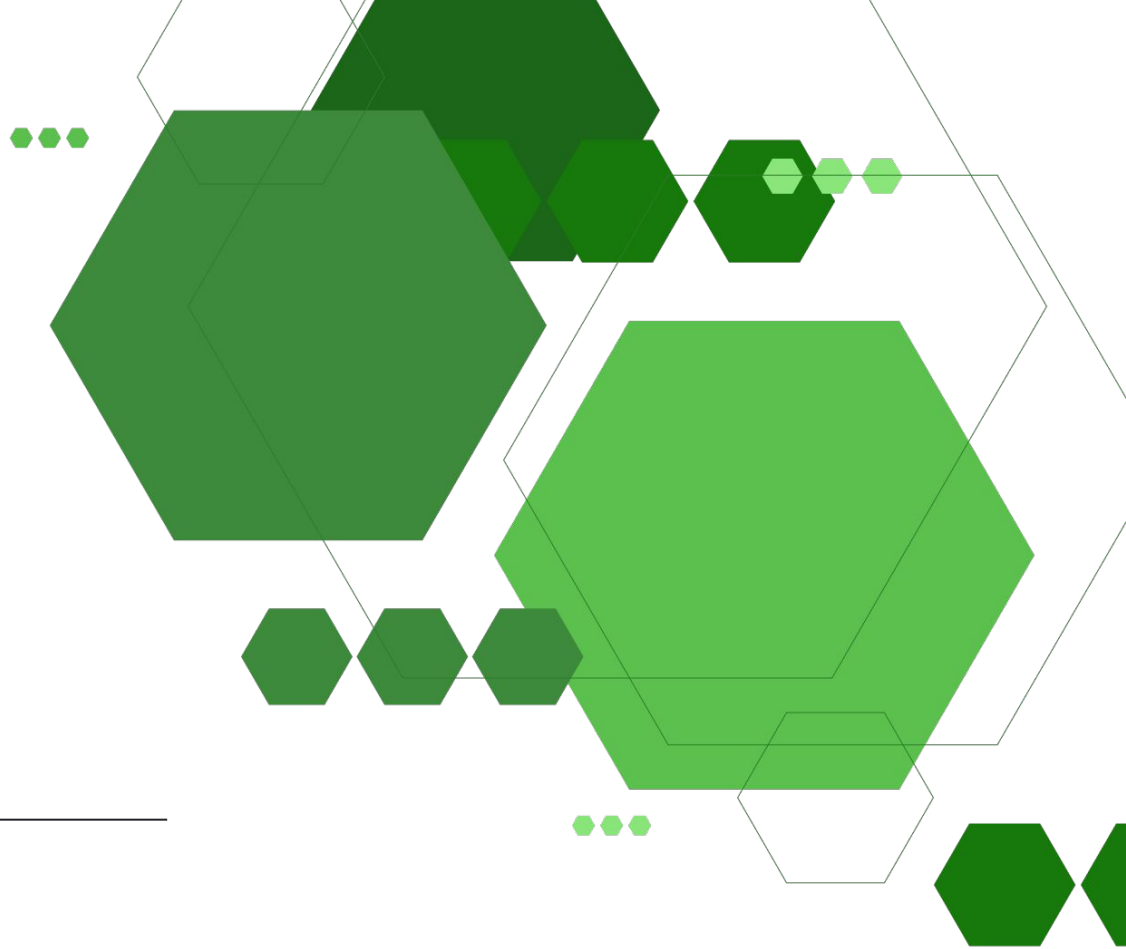
SkillFactory

Основы Javascript

Юлия Токаревская

Frontend-разработчик в Emplifi
Ментор разделов React, Архитектура приложений



Правила вебинара

- Длительность вебинара: 1 - 1.5 часа
- Возникающие вопросы можно задавать в чат в любой момент, или же голосом в специально отведенные промежутки времени
- Следите, чтобы во время объяснения материала ваш микрофон был выключен
- Запись вебинара появится на платформе через 2-3 рабочих дня

