JavaScript. Базовый уровень

Урок 2

Основные операторы JavaScript

Операторы и их приоритеты выполнения.
Условные операторы и циклы.

Разбор практического задания

План урока

- Бинарные и унарные операторы.
- Ветвления.
- Тернарный оператор.
- Оператор выбора switch.
- Функции: стандартные и стрелочные.
- Область видимости: let vs var
- Рекурсия


Операторы в JavaScript

Оператор

Это наименьшая автономная часть языка программирования, то есть команда.

Операторы бывают:

- унарные;
- бинарные.

Бинарный оператор

Применяется к двум операндам:

```
let a = 1, b = 2, c = 0, d = 0, k = false;

c = a + b; // сложение
d = b - a; // вычитание
k = d > a; // сравнение
d = a / b; // деление
d = a % b; // деление по модулю - возвращается остаток от деления
a *= c; // a = a * c;
c += b; // c = c + b;
```

Унарный оператор

Применяется к одному операнду:

```
4    a = +'2.5'; // 2,5
5    b = -''; // -0
6    c = +'qwerty'; // NaN
7    d = !true; // false
8    f = !(4 > 6); // true
```

Инкремент

Префиксная форма

```
let a = 9;
let res = ++a;
console.log(res); // 10
console.log(a); // 10
```

Постфиксная форма

```
let a = 9;
let res = a++;
console.log(res); // 9
console.log(a); // 10
```

Декремент

Префиксная форма

```
let a = 9;
let res = --a;
console.log(res); // 8
console.log(a); // 8
```

Постфиксная форма

```
let a = 9;
let res = a--;
console.log(res); // 9
console.log(a); // 8
```

Приоритеты операторов

Оператор	Описание
. [ ] ( )	Доступ к полям, индексация массивов, вызовы функций и группировка выражений
++ -- - ~ ! delete new typeof void	Унарные операторы, тип возвращаемых данных, создание объектов, неопределенные значения
*** / %**	Умножение, деление, деление по модулю
+ - +	Сложение, вычитание, объединение строк
<< >> >>>	Сдвиг битов

