О преподавателе

Басов Денис Алексеевич

- PHP
- JavaScript
- SQL
- HTML
- CSS

Описание курса

JavaScript. Уровень 1. Основы JavaScript

Тема	Часы
Модуль 1. Основы программирования *Операторы, переменные, типы данных…*	4
Модуль 2. Управляющие конструкции *if-else, for, while, switch…*	4
Модуль 3. Функции *Синтаксис функции, аргументы, области видимости…*	4
Модуль 4. Объектные типы *Свойства, методы, массивы…*	4
Модуль 5. Объектно-ориентированное программирование *Функция конструктор, прототипы, классы…*	4
Модуль 6. Дополнительная информация *Работа со строками, регулярные выражения, JSON…*	4

Программа курса предусматривает лабораторные работы по каждой теме, а также выполнение домашних заданий и контроль знаний

В рамках курса мы научимся:

- Понимать сущность программирования
- Использовать условные операторы
- Работать с циклическими операторами
- Создавать и использовать функции
- Понимать специфику веб-программирования
- Твердо знать язык JavaScript без привязки к среде исполнения

Бауманский учебный центр
Специалист.ru
Мировые стандарты обучения

Модуль 1 Основы программирования на JavaScript

Басов Денис

В этом модуле мы рассмотрим:

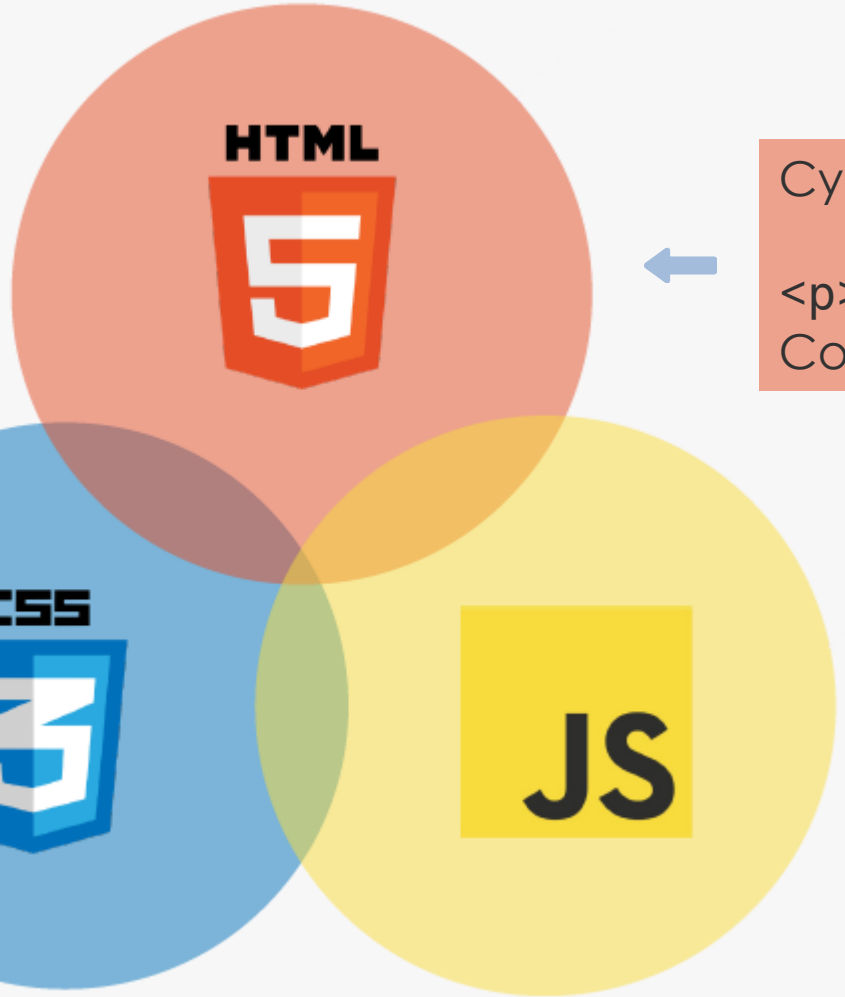
- Введение в JavaScript
- Типы данных
- Операторы
- Переменные и константы
- Работа с консолью браузера
- Выражения и инструкции
- Манипуляции с базовыми типами

Что такое JavaScript ?

