МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

(КАФЕДРА 43)

|  |
| --- |
| ОТЧЕТ ЗАЩИЩЁН С ОЦЕНКОЙ  РУКОВОДИТЕЛЬ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ассистент |  |  |  |  |  | Ю. В. Ветрова |
| (должность, учёная степень, звание) |  | (подпись) |  | (дата защиты) |  | (инициалы, фамилия) |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРАКТИЧЕСКОМУ ЗАДАНИЮ №3

Использование прототипов и классов в JavaScript

по дисциплине: [JavaScript, его библиотеки и фреймворки в Frontend-разработке](https://lms.guap.ru/course/view.php?id=1099" \l "section-1" \o "JavaScript, его библиотеки и фреймворки в Frontend-разработке)

|  |
| --- |
| РАБОТУ ВЫПОЛНИЛ |

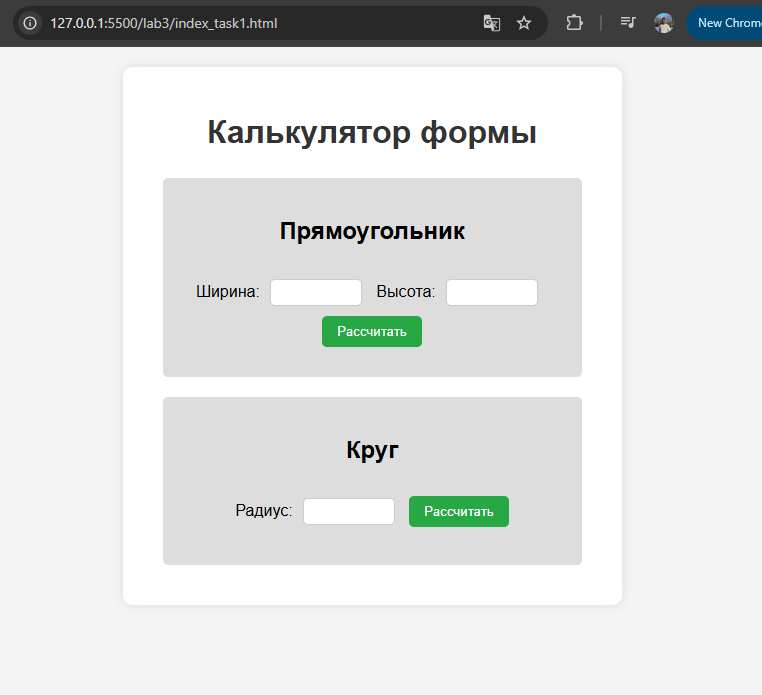
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4236 |  |  |  | Л. Мвале |
|  |  |  | подпись, дата |  | инициалы, фамилия |

1. **Цель**

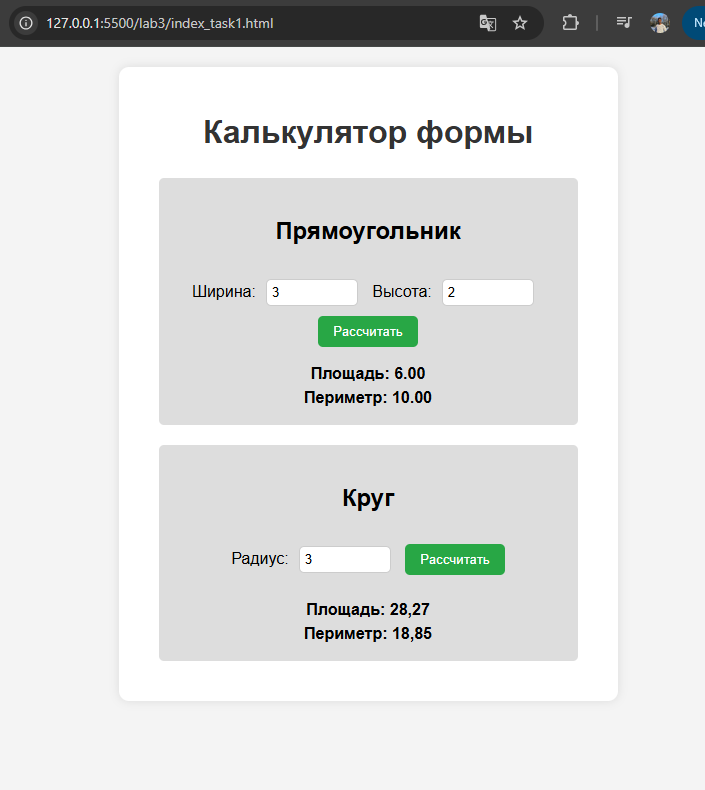
Цель работы – формирование практических навыков работы с прототипами и классами в JavaScript.

