# Тема 3. Операторы и выражения


хекслет колледж

# Цель занятия:

Сформировать у студентов понимание операторов JavaScript, их видов и порядка выполнения, а также научить составлять и анализировать выражения, используемые в условиях и вычислениях.

хекслет колледж

# Учебные вопросы:

1. Понятие оператора, операнда и выражения
2. Арифметические операторы
3. Операторы присваивания
4. Операторы сравнения
5. Логические операторы
6. Приоритет операторов
7. Math. Свойства и методы.

хекслет колледж

# 1. Понятие оператора, операнда и выражения

## Оператор.

Оператор в JavaScript — это специальный символ или ключевое слово, которое выполняет определённое действие над данными.

Операторы указывают:

- что именно нужно сделать с данными,
- какую операцию выполнить.

хекслет колледж

# Примеры операторов:

\+ — сложение

\- — вычитание

= — присваивание

> — сравнение

&& — логическое И

**let sum = 5 + 3;**

В данном примере операторы: **=** и **+**.

хекслет колледж

# Операнд

Операнд — это значение, переменная или выражение,
над которым оператор выполняет действие.

**5 + 3**

**+** — оператор
**5** и **3** — операнды

хекслет колледж

# Виды операторов по количеству операндов

- Унарные (1 операнд):

**!isAdmin**
**typeof age**

- Бинарные (2 операнда):

**a + b**
**x === y**

- Тернарные (3 операнда, будут изучены позже):

**condition ? value1 : value2**

хекслет колледж

# Выражение

Выражение — это комбинация операторов и операндов, которая вычисляется и возвращает значение.
Примеры выражений:

**10 + 5**
**age > 18**
**isLogged && isAdmin**

Каждое выражение имеет результат:

- число,
- строку,
- логическое значение.

хекслет колледж

# Связь понятий

Операторы выполняют действия
Операнды — данные, над которыми выполняются действия
Выражения — результат применения операторов к операндам
Пример:

**let result = (10 + 5) * 2;**
**(10 + 5) * 2** — выражение

**+** и ***** — операторы
**10, 5, 2** — операнды

хекслет колледж

# Итог

Оператор выполняет действие над операндами и формирует выражение, а выражение всегда можно вычислить и получить результат.

хекслет колледж

# 2. Арифметические операторы

Арифметические операторы используются для выполнения математических вычислений над числовыми значениями и переменными.

Они применяются:

- при расчётах,
- в формулах,
- при обработке числовых данных.

хекслет колледж

# Основные арифметические операторы

| Оператор | Назначение | Пример |
|---|---|---|
| + | сложение | 5 + 3 |
| - | вычитание | 10 - 4 |
| * | умножение | 6 * 2 |
| / | деление | 10 / 2 |

хекслет колледж

# Оператор остатка от деления %

Оператор **%** возвращает остаток от деления одного числа на другое.

```
10 % 3; // 1
```

Используется, например:

- для проверки чётности числа,

хекслет колледж

# Инкремент и декремент

Инкремент (**++**) — увеличивает значение на 1
Декремент (**--**) — уменьшает значение на 1

```
let a = 5;
a++; // 6
a--; // 5
```

хекслет колледж

# Префиксная и постфиксная формы

```
let x = 5;
console.log(++x); // 6

let y = 5;
console.log(y++); // 5
```

Префиксная форма (**++x**) — сначала увеличивает, потом возвращает значение.
Постфиксная форма (**x++**) — сначала возвращает значение, потом увеличивает.

хекслет колледж

# Арифметика со строками

Оператор + со строками выполняет конкатенацию:

```
"10" + 5; // "105"
```

Остальные арифметические операторы приводят значения к числу:

```
"10" - 5; // 5
"10" * 2; // 20
```

хекслет колледж

# Особые числовые значения

**NaN** — результат некорректного числового вычисления

**Infinity** — результат деления на ноль

```
10 / 0;       // Infinity
"abc" / 2;  // NaN
```

хекслет колледж

# Итог:

Арифметические операторы:

- выполняют математические действия,
- работают в выражениях,
- часто используются совместно с переменными и преобразованием типов.
- работают с числами, но оператор **+** может работать и со строками.

хекслет колледж

# 3. Операторы присваивания

Оператор присваивания (**=**) используется для:
сохранения значения в переменной,
изменения существующего значения переменной.
Примеры:

```
let x = 10; // присваиваем 10 переменной x
let y = x;  // присваиваем значение x переменной y
```

Левое значение (переменная) получает правое значение (число, строку, выражение).

