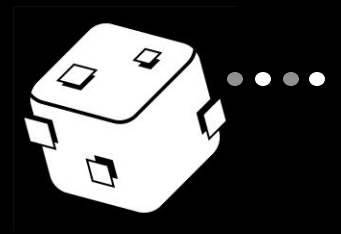
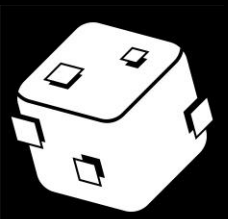
Условия и циклы в JavaScript

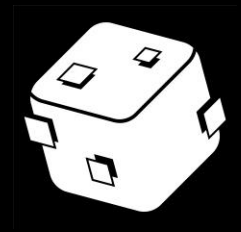
Contents





01

Введение и цели



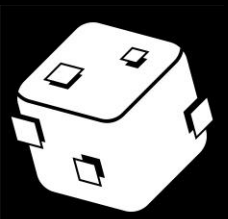
Условия и циклы в JavaScript

Научитесь управлять логикой программы

Условные конструкции
if / else, switch

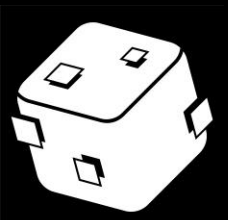
Циклы
for, while, do...while

Практика
Программа для проверки пароля



02

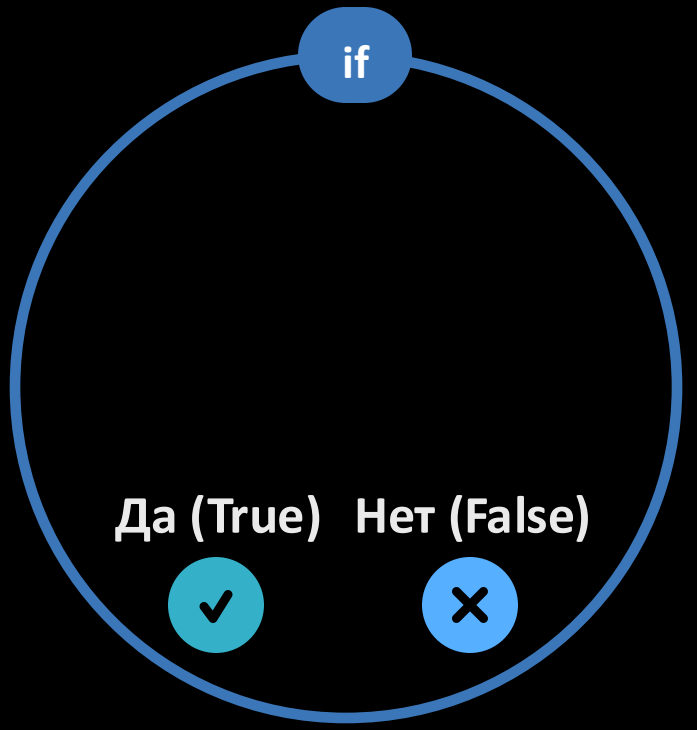
Условные конструкции



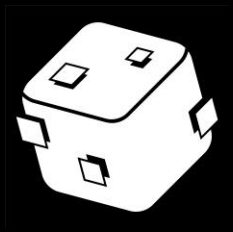
Что такое условие?

Условие — это проверка: **“да”** или **“нет”**.

Это основа управления логикой программы. Если условие выполняется (истинно), выполняется один блок кода, если нет (ложно) — другой. Это позволяет программе гибко реагировать на различные ситуации.



Конструкция if / else



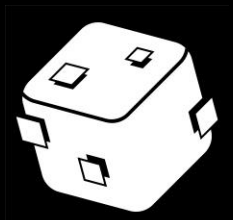
```
let age = 18; if (age >= 18) {
alert("Доступ разрешён"); } else {
alert("Доступ запрещён"); }
```

if

Если условие верно (истинно)

else

Если условие неверно (ложно)



Несколько условий: else if

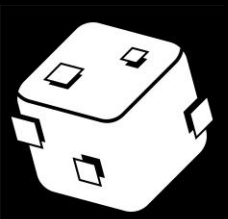
```
let score = 80;

if (score >= 90)
{
alert("Отлично!");
} else if (score >= 60)
{
alert("Хорошо!");
} else
{ alert("Попробуй ещё раз");
}
```

score >= 90	score >= 60	Иначе

Конструкция switch

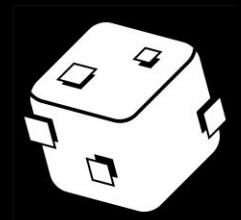
Удобна, когда нужно проверить одно значение на совпадение с множеством вариантов. Это делает код чище по сравнению с длинной цепочкой if...else if.

```
let color = "red";
switch (color)
{
case"red": alert("Стоп!");
            break;
case"yellow": alert("Жди!");
            break;
case"green": alert("Иди!");
            break;
default: alert("Неизвестный сигнал"); }
```



03

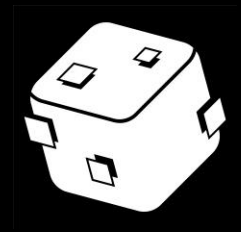
Циклы



Что такое цикл?

Цикл — это повторение блока кода, пока выполняется условие. Он позволяет автоматизировать однотипные операции и избежать дублирования кода.