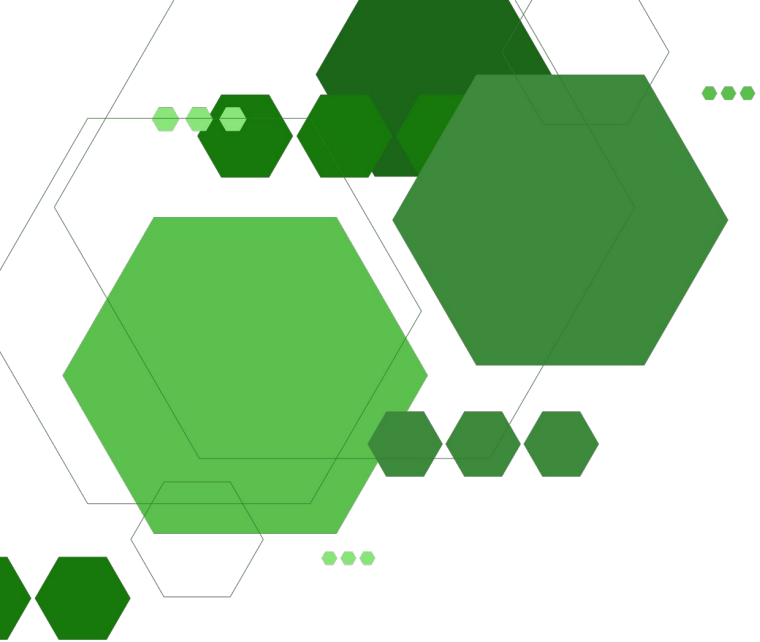
Спикер вебинара

Токаревская Юлия

Frontend-разработчик с опытом работы 7 лет
Стек: Javascript, React.js, Redux-Saga, Typescript

План вебинара:

1. Узнаем, для чего нужен язык Javascript и как подключить JS-скрипты к странице
2. Научимся работать с консолью в браузере
3. Узнаем, что такое переменные
4. Разберемся в типах данных и научимся преобразовывать типы данных
5. Выучим основные логические операторы
6. Научимся использовать условный оператор if и тернарный оператор



Введение в Javascript

Javascript и Ecmascript

Javascript - язык программирования, который позволяет создавать интерактивные и динамические веб-страницы

Программы на этом языке называются **скриптами**. Они встраиваются в HTML и выполняются браузером при загрузке страницы.

Javascript основан на стандарте **EcmaScript**. Актуальная версия - **ES6 (ES2015)**

Подключение скрипта к странице

Для подключения JS-скрипта к странице используется тег **<script>**

JS-код можно написать непосредственно внутри тега.

Этот способ подходит только для скриптов небольшого размера. Много строчек кода внутри HTML делает код нечитабельным и усложняет работу с ним.

```
<script>
    console.log(“This is JavaScript code”);
</script>
```

