

## Лабораторная работа 4

### Основы JavaScript

Цель: освоить базовый синтаксис javascript

- Лабораторная работа сдается преподавателю лично студентом.
- Сдача лабораторной работы заключается в том, что студент отвечает на вопросы преподавателя по заданиям, а также показывает и комментирует результаты самостоятельно выполненных заданий.
- Всё что написано в заданиях «изучите, узнайте и прочее» нужно изучить и для себя зафиксировать, так как преподаватель может спросить об этом при сдаче лабораторной работы.

(!) В качестве онлайн-справочников можно пользоваться <http://javascript.ru/> или <https://learn.javascript.ru/>

(!) В качестве редактора кода можно воспользоваться Visual Studio или Code Studio (Создать файл, выбрать нужный тип .html или .js) или ресурсом <https://jsfiddle.net/>

### ЧАСТЬ 1. Переменные

**Задание 1.1.** Вспомните особенности поведения **var** и **let**.

- а) Запустите следующий код. Объясните результат, почему **var** работает иначе, чем **let**?

```
function testVar() {  
  if (true) {  
    var x = 10;  
  }  
  console.log(x); // Что выведет?  
}  
  
function testLet() {  
  if (true) {  
    let y = 20;  
  }  
  console.log(y); // Что выведет?  
}  
  
testVar();  
testLet();
```

- б) Выполните следующий код и объясните, почему **var** и **let** позволяют переназначать значения, а **const** — нет. Добавьте пример объекта с **const** и покажите, что его содержимое можно изменять

```
var a = 10;  
let b = 20;  
const c = 30;  
  
a = 15; // Что произойдёт?  
b = 25; // Что произойдёт?  
c = 35; // Что произойдёт?  
  
console.log(a, b, c);
```

- с) Запустите следующий код. Объясните разницу в поведении **var** и **let** при использовании до их объявления

```
console.log(x); // Что выведет?  
var x = 10;  
  
console.log(y); // Что выведет?  
let y = 20
```

**Задание 1.2.** Создайте следующие ниже таблицы посредством JavaScript-кода и закрасьте их ячейки в соответствующие цвета. Можно воспользоваться методом **write()** объекта **document**. Метод **document.write()** используется для записи текста или HTML-кода непосредственно в документ. Метод добавляет содержимое в текущую точку HTML-документа, где находится парсер (обработчик).

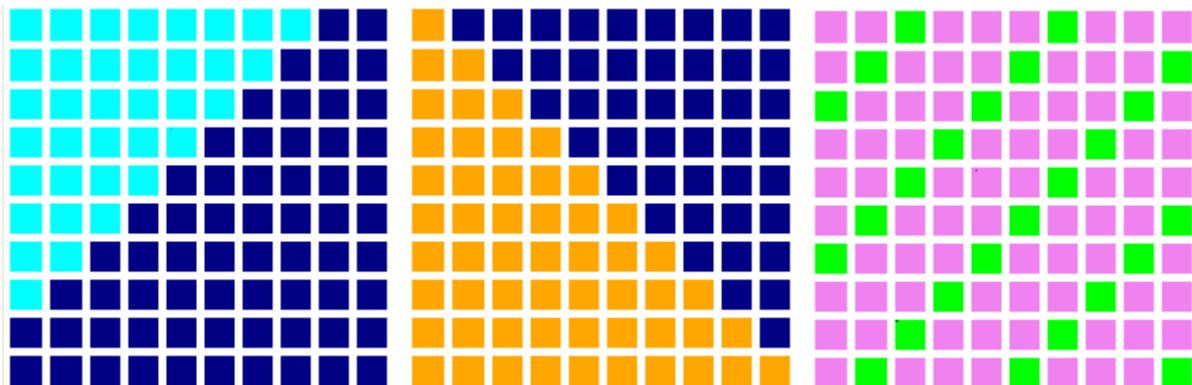
`document.write(строка);`

где **строка** - текст или HTML-код, который нужно записать в документе.