1. **Задание**
2. Создайте класс "Фигура", который будет иметь методы для вычисления площади и периметра фигуры. Напишите классы "Прямоугольник" и "Круг", которые наследуются от класса "Фигура" и реализуют соответствующие методы.
3. Создайте класс "Студент", который будет иметь защищённые свойства "имя", "возраст" и "средний балл". Напишите методы для изменения и получения значений этих свойств.
4. Создайте класс "Калькулятор", который будет иметь методы для выполнения основных математических операций (сложение, вычитание, умножение, деление).

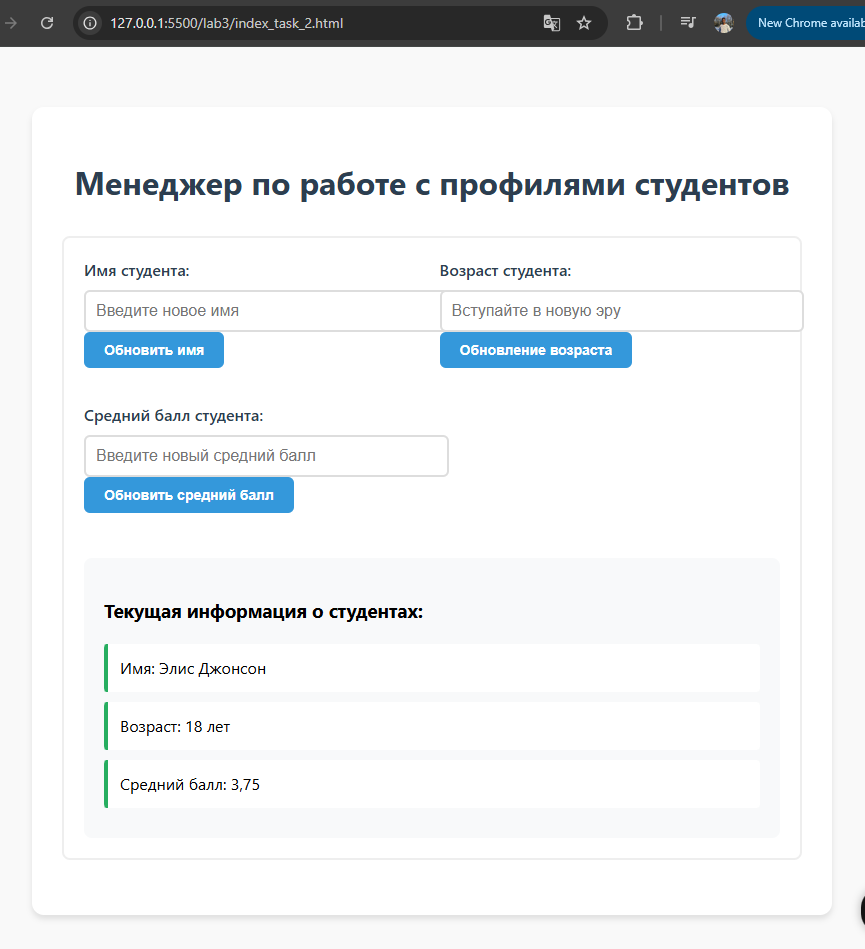
Скриншоты для первого задания



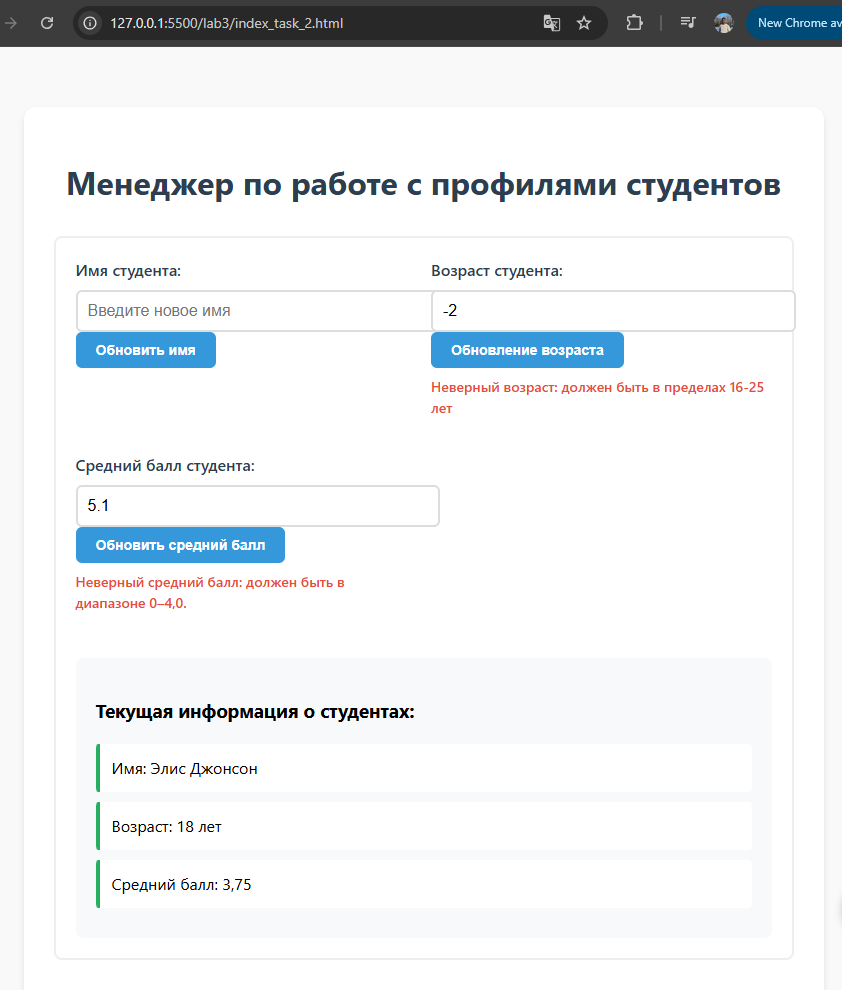
Скриншоты для выполннения первого задания