Подключение скрипта к странице

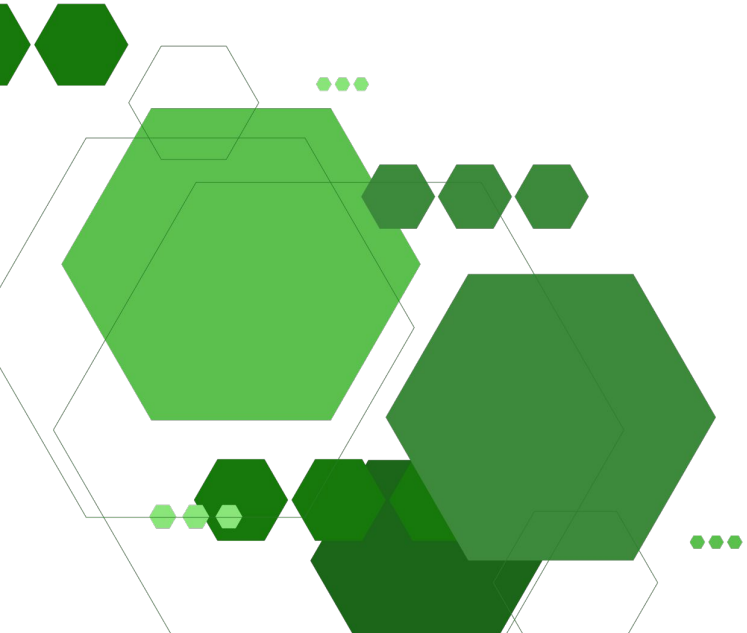
Тег <script> также может ссылаться на **отдельный файл с JS-скриптом**, тогда к тегу добавляется **атрибут src**, в котором указывается ссылка на файл, который нужно подключить.
Содержимое тега в таком случае оставляют пустым.

```
<script src="script.js"></script>
```

Этот способ - наиболее распространенный, потому что:

- Внешние файлы кэшируются браузером, таким образом, ускоряется загрузка страниц
- Меньше путаницы в коде

Работа с консолью

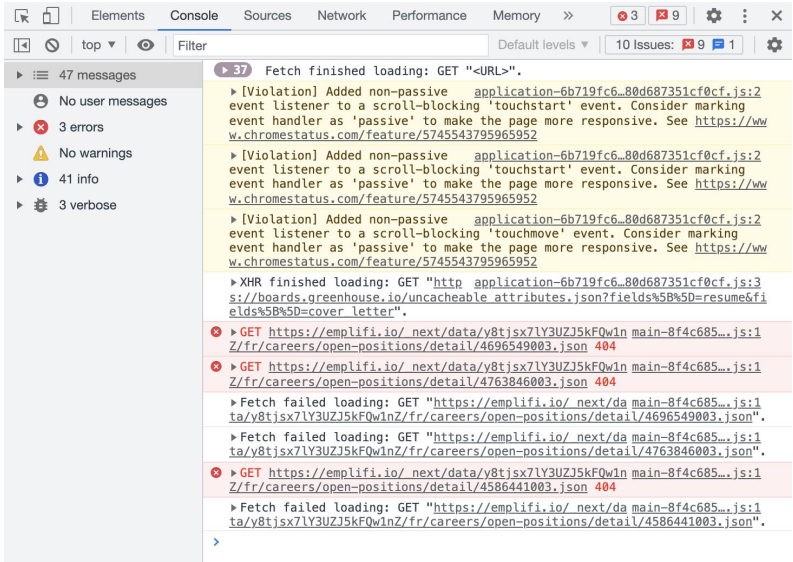


Чем полезна консоль браузера?

Консоль в панели разработчика - полезный инструмент для тестирования скриптов и поиска ошибок

В консоли отображаются ошибки, предупреждения и различная служебная информация

Открыть её можно, нажав **F12** (Windows) или **Cmd+Opt+J** (Mac)



Вывод информации в консоль

Вывести информацию в консоль можно с помощью команды console.log()

```
1  console.log(“Hello world”);
```

Можно также вывести сообщение об ошибке или предупреждение:

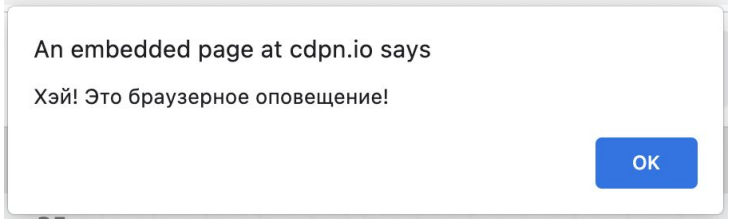
```
1  console.warn(“Это последнее предупреждение”);
2  console.error(“Вы совершили ошибку”);
```