- JavaScript - это прототипно-ориентированный, мультипарадигмальный, однопоточный, динамический язык, поддерживающий объектно-ориентированные, императивные и декларативные стили
- JavaScript был создан для того, чтобы сделать веб-страницы динамичными
- Наиболее широкое применение нашел в браузерах
- **JavaScript** не имеет никакого отношения к **Java**

Взаимодействие Front-end технологий

Контент



Существительное

```
<p>Hello!<p/>
```

Создаем параграф

Прилагательное

```
p{color: blue}
```

Задаем цвет

Глагол

```
p.hide()
```

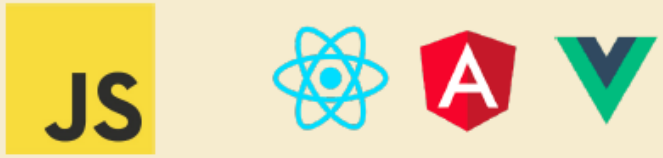
Скрываем параграф

Отображение

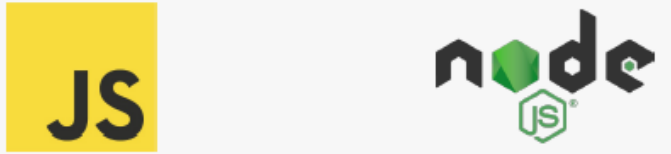
Поведение

Для чего применяется JavaScript

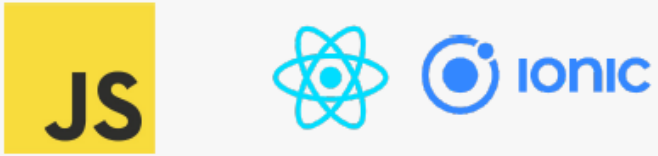
Динамические эффекты и Веб-приложения в браузере