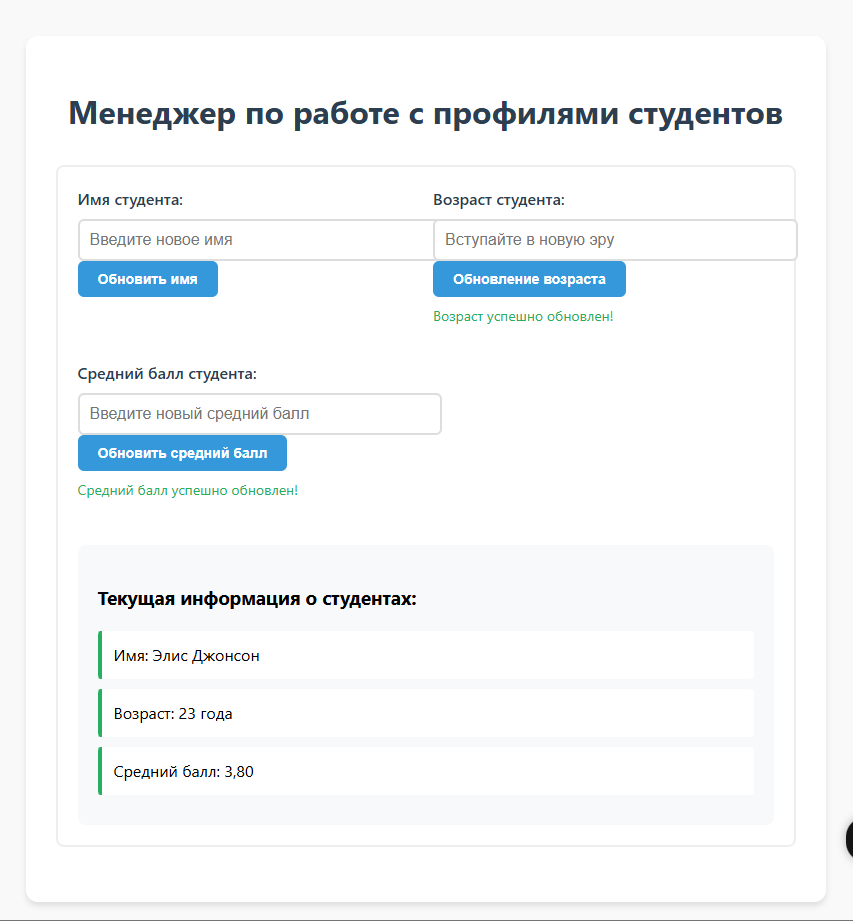
Скриншоты для второго задания



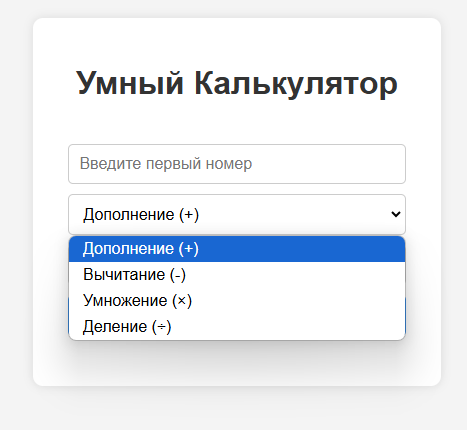
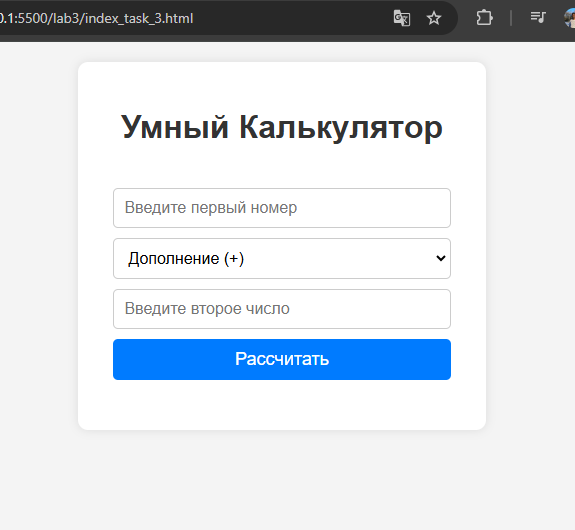
выполнение с неверными данными



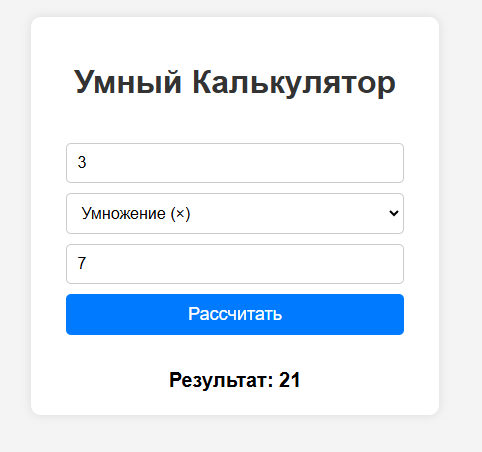
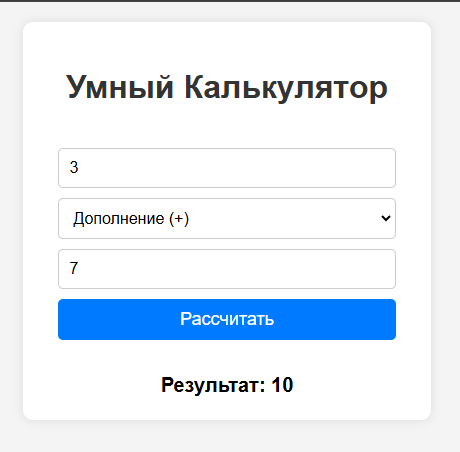
успешные обновления