Примечание

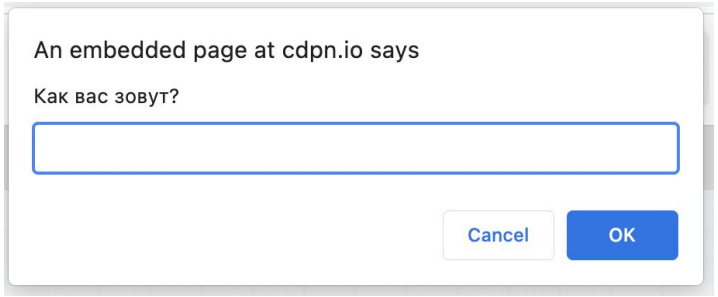
Информация в консоли предназначена в первую очередь для разработчиков. Не пытайтесь выводить в консоль информацию, предназначенную для пользователей. Скорее всего, они её не увидят :)

Другие полезные браузерные команды

alert - показывает пользователю модальное окно с каким-либо сообщением и кнопкой OK. Интерфейс остальной страницы будет заблокирован, пока пользователь не нажмет на OK.

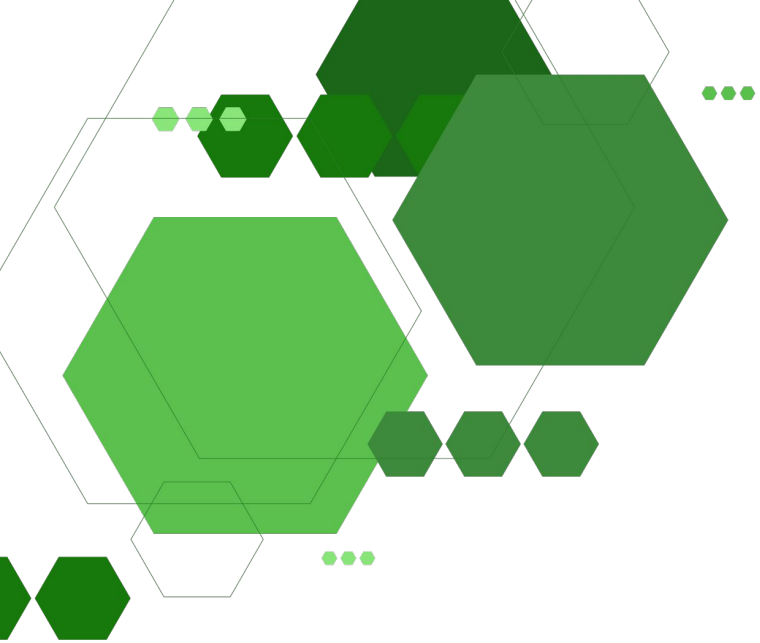


prompt - выводит модальное окно с заголовком, полем для ввода и кнопками OK и Отмена.