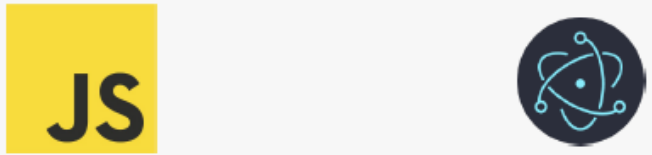
Серверные веб-приложения



Мобильные приложения



Приложения для ПК



История развития JavaScript

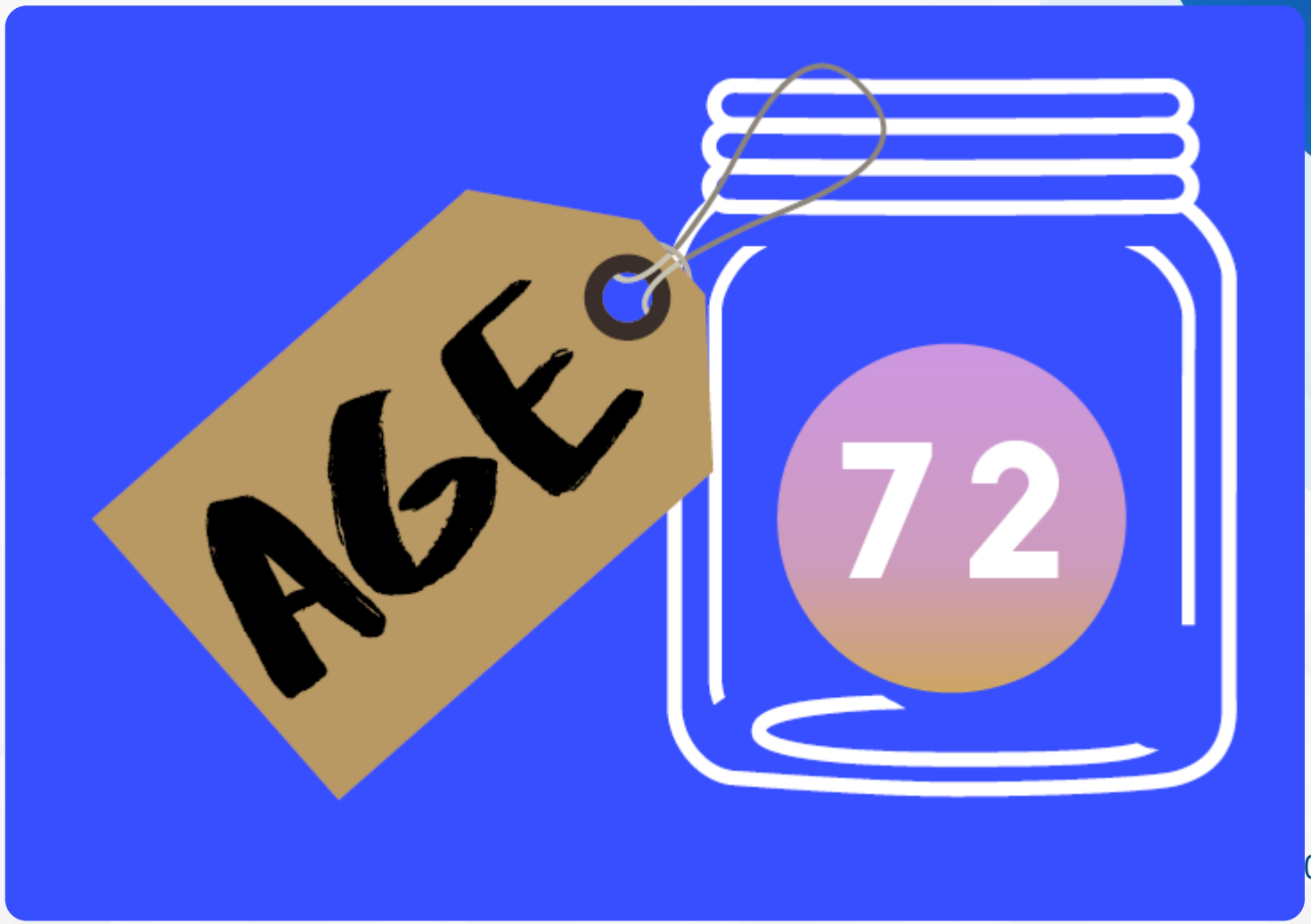
- **1995 год:** JavaScript был создан Бренданом Эйхом в компании Netscape
- **1996 год:** Язык был переименован в JavaScript, чтобы связать свою успешную судьбу с языком программирования Java
- **1997 год:** JavaScript стал стандартом ECMA-26214. После того, как Netscape передала JavaScript в ECMA, фонд Mozilla продолжил разработку JavaScript для браузера Firefox
- **1998 год:** ECMAScript 2
- **1999 год:** ECMAScript 3
- **2009 год:** ECMAScript 5
- **2015 год:** ECMAScript 6. Шестая версия стандарта, также известная как ECMAScript 2015
- С **2016** года каждый год выпускается новая версия стандарта

Переменные в JavaScript

Переменные похожи на «Именованные **контейнеры**» для значений в JavaScript

Мы можем сохранить значение и дать ему название, а еще…

- использовать
- изменить значение позже
- удалить



Базовый синтаксис объявления переменных

```
let someName = value;
```

Ключевое слово — let

Название переменной — someName

Значение переменной — value

```
let age = 55;
```

Создаем переменную с названием «age» и значением 55

Использование переменных

```
// объявляем переменную cats и присваиваем значение 4
let cats = 4;

// объявляем переменную dogs и присваиваем значение 6
let dogs = 6;

// складываем значения переменных
cats + dogs;
```

Типы данных

В JavaScript существует семь примитивных типов данных:

- **Number:** Используется для представления чисел. Например, 123, 120.50 и т.д
- **String:** используется для представления последовательности символов. Например, "Hello, World!"
- **Boolean:** Используется для представления логических значений. Он может принимать только два значения: true или false.
- **Undefined:** Если переменная объявлена, но не присвоено никакого значения, то ее значение будет undefined
- **Null:** Означает отсутствие значения
- **Symbol:** Используется для представления уникальных идентификаторов
- **BigInt:** Используется для представления целых чисел, которые больше, чем может хранить Number

Тип данных Number

Целые и десятичные числа: Числовой тип данных **number** представляет как целочисленные значения, так и числа с плавающей точкой

Специальные числовые значения: Кроме обычных чисел, существуют так называемые «специальные числовые значения», которые относятся к этому типу данных: Infinity, -Infinity и NaN

Number имеет ограничения на длину значения в пределах 64-битового формата. Это означает, что безопасно доступны целые числа в диапазоне от -(2^53 - 1) до (2^53 - 1) За пределами этого диапазона операции с целыми числами будут небезопасными, и возвращать приближённые значения.

Для работы с очень большими числами в JavaScript есть тип данных **BigInt**

Простые операции с числами

```
// сложение
2 + 2; // 4

// вычитание
6 - 3; // 3

// умножение
2 * 5; // 10

// деление
10 / 2; // 5

// возведение в степень
2 ** 3; // 8

// остаток от деления нацело
10 % 3; // 1
```

В JavaScript мы можем использовать все базовые математические операции

// - дает возможность написать комментарий, который игнорируется при выполнении программы

NaN – not a number, infinity

```
0 / 0; // NaN

"яблоко" / 5; // NaN

48 + NaN; // NaN

45 / 0; // Infinity

45 * 10000 ** 500; // Infinity
```

NaN – это **числовое** значение, которое означает, что это … не число

NaN часто получают при Некорректных математических операциях

Infinity – очень большое число, которое не может представить JS

Сохранение данных в новой переменной

```
let cats = 4;
let dogs = 6;

// складываем значения переменных
cats + dogs; // суммарное значение не сохранено

// для сохранения значения объявляем новую переменную
// и сохраняем в нее результат
let animals = cats + dogs;
```

Перезапись данных в той же переменной

```
let students = 33;

// пришел новый студент
students + 1; // 34
students; // все еще 33

// для сохранения измененного значения
// присваиваем результат той же переменной
students = students + 1;
students; // 34
```

Константы

```
// Объявим константы и попробуем изменить значения
const students = 30;
students = students + 1; // Assignment to constant variable

const G = 9.80665;
const PI = 3.141592;
G = 12; // Assignment to constant variable
PI = 8; // Assignment to constant variable
```

const работает также как let, только вы не можете изменить значение

Строки в JavaScript

В JavaScript любые текстовые данные являются строками

Строки можно создать с помощью одинарных, двойных или обратных кавычек.

Если использовать обратные кавычки ``, то в такую строку мы сможем вставлять произвольные выражения, обернув их в ${…}

Символы в строке имеют порядковые номера, мы можем получить символ из строки по его номеру

Длину строки (количество символов) можно получить с помощью свойства length

Строки в JavaScript имеют множество полезных методов для работы с текстом, такие как toUpperCase(), toLowerCase(), includes(), startsWith(), endsWith(), slice(), substring(), substr(), replace(), split(), trim(), и многие другие

Особенности применения кавычек в строках

```
let firstName = 'Иван';// строка в одинарных кавычках

let message = 'Однажды в студеную зимнюю пору...';

let animal = "Mountain goat";// строка в овойных кавычках

// открывающая и закрывающая кавычки должны быть одинаковыми
let error = "Это неправильно';
```

Шаблонные строки

Шаблонные строки это способ создания строк, который позволяет удобно комбинировать обычные символы и данные

```
let firstName = "Андрей";

let message = `Привет, ${firstName}`; // Привет, Андрей

let cats = 4,
  dogs = 6;

message = `Всего ${cats + dogs} животных`; // Всего 12 животных
```

