

**Ветвление**

## Инструкция «if»

Инструкция `if` вычисляет условие в скобках и, если результат `true`, то выполняет блок кода.

Например:

```
let year = prompt("Какой сейчас год?");
```

```
if (year == 2021) alert( "Вы правы!" );
```

Если мы хотим выполнить более одной инструкции, то нужно заключить блок кода в фигурные скобки:

```
if (year == 2021) {  
    alert( "Вы правы!" );  
    alert( "Вы такой умный!" );  
}
```

Рекомендуется использовать фигурные скобки {} всегда, когда вы используете инструкцию if, даже если выполняется только одна команда. Это улучшает читабельность кода.

## Преобразование к логическому типу

Инструкция `if` вычисляет выражение в скобках и преобразует результат к логическому типу.

Таким образом, код при таком условии никогда не выполнится:

```
if (0) {  
    // Какой-то код  
}
```

...а при таком – выполнится всегда:

```
if (1) {  
    // Какой-то код  
}
```

Мы также можем передать заранее вычисленное в переменной логическое значение в if, например так:

```
let cond = (year == 2021); // преобразуется к true или false
```

```
if ( cond ) {  
    // Какой-то код  
}
```

## Блок «else»

Инструкция if может содержать необязательный блок «else» («иначе»). Он выполняется, когда условие ложно.

Например:

```
let year = prompt("Какой сейчас год?");
```

```
if (year == 2021) {  
    alert( 'Правильно' );  
} else {  
    alert( 'А вот и неправильно!' ); // любое значение, кроме 2021  
}
```

## Несколько условий: «else if»

Иногда, нужно проверить несколько вариантов условия. Для этого используется блок else if.

Например:

```
let year = prompt("Какой сейчас год?");
```

```
if (year < 2021) {  
    alert( 'Нет это прошлое!' );  
} else if (year > 2021) {  
    alert( 'Это будущее' );  
} else {  
    alert( 'Верно!' );  
}
```

## Условный оператор “?”

Иногда нам нужно определить переменную в зависимости от условия.

Например:

```
let access;  
let age = prompt('Сколько вам лет?');  
  
if (age > 18) {  
    access = true;  
} else {  
    access = false;  
}  
alert(access);
```

Так называемый «условный» оператор «вопросительный знак» позволяет нам сделать это более коротким и простым способом.

Оператор представлен знаком вопроса ?. Его также называют «тернарный», так как этот оператор, единственный в своём роде, имеет три аргумента.

Синтаксис:

***результат = условие ? значение1 : значение2 ;***

Пример:

```
let age = prompt('Сколько вам лет?');  
let access = (age > 18) ? true : false;  
alert(access);
```

# Логические операторы

## || (ИЛИ)

Оператор «ИЛИ» выглядит как двойной символ вертикальной черты:

```
let result = a || b;
```

Традиционно в программировании ИЛИ предназначено только для манипулирования булевыми значениями: в случае, если какой-либо из аргументов true, он вернёт true, в противоположной ситуации возвращается false.

Существует всего четыре возможные логические комбинации:

```
alert( true || true ); // true  
alert( false || true ); // true  
alert( true || false ); // true  
alert( false || false ); // false
```

Как мы можем наблюдать, результат операций всегда равен true, за исключением случая, когда оба аргумента false.

```
let hour = prompt("Который час?");
```

```
if ( hour < 17 || hour > 20 ) {  
    alert( 'Курсы закончились!' );  
}
```

**&& (И)**

Оператор **И** пишется как два амперсанда &&:

```
let result = a && b;
```

В традиционном программировании И возвращает true, если оба аргумента истинны, а иначе – false:

```
alert( true && true ); // true
alert( false && true ); // false
alert( true && false ); // false
alert( false && false ); // false
```

```
let hour = 12;  
let minute = 30;  
  
if ( hour == 12 && minute == 30 ) {  
    alert( 'Сейчас 12:30' );  
}
```

# ! (НЕ)

Оператор НЕ представлен восклицательным знаком !.

```
let result = ! value;
```

Оператор принимает один аргумент и выполняет следующие действия:

1. Сначала приводит аргумент к логическому типу true/false.
2. Затем возвращает противоположное значение.

Например:

```
alert( !true ); // false
```

```
alert( !0 ); // true
```

# Циклы `while` и `for`

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 0 до 9 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены циклы.

# Цикл «while»

Цикл `while` имеет следующий синтаксис:

```
while ( /* какое-то условие */ ) {  
    // код  
    // также называемый "телом цикла"  
}
```

Код из тела цикла выполняется, пока условие ИСТИННО.

Цикл ниже выводит  $i$ , пока  $i < 3$ :

```
let i = 0;
while ( i < 3 ) { // выводит 0, затем 1, затем 2
    alert( i );
    i++;
}
```

Одно выполнение тела цикла по-научному называется *итерация*. Цикл в примере совершает три итерации.

## Цикл «do...while»

Проверку условия можно разместить под телом цикла, используя специальный синтаксис do..while:

```
do {  
    // тело цикла  
} while ( /* какое-то условие */ );
```

Цикл сначала выполнит тело, а затем проверит условие, и пока его значение равно true, он будет выполняться снова и снова.

## Цикл «for»

Более сложный, но при этом самый распространённый цикл — цикл for.

```
for ( /* начало */ ; /* условие */ ; /* шаг */ ) {  
    // тело  
}
```

```
for ( let i = 0 ; i < 3 ; i++ ) { // выведет 0, затем 1, затем 2  
    alert( i );  
}
```

## Прерывание цикла: «break»

Обычно цикл завершается при вычислении условия в `false`.

Но мы можем выйти из цикла в любой момент с помощью специальной директивы *break*.

Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт

```
let sum = 0;

while (true) {

    let value = +prompt("Введите число");

    if (!value) break;

    sum += value;

}

alert( 'Сумма: ' + sum );
```

## Переход к следующей итерации: continue

Директива `continue` – «облегчённая версия» `break`. При её выполнении цикл не прерывается, а переходит к следующей итерации (если условие все ещё равно `true`).

Например, цикл ниже использует `continue`, чтобы выводить только нечётные значения:

```
for ( let i = 0 ; i < 10 ; i++ ) {  
    if ( i % 2 == 0 ) continue; // если true, пропустить оставшуюся часть  
    alert(i); // 1, затем 3, 5, 7, 9  
}
```

# **Конструкция "switch"**

# Синтаксис

Конструкция switch имеет один или более блок case и необязательный блок default.

```
switch( x ) {  
    case 'value1': // if (x === 'value1')  
        // код  
        [break]  
    case 'value2': // if (x === 'value2')  
        // код  
        [break]  
    default:  
        // код  
        [break]  
}
```

```
const number = +prompt('Введите число 0 и 2', '');
```

```
if ( number === 0 ) {  
    alert('Вы ввели число 0');  
} else if ( number === 1 ) {  
    alert('Вы ввели число 1');  
} else if ( number === 2 ) {  
    alert('Вы ввели число 2');  
} else {  
    alert('Ошибка!');  
}
```