity() была переписана с использованием ключевых слов async/await для повышения читаемости кода.

Добавлена возможность обновления данных каждую минуту с помощью setTimeout, обеспечивая капитану Смиту постоянное обновление активности.

Функция getrandomactivity() была доработана для возврата данных, а также добавлена функция updateactivity() для отображения результатов на странице.

Эта реализация позволяет капитану Смиту находить новые занятия по своему вкусу, обеспечивая постоянное обновление активности и удобство использования.

**Вывод и ссылка на репозиторий Git**

При создании приложения "Капитан Смит" были применены разнообразные методы и концепции веб-разработки, включая работу с асинхронным кодом, обработку запросов к внешним API, использование Promise и async/await, а также добавление функции автоматического обновления данных с помощью setTimeout. Основным акцентом является использование async/await.

Необходимо подчеркнуть, что применение async/await улучшает читаемость и понимание асинхронного кода, делая его более простым и похожим на синхронный. Для обработки ошибок можно использовать блок try...catch для ясности.

Добавление функции обновления данных каждую минуту значительно улучшило пользовательский опыт, поскольку информация постоянно обновляется без необходимости перезагружать страницу.

В общем и целом, разработанное приложение позволяет Капитану Смиту наслаждаться любимым занятием благодаря удобному интерфейсу и автоматическому обновлению данных, предлагая новые варианты каждую минуту!

Ссылка: <https://github.com/Any-utka/-2>

**Ответы на контрольные вопросы**

1. Какое значение возвращает функция fetch?

Функция fetch возвращает объект Promise. Этот Promise разрешается при получении ответа от сервера.

1. Что представляет собой Promise?

Promise представляет асинхронную операцию, которая может завершиться успешно (resolve) или с ошибкой (reject).

1. Какие методы доступны у объекта Promise?

У объекта Promise доступны следующие методы:

- then (onFulfilled, onRejected): добавляет обработчик или обработчики, вызываемые, когда объект Promise имеет состояние fulfilled (успешное выполнение) или rejected (выполнение отклонено).

- catch(onRejected): добавляет обработчик или обработчики, которые будут вызваны при изменении состояния объекта Promise на rejected (выполнение отклонено).

- finally(onFinally): добавляет обработчик, который будет вызван вне зависимости от того, с каким результатом объект Promise был выполнен (rejected — выполнение отклонено или fulfilled — успешное выполнение).

**Источники**

1. <https://github.com/MSU-Courses/javascript_typescript/blob/main/lab/LI2/JS02.md>
2. <https://github.com/MSU-Courses/javascript_typescript/blob/main/lab/LI2/files/index.html>
3. <https://github.com/MSU-Courses/javascript_typescript/blob/main/lab/LI2/files/src/activity.js>