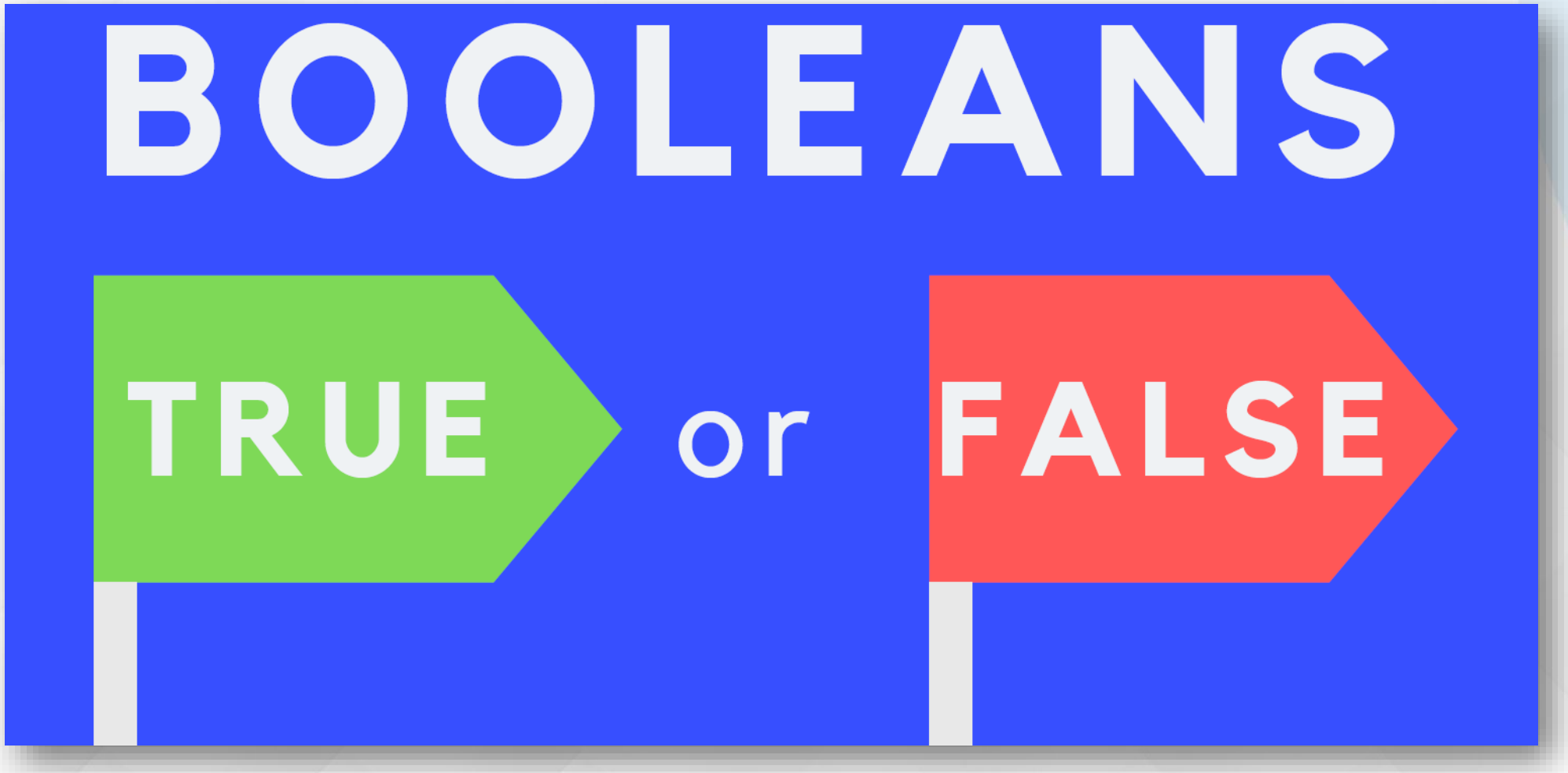
Шаблонные строки

```
let firstName = "Андрей";
let lastName = "Сидоров";
let age = 22;
let hobby = "Кататься на велосипеде";

let userInfo = `
  <div class="user-info">
    <h2>${firstName} ${lastName}</h2>
    <p>Возраст: ${age}</p>
    <p>Хобби: ${hobby}</p>
  </div>
`;
```