Пример:

Код	Отображение
<pre>&lt;p&gt;Первый параграф&lt;/p&gt; &lt;script&gt;   document.write('&lt;p&gt;Второй параграф&lt;/p&gt;'); &lt;/script&gt; &lt;p&gt;Третий параграф&lt;/p&gt;</pre>	Первый параграф Второй параграф Третий параграф

Метод является устаревшим и может замедлить загрузку страницы. Альтернативой является использование современных методов работы с DOM.



**Задание 1.3.** Дан массив целых чисел. Напишите код, который возвращает **true**, если в массиве встречаются дубликаты, и **false** – в обратном случае. Ввод входного массива сделайте через функцию `prompt()`.

**Задание 1.4.** Палиндромы в ДНК — это участки, которые читаются одинаково в обоих направлениях (например, "ATGCAT"). Напишите программу, которая находит все палиндромные подстроки заданной длины в последовательности ДНК.

**Задание 1.5.** Напишите код для сравнения двух последовательностей ДНК и выявления мутаций, то есть различий между ними. Для сравнения последовательностей используйте расстояние Левенштейна.

**Задание 1.6.** Дан массив объектов, где каждый объект описывает человека и его друзей. Каждый объект имеет следующую структуру:

```
{  
  name: "Имя", // Имя человека  
  friends: ["Имя друга 1", "Имя друга 2", ...] // Массив имён друзей
```

```
}
```

Напишите функцию, которая определяет по введенным двум именам и числу **n** обладают ли два человека дружбой в **n** степени (по аналогии с "n-м рукопожатием"). Степень дружбы определяется как минимальное количество "связей" (друзей) между двумя людьми. Функция возвращает **true**, если связь найдена за указанное **n** или меньшее число шагов. И возвращает **false**, если общих друзей нет за указанное число шагов.

**Задание 1.7.** Задается месяц и год, требуется вывести количество рабочих дней в этом месяце. День считается рабочим, если он не выходной и не праздничный. Праздничные дни задаются массивом (число и месяц), а выходные дни – субботы и воскресенья.

**Задание 1.8.** Компания занимается доставкой товаров. Задается список городов **cities**, между которыми можно перемещаться, и расстояния между ними. Также есть список заказов **orders**, каждый из которых содержит информацию о городе отправления, городе назначения и весе груза.

Необходимо написать функцию, которая:

1. Определяет, существует ли маршрут между городами отправления и назначения для каждого заказа.
2. Если маршрут существует, вычисляет **общее расстояние для доставки всех заказов**.
3. Если маршрут не существует, возвращает **список заказов, которые невозможно доставить**.

Функция должна возвращать объект с двумя полями:

- **totalDistance**: общее расстояние для доставки всех возможных заказов.
- **undeliverableOrders**: массив заказов, которые невозможно доставить.

(!) Обратите внимание, что каждая посылка перевозится отдельно.

**Пример входных данных:**

```
const cities = [
  { from: "Москва", to: "Санкт-Петербург", distance: 700 },
  { from: "Москва", to: "Нижний Новгород", distance: 400 },
  { from: "Санкт-Петербург", to: "Нижний Новгород", distance: 900 },
  { from: "Нижний Новгород", to: "Казань", distance: 300 }
];
const orders = [
  { id: 1, from: "Москва", to: "Нижний Новгород", weight: 50 },
  { id: 2, from: "Москва", to: "Казань", weight: 30 },
  { id: 3, from: "Санкт-Петербург", to: "Казань", weight: 20 }
];
```

Ожидаемый результат:

```
{
  totalDistance: 2300, // Москва → Нижний Новгород (400) + Москва → Казань (через Нижний Новгород, 400 + 300) + Санкт-Петербург → Казань (через Нижний Новгород, 900 + 300)
  undeliverableOrders: []
}
```

## ЧАСТЬ 2. Массивы и их методы

**Задача 2.1.** Напишите функцию `filterByRange(arr, a, b)`, которая принимает массив `arr`, ищет элементы со значениями больше или равными `a` и меньше или равными `b` и возвращает результат в виде массива. **Функция должна возвращать новый массив и не изменять исходный.**

**Задача 2.2.** Напишите функцию `filterByRange2(arr, a, b)`, которая принимает массив `arr` и удаляет из него все значения кроме тех, которые находятся между `a` и `b`, алогично как в предыдущей задаче, но **функция должна изменять принимаемый массив и ничего не возвращать**.