Инициализация → **Проверка** → **Тело цикла** → **Изменение**



Цикл for

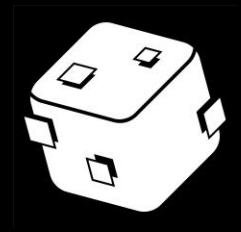
Самый распространенный цикл, когда известно, сколько раз нужно повторить действие. Он компактно объединяет инициализацию, условие и изменение счетчика.

```
for (let i = 1; i <= 5; i++)
 {
alert("Шаг " + i);
}
```

let i = 1 — начальное значение счетчика

i <= 5 — условие продолжения цикла

i++ — действие после каждой итерации

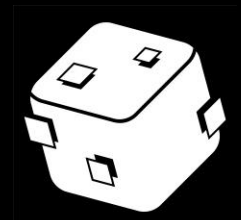


Цикл while

Выполняет блок кода **пока** условие истинно.

Подходит, когда количество итераций заранее неизвестно.

```
let i = 1; while (i <= 3) {
alert("Счётчик: " + i); i++; }
```

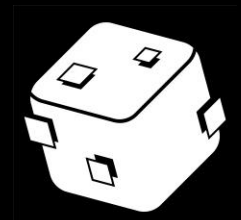


Цикл do...while

Отличается от while тем, что тело цикла выполняется **хотя бы один раз**, даже если условие изначально ложно. Проверка условия происходит после выполнения кода.

```
let i = 5; do { alert(i);
i++; } while (i < 5);
```

1. Выполняем тело цикла
2. Проверяем условие (i < 5)
3. Условие ложно, выходим



Управление циклом: break и continue

break

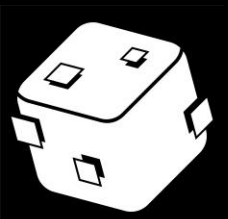
Прерывает выполнение цикла полностью и немедленно.

```
for (let i = 1; i <= 10; i++) { if (i === 5) break;
console.log(i); // 1, 2, 3, 4 }
```

continue

Пропускает текущую итерацию и переходит к следующей.

```
for (let i = 1; i <= 5; i++) { if (i === 3) continue;
console.log(i); // 1, 2, 4, 5 }
```



04

Практика

Практика 1: Проверка пароля

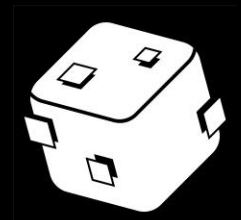
Простейшая задача: запросить пароль у пользователя и проверить его. Здесь используется строгое сравнение === для точного совпадения строк.

```
let password = prompt("Введите пароль:"); if (password
=== "1234") { alert("Доступ разрешён!"); } else {
alert("Неверный пароль!"); }
```

Практика 2: Ограничение попыток

Расширим задачу: дадим пользователю 3 попытки для ввода пароля.

```
let tries = 0; let password; while (tries < 3) { password =
prompt("Введите пароль:"); if (password === "1234") {
alert("Добро пожаловать!"); break; // Выходим из цикла }
else { alert("Неверно, попробуйте ещё раз"); tries++; //
Увеличиваем счетчик } }
```



Практика 3: Цикл for

Классическое упражнение: вывести числа от 1 до 10 в консоль. Это помогает понять, как работает счетчик.

```
for (let i = 1; i <= 10; i++) { console.log(i); }
console.log(i); }
```

Console Output

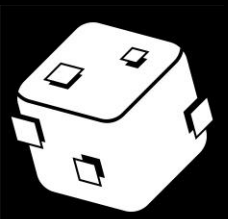
> 1

> 2

> 3

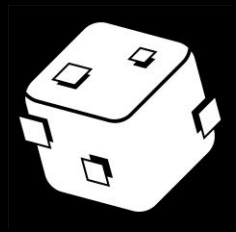
...

> 10



05

Закрепление и итог



Проверка понимания

1. Для чего используется `if / else`?

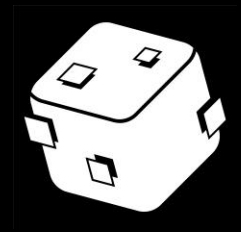
Для выполнения разных блоков кода в зависимости от того, истинно или ложно заданное условие.

2. Чем отличается `while` от `for`?

`for` используется, когда количество итераций известно. `while` — когда цикл выполняется до наступления какого-то события.

3. Когда используется `switch`?

Когда нужно сравнить одну переменную с множеством возможных значений, что делает код чище.



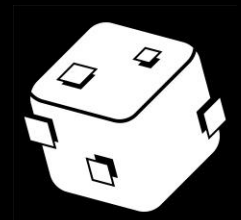
Типичные ошибки

Забытые скобки `{}`
Приводит к выполнению только первой строки после if.

Пропущенные `break`
В `switch` приводит к "проваливанию" в следующие case.

`==` вместо `===`
Может привести к неожиданному приведению типов.

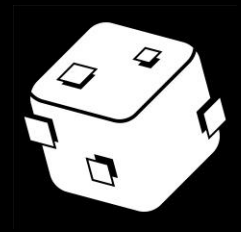
Бесконечные циклы
Забыли изменить переменную в условии?



Мини-проект: Программа проверки пароля

Создайте программу, которая:

1. Запрашивает пароль
2. Даёт 3 попытки
3. При успехе: "Добро пожаловать!"
4. Иначе: "Доступ заблокирован"



Итоги урока

- Как программа принимает решения с помощью **if/else** и **switch**.
- Как выполнять повторяющиеся действия с помощью **for**, **while** и **do...while**.
- Как управлять ходом выполнения цикла с **break** и **continue**.
- Как написать программу с проверкой пароля и избежать типичных ошибок.