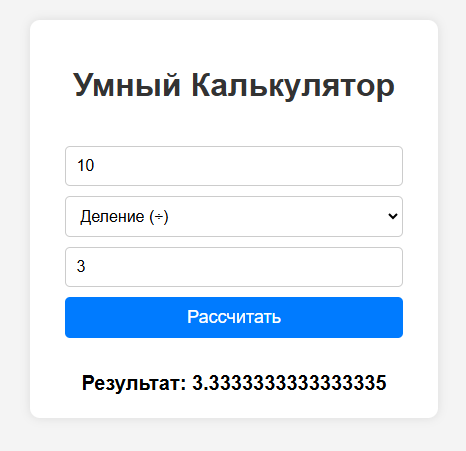
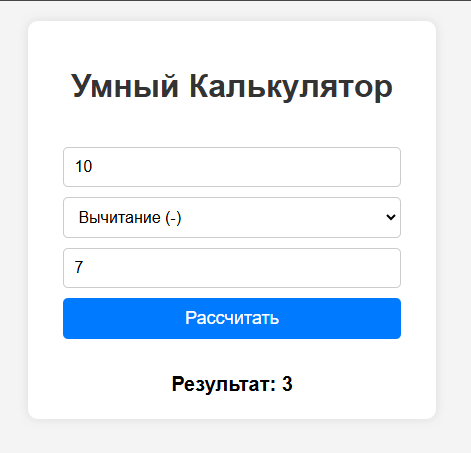


Скриншоты для третьего задания



выполнение





Приложение

Первый задание

script\_1.js

// Base Shape class

*class* Shape {

*constructor*() {

        if (this.constructor === Shape) {

            throw new Error("Cannot instantiate an abstract class.");

        }

    }

    area() {

        throw new Error("Method 'area()' must be implemented.");

    }

    perimeter() {

        throw new Error("Method 'perimeter()' must be implemented.");

    }

}

// Rectangle class (inherits from Shape)

*class* Rectangle extends *Shape* {

*constructor*(*width*, *height*) {

        super();

        this.width = width;

        this.height = height;

    }

    area() {

        return this.width \* this.height;

    }

    perimeter() {

        return 2 \* (this.width + this.height);

    }

}

// Circle class (inherits from Shape)

*class* Circle extends *Shape* {

*constructor*(*radius*) {

        super();

        this.radius = radius;

    }

    area() {

        return Math.PI \* Math.pow(this.radius, 2);

    }

    perimeter() {

        return 2 \* Math.PI \* this.radius;

    }

}

// Function to calculate rectangle properties

*function* calculateRectangle() {

*const* width = parseFloat(document.getElementById("rect-width").value);

*const* height = parseFloat(document.getElementById("rect-height").value);

    if (isNaN(width) || isNaN(height) || width <= 0 || height <= 0) {

        document.getElementById("rect-result").innerHTML = "Please enter valid dimensions.";

        return;

    }

*const* rectangle = new Rectangle(width, height);

    document.getElementById("rect-result").innerHTML = `

        Area: ${rectangle.area().toFixed(2)} <br>

        Perimeter: ${rectangle.perimeter().toFixed(2)}

    `;

}

// Function to calculate circle properties

*function* calculateCircle() {

*const* radius = parseFloat(document.getElementById("circle-radius").value);

    if (isNaN(radius) || radius <= 0) {

        document.getElementById("circle-result").innerHTML = "Please enter a valid radius.";

        return;

    }

*const* circle = new Circle(radius);

    document.getElementById("circle-result").innerHTML = `

        Area: ${circle.area().toFixed(2)} <br>

        Perimeter: ${circle.perimeter().toFixed(2)}

    `;

}

Index\_1.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Shape Calculator</title>

