МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Математики и Информатики

Департамент Информатики

Лабораторная работа № 1

Введение в JavaScript

Проверил: N.Calin

Выполнил: Erebacan Vitali I2502ru

Кишинев, 2026

Цель работы

Познакомиться с основами JavaScript, научиться писать и выполнять код в браузере и в локальной среде, разобраться с базовыми конструкциями языка.

Часть 1. Выполнение кода в браузере
1. Подготовка среды
2. Выполнение кода JavaScript в браузере
3. Создание первой HTML-страницы с JavaScript
4. Подключение внешнего JavaScript-файла
Часть 2. Работа с типами данных
1 Объявление переменных и работа с типами данных.
2 Управление потоком выполнения (условия и циклы)
Часть 3.Контрольные вопросы
Чем отличается var от let и const?
Что такое неявное преобразование типов в JavaScript?
Как работает оператор == в сравнении с ===?

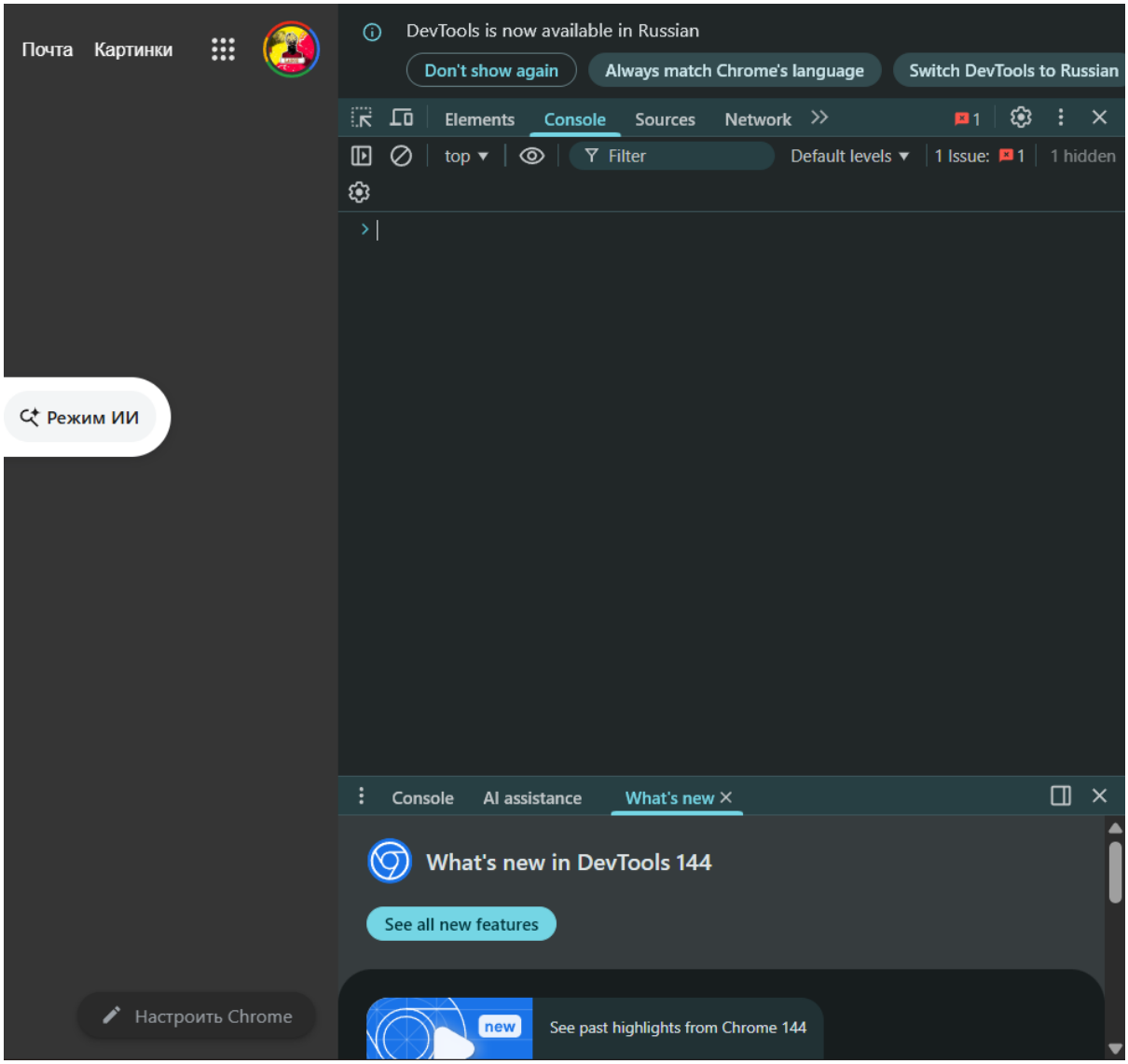
Часть1

Задание 1-2

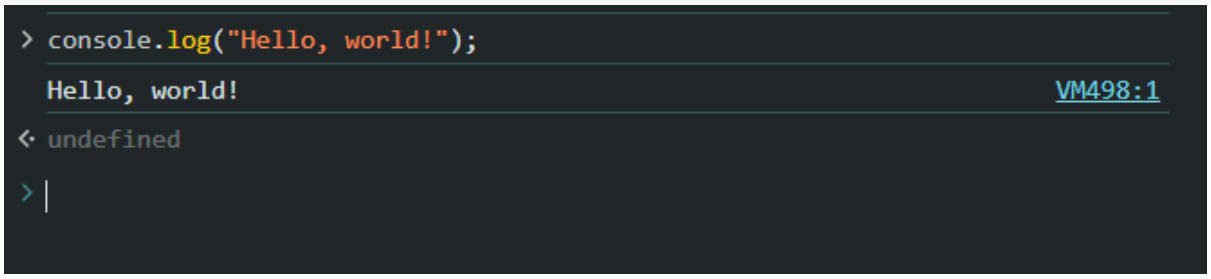
Установил Node.js

node_modules	17.02.2026 20:12	Папка с файлами	
corepack	07.11.2024 15:32	Файл	1 КБ
corepack	07.11.2024 15:32	Сценарий Windo...	1 КБ
install_tools	09.02.2026 22:31	Пакетный файл ...	4 КБ
node	09.02.2026 23:15	Приложение	89 265 КБ
nodevars	07.11.2024 15:33	Пакетный файл ...	1 КБ
npm	07.11.2024 15:37	Файл	3 КБ
npm	07.11.2024 15:37	Сценарий Windo...	1 КБ
npm	24.09.2025 6:03	Сценарий Windo...	2 КБ
npx	07.11.2024 15:37	Файл	3 КБ
npx	07.11.2024 15:37	Сценарий Windo...	1 КБ
npx	24.09.2025 6:03	Сценарий Windo...	2 КБ

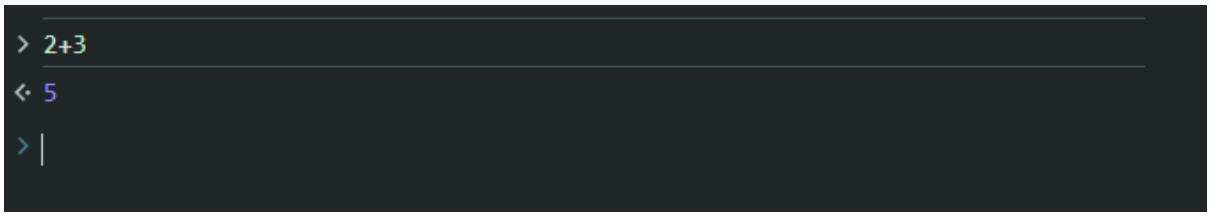
Открываю консоль разработчика



Пишу команду console.log("Hello, world!"); и нажимаю Enter.



Записываю в консоли 2 + 3 и смотрю результат.