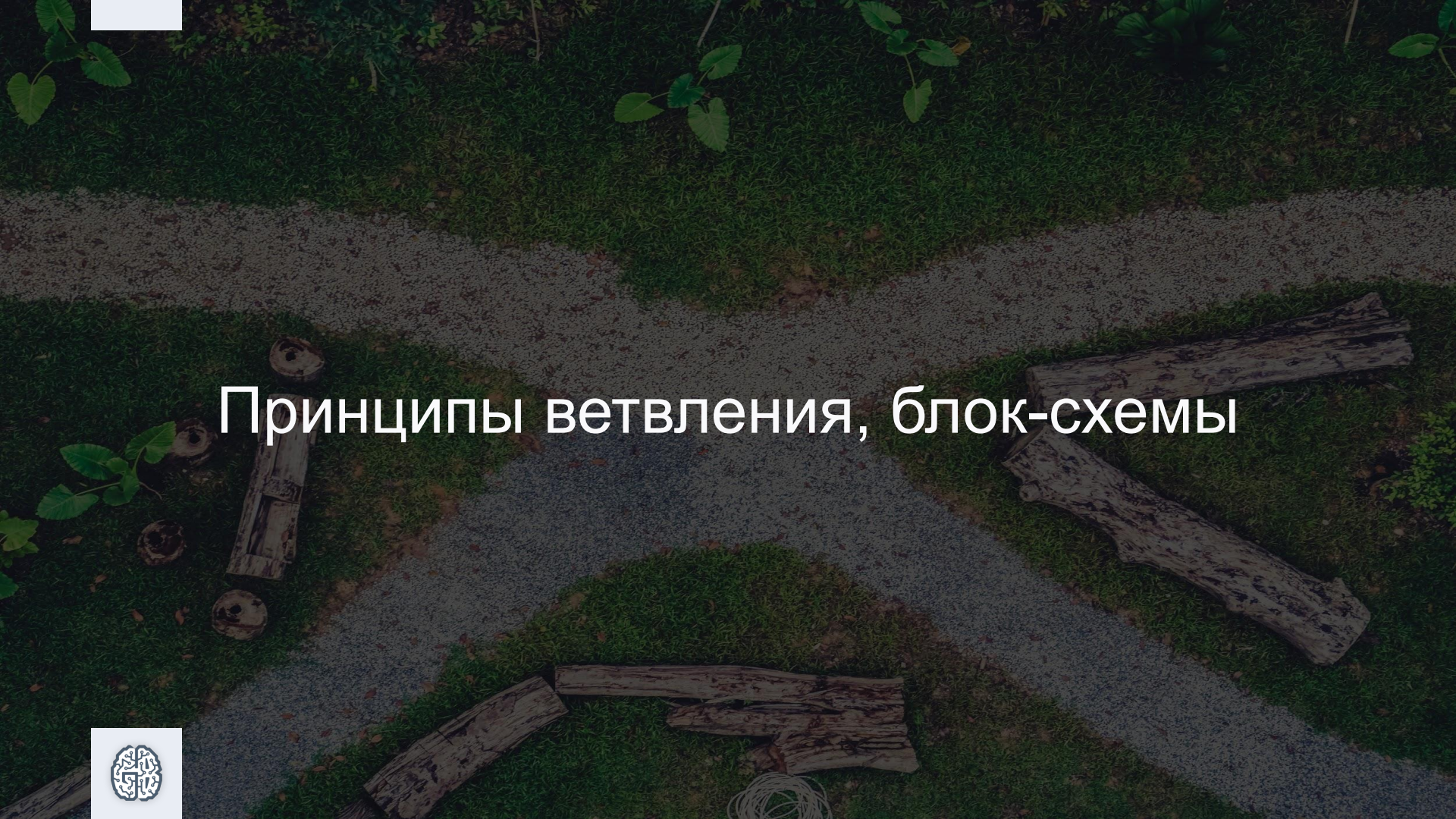
Приоритеты операторов

Оператор	Описание
< <= > >= instanceof	Меньше, меньше или равно, больше, больше или равно, instanceof
== != === !==	Равенство, неравенство, строгое равенство, строгое неравенство
&	Побитовое И
^	Побитовое исключающее ИЛИ
|	Побитовое ИЛИ

Приоритеты операторов

Оператор	Описание
&&	Логическое И
||	Логическое ИЛИ
?:	Условный оператор
= OP=	Присваивание, присваивание с операцией (например += и &=)
,	Вычисление нескольких выражений

Принципы ветвления, блок-схемы



Логические «И» и «ИЛИ»

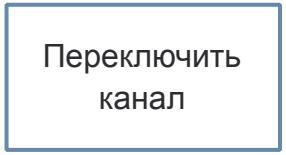
```
true && true && true // true
true && false && true // false
false && false && false // false

true || true || true // true
true || false || true // true
false || false || false // false
```

Ветвление

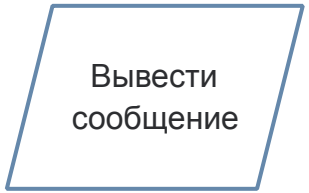
«Если случится событие А, то я выполню действие Б».

Для ветвления в программировании применяются специальные операторы, обеспечивающие выполнение определенной команды или набора команд только при условии истинности логического выражения или их группы.