### Задача 2.3. Есть массив объектов с полями name, age, from.

- Напишите функцию, которая вычисляет среднее значение поля `age`.
- Напишите функцию, которая возвращает значения поля `name` объектов массива, у которых значение поля `age` больше среднего значения `age`, вычисленного по всему массиву.
- Напишите функцию, которая возвращает значение `from`, наиболее часто встречающееся в массиве, и соответствующие значения поля `name` (те, у кого `from` равен этому наиболее часто встречающемуся значению).
- Напишите функцию, которая возвращает список объектов, отсортированный по полю, которое задает пользователь. Если пользователь указал неверное значение поля, то должно выводиться сообщение об этом.

Пример массива (можно скопировать)

```
[
    {"name": "Alex", "age": 30, "from": "Moscow"},
    {"name": "Sofa", "age": 22, "from": "Kazan"},
    {"name": "Ilya", "age": 26, "from": "Piter"},
    {"name": "Kolya", "age": 30, "from": "Kirov"},
    {"name": "Alex", "age": 18, "from": "Piter"},
    {"name": "Boris", "age": 38, "from": "Kirov"},
    {"name": "Olga", "age": 16, "from": "Kirov"},
    {"name": "Alex", "age": 42, "from": "Moscow"},
]
```

**Задача 2.4.** Дан массив (можно скопировать):

```
[{
  model: "Bert",
  architecture: "encoder",
  application: ["classification", "masked prediction", "embedding"]
},
{
  model: "GPT",
  architecture: "decoder",
  application: ["text generation", "question answering", "instructed generation", "classification"]
},
{
  model: "T5",
  architecture: "encoder-decoder",
  application: ["classification", "summarization", "question answering"]
}]
```

Требуется пробежаться по всем элементам массива и создать новый массив из строковых значений, которые встречаются в поле `application`, **исключая дублирование значений**.

То есть результатом должен быть массив (порядок элементов неважен):

```
["classification", "masked prediction", "embedding", "text generation", "question  
answering", "instructed generation", "summarization"]
```

