

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

Нижегородский государственный технический
университет им. Р.Е.Алексеева

Институт радиоэлектроники и информационных технологий
Кафедра прикладной математики и информатики

Отчёт № 5

«Язык программирования JavaScript. Основы»

по дисциплине

web-разработка

Выполнил:

студент группы 23-ПМ-1

Блинов А. С.

Проверил:

Курушин Е. А.

Нижний Новгород

2025 год

Оглавление

Цели и задачи.....	3
Теоретическая справка.....	4
Реализация.....	8
Вывод.....	11
Приложение	12

Цели и задачи

В ходе данной работы были поставлены следующая цель:

Освоить основы работы с переменными, массивами, функциями и объектами в JavaScript, включая их создание, модификацию и взаимодействие между ними.

Для из реализации нужно выполнить следующие задачи:

1. Создать несколько переменных с разными типами данных. Провести несколько манипуляций с каждой переменной (математические операции, вывод в консоль, alert и прочее).

2. Создайте массив со случайными числами (не менее 10) и «переберите» его разными способами (for, for ... in, forEach).

3. Создайте функцию, которая проверяет, имеется ли в массиве (созданном в шаге 2) переданное ей в качестве аргумента число и возвращает соответствующее значение true или false.

Если переданного числа в массиве нет, то его необходимо добавить.

4. Создайте объект с несколькими свойствами (не менее 3-х с разными типами данных).

5. Напиши стрелочную функцию, которая выводит в консоль значение свойства по переданному ключу объекта.

Теоретическая справка

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который широко используется для создания интерактивных веб-страниц. Он работает на стороне клиента (в браузере) и сервера (Node.js). В этом материале рассмотрим ключевые теоретические аспекты JavaScript, включая типы данных, переменные, функции и объекты.

1. Что такое ECMAScript?

ECMAScript — это стандарт, на котором основан JavaScript. Он определяет синтаксис, семантику и основные возможности языка.

- JavaScript — это реализация ECMAScript с дополнительными возможностями (например, DOM-взаимодействие в браузере).

- Новые версии ECMAScript (ES6, ES7 и т. д.) добавляют современные фичи (`let/const`, стрелочные функции, классы и др.).

2. Что такое типизация и какую типизацию использует JavaScript?

Типизация — это способ классификации данных в языке программирования.

- JavaScript использует динамическую слабую типизацию:

- Динамическая — тип переменной определяется во время выполнения.

- Слабая — язык автоматически преобразует типы при операциях (например: `"5" + 2 = "52"`).

3. Какие типы данных доступны в JavaScript?

В JavaScript есть примитивные и ссылочные типы данных:

Примитивные (immutable, передаются по значению):

- number (числа, включая NaN, Infinity)
- string (строки)
- boolean (true/false)
- null (отсутствие значения)
- undefined (неопределённое значение)
- symbol` (уникальные идентификаторы, ES6+)
- bigint (большие целые числа, ES11+)

Ссылочные (передаются по ссылке):

- object (объекты, массивы, функции, даты и др.)

4. В чем отличие let от var?

Критерий	var	let
Область видимости	Функция (или глобальная)	Блок («{ }»)
Поднятие (hoisting)	Инициализируется как «undefined»	Недоступна до объявления (временная мёртвая зона)
Переопределение	Можно объявить дважды	Ошибка при повторном объявлении

Таблица 1 – Отличие let и var

Пример:

```
if (true) {
    var x = 10; // видно вне блока
```

```
let y = 20; // видно только внутри блока  
}
```

```
console.log(x); // 10
```

```
console.log(y); // ReferenceError
```

5. Что такое стрелочная функция и чем она отличается от обычной?

Стрелочная функция (`=>`) — это сокращённый синтаксис для объявления функций в ES6+.

Отличия от обычной функции (`function`):

1. Нет своего «`this`» (берёт из внешнего контекста).
2. Нельзя использовать как конструктор (нет «`new`»).
3. Нет «`arguments`» (используется `rest`-оператор «`...args`»).
4. Более короткий синтаксис.

Пример:

```
const sum = (a, b) => a + b; // return не нужен, если одна строка
```

```
const greet = () => console.log("Hello!");
```

6. Что такое объект и как он хранит данные? Как добавить свойства и методы?

Объект — это структура данных, хранящая пары ключ-значение.

Создание объекта:

```
const user = {  
  name: "Alice", // свойство  
  age: 25,  
  sayHi() {      // метод
```

```
console.log(`Hi, I'm ${this.name}`);  
  
}  
  
};
```

Добавление свойств и методов:

1. Через точку:

```
user.city = "Berlin";  
  
user.logAge = function() { console.log(this.age); };
```

2. Через квадратные скобки (если ключ динамический):

```
const key = "isAdmin";  
  
user[key] = true;
```

Хранение данных:

- Объекты хранят данные по ссылке (изменение копии влияет на оригинал).

