

# JS: Operators

# НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

# Повторим;)

Какие способы объявления переменных есть в JavaScript?

Сколько примитивных типов в JavaScript?

Что входит в тип Object

С помощью какого оператора можно проверить тип переменной?

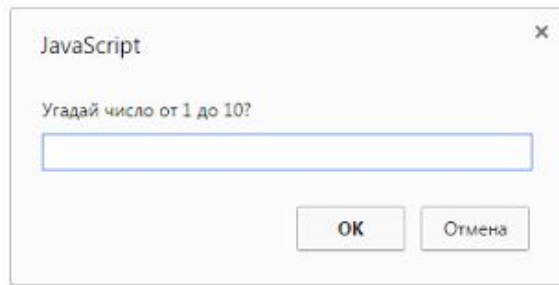
# ЦЕЛЬ

Изучить различные виды операторов в JavaScript

# ПЛАН ЗАНЯТИЯ

- Math operators
- Comparison operators
- Conditional operators

# Встроенные функции. Взаимодействие с пользователем



A screenshot of a JavaScript alert dialog box. The title bar at the top left says "JavaScript" and has a close button (X) at the top right. The main text inside the dialog asks "Угадай число от 1 до 10?". Below the text is a single-line text input field. At the bottom of the dialog are two buttons: "ОК" (OK) on the left and "Отмена" (Cancel) on the right.

# console.log()

Чтобы вывести что-то на консоль из нашего кода, существует функция `console.log`.

Обычный пользователь сайта не увидит такой вывод, так как он в консоли. Чтобы увидеть его, либо откройте консольную панель инструментов разработчика, либо нажмите Esc, находясь в другой панели: это откроет консоль внизу.

```
let message = "Привет, мир!";  
console.log(message);
```

# alert()

- alert используется для вывода всплывающего диалогового окна с сообщением.
- Принимает один параметр - текст сообщения.

Она показывает сообщение и ждёт, пока пользователь нажмёт кнопку «ОК».

```
1 alert("Hello");
```



# prompt()

- `prompt` используется для вывода всплывающего диалогового окна с полем для ввода текста.
- Принимает два параметра: текст сообщения и необязательное значение по умолчанию для поля ввода.

```
1 let age = prompt('Сколько тебе лет?', 100);  
2
```

# confirm()

Функция `confirm` отображает модальное окно с текстом вопроса `question` и двумя кнопками: ОК и Отмена.

```
1 let isBoss = confirm("Ты здесь главный?");  
2  
3 alert( isBoss ); // true, если нажата ОК
```

# Выражения

```
= += -= *= /= %= << >>=  
x != y; x %= 2;  
x = 5 + 12;
```

## Expressions

```
true || false j = i++;  
var result = 5 + 4 * 12 / 4;
```

**В JavaScript поддерживаются следующие математические операторы:**

**Сложение +,**

**Вычитание −,**

**Умножение \*,**

**Деление /,**

**Взятие остатка от деления %,**

**Возведение в степень \*\*.**

## Сложение

+

Объединение чисел в одно целое.

```
let a = 5;  
let b = 10;  
let result = a + b;  
console.log(result); // Выведет 15
```

## Вычитание

-

Вычитает правое число от левого.

```
let a = 10;  
let b = 2;  
let result = a - b;  
console.log(result); // Выведет 8
```

## Умножение

\*

Умножает два числа вместе.

```
let a = 3;  
let b = 7;  
let result = a * b;  
console.log(result); // Выведет 21
```

## Деление

/

Делит левое число на правое.

```
let a = 10;  
let b = 2;  
let result = a / b;  
console.log(result); // Выведет 5
```



## Взятие остатка %

Оператор взятия остатка %, несмотря на обозначение, никакого отношения к процентам не имеет.

Результат  $a \% b$  – это остаток от целочисленного деления  $a$  на  $b$ .

Например:

```
1 alert( 5 % 2 ); // 1, остаток от деления 5 на 2
2 alert( 8 % 3 ); // 2, остаток от деления 8 на 3
3 alert( 8 % 4 ); // 0, остаток от деления 8 на 4
```

## Возведение в степень \*\*

Оператор возведения в степень  $a ** b$  возводит  $a$  в степень  $b$ .  
В школьной математике мы записываем это как  $a^b$ .