хекслет колледж

# Составные операторы присваивания

Составные операторы позволяют одновременно изменять значение переменной и присваивать его.

| Оператор | Описание | Пример |
|---|---|---|
| += | сложение и присваивание | x += 5; // x = x + 5 |
| -= | вычитание и присваивание | x -= 3; // x = x - 3 |
| *= | умножение и присваивание | x *= 2; // x = x * 2 |
| /= | деление и присваивание | x /= 4; // x = x / 4 |
| %= | остаток от деления и присваивание | x %= 3; // x = x % 3 |

хекслет колледж

# Примеры использования:

```
let score = 10;


score += 5; // score = 15
score -= 3; // score = 12
score *= 2; // score = 24
score /= 4; // score = 6
score %= 4; // score = 2
```

хекслет колледж

# Для строк можно использовать += для конкатенации:

```
let greeting = "Hello";
greeting += " World";    // "Hello World"
```

хекслет колледж

# Итог:

- **=** присваивает значение переменной.
- **+=, -=, *=, /=, %=** — сокращённые формы присваивания с операцией.
- Составные операторы повышают удобство и читаемость кода.
- Операторы присваивания связывают переменную с выражением и могут одновременно изменять её значение.

хекслет колледж

# 4. Операторы сравнения

Операторы сравнения используются для:

- сравнения двух значений,
- получения логического результата: **true** или **false**.
- Они важны для условий, циклов и логики программы.

хекслет колледж

# Основные операторы сравнения

| Оператор | Описание | Пример |
|---|---|---|
| == | равно (с приведением типов) | 5 == "5" → true |
| === | строго равно (без приведения типов) | 5 === "5" → false |
| != | не равно (с приведением типов) | 5 != "6" → true |
| !== | строго не равно (без приведения типо | 5 !== "5" → true |
| > | больше | 10 > 5 → true |
| < | меньше | 3 < 7 → true |
| >= | больше или равно | 5 >= 5 → true |
| <= | меньше или равно | 4 <= 5 → true |

хекслет колледж

# Разница между == и ===

**==** — выполняет неявное преобразование типов, поэтому сравнение "5" == 5 вернёт **true**.
**===** — проверяет и значение, и тип, поэтому "5" === 5 вернёт false.

Рекомендуется использовать **===** и **!==**, чтобы избежать неожиданных результатов.

хекслет колледж

# Примеры использования:

```
let a = 10;
let b = "10";

console.log(a == b);    // true
console.log(a === b);  // false
console.log(a != 5);     // true
console.log(a > 7);      // true
console.log(a <= 10);  // true
```

хекслет колледж

# Сравнение строк

Строки сравниваются по Unicode-коду символов.

```
console.log("abc" > "abd");          // false
console.log("apple" < "banana");  // true
```

Когда JavaScript сравнивает строки, он смотрит на Unicode-код каждого символа.
Символы в таблице Unicode расположены примерно в алфавитном порядке для латиницы, а цифры идут раньше букв.
То есть 'a' < 'b' и 'A' < 'B'.
Сравнение идёт символ за символом слева направо, пока не найдётся первое различие.

хекслет колледж

# Итог:

- Используйте строгие операторы (**===**, **!==**) для надёжности.
- Сравнение всегда возвращает **true** или **false**.
- строки сравниваются посимвольно по кодам символов.
- Для чисел — сравнение как обычно в математике.

хекслет колледж

# 5. Логические операторы

Логические операторы позволяют:

- объединять или изменять логические значения (true / false),
- строить сложные условия для ветвлений и циклов.

В JavaScript логические операторы всегда возвращают логическое значение.

хекслет колледж

# Основные логические операторы

| Оператор | Назначение | Пример |
|---|---|---|
| && | логическое И | true && false → false |
| \|\| | логическое ИЛИ | true // false → true |
| ! | логическое НЕ | !true → false |

хекслет колледж

# Логическое И (&&)

Возвращает **true**, если оба значения истинны. Иначе возвращает **false**.

```
let a = true;
let b = false;
console.log(a && b); // false
console.log(true && true); // true
```

хекслет колледж

# Логическое ИЛИ (||)

Возвращает **true**, если хотя бы одно значение истинно.
Только если оба значения **false** — результат false.

```
let a = true;
let b = false;
console.log(a || b); // true
console.log(false || false); // false
```

хекслет колледж

# Логическое НЕ (!)

Меняет логическое значение на противоположное.

```
let isAdmin = true;
console.log(!isAdmin); // false
```

хекслет колледж

# Особенности логических операторов

JavaScript использует короткое замыкание (short-circuit):

- Для **&&** проверка останавливается при первом false.
- Для **||** проверка останавливается при первом true.

Логические операторы работают с любым типом, но результат всегда приводится к **true** или **false**.

хекслет колледж

# Итог:

- && — логическое И, оба значения должны быть истинными.
- || — логическое ИЛИ, достаточно одного истинного значения.
- ! — логическое НЕ, меняет значение на противоположное.
- Логические операторы позволяют строить условия, которые зависят от нескольких проверок одновременно.

хекслет колледж

# 6. Приоритет операторов

Приоритет операторов — это правило, которое определяет порядок выполнения операций в выражении.
Если выражение содержит несколько операторов, JavaScript сначала выполняет операции с более высоким приоритетом, а затем — с более низким.
Скобки () позволяют изменить порядок выполнения, повышая приоритет операций внутри них.

