



ФКН ВШЭ

Москва 2024

Введение в JavaScript



Брэндон Эйх разработал JavaScript для Netscape 1995





Основные факты

- Динамическая и слабая типизация
- Используется для разработки веб-сайтов, серверов, приложений и баз данных
- Более 60% разработчиков используют JavaScript (согласно данным опроса разработчиков Stack Overflow за 2024 год)



Используется для разработки веб-сайтов, серверов, приложений и баз данных

JavaScript — универсальный язык программирования. Благодаря этому он нашел применение во многих областях:

- Веб-сайты
- Серверы
- Приложения
- Базы данных





Инструменты разработки

Редакторы кода:



Отладка:



Стиль и
форматирование:





Установка Node.js



ОСНОВЫ ЯЗЫКА

Ключевые слова:

- **let**: переменная, значение которой можно изменять
- **const**: для неизменяемых значений
- **var**: устаревшее ключевое слово с глобальной или функциональной областью видимости



Ключевое слово let

Используется для объявления переменной с возможностью изменения значения. Область видимости — блок (между {}).

- ✓ `let name; // Объявление переменной`
- ✓ `name = "Петр"; // Присвоение значения переменной`
`// Чтение значения переменной`
`console.log(name); // "Петр"`
- ✓ `let surname = "Иванов"; // Можно объединить объявление и присваивание значения`
- ✗ `// Нельзя объявить ещё одну переменную с таким же названием`
`let name;`



Ключевое слово `const`

Для объявления констант, значение которых не может быть изменено. Область видимости — блок.

✓ `const SECONDS_IN_MINUTE = 60; // Объявление константы и присваивание значения`

✓ `// Чтение значения константы
console.log(SECONDS_IN_MINUTE); // 60`

✗ `// Объявить константу без значения нельзя
const SECONDS_IN_MINUTE;`

✗ `// Изменить значение константы нельзя
SECONDS_IN_MINUTE = 50;`

✗ `// Нельзя объявить ещё одну переменную с таким же названием
const SECONDS_IN_MINUTE = 50;`



Ключевое слово var

Используется для объявления переменной с функциональной или глобальной областью видимости. Рекомендуется избегать использования var, так как let и const безопаснее.

```
var name; // Объявление переменной
```

```
name = "Петр"; // Присвоение значения переменной
```

```
// Чтение значения переменной  
console.log(name); // "Петр"
```

```
var surname = "Иванов"; // Можно объединить объявление и присваивание значения
```

```
var name; // Можно объявить ещё одну переменную с таким же названием
```



	var	let	const
Можно объявить несколько раз	✗	✗	✓
Можно переприсвоить	✓	✗	✓
Обязательно задать значение	✗	✓	✗



Правила именования

- Имя должно начинаться с:
 - Буквы (a–z, A–Z).
 - Символов _ (подчеркивание) или \$ (знак доллара).
- Имя может содержать буквы, цифры (0–9), _ и \$.
- Имя не может быть зарезервированным словом языка (например, let, function, return).
- Регистр имеет значение (myVariable и myvariable — разные переменные).



Правила именования

```
let name = "Иван"; // Верно  
let _private = 42; // Верно  
let $dollar = 100; // Верно  
let age1 = 18; // Верно
```

```
let 1number = 10; // Ошибка: имя не может начинаться с цифры  
let my-variable = 5; // Ошибка: тире недопустимо  
let function = "test"; // Ошибка: зарезервированное слово
```







Зарезервированные слова

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield



Типы данных (часть 1)

	string	number	bigInt	boolean
Тип	Строка	Число	Большое целое	Логический
typeof	"string"	"number"	"bigint"	"boolean"
Пример	"привет"	3.14 1.1e32 0xFF	1924924124n	true false
Примитив				



Примитив?