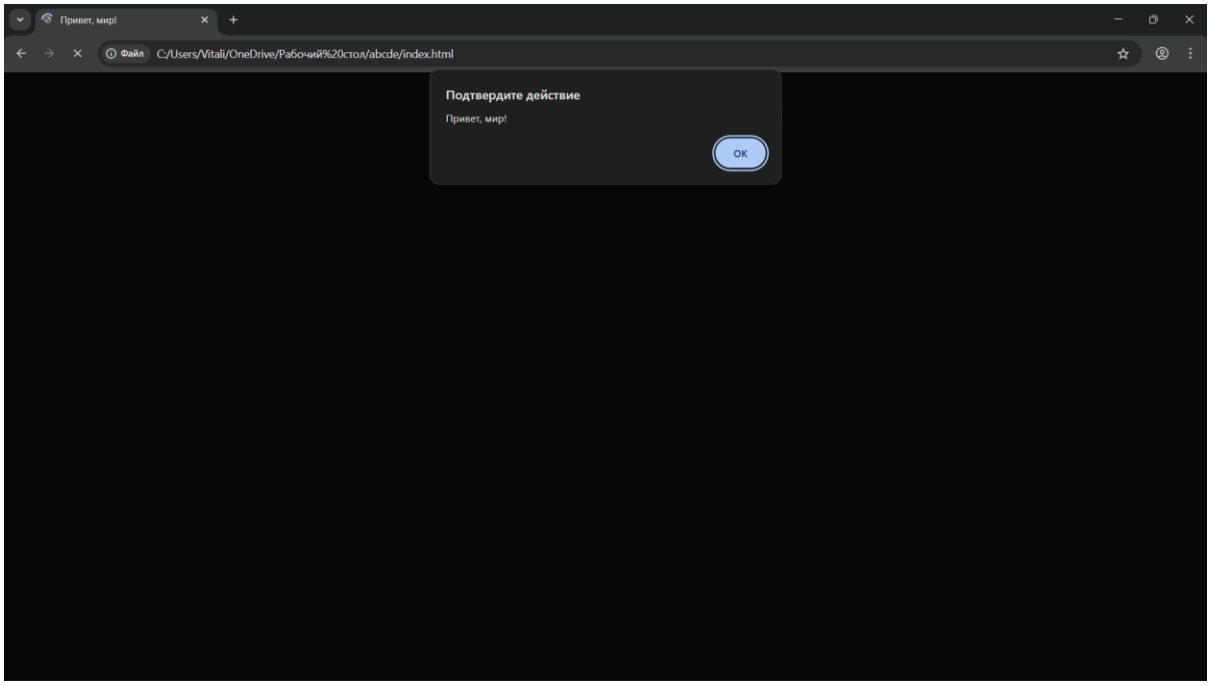


Задание 3 Создание первой HTML-страницы с JavaScript

Создаю файл index.html и вставляю в него следующий код:

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <title>Привет, мир!</title>
 </head>
 <body>
   <script>
     alert("Привет, мир!");
     console.log("Hello, console!");
   </script>
 </body>
</html>
```

Результат:



Внутри тега <script> был размещён JavaScript-код.
Команда alert("Привет, мир!") выводит диалоговое окно при загрузке страницы.
Команда console.log("Hello, console!") выводит сообщение в консоль браузера.
После открытия файла в браузере код успешно выполнился.

Задание 4 Подключение внешнего JavaScript-файла

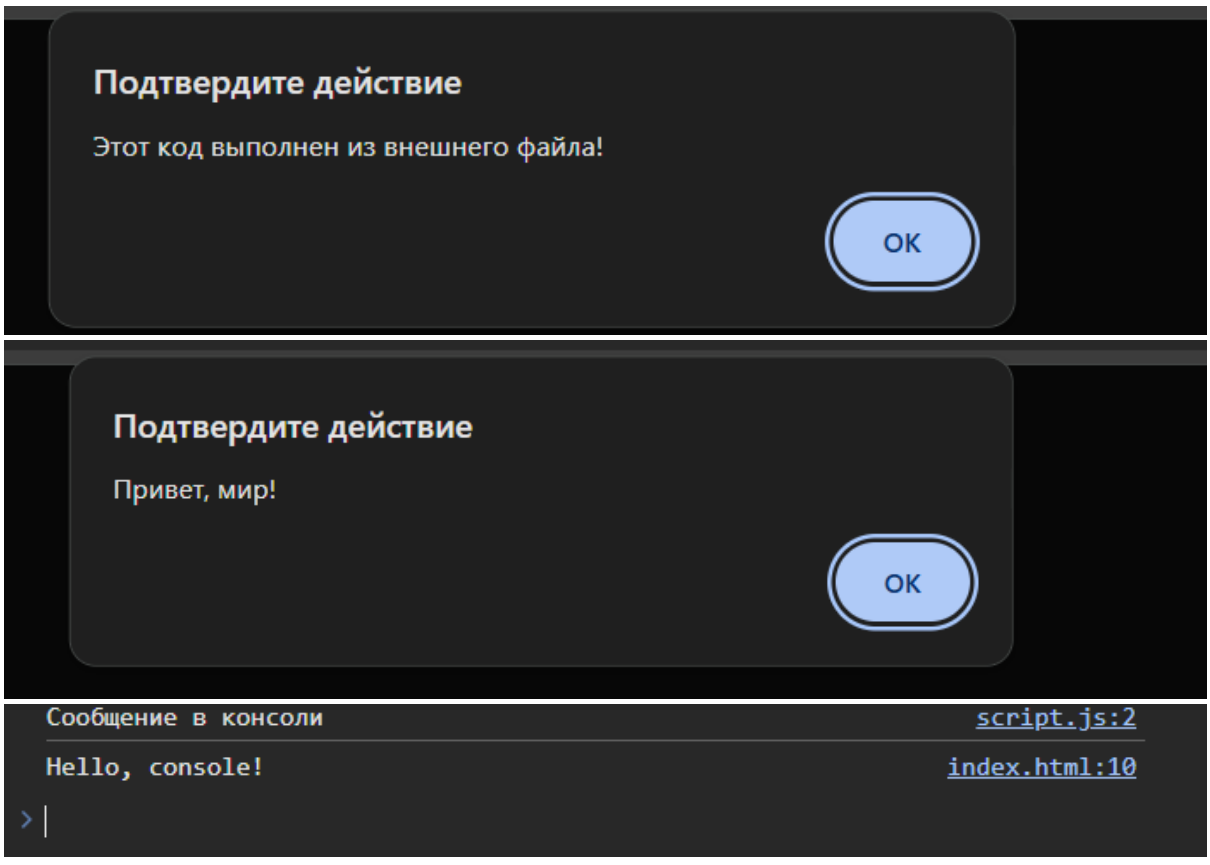
Создаю файл script.js и добавляю в него код:

```
abcde > JS script.js
1   alert("Этот код выполнен из внешнего файла!");
2   console.log("Сообщение в консоли");
```

Подключаю файл в index.html, добавив в <head> <script src="script.js"></script>

```
abcde > index.html > html > head > script
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <title>Привет, мир!</title>
5       <script src="script.js"></script>
6     </head>
7     <body>
8       <script>
9         alert("Привет, мир!");
10        console.log("Hello, console!");
11      </script>
12    </body>
13  </html>
```

Открываю страницу в браузере.



При открытии страницы код из внешнего файла успешно выполнился: появилось всплывающее окно и сообщение в консоли.

Часть 2 Работа с типами данных

Задание 1

В файле script.js создаю несколько переменных:

name - строка с моим именем.
birthYear - число, представляющее год вашего рождения.
isStudent - логическая переменная, указывающая, являетесь ли вы студентом.

```
abcde > JS script.js > ...
1   alert("Этот код выполнен из внешнего файла!");
2   console.log("Сообщение в консоли");
3   let name = "Vitali";
4   let birthYear = 2006;
5   let isStudent = true;
6
7   console.log("Имя:", name);
8   console.log("Год рождения:", birthYear);
9   console.log("Студент:", isStudent);
```

Результат:

```
Сообщение в консоли                                    script.js:2
Имя: Vitali                                            script.js:7
Год рождения: 2006                                     script.js:8
Студент: true                                          script.js:9
Hello, console!                                        index.html:10
>
```

В результате выполнения задания были созданы переменные разных типов и выведены их значения в консоль.

Задание 2

Добавляю следующий код в script.js:

```
abcde > JS script.js > ...
1   let score = prompt("Введите ваш балл:");
2   if (score >= 90) {
3     console.log("Отлично!");
4   } else if (score >= 70) {
5     console.log("Хорошо");
6   } else {
7     console.log("Можно лучше!");
8   }
9
10  for (let i = 1; i <= 5; i++) {
11    console.log(`Итерация: ${i}`);
12  }
```

Результат:

```
Можно лучше!                                    script.js:7
Итерация: 1                                     script.js:11
Итерация: 2                                     script.js:11
Итерация: 3                                     script.js:11
Итерация: 4                                     script.js:11
Итерация: 5                                     script.js:11
Hello, console!                                 index.html:10
>
```

В результате выполнения задания была изучена работа условных операторов и циклов в JavaScript. Пользовательский ввод через prompt() позволяет управлять логикой выполнения, а циклы позволяют многократно выполнять один и тот же код.

Часть 3

1. Чем отличается var от let и const?

Главные отличия касаются области видимости, всплытия (hoisting) и возможности переопределения.

var (устаревший способ):

Область видимости: Функциональная (function-scoped) или глобальная. Если переменная объявлена внутри функции, она видна только там. Если вне — она глобальна.

Всплытие (Hoisting): Переменные var "всплывают" в начало своей области видимости. Вы можете обратиться к переменной до её объявления (значение будет undefined).

Переопределение: Можно переобъявлять сколько угодно раз в одной области видимости.

Привязка к глобальному объекту: В браузере переменные var, объявленные глобально, становятся свойствами объекта window.

let:

Область видимости: Блочная (block-scoped). Переменная существует только внутри блока { ... }, где она объявлена (цикл, условие и т.д.).

Всплытие (Hoisting): Технически всплывает, но не инициализируется. Находится во "временной мёртвой зоне" (Temporal Dead Zone) до момента фактического объявления. Обращение к переменной до её строки вызовет ошибку ReferenceError.

Переопределение: Нельзя повторно объявить переменную с тем же именем в одной области видимости.

const:

Область видимости: Блочная (block-scoped), как и let.

Обязательная инициализация: Должна быть сразу присвоена значение.

Изменяемость: Ссылку изменить нельзя. Если это примитив (число, строка), то это константа. Если это объект или массив, то свойства объекта менять можно, но саму переменную переприсвоить (написать const a = {}; a = [];) нельзя.

Всплытие: Как и у let, есть TDZ, ошибка при доступе до объявления.

2. Что такое неявное преобразование типов в JavaScript?

Это автоматическое преобразование типа значения из одного в другой, которое JavaScript выполняет "под капотом", когда ожидается конкретный тип данных, а передан другой.

Примеры:

Строковое преобразование: Когда вы используете оператор + со строкой.

console.log("5" + 3); // "53" (число 3 стало строкой)

console.log("hello" + true); // "hellotrue" (true стало строкой)

Численное преобразование: Когда вы используете математические операторы (-, *, /) или сравнения (>, <).

console.log("10" - 5); // 5 (строка "10" стала числом 10)

console.log("10" * "2"); // 20 (обе строки стали числами)

console.log("яблоко" - 1); // NaN (Not a Number, строка не смогла стать числом)

console.log(true + true); // 2 (true преобразовалось в 1)

Логическое преобразование: В контексте условий (if, while, &&, ||).

Falsy значения (становятся false): false, 0, "" (пустая строка), null, undefined, NaN.

Truthy значения (становятся true): всё остальное (например, "hello", 1, [], {}).

```
if ("hello") { // "hello" преобразуется в true
    console.log("Это выполнится");
}
```

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

Главное отличие в том, учитывается ли тип данных при сравнении.

=== (Строгое равенство):

Сначала сравнивает типы операндов.

Если типы разные — сразу возвращает false.

Если типы одинаковые, тогда сравнивает значения.

Не выполняет неявное преобразование типов.

== (Нестрогое / Абстрактное равенство):

Если типы операндов одинаковы — работает как === (сравнивает значения).

Если типы разные — JavaScript пытается привести их к одному типу (обычно к числу) и затем сравнить.

Выполняет неявное преобразование типов.

```
console.log(5 == "5");   // true (строка "5" преобразовалась в число 5)
console.log(5 === "5");  // false (разные типы: number и string)

console.log(0 == false); // true (false стал 0, 0 == 0)
console.log(0 === false); // false (number vs boolean)

console.log(null == undefined); // true (спецправило: они равны друг другу)
console.log(null === undefined); // false (разные типы)

console.log(" " == 0); // true (пустая строка стала 0)
console.log(" " === 0); // false
```

Рекомендация: В современном JavaScript практически всегда рекомендуется использовать === (строгое сравнение), чтобы избежать неочевидных ошибок, связанных с неявным преобразованием.