    <link rel="stylesheet" href="style\_1.css">

</head>

<body>

    <div class="container">

        <h1>Shape Calculator</h1>

        <div class="shape-form">

            <h2>Rectangle</h2>

            <label>Width: <input type="number" id="rect-width" /></label>

            <label>Height: <input type="number" id="rect-height" /></label>

            <button onclick="calculateRectangle()">Calculate</button>

            <div id="rect-result"></div>

        </div>

        <div class="shape-form">

            <h2>Circle</h2>

            <label>Radius: <input type="number" id="circle-radius" /></label>

            <button onclick="calculateCircle()">Calculate</button>

            <div id="circle-result"></div>

        </div>

    </div>

    <script src="script\_1.js"></script>

</body>

</html>

style\_3.css

body {

*font-family*: Arial, sans-serif;

*background-color*: #f4f4f4;

*text-align*: center;

*padding*: 20px;

}

.container {

*background*: white;

*padding*: 20px;

*border-radius*: 10px;

*width*: 40%;

*margin*: auto;

*box-shadow*: 0px 0px 10px rgba(0, 0, 0, 0.1);

}

h1 {

*color*: #333;

}

.calculator {

*display*: flex;

*flex-direction*: column;

*gap*: 10px;

*padding*: 15px;

}

input, select {

*padding*: 10px;

*border-radius*: 5px;

*border*: 1px solid #ccc;

*font-size*: 16px;

}

button {

*padding*: 10px;

*border*: none;

*background-color*: #007bff;

*color*: white;

*border-radius*: 5px;

*cursor*: pointer;

*font-size*: 18px;

}

button:hover {

*background-color*: #0056b3;

}

#result {

*margin-top*: 15px;

*font-size*: 20px;

*font-weight*: bold;

}

script\_2.js

// Student Class Implementation

*class* Student {

*constructor*(*name*, *age*, *gpa*) {

        // Protected properties convention using underscore

        this.\_name = this.validateName(name);

        this.\_age = this.validateAge(age);

        this.\_gpa = this.validateGPA(gpa);

    }

    // Validation methods

    validateName(*name*) {

        if (typeof name === 'string' && name.trim().length >= 2) {

            return name.trim();

        }

        throw new Error('Invalid name: Must be at least 2 characters');

    }

    validateAge(*age*) {

*const* num = Number(age);

        if (Number.isInteger(num) && num >= 16 && num <= 25) {

            return num;

        }

        throw new Error('Invalid age: Must be between 16-25');

    }

    validateGPA(*gpa*) {

*const* num = Number(gpa);

        if (!isNaN(num) && num >= 0 && num <= 4.0) {

            return Math.round(num \* 100) / 100; // Round to 2 decimals

        }

        throw new Error('Invalid GPA: Must be between 0-4.0');

    }

    // Accessor methods

*get* name() {

        return this.\_name;

    }

*set* name(*newName*) {

        this.\_name = this.validateName(newName);

    }

*get* age() {

        return this.\_age;

    }

*set* age(*newAge*) {

        this.\_age = this.validateAge(newAge);

    }

*get* gpa() {

        return this.\_gpa;

    }

*set* gpa(*newGPA*) {

        this.\_gpa = this.validateGPA(newGPA);

    }

    // Utility methods

    getStudentInfo() {

        return {

            name: this.\_name,

            age: this.\_age,

            gpa: this.\_gpa

        };

    }

}

// Create initial student instance

*let* student = new Student('Alice Johnson', 18, 3.75);

// Display initial values

*function* refreshDisplay() {

*const* info = student.getStudentInfo();

    document.getElementById('nameDisplay').textContent =

        `Name: ${info.name}`;

    document.getElementById('ageDisplay').textContent =

        `Age: ${info.age} years`;

    document.getElementById('gpaDisplay').textContent =

        `GPA: ${info.gpa.toFixed(2)}`;

}

// Update functions with error handling

*function* updateName() {

*const* input = document.getElementById('nameInput');

*const* message = document.getElementById('nameMessage');

    try {

        student.name = input.value;

        message.textContent = 'Name updated successfully!';

        message.className = 'success';

        input.value = '';

        refreshDisplay();

    } catch (error) {

        message.textContent = error.message;

        message.className = 'error';