Это не объект, а конкретное значение

**Не может быть изменен
(при изменении создается новый)**

**Не имеет методов.
При вызове метода создается объект-обертка**

```
const name = "batman";  
console.log(name); // "batman"
```

// Методы строки не мутируют строку

```
name.toUpperCase();
```

```
console.log(name); // "batman"
```

// Методы строки возвращают новую строку

```
const upperName = name.toUpperCase();
```

```
console.log(upperName); // "BATMAN"
```





// Под капотом создаётся объект-обёртка

```
const upperName2 = new String(name).toUpperCase();
```

```
console.log(upperName2); // "BATMAN"
```




Типы данных (часть 2)

	undefined	null	symbol	object
Тип	Неопределенный	Null	Символ	Структура
typeof	"undefined"	"object"	"symbol"	"object"
Пример	undefined	null	Symbol()	{ name: "Batman" }
Примитив				



Undefined

Неопределенное значение

Автоматически присваивается переменным, для которых не указано значение

Автоматически присваивается аргументам функции, для которых не были переданы значения

Если функция ничего не возвращает, то она возвращает undefined

Несуществующие свойства объекта имеют значение undefined

```
let value;  
console.log(typeof value); // "undefined"  
  
// Переменная name нигде не определена  
if (typeof name === "undefined") // "OK"  
if (name === undefined) // "ReferenceError"
```



Null

Отсутствие значения

Не присваивается автоматом

Его может вручную присвоить разработчик

```
console.log(typeof null === "object"); // true
```



Boolean

Логическое значение

Примитив `boolean` и объект-обёртка `Boolean`

Принимает значение `true` (истина) или `false` (ложь)

```
const isValid = true;
```

```
const isOk = false;
```



Symbol

Уникальный идентификатор

Используется как идентификатор для скрытых свойств объектов

```
console.log(Symbol.for("a")); // Symbol(a)
```

```
console.log(Symbol.for("b")); // Symbol(b)
```

```
console.log(Symbol("a")); // Symbol(a)
```

```
console.log(Symbol("b")); // Symbol(b)
```



String

Последовательность символов для представления текста

```
// Кавычки двух видов
const singleQuotes = 'Привет';
const doubleQuotes = "Привет";
const quote = "'Привет' - это слово";

console.log(singleQuotes); // "Привет"
console.log(doubleQuotes); // "Привет"
console.log(quote); // "'Привет' - это слово"

// Шаблонные строки
const name = 'Пользователь';
const greeting = `Привет, ${name}`;

console.log(greeting); // "Привет, Пользователь"
```

Примитив string и объект-обёртка String

Значение пишется в кавычках



Работа со строками

```
// Конкатенация
const concatenatedString = "Собираем" + "строку";
console.log(concatenatedString); // "Собираемстроку"
```

```
// Методы
const login = "MyLogin";

console.log(login.length); // 7

console.log(login[0]); // "M"

console.log(login.toUpperCase()); // "MYLOGIN"
console.log(login.toLowerCase()); // "mylogin"
```



Number

Примитив number и объект-обёртка Number

Числовой тип в формате 64-битного числа двойной точности с плавающей запятой

64 бита, от -10307 до +10307

```
const result = 0.1 + 0.2;  
console.log(result);
```




Арифметические операции

```
console.log(3 + 2); // 5  
console.log(3 - 2); // 1  
console.log(3 * 2); // 6  
console.log(3 / 2); // 1.5  
console.log(3 % 2); // 1  
  
console.log(3 / 0); // ??
```



Infinity

```
console.log(3 / 0); // Infinity
console.log(-3 / 0); // -Infinity

console.log(3 / Infinity); // 0
console.log(-3 / Infinity); // -0

console.log(Infinity * Infinity); // Infinity
console.log(Infinity * 0); // ??
```



NaN (Not a Number)

```
console.log(Infinity * 0); // NaN  
console.log(Infinity - Infinity); // NaN  
console.log(Infinity / Infinity); // NaN  
console.log(Infinity % 5); // NaN  
  
console.log(typeof NaN); // ??  
console.log(NaN === NaN); // ??
```



NaN (Not a Number)

```
console.log(Infinity * 0); // NaN
console.log(Infinity - Infinity); // NaN
console.log(Infinity / Infinity); // NaN
console.log(Infinity % 5); // NaN

console.log(typeof NaN); // "number"
console.log(NaN === NaN); // false

console.log(Number.isNaN(NaN)); // true
console.log(Number.isNaN("строка")); // false
```