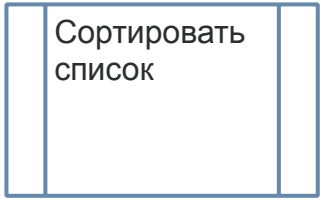
Блок-схемы



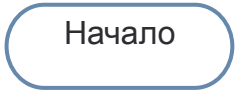
процесс



данные

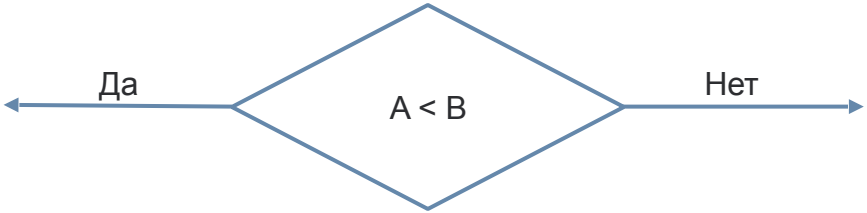


предопределенный процесс



терминатор

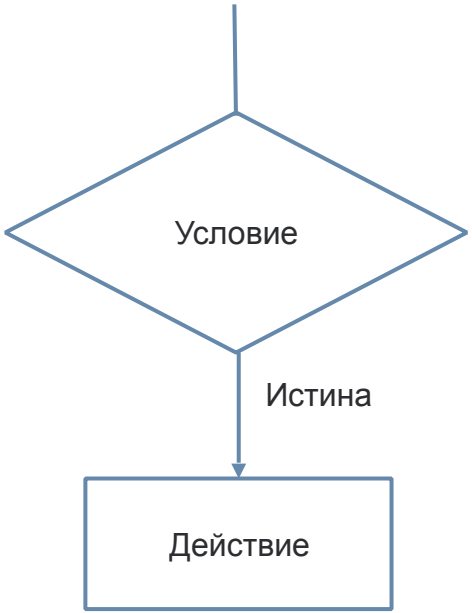
Блок-схемы



Ветвление

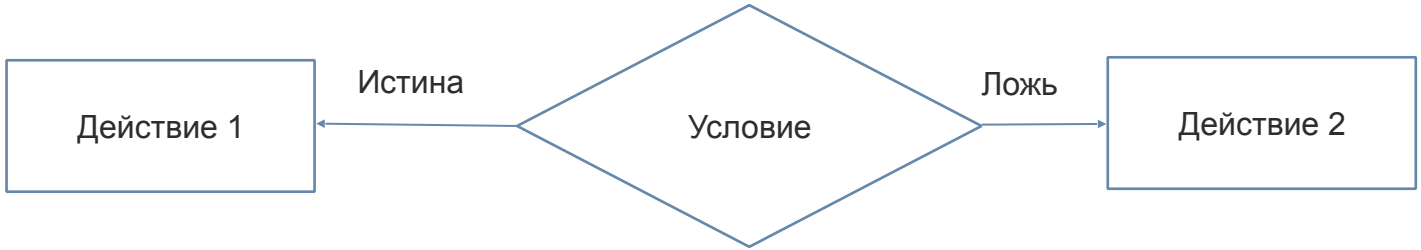
Конструкция if-else

Оператор if



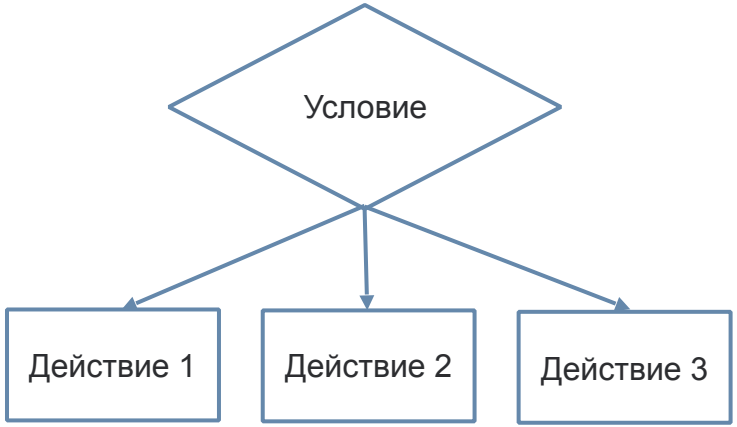

```
let question = prompt( message: "Висит груша – нельзя скушать");

// неполная форма
if (question == "груша") {
    alert("Правильно:)");
}
```

