Лабораторная работа № 12.

Тема: Async-Await и обработка ошибок.

Цель: Ознакомиться с синтаксисом async/await в JavaScript, научиться их применять данный синтаксис для упрощения работы с асинхронным кодом, а также эффективно обрабатывать ошибки с помощью конструкции try/catch.

Вариант 1

1. Простая async-функция.
Написать функцию async function hello(), которая возвращает строку "Hello, async!". Проверить работу через .then() и через await внутри другой функции.

2. Последовательные операции с await.
Создать функцию delay(ms, value), возвращающую промис.
В async-функции трижды вызвать её с разными задержками и значениями. Вывести результаты по порядку.

3. Возврат промиса из async.
Реализовать async-функцию, которая возвращает Promise.resolve(100). Проверить, что результат можно получить через .then().

4. try/catch в async.
Написать функцию, которая через 1 секунду отклоняет промис (reject). Вызвать её через await и обработать ошибку в try/catch.

5. Переписывание с then на async/await.
Дан код:

```
Promise.resolve("start")
  .then(v => v + " → step1")
  .then(v => v + " → step2")
  .then(v => console.log(v));
```

Переписать его с использованием async/await.

🔷 Вариант 2

1. Возврат значения из async.
Написать async-функцию, которая возвращает число 10. Проверить результат через await в другой функции.

2. Асинхронный расчёт.
Создать функцию async calculate(), которая ждёт два промиса (с числами) через await, складывает их и выводит результат.

3. Локальная обработка ошибок.
Сделать функцию с двумя await. Первый промис успешный, второй — с ошибкой. Использовать два отдельных блока try/catch для каждого шага.

4. Сравнение подходов.
Реализовать вывод "A" → "B" → "C" с задержкой 1 секунда между сообщениями:
один раз через .then(),
второй раз через async/await.

5. Обработка ошибок в цепочке.
Создать async-функцию, где первый await возвращает строку, второй выбрасывает ошибку. В catch вернуть запасное значение и вывести его в следующем шаге.

⚡ Отчет должен содержать (см. образец):

- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна VSC с исходным кодом программ;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;
- выводы.

Отчеты в формате **pdf** отправлять на email:
colledge20education23@gmail.com

Шпаргалка по async/await

1. async всегда возвращает промис

Даже если вернуть обычное значение — оно оборачивается в Promise.resolve().

👉 Пример:

```
async function greet() {
  return "Привет!";
}

greet().then(value => console.log(value));
// Выведет: "Привет!"
```

Использование await для работы с промисом

await «распаковывает» результат промиса.

👉 Пример:

```
function getNumber() {
  return new Promise(resolve => {
    setTimeout(() => resolve(7), 1000);
  });
}

async function demo() {
  const num = await getNumber();
  console.log("Число:", num);
}

demo();
```

Ошибка в async превращается в reject

Если в async выбросить исключение, оно станет отклонённым промисом.

👉 Пример:

```
async function fail() {
  throw new Error("Что-то пошло не так!");
}
```

```
fail().catch(err => console.error("Ошибка:", err.message));
```

try/catch для обработки ошибок

Можно использовать привычный синтаксис try/catch для работы с ошибками в асинхронных функциях.

👉 Пример:

```
async function fetchData() {
  try {
    let data = await Promise.reject("Сервер недоступен");
    console.log(data);
  } catch (err) {
    console.error("Перехвачено:", err);
  }
}

fetchData();
```

Несколько await подряд

Можно выполнять несколько асинхронных операций по шагам.

👉 Пример:

```
function delay(ms, value) {
  return new Promise(resolve => setTimeout(() =>
resolve(value), ms));
}

async function steps() {
  console.log(await delay(500, "Первый шаг"));
  console.log(await delay(500, "Второй шаг"));
  console.log(await delay(500, "Третий шаг"));
}

steps();
```

Сравнение: .then() и async/await

Один и тот же код можно писать по-разному.

👉 С промисами:

```
Promise.resolve("Начало")
  .then(v => v + " → шаг1")
  .then(v => v + " → шаг2")
  .then(v => console.log(v));
```

👉 С async/await:

```
async function run() {
  let value = "Начало";
  value += " → шаг1";
  value += " → шаг2";
  console.log(value);
}

run();
```