Работа с числами

```
// Статические методы Number

console.log(Number.MIN_VALUE); // 5e-324

console.log(Number.MAX_VALUE); // 1.7976931348623157e+308

// Math - математические операции и функции

console.log(Math.PI); // 3.141592653589793

console.log(Math.sqrt(16)); // 4

console.log(Math.pow(2, 10)); // 1024

// Методы
const price = 10;

console.log(price.toFixed(1)); // "10.0"

console.log(price.toFixed(2)); // "10.00"

console.log(price.toFixed(3)); // "10.000"
```



BigInt

Только целые числа

Диапазон зависит от конкретной реализации и окружения

Нельзя смешивать с Number и использовать с Math

Используется в финансовых технологиях, в высокоточных метках времени

```
const bigInt = 10n;  
const anotherBigInt = BigInt(10);  
  
console.log(bigInt === anotherBigInt); // true  
const invalidBigInt = BigInt(10.5);
```



Object

Не примитив

Набор свойств различных типов (включая вложенные объекты)

Получает методы не через объект- обёртку, а через прототип

Значение в памяти, можно ссылаться

```
const hero = {  
  name: "Batman",  
  
  tools: [  
    { name: "Batarang" },  
    { name: "Batmobile" },  
  ],  
  isReady: function() {  
    return true;  
  },  
  
  birthDay: new Date(1915, 03, 07)  
};  
  
console.log(hero.name); // "Batman"  
console.log(hero.isReady()); // true  
console.log(hero.birthDay.toDateString()); // "Apr 07 1915"
```



Задача

Объявить переменные со значениями – числами с плавающей точкой.

Вычислить значение по формуле.

Вывести результат в консоль.

$$z = \frac{(a + b) \cdot c}{d}$$



Функция

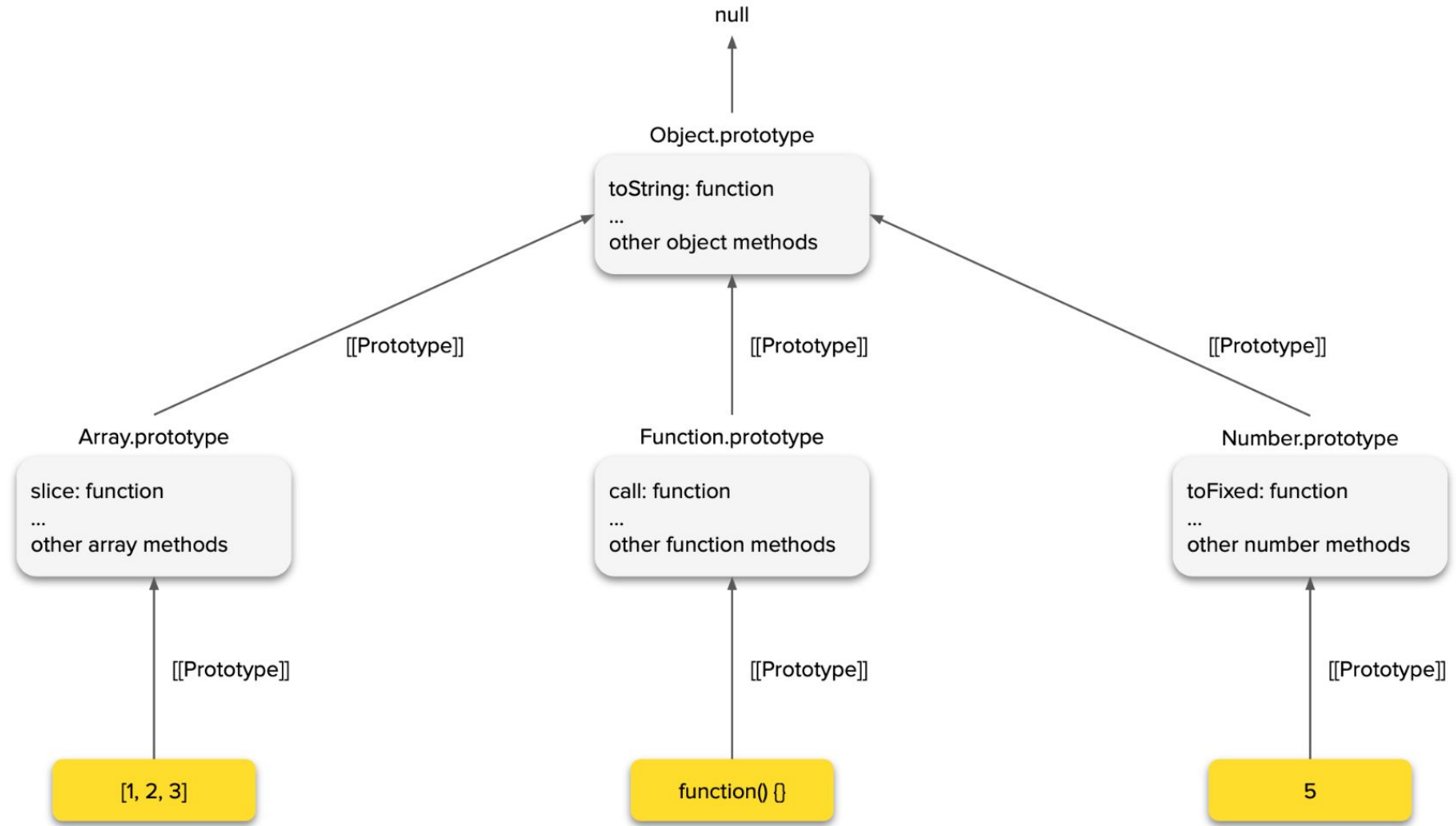
```
function getHelloMessage(name) {  
    return `Привет, ${name}!`;  
}
```

```
function displayMessage(message) {  
    console.log(message);  
}
```

```
displayMessage(getHelloMessage("Иван")); // "Привет, Иван!"
```