    }

}

*function* updateAge() {

*const* input = document.getElementById('ageInput');

*const* message = document.getElementById('ageMessage');

    try {

        student.age = input.value;

        message.textContent = 'Age updated successfully!';

        message.className = 'success';

        input.value = '';

        refreshDisplay();

    } catch (error) {

        message.textContent = error.message;

        message.className = 'error';

    }

}

*function* updateGPA() {

*const* input = document.getElementById('gpaInput');

*const* message = document.getElementById('gpaMessage');

    try {

        student.gpa = input.value;

        message.textContent = 'GPA updated successfully!';

        message.className = 'success';

        input.value = '';

        refreshDisplay();

    } catch (error) {

        message.textContent = error.message;

        message.className = 'error';

    }

}

// Initial display

refreshDisplay();

index\_2.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Student Information System</title>

    <link rel="stylesheet" href="style\_2.css">

</head>

<body>

    <div class="dashboard">

        <h1>Student Profile Manager</h1>

        <div class="student-card">

            <div class="control-panel">

                <!-- Name Control -->

                <div class="input-group">

                    <label for="nameInput">Student Name:</label>

                    <input type="text" id="nameInput" placeholder="Enter new name">

                    <button onclick="updateName()">Update Name</button>

                    <div id="nameMessage" class="message"></div>

                </div>

                <!-- Age Control -->

                <div class="input-group">

                    <label for="ageInput">Student Age:</label>

                    <input type="number" id="ageInput" placeholder="Enter new age">

                    <button onclick="updateAge()">Update Age</button>

                    <div id="ageMessage" class="message"></div>

                </div>

                <!-- GPA Control -->

                <div class="input-group">

                    <label for="gpaInput">Student GPA:</label>

                    <input type="number" step="0.1" id="gpaInput" placeholder="Enter new GPA">

                    <button onclick="updateGPA()">Update GPA</button>

                    <div id="gpaMessage" class="message"></div>

                </div>

            </div>

            <!-- Display Section -->

            <div class="display-section">

                <h3>Current Student Information:</h3>

                <div class="student-info" id="nameDisplay"></div>

                <div class="student-info" id="ageDisplay"></div>

                <div class="student-info" id="gpaDisplay"></div>

            </div>

        </div>

    </div>

    <script src="script\_2.js"></script>

</body>

</html>

style\_2.css

:root {

    --primary-color: #2c3e50;

    --secondary-color: #3498db;

    --success-color: #27ae60;

    --error-color: #e74c3c;

}

body {

*font-family*: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

*max-width*: 800px;

*margin*: 40px auto;

*padding*: 20px;

*background-color*: #f9f9f9;

}

.dashboard {

*background*: white;

*border-radius*: 12px;

*box-shadow*: 0 4px 6px rgba(0, 0, 0, 0.1);

*padding*: 30px;

}

.student-card {

*border*: 2px solid #eee;

*border-radius*: 8px;

*padding*: 20px;

*margin-bottom*: 25px;

}

.control-panel {

*display*: grid;

*gap*: 15px;

*grid-template-columns*: repeat(auto-fit, minmax(250px, 1fr));

}

.input-group {

*margin-bottom*: 20px;

}

label {

*display*: block;

*margin-bottom*: 8px;

*color*: var(--primary-color);

*font-weight*: 600;

}

input {

*width*: 100%;

*padding*: 10px;

*border*: 2px solid #ddd;

*border-radius*: 6px;

*font-size*: 16px;

*transition*: border-color 0.3s ease;

}

input:focus {

*border-color*: var(--secondary-color);

*outline*: none;

}

button {

*background-color*: var(--secondary-color);

*color*: white;

*padding*: 10px 20px;

*border*: none;

*border-radius*: 6px;

*cursor*: pointer;

*transition*: background-color 0.3s ease;

*font-size*: 14px;

*font-weight*: 600;

}

button:hover {

*background-color*: #2980b9;

}

.display-section {

*margin-top*: 25px;

*padding*: 20px;

*background-color*: #f8f9fa;

*border-radius*: 8px;

}

.student-info {

*font-size*: 16px;

*margin*: 10px 0;

*padding*: 12px;