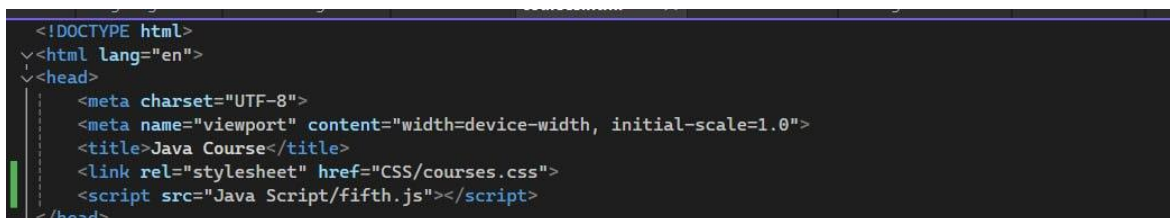
- Для клонирования используется «Object.assign()» или «...spread»:

```
const clone = { ...user };
```

Реализация

В этой работе были выполнены задачи на JavaScript, включая работу с переменными, массивами, функциями и объектами. Каждый шаг сопровождается комментариями, объясняющими логику кода.

Сначала был создан html-файл и подключён js-файл. Представлено на рисунке 1.

A screenshot of a code editor showing the structure of an HTML file. The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Java Course</title>
    <link rel="stylesheet" href="CSS/courses.css">
    <script src="Java Script/fifth.js"></script>
  </head>
```

Рисунок 1 – html-файл

1. Объявление переменных и операции с ними

Были созданы переменные разных типов и выполним с ними базовые операции.

2. Создание и перебор массива случайных чисел

Был сгенерирован массив из 10 случайных чисел и выполнен перебор его тремя способами.

for — классический цикл с доступом по индексу.

for...in — перебирает ключи (индексы), но не рекомендуется для массивов (лучше для объектов).

forEach — удобный метод для итерации без управления индексом.

3. Функция для проверки и добавления числа в массив

Была написана функция, которая проверяет наличие числа в массиве и добавляет его, если отсутствует.

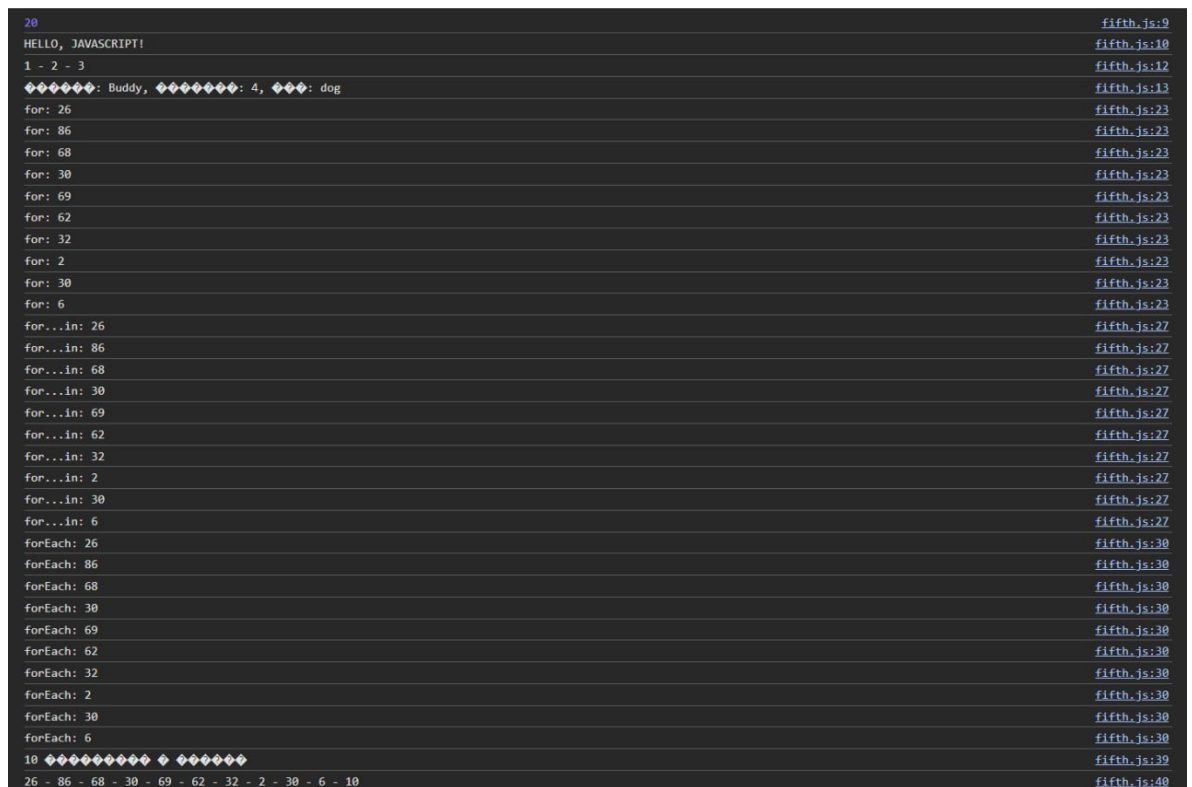
4. Создание объекта с разными свойствами

Был создан объект, описывающий питомца.

5. Стрелочная функция для получения значения свойства

Была написана функция, которая выводит значение свойства объекта по ключу.

Программа была открыта в браузере и пример её работы представлен на рисунках 2 и 3.



