Департамент анализа данных и машинного обучения

Факультета информационных технологий и анализа больших данных

**ЛАБОРАТОРНАЯ РАБОТА №7**

**на тему**

**«Введение в JavaScript»**

Дисциплина «Технологии разработки web-приложений»

Выполнил:

Студент группы ЗБ-ПИ20-2

Института онлайн-образования

Миловидов Владислав Игоревич

Проверил:

Ассистент ДАДиМО ФИТиАБД

Клочков Евгений Юрьевич

Москва 2023

### Контрольные вопросы

1. **Есть ли у стрелочной функции метод this?**

Да, у стрелочной функции в JavaScript есть метод this. Однако, значение this в стрелочных функциях определяется не контекстом вызова, а контекстом, в котором она была определена. В отличие от обычных функций, в стрелочных функциях this не является изменяемым и не может быть привязано к другому контексту с помощью методов call(), apply() или bind().

1. **Может ли стрелочная функция быть без аргументов?**

Да, стрелочная функция может быть без аргументов в JavaScript, но в этом случае ее необходимо вызывать со скобками () без параметров.

1. **Из-за чего появилась стрелочная функция?**

Стрелочные функции (arrow functions) были добавлены в язык JavaScript в стандарте ECMAScript 6 (ES6) в 2015 году. Они были введены для упрощения написания кода, улучшения его читабельности и более сжатого синтаксиса, а также для устранения некоторых проблем в области видимости переменных и использования ключевого слова this.

1. **Объяснить работу алгоритма:**
2. // Стрелочные функции без параметров, которые визуально легче разбирать
3. setTimeout( () => {
4. console.log('Я буду раньше');
5. setTimeout( () => {
6. // deeper code
7. console.log('Я буду позже');
8. }, 1);
9. }, 1);

Данный алгоритм представляет собой две отложенные функции, одна из которых запускается через 1 миллисекунду после её вызова. Внутри первой функции происходит вывод сообщения "Я буду раньше" в консоль. Внутри второй функции, запущенной через 1 миллисекунду после первой, происходит более глубокий код (который указан комментарием "deeper code") и выводится сообщение "Я буду позже" в консоль.

Таким образом, данный алгоритм демонстрирует отложенное выполнение кода в JavaScript и порядок выполнения функций setTimeout(). Первая отложенная функция вызывается раньше, но вторая вызывается сразу после первой и может содержать более глубокий код, который выполнится после первой функции.