Оператор else




```
let question = prompt( message: "Висит груша – нельзя скушать");

// полная форма
if (question == "груша") {
    alert("Правильно:)");
} else {
    alert("Неправильно :(");
}
```

Вложенное ветвление




```
let age = 6;

if (age >= 18) {
    alert("Вы можете смотреть любые фильмы");
} else if (age < 16 && age >= 6) {
    alert("Вы можете смотреть только фильмы с рейтингом 6+");
}
```

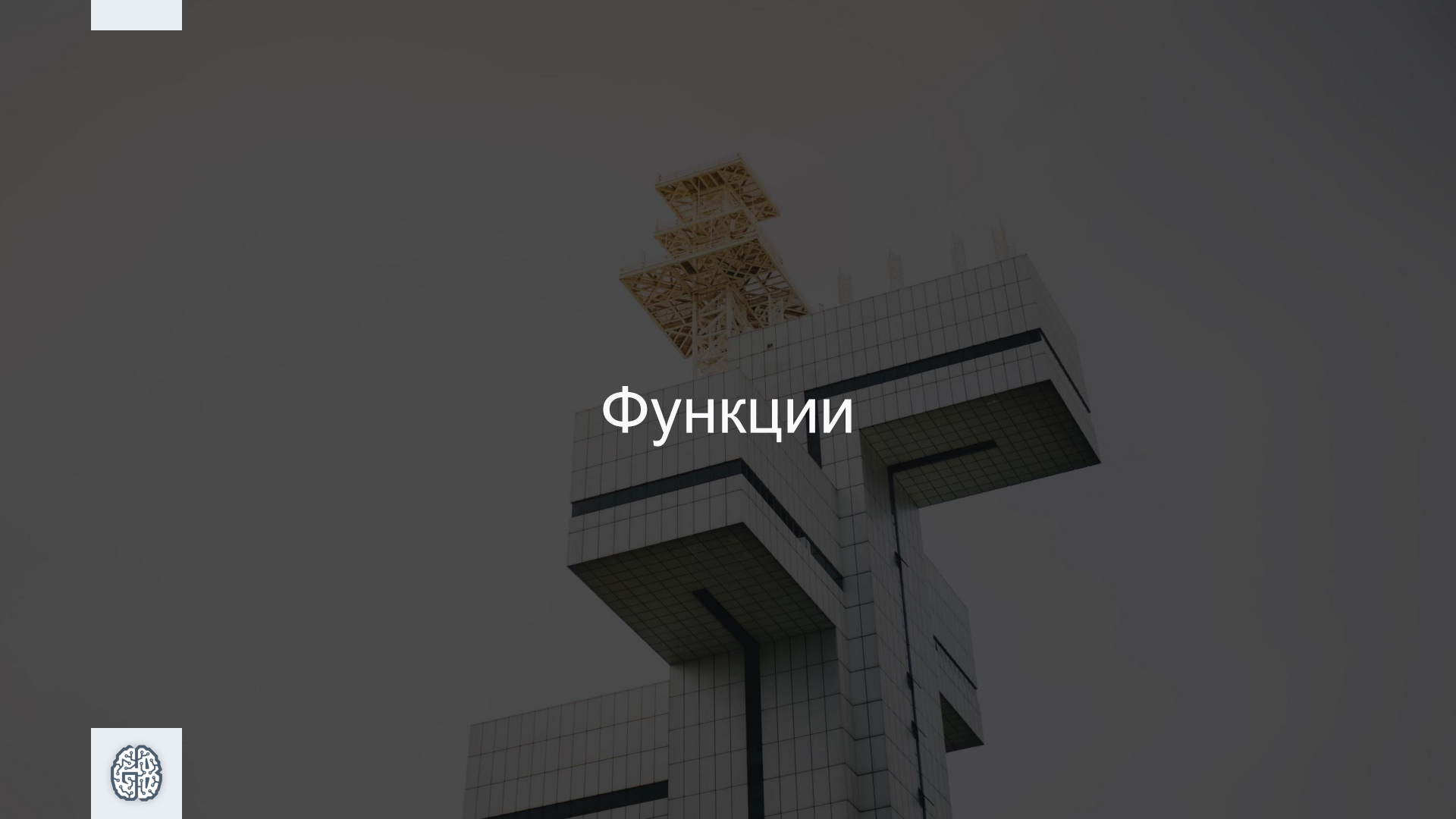
Тернарный оператор

```
let ticketPrice = 200;
let answer = confirm("Пойдем в кино вместе?");

let totalTicketsPrice = answer ? ticketPrice*2 : ticketPrice;
```

