Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ

ПО Лабораторной работе №22

ОП Т.094001

Надо обязательно дойти до конца

Руководитель практики

от колледжа (Рогалевич А. В.)

Учащийся (Бакулин А. И.)

2024

**Задание 1.**

Сравнить два фрагмента кода.

Первый использует finally для выполнения кода после try…catch:

try {

    начать работу

    работать

} catch (e) {

    обработать ошибку

} finally {

    очистить рабочее пространство

}

Второй фрагмент просто ставит очистку после try..catch:

try {

    начать работу

    работать

} catch (e) {

    обработать ошибку

}

очистить рабочее пространство Нужна очистка после работы, неважно возникли ошибки или нет. Есть ли здесь преимущество в использовании finally или оба фрагмента кода одинаковы? Если такое преимущество есть, то дать пример, когда оно проявляется.

**Код**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Task 1</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

  </style>

</head>

<body>

  <script>

    // Пример использования finally

    try {

      // Начать работу

      // Работать

    } catch (e) {

      // Обработать ошибку

    } finally {

      // Очистить рабочее пространство

      console.log("Cleaning up with 'finally'");

    }

    // Пример без finally

    try {

      // Начать работу

      // Работать

    } catch (e) {

      // Обработать ошибку

    }

    // Очистить рабочее пространство

    console.log("Cleaning up without 'finally'");

  </script>

</body>

</html>

**Задание 2.**

Написать функцию, которая будет извлекать корень из числа и при этом выбрасывать исключение, если корень извлекается из отрицательного числа.

**Код**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Task 2</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

  </style>

</head>

<body>

  <script>

    function sqrtNumber(num) {

      if (num < 0) {

        throw new Error("Cannot take square root of a negative number");

      }

      return Math.sqrt(num);

    }

    try {

      const userInput = parseFloat(prompt("Введите число для извлечения корня:"));

      if (isNaN(userInput)) {

        throw new Error("Введено некорректное число.");

      }

      const result = sqrtNumber(userInput);

      console.log("Корень числа:", result);

    } catch (error) {

      console.error(error.message);

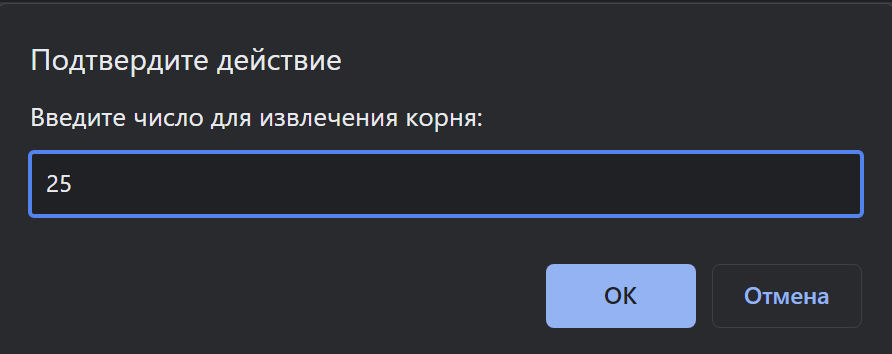
    }

  </script>

</body>

</html>

**Реализация**





**Задание 3.**

Пусть JSON выдается следующим образом:

let json = `[

    {

        &quot;name&quot;: &quot;user1&quot;,

        &quot;age&quot;: 25,

        &quot;salary&quot;: 1000

11

    },

    {

        &quot;name&quot;: &quot;user2&quot;,

        &quot;age&quot;: 26,

        &quot;salary&quot;: 2000

    },

    {

        &quot;name&quot;: &quot;user3&quot;,

        &quot;age&quot;: 27,

        &quot;salary&quot;: 3000

    }

]`;

Проверить этот JSON на общую корректность при разборе, а после разбора проверить, что в результате получается массив, а не что-то другое. Если в результате получается не массив выбросить исключение.