Object





Массив

Особый тип объекта, предназначенный для работы с упорядоченным набором элементов

Length возвращает последний индекс + 1

toString() возвращает перечисление элементов через запятую

Обращение к элементам по индексу через квадратные скобки

```
const array = ["first", "second"];  
const anotherArray = new Array("first", "second");  
  
console.log(array.length); // 2  
console.log(array[0]); // "first"
```



Получаем элементы

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
ships[0]; // Бесстрашный
```

```
ships[1]; // Решительный
```

```
ships[100]; // Undefined
```



Разбираем массив на элементы

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
const [dauntless, resolute, defender] = ships;
```

```
dauntless; // Бесстрашный
```

```
resolute; // Решительный
```

```
defender; // Защитник
```



Разбираем массив на элементы 2

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
const [, resolute] = ships;
```

```
resolute; // Решительный
```

```
const [dauntless,, defender] = ships;
```

```
dauntless; // Бесстрашный
```

```
defender; // Защитник
```



..Rest

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
const [dountless, ...otherShips] = ships;
```

```
dountless; // Бесстрашный
```

```
otherShips; // ['Решительный', 'Защитник']
```



Задача

Создайте функцию, которая принимает первый элемент массива и оставшиеся элементы как отдельные параметры.

- Выведите первый элемент массива отдельно.
- Выведите оставшиеся элементы в виде нового массива.

Подсказка: использовать Rest



Добавление/удаление

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
const lastShip = ships.pop();
```

```
lastShip; // 'Защитник'
```

```
ships; // ['Бесстрашный', 'Решительный']
```

```
ships.push('Спаситель');
```

```
ships; // ['Бесстрашный', 'Решительный', 'Спаситель']
```



Задача

Напишите программу, которая:

1. Создает массив чисел от 1 до 5.
2. Удаляет последний элемент массива (используйте `pop`).
3. Добавляет в конец массива два числа (используйте `push`).
4. Выводит измененный массив.



Работаем с первым элементом

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
const firstShip = ships.shift();
```

```
firstShip; // 'Бесстрашный'
```

```
ships; // ['Решительный', 'Защитник']
```

```
ships.unshift('Спаситель');
```

```
ships; // ['Спаситель', 'Решительный', 'Защитник']
```



Задача

Напишите программу, которая:

1. Создает массив из слов: `['JavaScript', 'is', 'awesome']`.
2. Удаляет первый элемент массива (используйте `shift`).
3. Добавляет два новых слова в начало массива (используйте `unshift`).
4. Выводит измененный массив.



