Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики

Отчет по лабораторной работе № 1

по предмету „JavaScript”

Выполнил: студент группы IA2403

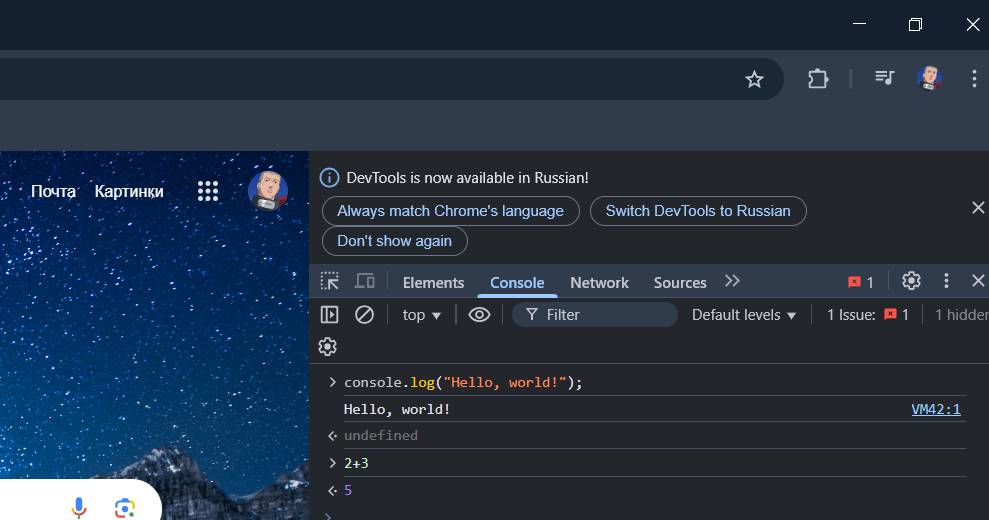
Сакалы Дмитрий

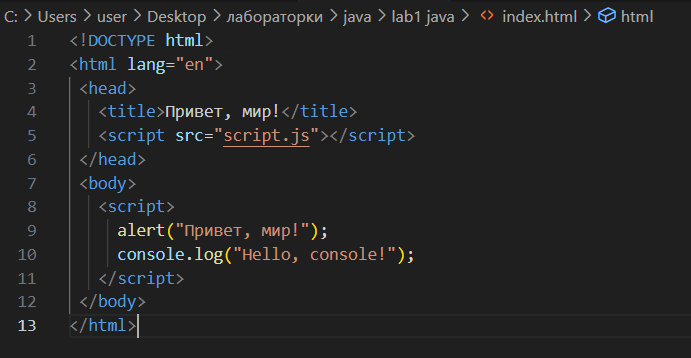
Проверил преподаватель:

Mirinet A., Dr., Conf.univ.

Кишинев, 2024

1 Работа в браузере



2 Работа в коде HTML

Стандартный код для html документа в который подключаем:

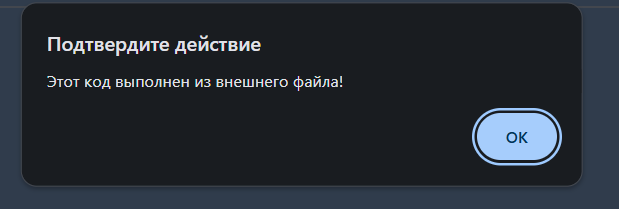
<script src="script.js"></script> - ссылка на js файл со скриптом

alert("Привет, мир!") – выводит сообщение с текстом

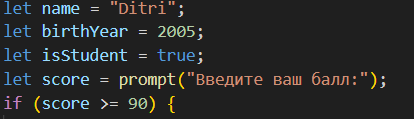
console.log("Hello, console!") – выводит в консоле

Создаем переменные в который вводим имя, год рождения и указываем true если мы студент. Затем выводим на в консоль все переменные.

