JavaScript про ECMAScript (семинары)

Урок 4. Асинхронность в JavaScript

Задание 1: "Получение данных о пользователе"

Реализуйте асинхронную функцию для получения данных о пользователе с удаленного сервера:

Функция `getUserData`

- **Описание:**
  1. Функция принимает идентификатор пользователя (ID) в качестве аргумента и использует `fetch` для получения данных с удаленного сервера.
  2. Функция возвращает промис, который разрешается с объектом данных о пользователе, если запрос был успешным.
  3. Если пользователь с указанным ID не найден, промис должен быть отклонен с соответствующим сообщением об ошибке.
- **Последовательность действий:**
  1. Вызовите `fetch`, передав URL с нужным ID пользователя.
  2. Если ответ успешен (код 200), извлеките данные с помощью `response.json()`.
  3. Верните объект с данными о пользователе.
  4. Если ответ не успешен, отклоните промис с сообщением об ошибке.

Пример использования функции

```
getUserData(1)

  .then(user => console.log(user))

  .catch(error => console.error(error));
```

Эталонное решение:

```
async function getUserData(userId) {

  try {
```

```
    const response = await
fetch(`https://api.example.com/users/${userId}`);

    if (!response.ok) {

      throw new Error('User not found');

    }

    const userData = await response.json();

    return userData;

  } catch (error) {

    return Promise.reject(error.message);

  }

}
```

Задание 2: "Отправка данных на сервер"

Реализуйте функцию для отправки данных о пользователе на сервер:

Функция `saveUserData`

- **Описание:**
  - Функция принимает объект с данными о пользователе и использует `fetch` для отправки данных на удаленный сервер.
  - Функция возвращает промис, который разрешается, если данные успешно отправлены.
  - Если запрос неуспешен, промис должен быть отклонен с соответствующим сообщением об ошибке.
- **Подсказка:**
  - Используйте метод `POST` и задайте заголовок `Content-Type` как `application/json`.
  - Объект с данными о пользователе необходимо сериализовать в JSON-строку с помощью `JSON.stringify()`.

Пример использования функции

```
const user = {
  name: 'John Smith',
  age: 30,
  email: 'john@example.com'
};

saveUserData(user)
  .then(() => {
    console.log('User data saved successfully');
  })
  .catch(error => {
    console.error(error);
  });
```

Эталонное решение:

```
async function saveUserData(user) {
  try {
    const response = await fetch('https://api.example.com/users', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(user),
    });

    if (!response.ok) {
      throw new Error('Failed to save user data');
    }
  } catch (error) {
    return Promise.reject(error.message);
  }
}
```

Задание 3: "Изменение стиля элемента через заданное время"

Реализуйте функцию, которая изменяет стиль элемента с задержкой:

Функция `changeStyleDelayed`

- **Описание:**
  - Функция принимает идентификатор элемента (`id`) и время задержки в миллисекундах (`delay`).
  - После истечения времени задержки, функция должна изменить стиль элемента.

Пример использования функции

```
changeStyleDelayed('myElement', 2000); // Через 2 секунды изменяет стиль элемента с id 'myElement'
```

Эталонное решение:

```
function changeStyleDelayed(elementId, delay) {
  setTimeout(() => {
    const element = document.getElementById(elementId);
    if (element) {
      element.style.color = 'red'; // Пример изменения стиля
    }
  }, delay);
}
```