Оператор выбора switch

```
let day = prompt( message: "Какой сегодня день недели?");

switch (day) {
    case 'суббота':
        alert('Ура, завтра не на работу!');
        break;
    case 'воскресенье':
        alert('Завтра на работу:(');
        break;
    case 'пятница':
        alert('2 выходных впереди!');
        break;
    default:
        alert('Хочу отдыхать!');
}
```

Функции



Функции

Функция – это блок кода, к которому можно обращаться из разных частей скрипта.

```
// function declaration
function sayHello() {
    alert('Привет!');
    alert('Hello!');
    alert('Hola!');
}
```

```
// expression declaration
let sayHello = function() {
    alert('Привет!');
    alert('Hello!');
    alert('Hola!');
};
```

Имена функций

Неправильно

```
function 1word() {

}
function hello#$worl^() {

}
function всемПривет() {

}
function switch() {

}
```

Корректно

```
function word1() {

}
function $_hello_world() {

}
function sayHello() {

}
```

Отличия между declaration и expression

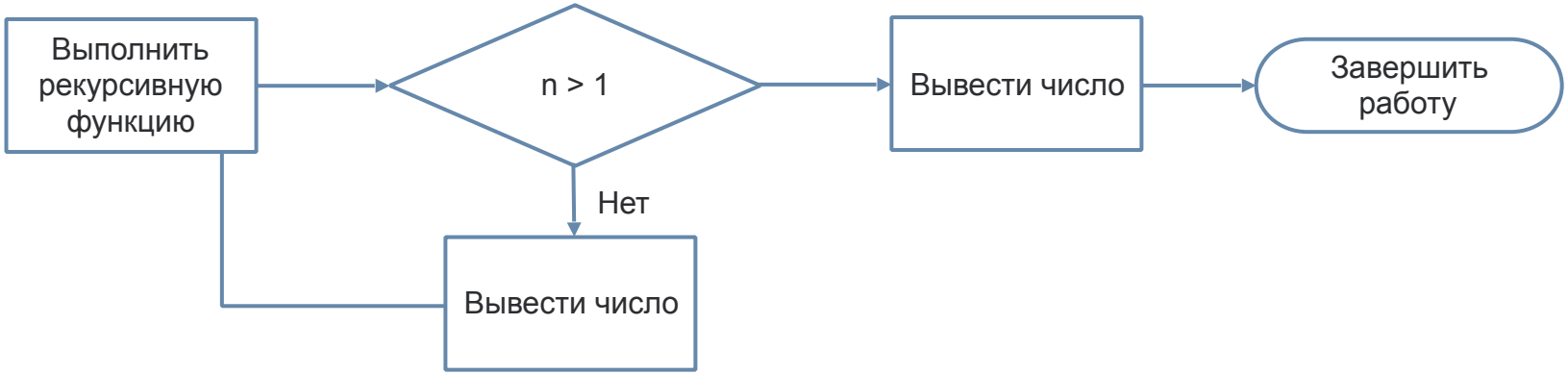
	Function declaration	**Function expression**
Когда браузер узнает о существовании функции	До начала выполнения кода, т.к. просматривает весь код целиком заранее	Когда выполнение доходит до определения функции
Можно вызвать до объявления функции	Да	Нет