**Задание 2.5.** Дан массив (можно скопировать)

```
[{
  name: "Alice",
  code: 5005,
  home: {roof:"yellow", door:"yellow", wall:"yellow"}
},
{
  name: "Bob",
  code: 5105,
  home: {roof:"green", door:"grey", wall:"grey"}
},
{
  name: "Charlie",
  code: 5205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Diana",
  code: 5305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Eve",
  code: 5405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Frank",
  code: 5505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Grace",
  code: 5605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Heidi",
  code: 5705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Ivan",
  code: 5805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Julia",
  code: 5905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Kevin",
  code: 6005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Liam",
  code: 6105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Mia",
  code: 6205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Noah",
  code: 6305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Olivia",
  code: 6405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Peter",
  code: 6505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Quinn",
  code: 6605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Rory",
  code: 6705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Sam",
  code: 6805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Tina",
  code: 6905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Uma",
  code: 7005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Victor",
  code: 7105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Wendy",
  code: 7205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Xavier",
  code: 7305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Yara",
  code: 7405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Zoe",
  code: 7505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Adam",
  code: 7605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Ben",
  code: 7705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Chloe",
  code: 7805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "David",
  code: 7905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Emily",
  code: 8005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Fiona",
  code: 8105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "George",
  code: 8205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Hannah",
  code: 8305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Ian",
  code: 8405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Jack",
  code: 8505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Karen",
  code: 8605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Leo",
  code: 8705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Megan",
  code: 8805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Nathan",
  code: 8905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Oliver",
  code: 9005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Patricia",
  code: 9105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Quinn",
  code: 9205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Robert",
  code: 9305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Sophia",
  code: 9405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Theodore",
  code: 9505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Uma",
  code: 9605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Victor",
  code: 9705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Wendy",
  code: 9805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Xavier",
  code: 9905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Yara",
  code: 10005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Zoe",
  code: 10105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Adam",
  code: 10205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Ben",
  code: 10305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Chloe",
  code: 10405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "David",
  code: 10505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Emily",
  code: 10605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Frank",
  code: 10705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Grace",
  code: 10805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Heidi",
  code: 10905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Ivan",
  code: 11005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Julia",
  code: 11105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Kevin",
  code: 11205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Liam",
  code: 11305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Mia",
  code: 11405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Noah",
  code: 11505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Olivia",
  code: 11605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Peter",
  code: 11705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Quinn",
  code: 11805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Rory",
  code: 11905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Sam",
  code: 12005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Tina",
  code: 12105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Uma",
  code: 12205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Victor",
  code: 12305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Wendy",
  code: 12405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Xavier",
  code: 12505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Yara",
  code: 12605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Zoe",
  code: 12705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Adam",
  code: 12805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Ben",
  code: 12905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Chloe",
  code: 13005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "David",
  code: 13105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Emily",
  code: 13205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Frank",
  code: 13305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Grace",
  code: 13405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Heidi",
  code: 13505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Ivan",
  code: 13605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Julia",
  code: 13705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Kevin",
  code: 13805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Liam",
  code: 13905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Mia",
  code: 14005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Noah",
  code: 14105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Olivia",
  code: 14205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Peter",
  code: 14305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Quinn",
  code: 14405,
  home: {roof:"purple", door:"purple", wall:"purple"}
},
{
  name: "Rory",
  code: 14505,
  home: {roof:"brown", door:"brown", wall:"brown"}
},
{
  name: "Sam",
  code: 14605,
  home: {roof:"pink", door:"pink", wall:"pink"}
},
{
  name: "Tina",
  code: 14705,
  home: {roof:"grey", door:"grey", wall:"grey"}
},
{
  name: "Uma",
  code: 14805,
  home: {roof:"black", door:"black", wall:"black"}
},
{
  name: "Victor",
  code: 14905,
  home: {roof:"white", door:"white", wall:"white"}
},
{
  name: "Wendy",
  code: 15005,
  home: {roof:"orange", door:"orange", wall:"orange"}
},
{
  name: "Xavier",
  code: 15105,
  home: {roof:"green", door:"green", wall:"green"}
},
{
  name: "Yara",
  code: 15205,
  home: {roof:"blue", door:"blue", wall:"blue"}
},
{
  name: "Zoe",
  code: 15305,
  home: {roof:"red", door:"red", wall:"red"}
},
{
  name: "Adam",
  code: 1540
```

```
    name: "Carol",
    code: 5015,
    home: {roof:"yellow", door:"white", wall:"yellow"}
  },
  {
    name: "Dave",
    code: 5115,
    home: {roof:"pink", door:"red", wall:"yellow"}
  }
]
```

Требуется сформировать массив из значений поля `code` тех объектов, у которых в поле `home` объект имеет два одинаковых цвета (то есть совпадают минимум два цвета из полей `roof`, `door` и `wall`).

**Задание 2.6.** Дан массив аналогичный массиву из предыдущего задания, удалите из массива элементы, у которых в поле `home` встречается цвет, заданный пользователем (через диалоговое окно). Если цвета не будет, то требуется вывести окно с сообщением, что такого объекта в массиве нет.

**Задание 2.7.** Пусть есть массив чисел или строк. Например:

```
const data = [1, 2, 3, 2, 4, 2, 5, 3, 3, 3];
```

Найдите элемент, который встречается чаще всего. Если таких элементов несколько, то выведите их все в виде списка, где каждый элемент списка содержит значение и свои индексы в исходном массиве.

**Задание 2.8.** Пусть есть массив объектов, представляющих фильмы. Каждый объект содержит следующие свойства:

```
title (название фильма),
genre (жанр фильма, например, "Драма", "Комедия", "Боевик"),
rating (рейтинг фильма от 1 до 10).
```

Выполните:

- Отсортировать фильмы по убыванию рейтинга.
- Создать новый массив, содержащий только фильмы с рейтингом выше 7.
- Сгруппировать фильмы по жанрам.