Проход по элементам массива. ForEach

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];
```

```
ships.forEach(ship => console.log(ship)); //
```

```
Бесстрашный
```

```
// Решительный
```

```
// Защитник
```



Проход по элементам массива. Map

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];  
const capitalized = ships.map(ship => ship.toUpperCase());  
capitalized; // ['БЕССТРАШНЫЙ', 'РЕШИТЕЛЬНЫЙ', 'ЗАЩИТНИК']
```



Задача

Напишите программу, которая:

1. Создает массив чисел.
2. С помощью `forEach` подсчитывает сумму всех чисел в массиве.
3. Выводит сумму в консоль.



Задача

Напишите программу, которая:

1. Создает массив из чисел от 1 до 5.
2. С помощью `map` создает новый массив, где каждый элемент — это квадрат исходного числа.
3. Выводит новый массив в консоль.



Проход по элементам массива. Filter

```
const ships = ['Бесстрашный', 'Решительный', 'Защитник'];  
const adjectives = ships.filter(ship => ship.endsWith('ный'));  
adjectives; // ['Бесстрашный', 'Решительный']
```



Проход по элементам массива. Reduce

```
const jedi = [  
  { name: 'Люк', age: 19 },  
  { name: 'Йода', age: 896 },  
  { name: 'Энакин', age: 42 }  
]  
  
const oldestJedi = jedi.reduce((result, nextJedi, jedis) => {  
  return result.age > nextJedi.age ? result : nextJedi;  
}, jedi[0]);  
  
// OR  
  
const oldestJedi = jedi.reduce((result, nextJedi, jedis) => {  
  return result.age > nextJedi.age ? result : nextJedi;  
});
```



Задача

Напишите программу, которая:

1. Создает массив из случайных чисел от 1 до 20.
2. С помощью `filter` создает новый массив, содержащий только четные числа.
3. С помощью `reduce` подсчитывает сумму четных чисел.
4. Выводит новый массив и сумму в консоль.



Динамическая и слабая типизация

Динамическая типизация:

- Тип переменной определяется во время выполнения программы, а не во время ее объявления.
- Одна и та же переменная может хранить данные разных типов в разное время.

```
let data; // Тип: undefined  
console.log(typeof data); // "undefined"
```

```
data = 42; // Тип: number  
console.log(typeof data); // "number"
```

```
data = "Привет"; // Тип: string  
console.log(typeof data); // "string"
```



Динамическая и слабая типизация

Слабая типизация:

- JavaScript автоматически преобразует типы данных, если это возможно, что иногда может приводить к неожиданным результатам.
- Типы данных можно смешивать в выражениях.

```
console.log(10 + "5"); // "105"
```

(число превращается в строку)

```
console.log("5" - 2); // 3
```

(строка превращается в число)

```
console.log(true + 1); // 2
```

(true превращается в 1)



Слабая типизация 2

```
// Приведение типов  
const hour = 10; // Число  
  
const name = "Олег"; // Строка  
  
console.log(hour + name); // ??  
  
const hero = { name: "Batman" };  
  
console.log(hour + hero); // ??
```



Преобразования к строке

```
// Явное приведение
console.log(String(null)); // "null"

// Неявное приведение
console.log("Строка" + undefined); // "Строкаundefined"

console.log(10 + {}); // "10[object Object]"

console.log(10 + []); // "10"
console.log(10 + [1]); // "101"
console.log(10 + ["Строка"]); // "10Строка"
```



Преобразование к числу

// Явное приведение

```
console.log(Number(null)); // ??
```

```
console.log(Number("")); // ??
```

```
console.log(Number(undefined)); // ??
```

```
console.log(Number({})); // ??
```

```
console.log(+ "10"); // ??
```

```
console.log(- "1"); // ??
```

// Неявное приведение

// Рассмотрим дальше



Преобразование к числу – ответ

```
// Явное приведение  
  
console.log(Number(null)); // 0  
  
console.log(Number("")); // 0  
  
console.log(Number(undefined)); // NaN  
  
console.log(Number({})); // NaN  
  
console.log(+ "10"); // 10  
  
console.log(- "1"); // -1  
  
// Неявное приведение  
  
// Рассмотрим дальше
```



Преобразование к логическому значению

```
// Явное приведение  
  
console.log(Boolean(0)); // false  
  
console.log(Boolean(undefined)); // false  
  
console.log(Boolean(null)); // false  
  
console.log(Boolean("")); // false  
  
console.log(Boolean(NaN)); // false  
  
console.log(!0); // true  
  
console.log (!!0); // false  
  
// Неявное приведение  
  
// Рассмотрим дальше
```