Стрелочные функции

```
let sayHello = () => console.log('Hello!');
sayHello();

let sayHelloAnyTimes = () => {
    console.log('Hello');
    console.log('Привет');
    console.log('Hola');
};

sayHelloAnyTimes();
```

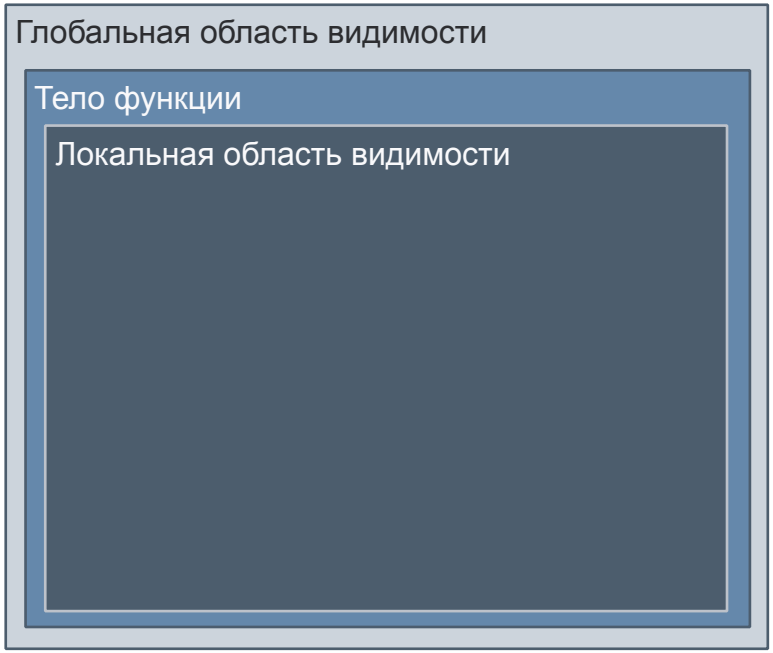
Рекурсия

«Чтобы понять рекурсию, нужно сначала понять рекурсию» (автор неизвестен)



Рекурсия

```
// Рекурсия
let number = parseInt( s: Math.random() * 100);
guess(number);

function guess(num) {
    let answer = parseInt(prompt( message: "Угадайте число от 0 до 100"));
    if (answer > num) {
        alert("Ваше число слишком большое");
    } else if (answer < num) {
        alert("Ваше число слишком маленькое");
    } else if (answer == num) {
        alert("Вы угадали");
        return;
    } else {
        alert("Необходимо ввести число!");
    }
    guess(num);
}
```