хекслет колледж

# Примеры приоритета:

```
let result = 2 + 3 * 4;
console.log(result); // 14
```

***** имеет высший приоритет, чем **+**
Сначала выполняется 3 * 4 = 12
Потом 2 + 12 = 14

```
let result = (2 + 3) * 4;
console.log(result); // 20
```

Скобки изменили порядок: сначала 2 + 3 = 5, потом 5 * 4 = 20

хекслет колледж

# Уровни приоритета операторов

| Уровень | Операторы | Пример |
|---|---|---|
| Высокий | (), ++, -- | (2 + 3), x++ |
| Высокий | Унарные +, -, ! | -a, !isAdmin |
| Средний | *, /, % | 5 * 2, 10 / 2 |
| Средний-низкий | +, - | 2 + 3, 5 - 1 |
| Низкий | <, >, <=, >= | a > b |
| Низкий | ==, ===, !=, !== | x === y |
| Очень низкий | && | true && false |
| Самый низкий | ` | let greeting = `Привет, ${name}!` |
| Присваивание | =, +=, -=, *= и т.д. | x = 5, x += 2 |

Чем выше в таблице — тем раньше выполняется операция.
Операции на одном уровне выполняются слева направо.

хекслет колледж

# Практические рекомендации:

- Используйте скобки для ясности и контроля порядка выполнения.
- Помните, что арифметические операции выполняются сначала, потом сравнения, потом логические операции.
- При сложных выражениях лучше разделять их на несколько переменных для читаемости.

хекслет колледж

# Примеры сложных выражений:

```
let a = 5, b = 10, c = 2;
let result = a + b * c > 20 && b > c;
console.log(result); // true
```

Разбор по приоритету:
b * c = 10 * 2 = 20
a + (b * c) = 5 + 20 = 25
25 > 20 → true
b > c = 10 > 2 → true
true && true → true

хекслет колледж

# Итог:

- Приоритет операторов определяет, какая операция выполняется раньше.
- Скобки всегда дают возможность явно задать порядок.
- Правильное понимание приоритета предотвращает неожиданные результаты вычислений.

хекслет колледж

# 7. Math. Свойства и методы.

Math — это встроенный объект в JavaScript, который предоставляет набор математических функций и констант.
Его методы и свойства можно использовать для выполнения различных математических операций, таких как округление, нахождение максимума/минимума, генерация случайных чисел и многое другое.

хекслет колледж

# Свойства Math

**Math.PI** — значение числа π (пи), приблизительно 3.14159.
**console.log(Math.PI); // 3.141592653589793**

**Math.E** — основание натурального логарифма (число Эйлера), приблизительно 2.718.

хекслет колледж

# Методы Math

## Округление чисел

**Math.round(x)** — округление числа до ближайшего целого.

```
console.log(Math.round(4.7)); // 5
console.log(Math.round(4.3)); // 4
```

**Math.floor(x)** — округление вниз (в меньшую сторону).
**Math.ceil(x)** — округление вверх (в большую сторону).
**Math.trunc(x)** — отбрасывает дробную часть числа (не округляет).

хекслет колледж

# Минимум и максимум

**Math.min(a, b, ...)** — возвращает наименьшее из переданных чисел.

**Math.max(a, b, ...)** — возвращает наибольшее из переданных чисел.

хекслет колледж

# Случайные числа

**Math.random()** — возвращает псевдослучайное число от 0 (включительно) до 1 (не включительно).

**console.log(Math.random()); // Например, 0.7324040990734169**

Генерация случайных чисел в диапазоне:

```
// Случайное число от 0 до 10 (целое)
const randomInt = Math.floor(Math.random() * 10);
```

хекслет колледж

# Степени и корни

**Math.pow(base, exponent)** — возведение в степень.

**Math.sqrt(x)** — квадратный корень.

**Math.cbrt(x)** — кубический корень.

хекслет колледж

# Абсолютное значение

**Math.abs(x)** — возвращает абсолютное значение числа (модуль числа).

```
console.log(Math.abs(-10));   // 10
```

хекслет колледж

# Тригонометрические функции

**Math.sin(x), Math.cos(x), Math.tan(x)** — тригонометрические функции (аргумент в радианах).

```
console.log(Math.sin(Math.PI / 2)); // 1
console.log(Math.cos(0)); // 1
```

хекслет колледж

# Логарифмы

**Math.log(x)** — натуральный логарифм (по основанию e).

**Math.log10(x)** — десятичный логарифм (по основанию 10).

**Math.log2(x)** — логарифм по основанию 2.

хекслет колледж

# Контрольные вопросы:

- Что такое оператор в JavaScript? Приведите пример.
- Что такое операнд? Приведите пример.
- Какие арифметические операторы вы знаете? Для чего используется %?
- В чём разница между x++ и ++x?
- Что произойдет при сложении числа и строки с помощью +?
- Как работают составные операторы присваивания (+=, -=, *=)? Приведите пример.
- Можно ли использовать += с константой (const)? Почему?
- В чём разница между == и ===?
- Что будет результатом сравнения "10" > 5? Почему?
- Что делает оператор !?
- Что такое приоритет операторов?
- Какие операторы имеют самый низкий приоритет?
- Как скобки () влияют на порядок выполнения операций?.

хекслет колледж

# Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

хекслет колледж

@HEXLY.KZ