Лабораторная работа № 11

Тема: Ошибки и отладка в JavaScript.

Вариант 1

Задание 1: Анализ ошибок.
Даны фрагменты кода с ошибками. Для каждого определите:

- Тип ошибки (синтаксическая, ошибка времени выполнения, логическая)
- Причину ошибки
- Способ исправления

Код 1:

```
function calculateAverage(numbers) {
    let sum = 0;
    for (let i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    return sum / numbers.lenght;
}

console.log(calculateAverage([10, 20, 30]));
```

Код 2:

```
const user = {
    name: "Алексей",
    age: 25
};

console.log(user.adress.city);
```

Код 3:

```
function divide(a, b) {
    return a / b;
}

console.log(divide(10, 0));
```

Код 4:

```
const data = JSON.parse('{name: "Test", value: 42}');
console.log(data);
```

Код 5:

```
let x = 10;
const y = 20;
y = 30;
console.log(x + y);
```

Задание 2: try...catch и пользовательские ошибки.
Создайте функцию validateUserData(userData), которая:

- Принимает объект с данными пользователя
- Проверяет наличие обязательных полей: name, email, age
- Проверяет, что email содержит '@'
- Проверяет, что age - число от 18 до 120

Используйте try...catch для обработки ошибок

Задание 3*: Инструменты DevTools.
Дана страница с ошибками:

html

```
<!DOCTYPE html>
<html>
<head>
   <title>Отладка</title>
</head>
<body>
   <button id="calculate">Рассчитать</button>
   <div id="result"></div>

   <script>
      document.getElementById('calculate').addEventListener('click',
function() {
         const numbers = [10, 20, 30, '40', 50];
         let sum = 0;

         for (let i = 0; i <= numbers.length; i++) {
```

```
            sum += parseInt(numbers[i]);
        }

        document.getElementById('result').innerHTML =
            'Среднее: ' + (sum / numbers.length);
    });
  </script>
</body>
</html>
```

Используя DevTools:

- Найдите и исправьте все ошибки
- Установите точки останова в критических местах
- Используйте debugger для пошагового выполнения
- Проверьте значения переменных в процессе выполнения

Используйте console.table() для отображения массива

Вариант 2

Задание 1: Диагностика и классификация ошибок.
Проанализируйте код, найдите ошибки и классифицируйте их:

Код 1:

```
function findMax(arr) {
    let max = arr[0];
    for (let i = 1; i < arr.length; i++) {
        if (arr[i] > max) {
            max == arr[i];
        }
    }
    return max;
}

console.log(findMax([5, 2, 9, 1]));
```

Код 2:

```
const config = {
    apiUrl: "https://api.test.com",
    timeout: 5000
```

```
};

function makeRequest() {
    fetch(config.apiUrll)
        .then(response => response.json())
        .then(data => console.log(data));
}

makeRequest();
```

Код 3:

```
class Calculator {
    constructor() {
        this.history = [];
    }

    add(a, b) {
        const result = a + b;
        this.history.push(`${a} + ${b} = ${result}`);
        return result;
    }

    getHistory() {
        return this.history.join('\n');
    }
}

const calc = new Calculator();
console.log(calc.add(5, "10"));
console.log(calc.getHistory());
```

Код 4:

```
async function loadData() {
    const response = await fetch('/api/data');
    const data = response.json();
    return data;
}

loadData().then(data => {
    console.log(data.value.toUpperCase());
```

```
});
```

Код 5:

```
const items = [
    { id: 1, name: "Item 1", price: 100 },
    { id: 2, name: "Item 2", price: 200 },
    { id: null, name: "Item 3", price: 300 }
];

const total = items.reduce((sum, item) =>
    sum + item.price, 0
);

console.log(`Общая стоимость: ${total}`);
```

Задание 2: try...catch и пользовательские ошибки.
Создайте функцию validatePassword, которая:

- принимает строку с паролем пользователя
- проверяет требования к паролю:
- Минимальная длина: 8 символов
- Должна содержать хотя бы одну цифру
- Должна содержать хотя бы один специальный символ

Используйте try...catch для обработки ошибок

Задание 3*: Практика с DevTools.
Дана страница с интерактивным калькулятором:

html

```
<!DOCTYPE html>
<html>
<head>
    <title>Калькулятор</title>
    <style>
        .error { color: red; }
        .success { color: green; }
    </style>
</head>
<body>
    <input type="number" id="num1" placeholder="Число 1">
    <select id="operation">
```

```
        <option value="+">+</option>
        <option value="-">-</option>
        <option value="*">*</option>
        <option value="/">/</option>
    </select>
    <input type="number" id="num2" placeholder="Число 2">
    <button id="calculate">=</button>
    <div id="output"></div>

    <script>
        document.getElementById('calculate').addEventListener('click',
function() {
            const num1 = document.getElementById('num1').value;
            const num2 = document.getElementById('num2').value;
            const operation = document.getElementById('operation').value;

            let result;
            switch(operation) {
                case '+': result = num1 + num2; break;
                case '-': result = num1 - num2; break;
                case '*': result = num1 * num2; break;
                case '/': result = num1 / num2; break;
            }

            const output = document.getElementById('output');
            if (isNaN(result)) {
                output.className = 'error';
                output.textContent = 'Ошибка вычисления';
            } else {
                output.className = 'success';
                output.textContent = `Результат: ${result}`;
            }
        });
    </script>
</body>
</html>
```

Используя DevTools:

- Найдите и исправьте логические ошибки
- Установите точку останова в обработчике события
- Используйте conditional breakpoint при делении на ноль

⚡ **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