```
20
HELLO, JAVASCRIPT!
1 - 2 - 3
Buddy, 4, dog
for: 26
for: 86
for: 68
for: 30
for: 69
for: 62
for: 32
for: 2
for: 30
for: 6
for...in: 26
for...in: 86
for...in: 68
for...in: 30
for...in: 69
for...in: 62
for...in: 32
for...in: 2
for...in: 30
for...in: 6
forEach: 26
forEach: 86
forEach: 68
forEach: 30
forEach: 69
forEach: 62
forEach: 32
forEach: 2
forEach: 30
forEach: 6
10
26 - 86 - 68 - 30 - 69 - 62 - 32 - 2 - 30 - 6 - 10
```

Рисунок 2 – Пример работы программы

26 - 86 - 68 - 30 - 69 - 62 - 32 - 2 - 30 - 6 - 10	fifth.js:40
Buddy	fifth.js:55
4	fifth.js:55
dog	fifth.js:55

Рисунок 3 – Пример работы программы

Вывод

В ходе выполнения задач мы закрепили ключевые аспекты JavaScript:

- Работу с переменными разных типов (числа, строки, булевы значения, массивы, объекты).
- Манипуляции с массивами (генерация, перебор, модификация).
- Создание функций для проверки и изменения данных.
- Использование объектов для структурированного хранения информации.
- Применение стрелочных функций для более лаконичного кода.

Этот практический пример показывает, как эффективно использовать базовые конструкции JavaScript для решения типовых задач. Полученные навыки станут основой для изучения более сложных тем, таких как асинхронность, DOM-манипуляции и работа с API.

Приложение

// 1. Объявление переменных с разными типами данных

```
let numValue = 10;
```

```
let textValue = "Hello, JavaScript!";
```

```
let boolValue = true;
```

```
let numArray = [1, 2, 3];
```

```
let petInfo = { name: "Buddy", age: 4, type: "dog" };
```

// Операции с переменными

```
console.log(numValue * 2);
```

```
console.log(textValue.toUpperCase());
```

```
alert(boolValue ? "Верно!" : "Неверно!");
```

```
console.log(numArray.join(" - "));
```

```
console.log(Кличка: ${petInfo.name}, Возраст: ${petInfo.age}, Тип:  
${petInfo.type});
```

// 2. Создание массива случайных чисел

```
let randomNums = [];
```

```
for (let i = 0; i < 10; i++) {
```

```
    randomNums.push(Math.floor(Math.random() * 100));
```

```
}
```

// Различные способы перебора массива

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

```
    console.log(for: ${randomNums[i]});
```

```
}
```

```
for (let idx in randomNums) {
```

```
    console.log(for...in: ${randomNums[idx]});  
  }  
}
```

```
randomNums.forEach(value => console.log(forEach: ${value}));
```

```
// 3. Функция для проверки и добавления числа в массив
```

```
const findOrAddNumber = (arr, num) => {  
  if (arr.includes(num)) {  
    console.log(`${num} уже присутствует);  
    return true;  
  } else {  
    arr.push(num);  
    console.log(`${num} добавлено в массив);  
    console.log(randomNums.join(" - "));  
    return false;  
  }  
};
```

```
findOrAddNumber(randomNums, 10);
```

```
// 4. Создание объекта с различными свойствами
```

```
let pet = {  
  name: "Buddy",  
  age: 4,  
  type: "dog"  
};
```

```
// 5. Стрелочная функция для получения значения свойства по ключу
```

```
const getValue = (obj, prop) => console.log(obj[prop]);
```

```
getValue(pet, "name");
```

```
getValue(pet, "age");
```

```
getValue(pet, "type");
```