Например:

```
1 alert( 2 ** 2 ); // 22 = 4
2 alert( 2 ** 3 ); // 23 = 8
3 alert( 2 ** 4 ); // 24 = 16
```

## Сложение строк при помощи +

Давайте рассмотрим специальные возможности операторов JavaScript, которые выходят за рамки школьной арифметики.

Обычно при помощи плюса '+' складывают числа.

Но если оператор '+' применить к строкам, то он их объединяет в одну:

```
1 let s = "моя" + "строка";  
2 alert(s); // моястрока
```

## Сложение строк и чисел при помощи +

Обратите внимание, если хотя бы один операнд является строкой, то второй будет также преобразован в строку.

Например:

```
1 alert( '1' + 2 ); // "12"  
2 alert( 2 + '1' ); // "21"
```

Какой будет результат?

```
1 alert(2 + 2 + '1' );
```

Другие арифметические операторы работают только с числами и всегда преобразуют операнды в числа.

Например, вычитание и деление:

```
1 alert( 6 - '2' ); // 4, '2' приводится к числу
2 alert( '6' / '2' ); // 3, оба операнда приводятся к числам
```

## Строковое преобразование

Строковое преобразование происходит, когда требуется представление чего-либо в виде строки, мы можем использовать функцию **String(value)**, чтобы преобразовать значение к строке:

```
1 let value = true;
2 alert(typeof value); // boolean
3
4 value = String(value); // теперь value это строка "true"
5 alert(typeof value); // string
```

## Численное преобразование

Мы можем использовать функцию **Number(value)**, чтобы явно преобразовать value к числу:

```
1 let str = "123";  
2 alert(typeof str); // string  
3  
4 let num = Number(str); // становится числом 123  
5  
6 alert(typeof num); // number
```

## Логическое преобразование

Логическое преобразование может быть выполнено явно с помощью функции **Boolean(value)**.

```
1 alert( Boolean(1) ); // true
2 alert( Boolean(0) ); // false
3
4 alert( Boolean("Привет!") ); // true
5 alert( Boolean("") ); // false
```



## Инкремент/декремент

Одной из наиболее частых числовых операций является увеличение или уменьшение на единицу.

Для этого существуют даже специальные операторы:

**Инкремент ++ увеличивает переменную на 1:**

```
1 let counter = 2;  
2 counter++;      // работает как counter = counter + 1,  
3 alert( counter ); // 3
```

**Декремент -- уменьшает переменную на 1:**

```
1 let counter = 2;  
2 counter--;      // работает как counter = counter - 1,  
3 alert( counter ); // 1
```

# Операторы сравнения

`== & ===`

## В JavaScript существуют следующие операторы сравнения:

- **Больше/меньше:**  $a > b$ ,  $a < b$ .
- **Больше/меньше или равно:**  $a \geq b$ ,  $a \leq b$ .
- **Равно:**  $a == b$ . Обратите внимание, для сравнения используется двойной знак равенства  $==$ . Один знак равенства  $a = b$  означал бы присваивание.
- **Строгое равно:**  $a === b$
- **Не равно:**  $a \neq b$ .

Все операторы сравнения возвращают значение логического типа:

`true` – означает «да», «верно», «истина».

`false` – означает «нет», «неверно», «ложь».



Больше, меньше >, <

Больше или равно >=

Меньше или равно <=

```
let num1 = 8;
```

```
let num2 = 12;
```

```
console.log(num1 > num2); // false
```

```
console.log(num1 < num2); // true
```

```
console.log(num1 >= num2); // false
```

```
console.log(num1 <= num2); // true
```

Оператор нестрогого равенства `==` проверяет равенство с приведением к общему типу.

```
1 alert( 0 == false ); // true
```

```
1 alert( '' == false ); // true
```

**Оператор строгого равенства === проверяет равенство без приведения типов.**

Другими словами, если a и b имеют разные типы, то проверка a === b немедленно возвращает false без попытки их преобразования.

```
1 alert( 0 === false ); // false, так как сравниваются разные типы
```

## != - оператор нестрогого неравенства

Проверяет неравенство значений

Нестрогое неравенство, не выполняет приведение типов перед сравнением - !=.

```
let x = 5;
```

```
let y = "5";
```

```
console.log(x != y); // false
```