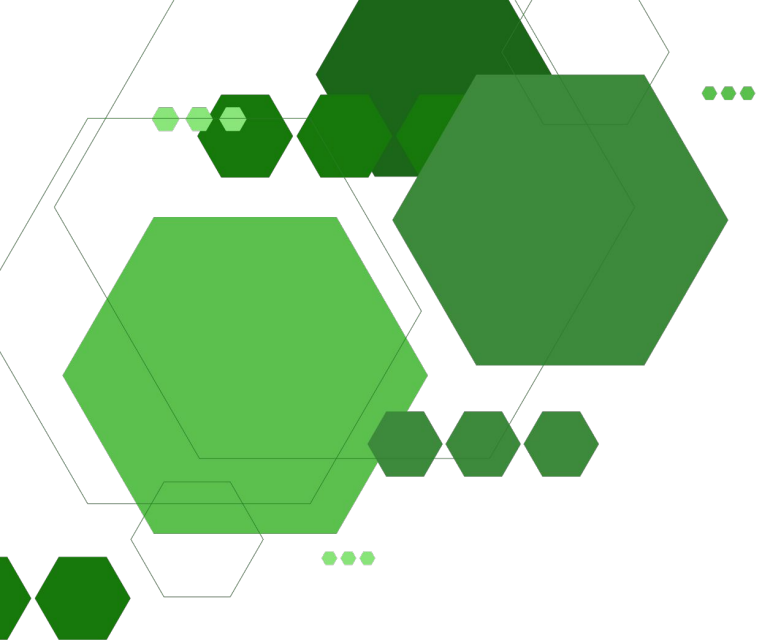
Примечание

В современном вебе эти методы взаимодействия с пользователем являются устаревшими и используются в основном в целях тестирования или отладки.
Рекомендуется не использовать их в реальных проектах.



Время для ваших вопросов

🤔❓



Переменные

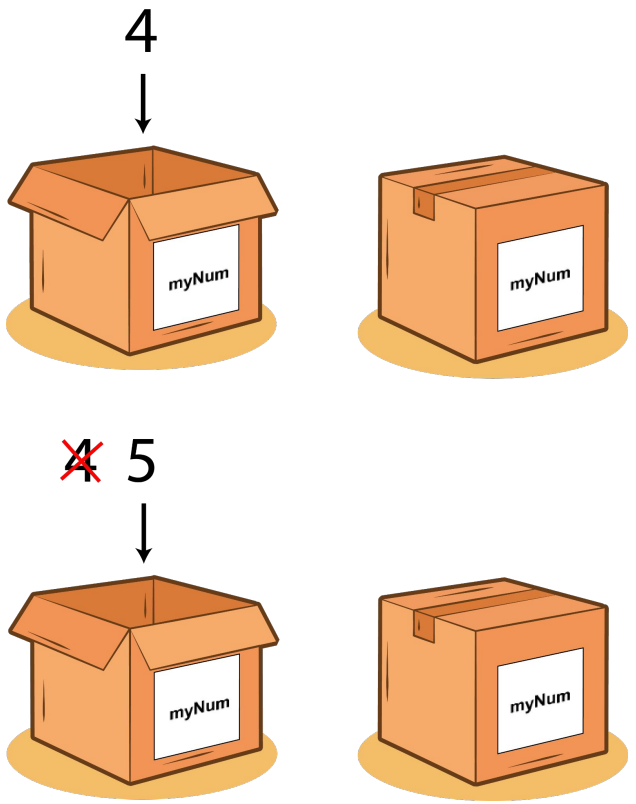
Что такое переменная?

Переменная - это именованная область памяти, в которой хранятся данные

Обычно работа с переменными в коде выглядит следующим образом:

1. Объявили (создали) переменную
2. Записали в переменную значение
3. Использовали переменную в коде

Значение переменной можно также **перезаписать**



Объявление переменной

В Javascript есть 3 ключевых слова для объявления переменных:

var (устаревший синтаксис)

```
var price;
```

let - объявление локальной переменной

```
let price;
```

const - объявление константы. Значение константы должно быть задано при объявлении и не может быть изменено

```
const PRICE = 30;
```

Имена переменных

- Имя переменной должно содержать только буквы, цифры или символы $ и _.
- Первый символ не должен быть цифрой.
- Имя переменной не должно совпадать с зарезервированным ключевым словом (class, var, for, is, function и т.д.)