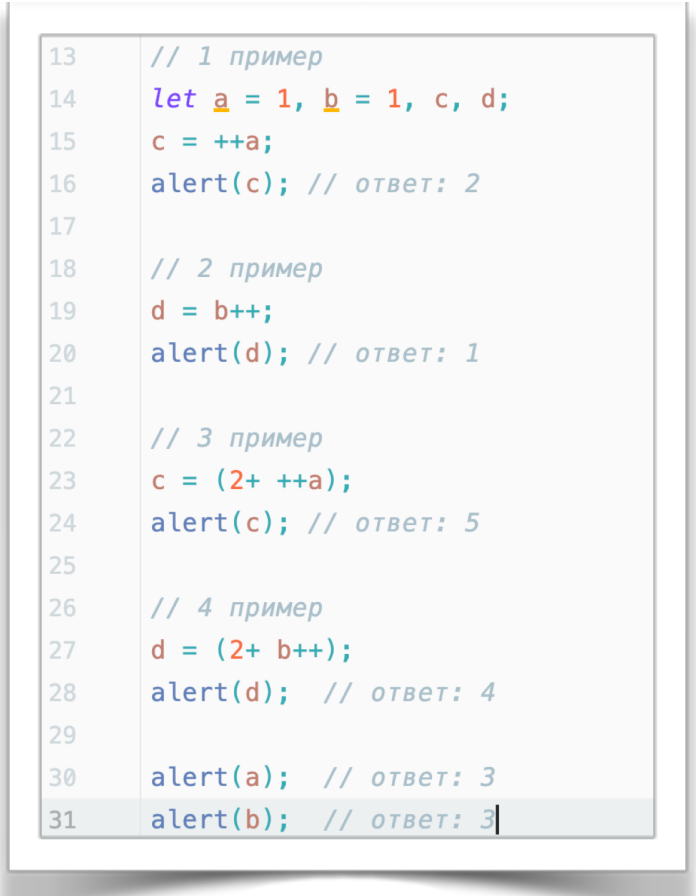
Области видимости

- **глобальные** переменные «видно» везде
- переменные **var** «видно» в пределах функции
- переменные **let** «видно» в пределах блока

Практическое задание

Практическое задание

1.Почему код дает именно такие результаты?



```
// 1 пример
let a = 1, b = 1, c, d;
c = ++a;
alert(c); // ответ: 2

// 2 пример
d = b++;
alert(d); // ответ: 1

// 3 пример
c = (2+ ++a);
alert(c); // ответ: 5

// 4 пример
d = (2+ b++);
alert(d);  // ответ: 4

alert(a);  // ответ: 3
alert(b);  // ответ: 3
```

Практическое задание

2.Чему будет равен **x**?

```
let a = 2;
let x = 1 + (a *= 2);
```

Практическое задание

3. Объявить две целочисленные переменные — **a** и **b** и задать им произвольные начальные значения. Затем написать скрипт, который работает по следующему принципу:
   - если **a** и **b** положительные, вывести их разность;
   - если **a** и **b** отрицательные, вывести их произведение;
   - если **a** и **b** разных знаков, вывести их сумму;

Ноль можно считать положительным числом.

Практическое задание

4. Реализовать четыре основные арифметические операции в виде функций с двумя параметрами. Обязательно использовать оператор **return**.

5. Реализовать функцию с тремя параметрами: **function mathOperation(arg1, arg2, operation)**, где **arg1**, **arg2** — значения аргументов, operation — строка с названием операции. В зависимости от переданного значения выполнить одну из арифметических операций (использовать функции из пункта 4 и вернуть полученное значение (применить **switch**).

Практическое задание

6.*С помощью рекурсии организовать функцию возведения числа в степень. Формат: **function power(val, pow), где val** — заданное число, **pow** –— степень.

Практическое задание

7.*Написать функцию для проверки пароля. На вход должен подаваться аргумент **path** – строка с паролем. Требования к паролю:

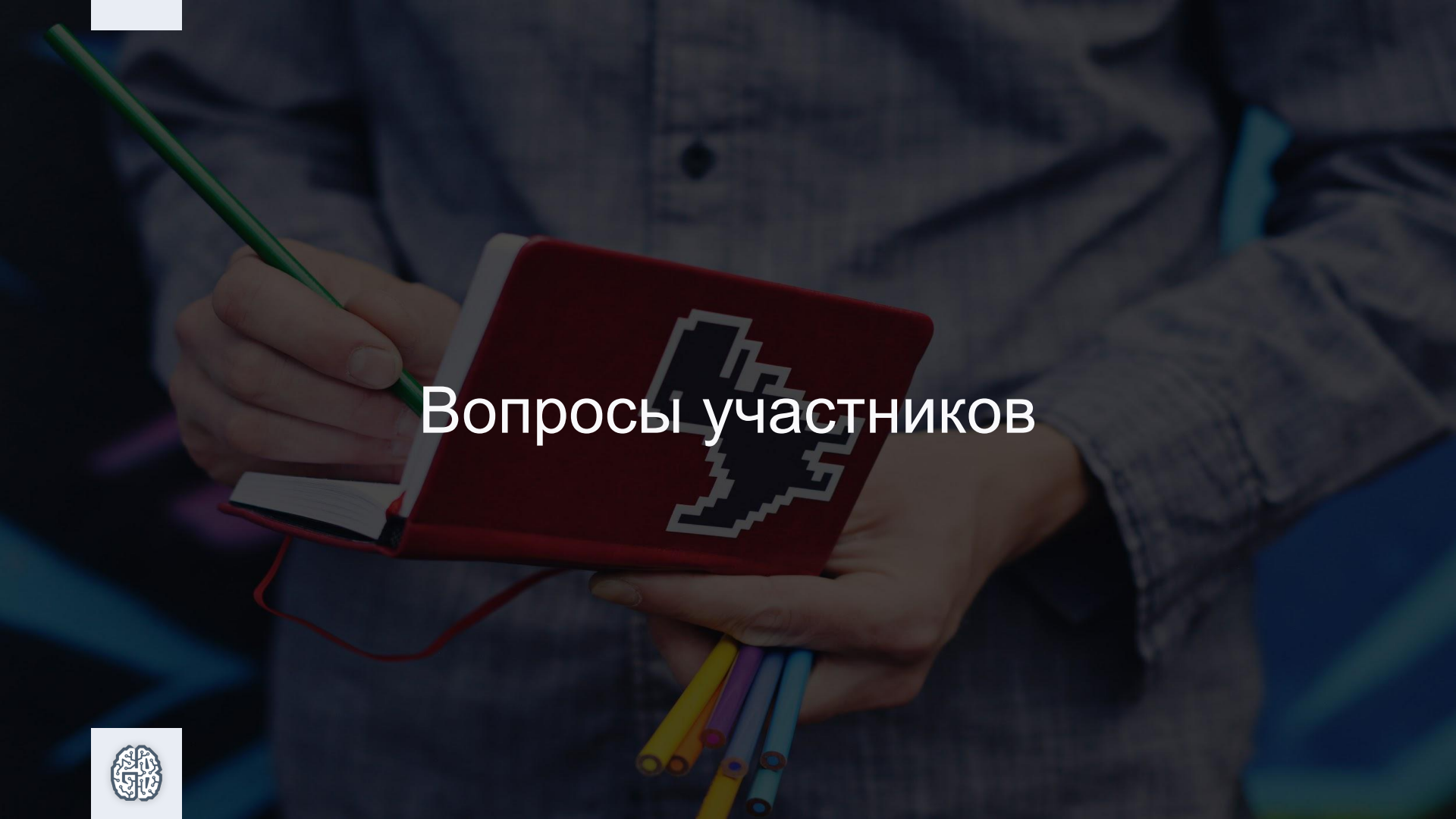
- длина должна быть не меньше 3-х символов и не больше 12;
- пароль должен содержать хотя бы один из следующих символов: &, $, %, *;
- пароль должен либо начинаться с буквы ‘q’, либо заканчиваться буквой ‘z’, либо оба условия сразу.

На выходе функция должна возвращать строку «Корректный пароль» в том случае, если пароль удовлетворяет всем условиям, и строку «Некорректный пароль» в том случае, если пароль не удовлетворяет условиям.

Для реализации проверки условий необходимо использовать следующие встроенные инструменты:

- свойство length https://javascript.ru/string/length
- функцию indexOf https://javascript.ru/string/indexof
- функцию lastIndexOf http://javascript.ru/string/lastindexof
- составные проверки с применением логического «и» / «или»

P.S. Функция проверки пароля может содержать в себе дополнительные подфункции, постарайтесь выделить отдельные логические блоки внутри функции и вынести их в отдельно.


Вопросы участников