## **!== - оператор строгого неравенства**

Проверяет неравенство значений

Строгое неравенство, выполняет приведение типов перед сравнением -  
!==.

```
let x = 5;
```

```
let y = "5";
```

```
console.log(x !== y); // true
```

## Сравнение разных типов

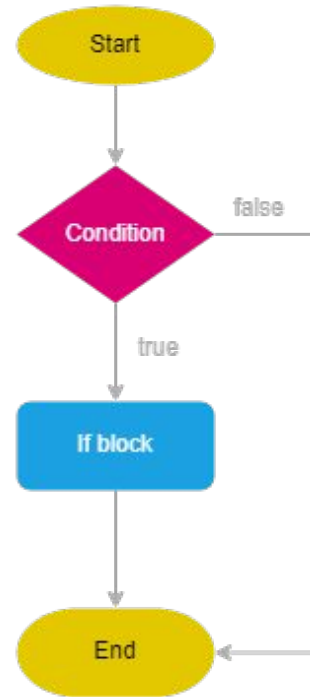
При сравнении значений разных типов JavaScript приводит каждое из них к числу.

```
1 alert( '2' > 1 ); // true, строка '2' становится числом 2
2 alert( '01' == 1 ); // true, строка '01' становится числом 1
```

Логическое значение true становится 1, а false – 0.

```
1 alert( true == 1 ); // true
2 alert( false == 0 ); // true
```

# Условные операторы



Иногда нам нужно выполнить различные действия в зависимости от условий. Для этого мы можем использовать инструкцию `if`

## Инструкция «if»

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

```
1  if (year == 2015) {  
2      alert( "Правильно!" );  
3      alert( "Вы такой умный!" );  
4  }
```

## Преобразование к логическому типу

Инструкция **if (...)** вычисляет выражение в скобках и преобразует результат к логическому типу.

Преобразование типов:

- Число 0, пустая строка "", null, undefined и NaN становятся false. Из-за этого их называют «ложными» («falsy») значениями.
- Остальные значения становятся true, поэтому их называют «правдивыми» («truthy»).

## Блок «else»

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

Например:

```
if (year == 2015) {  
    alert( 'Да вы знаток!' );  
} else {  
    alert( 'А вот и неправильно!' ); // любое значение, кроме 2015  
}
```

## Несколько условий: «else if»

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

```
if (year < 2015) {  
    alert( 'Это слишком рано...' );  
} else if (year > 2015) {  
    alert( 'Это поздновато' );  
} else {  
    alert( 'Верно!' );  
}
```

## Условный оператор „?“ (тернарный)

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.

```
1 let result = условие ? значение1 : значение2;
```



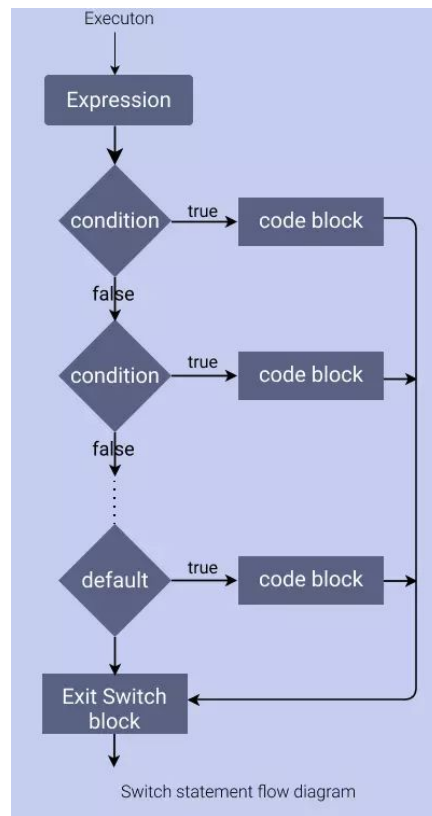
```
1 let accessAllowed = (age > 18) ? true : false;
```



## Конструкция "switch"

Конструкция **switch** заменяет собой сразу несколько if.

Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.