Рекомендации по выбору имени для переменной:

Имя должно быть **осмысленным**, т.е. отображать суть значения, которое содержится в переменной

Имена переменных записываются в **нотации camel case**: isUserLogged (кроме констант)

Упражнение

Какие из следующих значений могут быть использованы в качестве имен переменных?

userName → **Да**

$folder → **Да**

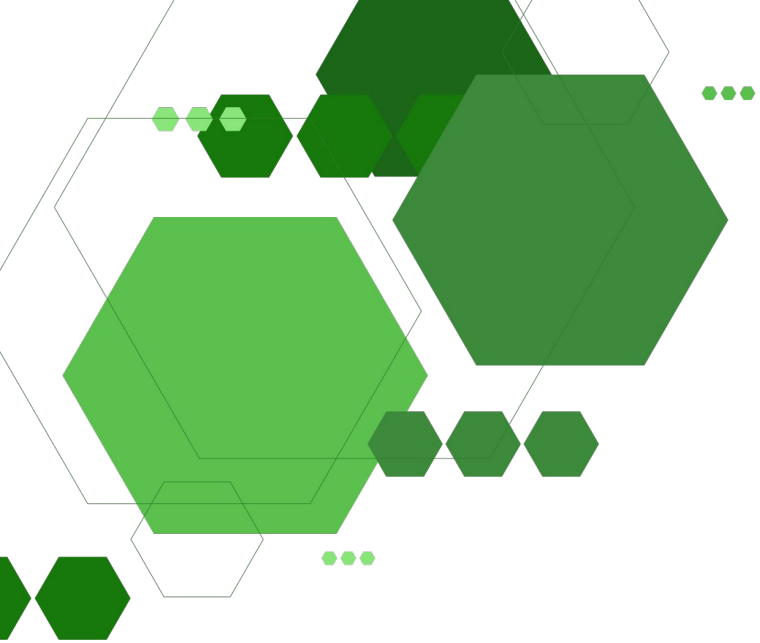
const → **Нет, const это зарезервированное слово**

LikesCount → **Нежелательно, имена переменных должны использовать нотацию camelCase**

3user → **Нет, первый символ имени не может быть цифрой**

_mainEntity → **Да**

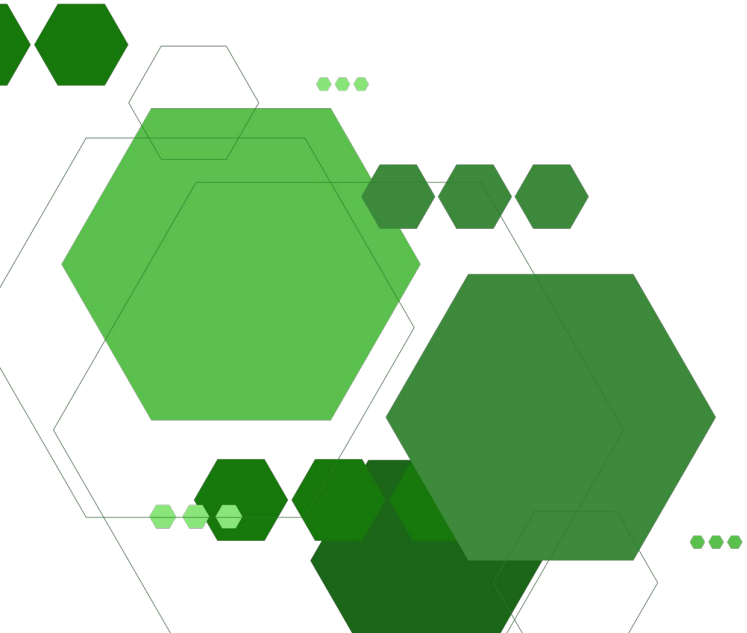
abc → **Нежелательно, это имя не отражает сути значения, хранящегося в переменной**