*background*: white;

*border-left*: 4px solid var(--success-color);

*border-radius*: 4px;

}

.error {

*color*: var(--error-color);

*margin-top*: 8px;

*font-size*: 14px;

*font-weight*: 500;

}

.success {

*color*: var(--success-color);

*margin-top*: 8px;

*font-size*: 14px;

}

h1 {

*color*: var(--primary-color);

*text-align*: center;

*margin-bottom*: 30px;

}

script\_3.js

// Calculator Class

*class* Calculator {

    add(*a*, *b*) {

        return a + b;

    }

    subtract(*a*, *b*) {

        return a - b;

    }

    multiply(*a*, *b*) {

        return a \* b;

    }

    divide(*a*, *b*) {

        if (b === 0) {

            return "Error: Division by zero!";

        }

        return a / b;

    }

}

// Function to perform calculation

*function* calculate() {

*const* num1 = parseFloat(document.getElementById("num1").value);

*const* num2 = parseFloat(document.getElementById("num2").value);

*const* operation = document.getElementById("operation").value;

*const* calculator = new Calculator();

    if (isNaN(num1) || isNaN(num2)) {

        document.getElementById("result").innerHTML = "Please enter valid numbers!";

        return;

    }

*let* result;

    switch (operation) {

        case "add":

            result = calculator.add(num1, num2);

            break;

        case "subtract":

            result = calculator.subtract(num1, num2);

            break;

        case "multiply":

            result = calculator.multiply(num1, num2);

            break;

        case "divide":

            result = calculator.divide(num1, num2);

            break;

        default:

            result = "Invalid operation!";

    }

    document.getElementById("result").innerHTML = `Result: ${result}`;

}

index\_3.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Smart Calculator</title>

    <link rel="stylesheet" href="style\_3.css">

</head>

<body>

    <div class="container">

        <h1>Smart Calculator</h1>

        <div class="calculator">

            <input type="number" id="num1" placeholder="Enter first number">

            <select id="operation">

                <option value="add">Addition (+)</option>

                <option value="subtract">Subtraction (-)</option>

                <option value="multiply">Multiplication (×)</option>

                <option value="divide">Division (÷)</option>

            </select>

            <input type="number" id="num2" placeholder="Enter second number">

            <button onclick="calculate()">Calculate</button>

        </div>

        <div id="result"></div>

    </div>

    <script src="script\_3.js"></script>

</body>

</html>

style\_3.css

body {

*font-family*: Arial, sans-serif;

*background-color*: #f4f4f4;

*text-align*: center;

*padding*: 20px;

}

.container {

*background*: white;

*padding*: 20px;

*border-radius*: 10px;

*width*: 40%;

*margin*: auto;

*box-shadow*: 0px 0px 10px rgba(0, 0, 0, 0.1);

}

h1 {

*color*: #333;

}

.calculator {

*display*: flex;

*flex-direction*: column;

*gap*: 10px;

*padding*: 15px;

}

input, select {

*padding*: 10px;

*border-radius*: 5px;

*border*: 1px solid #ccc;

*font-size*: 16px;

}

button {

*padding*: 10px;

*border*: none;

*background-color*: #007bff;

*color*: white;

*border-radius*: 5px;

*cursor*: pointer;

*font-size*: 18px;

}

button:hover {

*background-color*: #0056b3;

}

#result {

*margin-top*: 15px;

*font-size*: 20px;

*font-weight*: bold;

}

1. **Выводы**

В ходе выполнения данной работы были приобретены практические навыки работы с прототипами и классами в JavaScript. Были созданы и реализованы классы с использованием механизмов наследования, что позволило лучше понять принципы объектно-ориентированного программирования в JavaScript.

В частности:

* Освоены принципы создания классов и конструкторов.

Реализованы механизмы наследования с помощью extends и прототипов.

* Изучены и применены на практике методы для переопределения и расширения функциональности классов.
* Приобретены навыки структурирования кода и организации объектно-ориентированных решений.

Таким образом, работа способствовала закреплению теоретических знаний и развитию практических умений, необходимых для разработки программ с использованием прототипов и классов в JavaScript.