Лабораторная работа № 8

Тема: Основы асинхронности.

Цель: научиться различать и использовать синхронный и асинхронный код в JS. Понять, как работают setTimeout, setInterval, события и HTTP-запросы.

Вариант 1

1. Синхронный и асинхронный код.

Напишите код, который выводит в консоль синхронное сообщение, а затем использует setTimeout с нулевой задержкой для вывода асинхронного сообщения. Используйте console.trace() в обеих частях кода, чтобы показать разницу в вызове.

2. Работа с Call Stack.

Создайте три функции: greeting(), sayHello(), logMessage(). Пусть greeting() вызывает sayHello(), а sayHello() вызывает logMessage(). Поместите console.trace() в каждую функцию, чтобы наглядно продемонстрировать, как они добавляются в Call Stack.

3. Взаимодействие с Web API.

Напишите код, в котором два setTimeout вызываются подряд, но с разными задержками: первый с 2000 мс, второй с 1000 мс. Проследите, в каком порядке их колбэки попадут в очередь макрозадач.

4. Вложенные макрозадачи.

Создайте setTimeout с нулевой задержкой. Внутри его колбэка добавьте ещё один setTimeout с задержкой 500 мс. Используйте console.trace() в обоих колбэках. Проанализируйте, почему первый колбэк выполнится в одном цикле Event Loop, а второй — в следующем.

5. Очередь макрозадач.

Напишите код, который включает в себя:

```
console.log("start");
setTimeout(() => console.log("один"), 0);
setTimeout(() => console.log("два"), 0);
console.log("end");
```

Проанализируйте и запишите, в каком порядке сообщения появятся в консоли.

Вариант 2

1. Синхронный и асинхронный код.

Напишите код, который содержит синхронный цикл for и асинхронный setTimeout с нулевой задержкой. Расположите console.log до и после цикла, а также в колбэке таймера. Используйте console.trace() в колбэке, чтобы увидеть, когда он был вызван.

2. Работа с Call Stack.
Создайте три функции: calc(), add(), showResult(). Пусть calc() вызывает add(), а add() вызывает showResult(). Используйте console.trace() в каждой функции, чтобы проследить, как они поочередно добавляются в стек.

3. Взаимодействие с Web API.
Напишите код, в котором вы используете setTimeout с задержкой 1000 мс. Перед этим setTimeout добавьте синхронный цикл, который выполняется примерно 2 секунды. Проанализируйте, через сколько времени сработает таймер.

4. Макрозадачи и порядок выполнения.
Напишите код, который содержит синхронный console.log и два setTimeout. Один setTimeout с нулевой задержкой, а второй — с задержкой 1000 мс. Расположите их между двумя синхронными выводами в консоль. Проанализируйте порядок появления сообщений.

5. Очередь макрозадач.
Напишите код, который включает:
```
console.log("start");
setTimeout(() => console.log("первый таймер"), 1000);
setTimeout(() => console.log("второй таймер"), 0);
console.log("end");
```
Определите последовательность вывода в консоли и объясните, как на это повлияла задержка таймеров.

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