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

Выглядит она так:

```
1  switch(x) {  
2      case 'value1': // if (x === 'value1')  
3          ...  
4          [break]  
5  
6      case 'value2': // if (x === 'value2')  
7          ...  
8          [break]  
9  
10     default:  
11         ...  
12         [break]  
13 }
```

Переменная **x** проверяется на строгое равенство первому значению **value1**, затем второму **value2** и так далее.

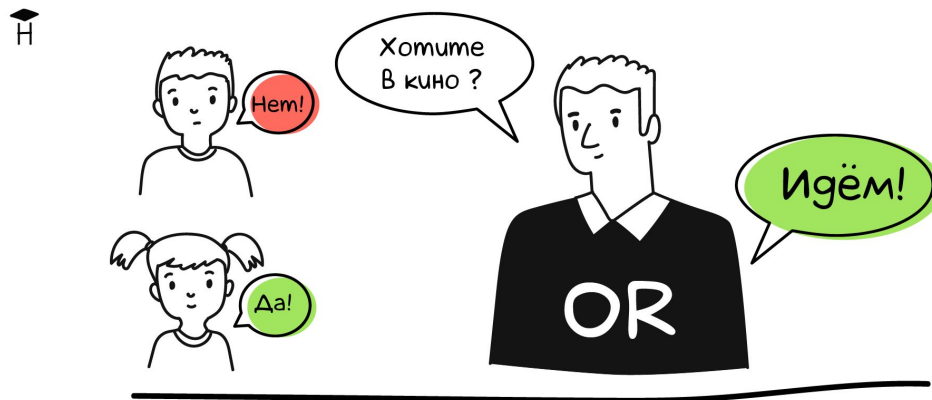
Если соответствие установлено – **switch** начинает выполняться от соответствующей директивы **case** и далее, до ближайшего **break** (или до конца **switch**).

Если ни один **case** не совпал – выполняется (если есть) вариант **default**.

## Пример использования switch (сработавший код выделен):

```
1  let a = 2 + 2;  
2  
3  switch (a) {  
4      case 3:  
5          alert( 'Маловато' );  
6          break;  
7      case 4:  
8          alert( 'В точку!' );  
9          break;  
10     case 5:  
11         alert( 'Перебор' );  
12         break;  
13     default:  
14         alert( "Нет таких значений" );  
15 }
```

# Логические операторы



Папа, живет по принципу  
"Если хотя бы кто-нибудь"

## || (ИЛИ)

Оператор «ИЛИ» выглядит как двойной символ вертикальной черты:

```
1 result = a || b;
```

Существует всего четыре возможные логические комбинации:

```
1 alert( true || true ); // true
2 alert( false || true ); // true
3 alert( true || false ); // true
4 alert( false || false ); // false
```

## || (или)

Обычно оператор || используется в if для проверки истинности любого из заданных условий.

```
1 let hour = 9;  
2  
3 if (hour < 10 || hour > 18) {  
4   alert( 'Офис закрыт.' );  
5 }
```

Можно передать и больше условий:

```
1 let hour = 12;  
2 let isWeekend = true;  
3  
4 if (hour < 10 || hour > 18 || isWeekend) {  
5   alert( 'Офис закрыт.' ); // это выходной  
6 }
```

## && (И)

Оператор И пишется как два амперсанда **&&** и возвращает true, если оба аргумента истинны, а иначе – false:

```
1 alert( true && true );    // true
2 alert( false && true );   // false
3 alert( true && false );   // false
4 alert( false && false );  // false
```

## && (И)

Пример с if:

```
1 let hour = 12;  
2 let minute = 30;  
3  
4 if (hour == 12 && minute == 30) {  
5     alert( 'Время 12:30' );  
6 }
```



## ! (НЕ)

Оператор **НЕ** представлен восклицательным знаком !

Оператор принимает один аргумент и выполняет следующие действия:

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

```
1 alert( !true ); // false
2 alert( !0 ); // true
```

## Шаблонные строки (Template Strings)

Шаблонные строки предоставляют удобный способ вставки переменных и выражений в строки.

В шаблонных строках можно использовать выражения, заключенные в `${}`

```
let myName = "Alice";  
let greeting = `Hello, ${myName}!`;   
console.log(greeting); // "Hello,  
Alice!"
```



# **Ваша новая IT-профессия – Ваш новый уровень жизни**

Программирование с нуля в  
немецкой школе AIT TR GmbH