Операторы. Инкремент и декремент

Увеличивают или уменьшают на единицу значение переменной и возвращают новое (префиксный) либо исходное (постфиксный) значение

```
let x = 0;
let y = x++;
console.log(x, y); // 1, 0

let x = 0;
let y = x--;
console.log(x, y); // -1, 0

let x = 0;
let y = ++x;
console.log(x, y); // 1, 1

let x = 0;
let y = --x;
console.log(x, y); // -1, -1
```



Операторы. Операторы сравнения

Конвертируются в примитив number с хинтом
“number” (Symbol.ToPrimitive(hint))

Если оба операнда - строки, то сравниваются
как строки

Если хотя бы один операнд не является строкой,
то операнды приводятся к числам

Если хотя бы один NaN, возвращается false

```
console.log("a" <= "b"); // true  
console.log("a" < "a"); // false  
console.log("a" < "3"); // false
```

```
console.log("5" <= 3); // false  
console.log("3" <= 5); // true
```

```
console.log("hello" >= 5); // false  
console.log(5 >= "hello"); // false
```

```
console.log(false > true); // false  
console.log(true > false); // true
```

Math.trunc()



Операторы. Операторы сравнения – 2

	===	!==	=	!=
Тип	Строгое	Строгое	Нестрогое	Нестрогое
Приводит типы	✗	✗	✓	✓
Используем?	✓	✓	✗	✗

```
console.log(false == ""); // true  
console.log(true == 1); // true  
console.log("1" == 1); // true
```

```
console.log(false === ""); // false  
console.log(true === 1); // false  
console.log("1" === 1); // false
```



Операторы. Логические операторы

&& - логическое И

|| - логическое ИЛИ

! - логическое НЕ

A	B	A && B	A B	!A	!B
true	true	true	true	false	false
true	false	false	true	false	true
false	true	false	true	true	false
false	false	false	false	true	true



Операторы. Логические операторы

&& - возвращает первое falsy значение или последнее

|| - возвращает первое truthy значение или последнее

```
console.log("строка" && null); // null
console.log("строка" && 0); // 0
console.log(null && stringValue); // null
console.log(false && nullValue); // false
console.log(true && "строка"); // "строка"
console.log("строка" && true); // true

console.log("строка" || 0); // "строка"
console.log(0 || "строка"); // "строка"
console.log(0 || null); // null
console.log(null || 0); // 0
console.log(true || "строка"); // true
console.log("строка" || true); // "строка"
```



Задачи

Задача 1: Создайте массив строк ["1", "2", "3", "4", "5"]. С помощью `map()` конвертируйте массив строк в массив чисел. Выведите результат на консоль.

Задача 2: Создайте массив строк ["apple", "pear", "banana", "kiwi", "grape"]. С помощью `filter()` создайте новый массив, который будет содержать слова длины больше 4 символов. Выведите новый массив на консоль.

Задача 3: Создайте массив чисел [10, 20, 30, 40, 50]. С помощью `reduce()` вычислите среднее значение всех элементов массива. Выведите результат на консоль.

Задача 4: Создайте массив объектов `{name: "", lastName: ""}`. С помощью `forEach()` выведите на консоль информацию о каждом объекте в формат: "Индекс: 0, Имя: , Фамилия: ".



if

Выполняет код в случае выполнения условия

```
1  const count = 0;  
2  
3  if (count > 0) {  
4      console.log("Привет"); // Этот код не выполнится  
5  }
```



if + else

Выполняет одну из веток, в зависимости от выполнения условия

```
1  const count = 0;
2
3  if (count > 0) {
4      console.log("Привет"); // Этот код не выполнится
5  } else {
6      console.log("Привет"); // Этот код выполнится
7  }
```



if + else if + else

Выполняет одну из веток, в зависимости от выполнения условия

```
1  const count = 0;
2
3  if (count > 0) {
4      console.log("Привет"); // Этот код не выполнится
5  } else if (count > 10) {
6      console.log("Привет"); // Этот код не выполнится
7  } else {
8      console.log("Привет"); // Этот код выполнится
9  }
```



Задача

Напишите функцию `checkAge(age)`, которая принимает в качестве аргумента возраст человека (число). Функция должна проверять возраст и выводить соответствующее сообщение на консоль:

- Если возраст меньше 18, вывести: "Вы ещё слишком молоды!"
- Если возраст от 18 до 65, вывести: "Добро пожаловать!"
- Если возраст больше 65, вывести: "Вам полагается скидка!"