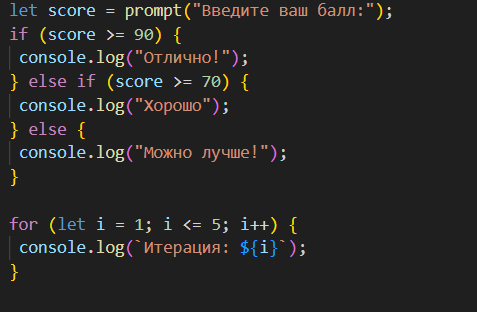


Результат

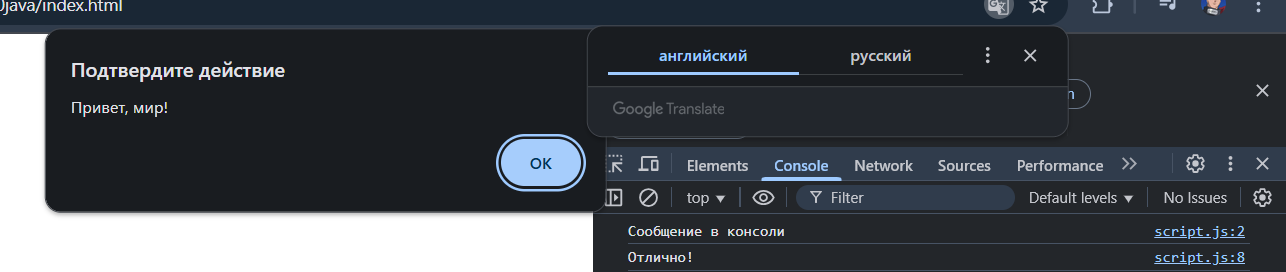
3 Работа с типами данных



Вставляем код который был нам выдан



Результат



После написания своей оценки мне выдало отлично видно что я молодец.

4 Ответы на контрольные вопросы

Вот ответы на ваши контрольные вопросы:

### 1. Чем отличается `var` от `let` и `const`?

- \*\*`var`\*\*:

- Область видимости: переменные, объявленные с помощью `var`, имеют функциональную область видимости (если объявлены внутри функции) или глобальную область видимости (если объявлены вне функции).

- Инициализация: переменная, объявленная через `var`, может быть использована до того, как она была инициализирована (т.е. происходит \*\*hoisting\*\*, то есть поднятие объявления переменной в верхнюю часть области видимости).

- Переназначение: переменные, объявленные через `var`, могут быть переназначены в любом месте.

- \*\*`let`\*\*:

- Область видимости: переменные, объявленные с помощью `let`, имеют блочную область видимости (область видимости ограничена блоком кода, например, внутри фигурных скобок `{}`).

- Инициализация: переменные, объявленные через `let`, не могут быть использованы до их инициализации (так как они не подвержены \*\*hoisting\*\*).

- Переназначение: переменные, объявленные через `let`, могут быть переназначены в пределах своей области видимости.

- \*\*`const`\*\*:

- Область видимости: также имеет блочную область видимости, как и `let`.

- Инициализация: переменные, объявленные через `const`, должны быть инициализированы сразу при объявлении.

- Переназначение: переменные, объявленные через `const`, не могут быть переназначены после их инициализации. Это означает, что саму переменную изменить нельзя, но если она ссылается на объект или массив, то его содержимое может быть изменено.

### 2. Что такое неявное преобразование типов в JavaScript?

Неявное преобразование типов в JavaScript происходит, когда операторы или выражения автоматически приводят значения к нужному типу данных. Это может происходить, например, при использовании операторов сравнения или арифметических операций. JavaScript пытается привести значения к соответствующему типу для выполнения операции.

Пример:

```javascript

let result = '5' + 1; // Строка '5' и число 1 приводятся к строкам, и результат будет '51'

console.log(result); // '51'

let result2 = '5' - 1; // Строка '5' приводится к числу, и результат будет 4

console.log(result2); // 4

```

В данном примере:

- В первом случае, при сложении строки и числа, JavaScript приводит число к строке, а затем выполняет конкатенацию.

- Во втором случае, при вычитании, строка автоматически приводится к числу.

### 3. Как работает оператор `==` в сравнении с `===`?

- \*\*`==` (оператор нестрогого сравнения)\*\*:

- Выполняет \*\*неявное преобразование типов\*\* перед сравнением. Это значит, что если операнды имеют разные типы, JavaScript попытается привести их к одному типу перед сравнением.

Пример:

```javascript

console.log(5 == '5'); // true, так как строка '5' приводится к числу 5

console.log(0 == false); // true, так как 0 приводится к false

```

- \*\*`===` (оператор строгого сравнения)\*\*:

- Выполняет \*\*сравнение без преобразования типов\*\*. То есть, он возвращает `true` только в том случае, если операнды имеют одинаковые значения \*\*и типы\*\*.

Пример:

```javascript

console.log(5 === '5'); // false, типы разные (число и строка)

console.log(0 === false); // false, типы разные (число и логическое значение)

```

Таким образом, `==` может приводить значения к одинаковому типу, а `===` сравнивает значения без изменений типов. В большинстве случаев рекомендуется использовать `===` для более строгих и предсказуемых сравнений.