С помощью шаблонных строк очень удобно формировать HTML-разметку со вставкой данных из переменных

Булев тип данных



Применение булева типа данных

```
let truthyValue = true; // "Истина"
let falsyValue = false; // "Ложь

let isAdmin = true;
let isSunnyNow = false;

isSunnyNow = true;
```

Булев тип имеет всего два значения – это true и false. True – истина, false – ложь.

Undefined

```
let message; // значение переменной не задано
message; // undefined
message = 'Копирование завершено';

let age;
age; // undefined
age = 11;
```

Значение undefined имеют переменные, которые объявлены, но которым еще не присвоено значение.

Null

```
let message = null; // указываем, что значения нет
message; // null
message = "Копирование завершено";

let userLogin = null; // пользователь не авторизован
userLogin = "darkMan999"; // получили логин
```

В JavaScript, null является особенным значением, которое представляет собой отсутствие какого-либо объектного значения

Операторы сравнения

```
5 < 7; // меньше

3 > 9; // больше

2 <= 1; // меньше, либо равно

9 >= 1; // больше, либо равно
```

Как в математике, сравниваем 2 значения и всегда получаем простой ответ: да или нет (true / false)

Операторы сравнения

```
10 > 1; // true
0.2 > 0.3; // false
-10 < 0; // true
50.5 < 2; // false
99 >= 5; // true
54 <= 4; // false
"a" < "b"; // true
"A" > "a"; // false
```

Операторы сравнения возвращают значения булева типа!)

Хоть это и необычно, но вы можете сравнивать строки. Просто будьте осторожны, результат может быть неожиданный, особенно когда вы используете специальные символы.

Операторы сравнения

```
let age = 5; // ПРИСВАИВАНИЕ

3 == 5; // проверка на равенство
5 != 3; // проверка на неравенство

6 === 6; // строгое равенство
4 !== 5; // строгое неравенство
```

Часто путают знак присваивания и сравнения. Будьте бдительны

Строгое равенство и неравенство учитывают тип данных сравниваемых значений, а нестрогое нет.

Операторы сравнения

```
3 == 5; // false
5 == 5; // true
'4' == 4; // true
0 == ''; // true
true == false; // false
5 != 3; // true
4 != '4'; // false
null == undefined; // true
```

Нестрогое равенство приводит значения к одному типу, а затем сравнивает

Операторы сравнения

```
3 === 5; // false
5 === 5; // true
'4' === 4; // false
0 === ''; // false
true === false; // false
5 !== 3; // true
4 !== '4'; // true
null === undefined; // false
```

Операторы строгого равенства и неравенства сравнивают значения без преобразования типов.

Если типы данных разные, сразу false

Булева логика, операторы И(&&), ИЛИ(||), НЕ(!)

A И B

У меня есть водительские права И у меня есть автомобиль

B \ A	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

👇

Выражение истинно, когда оба значения истинны

A ИЛИ B

У меня есть еда ИЛИ деньги на её покупку

B \ A	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

👇

Выражение истинно, когда хотя бы одно значение истинно

НЕ A, НЕ B

👇

Изменяет значение на противоположное. Истина становится ложью, а ложь истиной.

Если значение A было Истинно, оно меняется на ложно, то же самое в обратную сторону.

Псевдо-истинные и псевдо-ложные значения

Все значения в логических операциях приводятся к булевому типу, TRUE либо FALSE.

Псевдо-ложные значения:

- false
- 0
- “” (пустая строка)
- null
- undefined
- NaN

Все остальные значения преобразуются в true.

Спасибо
за внимание!

Ваши вопросы...

Учебный центр «СПЕЦИАЛИСТ» –

Ваш путь к успеху

info@specialist.ru

+7 (495) 232-32-16