Задача 1: Использование оператора `if...else`

Решите задачу, используя обычный оператор `if`.



Тернарный оператор

условие ? выражение1 : выражение2

```
console.log(true ? "Истина" : "Ложь"); // "Истина"  
const isReady = true;  
const status = isReady ? "Готов" : "Не готов";  
console.log(status); // "Готов"
```



Задача

Напишите функцию `checkAge(age)`, которая принимает в качестве аргумента возраст человека (число). Функция должна проверять возраст и выводить соответствующее сообщение на консоль:

- Если возраст меньше 18, вывести: "Вы ещё слишком молоды!"
- Если возраст от 18 до 65, вывести: "Добро пожаловать!"
- Если возраст больше 65, вывести: "Вам полагается скидка!"

Задача 2: Использование тернарного оператора

Решите задачу, используя тернарные операторы



switch

```
1  const count = 0;
2  let message;
3
4  switch (count) {
5      case 1: // Если count === 1
6          message = "Один";
7          break;
8      case 2:
9      case 3:
10     case 4: // Если count === 2 || count === 3 || count === 4
11         message = "Несколько";
12         break;
13     default: // Во всех остальных случаях
14         message = "Много";
15         break;
16 }
17
18 console.log(message); // "Много"
```



switch

Напишите функцию, которая:

1. Использует оператор `switch` для определения названия дня недели по числу (от 1 до 7).
2. Выводит соответствующее название дня в консоль.
3. Если число не входит в диапазон от 1 до 7, выводит сообщение: "Некорректный номер дня".



Циклы while и do-while

```
1  // Выполняем действие до проверки условия
2  // do { ... } while (...);
3
4  // Выполняем действие после проверки условия
5  // while (...) do { ... };
6
7  let count = 0;
8
9  while (count < 5) {
10     count++;
11     console.log(count); // 1 2 3 4 5
12 }
```



Цикл for

for ([инициализация]; [условие выхода]; [финальное выражение]) { ... }

```
1  const count = 5;  
2  
3  for (let index = 1; index <= count; index++) {  
4      console.log(index); // 1 2 3 4 5  
5  }
```



Задача

Напишите функцию `processGrades(grades)`, которая принимает массив оценок студентов в качестве аргумента.

Функция должна найти минимальную и максимальную оценку в массиве и вывести их в консоль



Оператор “?.”

```
const config = {  
  ui: {  
    size: {  
      width: 400  
    }  
  }  
};
```

// Сейчас

```
console.log(config.ui?.size?.width); // 400  
console.log(config.ui?.size?.height); // undefined  
console.log(config.default?.size?.height); // undefined
```

// Раньше (<2020)

```
console.log(config.default && config.default.size && config.default.size.height); // undefined
```



Область видимости

```
1  const count = 0;
2
3  if (true) {
4      const innerCount = 1;
5
6      // Тут доступны обе переменные: count и innerCount
7      console.log(count); // 0
8      console.log(innerCount); // 1
9  }
10
11 // Тут доступна только одна переменная: count
12 console.log(count); // 0
13 console.log(innerCount); ❌
```



Захват переменной

```
1  const createCacheable = function(heavyFunction) {
2      let result;
3
4      return function() {
5          if (!result) {
6              result = heavyFunction();
7          }
8          return result;
9      }
10 };
11 const myHeavyFunction = function() {
12     console.log("Вычисление");
13     return 12;
14 }
15 const cacheable = createCacheable(myHeavyFunction);
16 console.log(cacheable()); // "Вычисление" 12
17 console.log(cacheable()); // 12
18 console.log(cacheable()); // 12
```



TypeError

Невозможность выполнить операцию, чаще всего из за некорретного типа

```
Uncaught TypeError: Assignment to constant variable  
at script.js:3:7
```



ReferenceError

Обращение к несуществующей переменной

```
Uncaught ReferenceError: MAX_VALUE is not defined  
at script.js:3:13
```