**Код**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Task 3</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

  </style>

</head>

<body>

  <script>

    let json = `[

      {

          "name": "user1",

          "age": 25,

          "salary": 1000

      },

      {

          "name": "user2",

          "age": 26,

          "salary": 2000

      },

      {

          "name": "user3",

          "age": 27,

          "salary": 3000

      }

    ]`;

    try {

      const parsedJson = JSON.parse(json);

      if (!Array.isArray(parsedJson)) {

        throw new Error("Invalid JSON format. Expected an array.");

      }

      console.log("Parsed JSON:", parsedJson);

    } catch (error) {

      console.error(error.message);

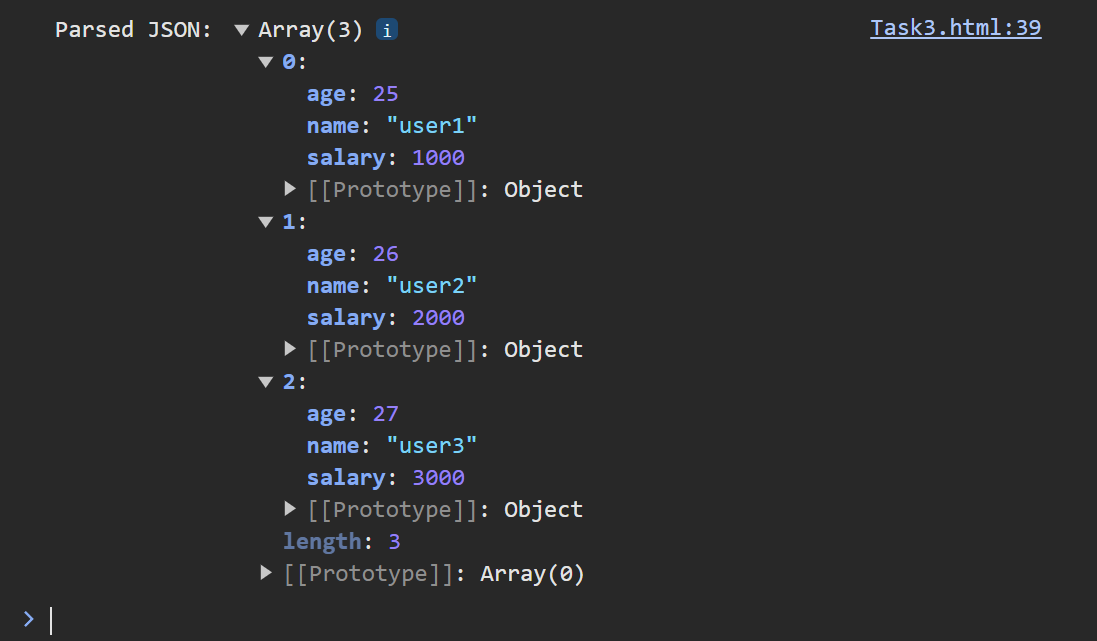
    }

  </script>

</body>

</html>

**Реализация**

****

**Задание 4.**

Пользователь вводит три числа. Если одно из чисел меньше ноля, вернуть и обработать первое исключение, если одно из чисел равно нолю, вернуть и обработать второе исключение, если пользователь не ввёл одно или несколько чисел, вернуть и обработать третье исключение. По завершении работы в любом случае вывести сообщение о завершении работы алгоритма.

