JS Boolean

Логический тип данных может иметь только два значения: true и false.

let x = true

console.log(typeof x) //вернет boolean

Условный оператор if

**Оператор**- автономная конструкция в языках программирования, содержащая команду или набор команд.

**Условный оператор** - оператор реализующий выполнение определённых команд при выполнении заданного условия.

Если говорить совсем простым языком, это определенный блок команд, который будет выполняться только при получении true внутри условия.

**Конструкция:**

if (<условие>) {

<выражение 1>

} else {

<выражение 2>

}

**Читается эта конструкция следующим образом:**

*Если условие выполняется (true), то выполняется выражение №1, иначе выполняется выражение №2*

Рассмотрим на примере

let x = 1

let y = 1

if (x === y) {

console.log('X равен Y')

} else {

console.log('X не равен Y')

}

x и y сравниваются между собой по значению и если их значения совпадают, то в консоль выводится 'X равен Y', если же значения отличаются, то выводится 'X не равен Y'.

Так же стоит запомнить, что условный оператор может ограничиться лишь одним выражением.

if (<условие>) {

<выражение>

}

В таком случае, в зависимости от выполнения условия выражение будет либо срабатывать, либо блок будет просто пропускаться.

Запомните!  
Никогда не стоит составлять условие следующим образом:

if (a === true) {

<какие-то действия>

}

Это будет показывать, что вы совершенно не понимаете как работает JS на базовом уровне и у вас нет опыта.

Внутри условия if всегда ожидается ответ **true**, потому сравнивать переменную с true или false не имеет никакого смысла.

Условный оператор может иметь вложенные условия любой глубины.

if (x === y) {

if (y === z) {

<выражение 1>

} else {

<выражение 2>

}

} else {

<выражение 3>

}

Кроме того, можно делать цепочку из уловных операторов следующиего вида:

if (x === y) {

<выражение 1>

} else if (x === z) {

<выражение 2>

} else {

<выражение 3>

}

Данная конструкция не ограничена по длине и может содержать столько условий, сколько вам необходимо.

Тернарный оператор

**Тернарный оператор** - это довольно элегантная запись обычного условного оператора if () {} и работает он таким же образом. Но с одной оговоркой - в нём всегда нужно указывать значение else, в отличие от обычного if, где else можно опускать.  
Конструкция:

<условие> ? <выражение 1> : <выражение 2>

(а === b) ? console.log('A = B') : console.log('A != B')

То есть при получении в условии результата true будет срабатывать выражение 1, иначе будет срабатывать выражение 2.

Так зачем же нам нужен тернарный оператор?

Всё дело в том, что обычный оператор if может выполнять какие-то действия, а тернарный оператор больше служит для присвоения значений.

Да, обычный оператор if может делать абсолютно всё, что может делать тернарный оператор, но для элегантной записи небольшого условия лучше подойдет именно тернарный.

*Пример:*

Вариант с if

if (isUserHadBreakfast) {

z = 'had breakfast'

} else {

z = 'no'

}

Вариант с тернарным оператором

z = isUserHadBreakfast ? 'had breakfast' : 'no'

Оба данных варианта записи делают одно и то же, но запись тернарного оператора куда компактнее и занимает всего 1 строчку кода.

Операторы сравнения

Значения в JS могут сравниваться двумя типами:

>>>>>>>>>> обычное равенство

>>>>>>>>>> строгое равенство  
  
Разница между ними заключается в том, что оператор строгого равенства === проверяет равенство без приведения типов.  
А это значит следующее

console.log( '1' == 1 ) //вернет true

console.log( '1' === 1 ) //вернет false

В первом случае значения приведутся к одному типу (строке или числу), а во втором случае сраниваться будет не только значение, но и сам тип. Очевидно, что строка '1' не может быть равна числу 1.

Всегда используйте строгое сравнение, это защитит вас от проблем в будущем.

**Отрицание**

Отрицание (логическое **НЕ**) записывается добавлением восклицательного знака перед равенством.

console.log( 1 + 1 !== 2 ) //вернет false

console.log( 1 + 2 !== 2 ) //вернет true

**Логические операции**

Для понимания логических операций будем считать, что **true = 1**, **false = 0**.

Стоит усвоить:

логическое И (&&) - это операция логического умножения.  
логическое ИЛИ (||) - это операция логического сложения.

Обратимся к элементарной арифметике, только в двоичной системе, где ответ может быть только 0 или 1

Конъюнкция - логическое умножение

true && true = 1 \* 1 = 1

true && false = 1 \* 0 = 0

false && true = 0 \* 1 = 0

false && false = 0 \* 0 = 0

Дизъюнкция - логическое сложение

true || true = 1 + 1 = 1

true || false = 1 + 0 = 1

false || true = 0 + 1 = 1

false || false = 0 + 0 = 0