

Молдавский Государственный Университет  
Факультет Математики и Информатики  
Департамент Информатики

Отчет по лабораторной работе по дисциплине  
„JavaScript”

Выполнила: студентка группы **IAFR2403 R**  
**Alexandra Ivarovscaia**

Проверил преподаватель:  
Nartea Nichita, lector univ.

Кишинев, 2025

## Индивидуальная (лабораторная) работа №2

**Цель:** ознакомиться с продвинутыми функциями JavaScript, включая асинхронный JavaScript, модули и обработку ошибок.

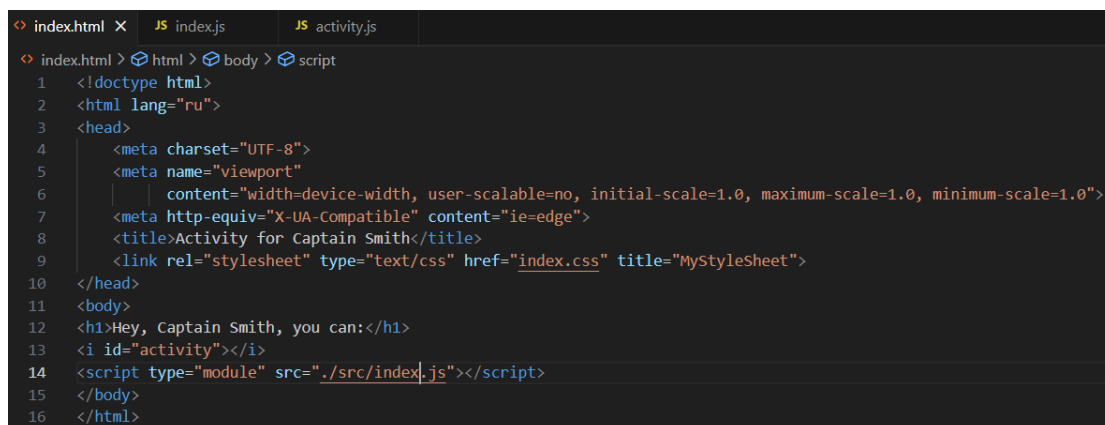
**Условие:** создание мини-приложения, которое будет предлагать капитану Смитту новое занятие при каждом обновлении.

**Структура проекта:**

1. файл index.html с основной структурой веб-страницы
2. файл index.css для определения стилей страницы
3. директория /src, где размещены файлы JavaScript
4. файл index.js (в директории /src) с основным кодом JavaScript
5. файл activity.js (в директории /src), в котором будет содержаться логика для получения данных со сторонних ресурсов.

### Выполнение работы:

Создание HTML-страницы со следующим содержанием:



```
index.html X JS index.js JS activity.js
index.html > html > body > script
1 <!doctype html>
2 <html lang="ru">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>Activity for Captain Smith</title>
9   <link rel="stylesheet" type="text/css" href="index.css" title="MyStyleSheet">
10 </head>
11 <body>
12 <h1>Hey, Captain Smith, you can:</h1>
13 <i id="activity"></i>
14 <script type="module" src="./src/index.js"></script>
15 </body>
16 </html>
```

Рисунок 1: код HTML страницы

Функция getRandomActivity(), которая делает запрос и получает данные со стороннего ресурса. Полученная активность отображается на странице index.html. Добавлена обработка ошибок в функцию getRandomActivity(). В случае ошибки выводится следующий текст в файл index.html: "К сожалению, произошла ошибка". Используются ключевые слова async / await.

```

/**
 * @description function to get random activity from API and update HTML doc
 * @throws {Error} in case of error updates HTML doc with error message
 * @returns {String} a random activity option string
 */

export async function getRandomActivity() {
  try {
    const response = await fetch("https://bored-api.appbrewery.com/random");
    if (!response.ok) {
      throw new Error("❌ сожалению, произошла ошибка");
    }

    const json = await response.json();
    console.log(json.activity);
    // document.getElementById("activity").textContent = json.activity;
    return json.activity;
  } catch (error) {
    console.error(error.message);
    // document.getElementById("activity").textContent = error.message;
  }
}

// getRandomActivity();

```

Рисунок 2: функция `getRandomActivity()`

Добавлена функция `updateActivity()`, которая отображает полученные данные, которые возвращает `getRandomActivity()`. Добавлен функционал обновления данных каждую минуту, используя функцию `setTimeout()`.

```

import {getRandomActivity} from "../activity.js";

/**
 * @description function to call getRandomActivity and update HTML doc and refresh page
 * @throws {error} in case of error updates HTML doc with error message
 */

function updateActivity() {
  try {
    const updatedData = getRandomActivity();
    console.log(updatedData);
    document.getElementById("activity").textContent = updatedData;
  } catch (error) {
    document.getElementById("activity").textContent = "❌ сожалению, произошла ошибка";
  } finally {
    setTimeout(updateActivity, 60000);
  }
}

updateActivity();

```

Рисунок 3: функция `updateActivity()`

Ссылка на GIT репозиторий: <https://github.com/>

## Контрольные вопросы

1. Какое значение возвращает функция `fetch`?  
Функция `fetch` используется для получения данных со сторонних ресурсов. Метод `fetch` возвращает промис, что делает необходимым использование `then`, `catch` или `async/await`.
2. Что представляет собой `Promise`?  
`Promise` в JavaScript – это объект-обёртка, который обеспечивает возможность асинхронного выполнения функций, переданных в него. Он был создан для организации последовательного выполнения асинхронного кода.
3. Какие методы доступны у объекта `Promise`?  
Метод `then` объекта `Promise` используется для добавления обработчиков, которые будут выполнены после того, как `Promise` будет разрешён или отклонён. Ещё одной конструкцией является обработка ошибок через `catch`. Также присутствует метод `finally`, который выполняет действия независимо от результата выполнения `Promise`.
4. Каковы основные различия между использованием `async / await` и `Promise`?  
`async/await` позволяют писать асинхронный код, который легче читать и поддерживать. Ключевое слово `async`, превращает обычную функцию в асинхронную и результат вызова функции оборачивает в `Promise`. Также асинхронная функция позволяет использовать в своём теле ключевое слово `await`. `Await` используется перед `promise-based` функцией, чтобы остановить поток выполнения и дождаться результата её выполнения (результат `Promise`). В то же время, остальной код не блокируется и продолжает работать. `Await` заменяет метод `then`.

## Список литературы

1. [MSU-Courses GitHub](#) – курс JavaScript Государственного Университета Молдовы
2. [W3Schools JavaScript Tutorial](#) – базовый курс JavaScript
3. [Freecodecamp](#) – курс JavaScript для начинающих
4. «Программирование на JavaScript», Васильев А.Н. – учебное пособие для самостоятельного обучения