**Код**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Task 4</title>

  <style>

    body {

      font-family: Arial, sans-serif;

    }

  </style>

</head>

<body>

  <script>

    try {

      const num1 = parseFloat(prompt("Введите первое число:"));

      const num2 = parseFloat(prompt("Введите второе число:"));

      const num3 = parseFloat(prompt("Введите третье число:"));

      if (isNaN(num1) || isNaN(num2) || isNaN(num3)) {

        throw new Error("Не введено одно или несколько чисел.");

      }

      if (num1 < 0) {

        throw new Error("Первое число меньше ноля.");

      } else if (num2 === 0) {

        throw new Error("Второе число равно нолю.");

      }

      console.log("Введенные числа:", num1, num2, num3);

    } catch (error) {

      console.error(error.message);

    } finally {

      console.log("Завершение работы алгоритма.");

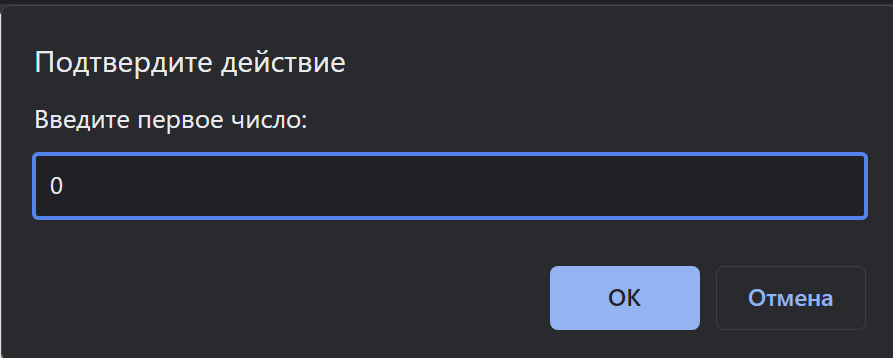
    }

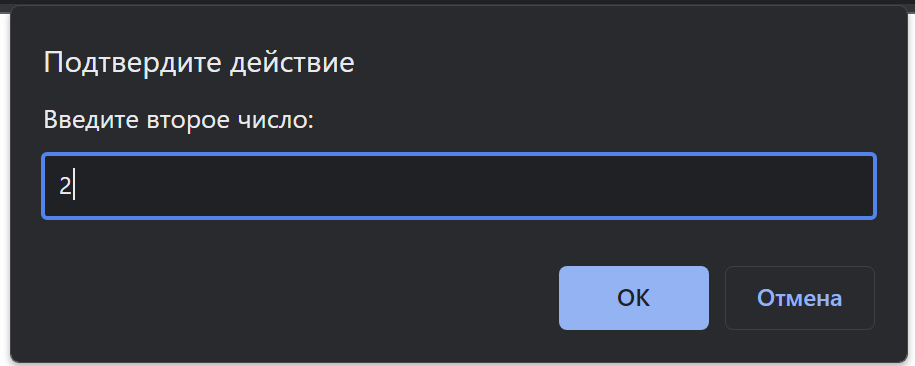
  </script>

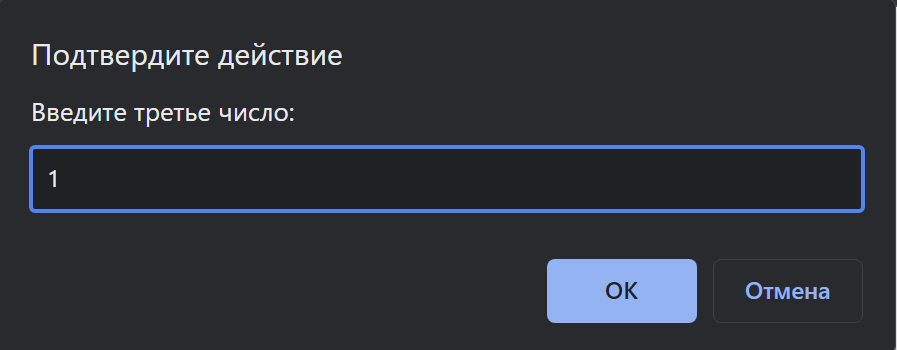
</body>

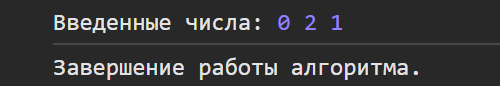
</html>

**Реализация**







****

**Шутка минутка**