Время для ваших вопросов

🤔❓

Типы данных. Преобразование типов данных



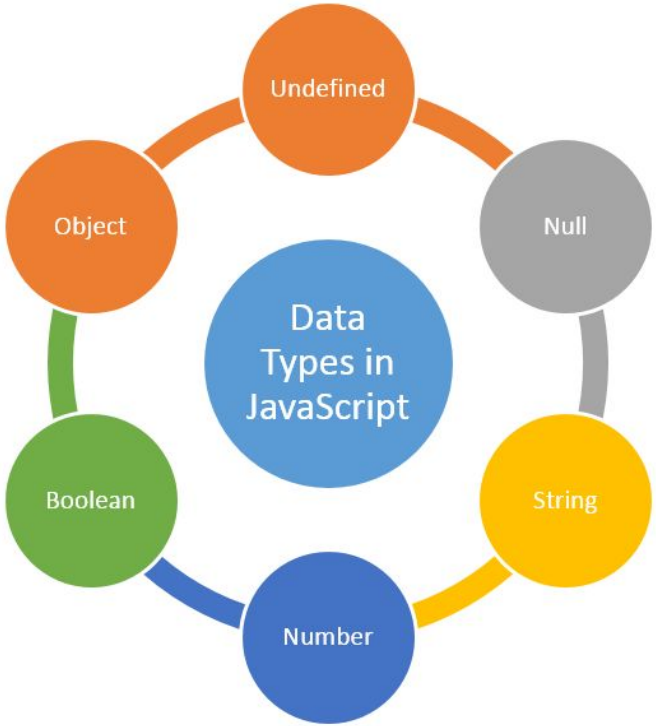
Типы данных

В Javascript существует **8 типов данных**:

- null
- undefined
- boolean (true, false)
- number (15, 129, 14.34)
- string (“Javascript”, ‘HTML’, `Hello`)
- symbol
- object
- bigInt

Для определения типа данных используется оператор **typeof**

```
let name = "Helen";
console.log(typeof name); // "string"
```

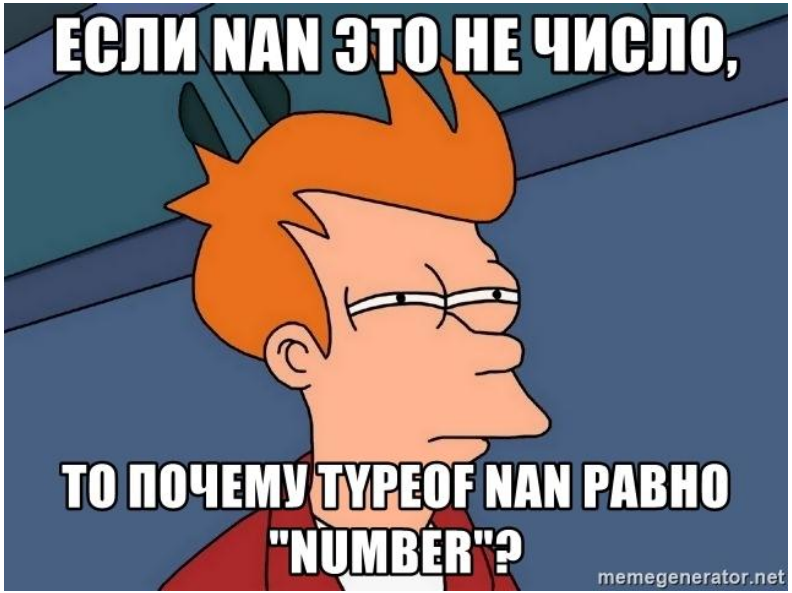


Специальные числовые значения

NaN (Not a Number) - “не число”, как правило возникает вследствие ошибочных операций с числами, например:
1 * undefined
5 - "abc"

Определить значение NaN можно только с помощью функции isNaN

Infinity - значение, обозначающее бесконечность. Может получиться в результате деления числа на ноль или если результат вычислений выходит за допустимый диапазон чисел Javascript (2^{53})



Преобразование типов данных

Существует несколько способов преобразовать типы данных:

1. Boolean(), Number(), String()
2. parseInt(), parseFloat() - преобразует строки в числа
3. toString() - преобразует объекты в строки
4. Унарный плюс
   +”123” => 123
   +”aa123” => NaN

Упражнение

Что получится в результате выполнения данных выражений?

5 + “5” → **“55”**

10 - undefined → **NaN**

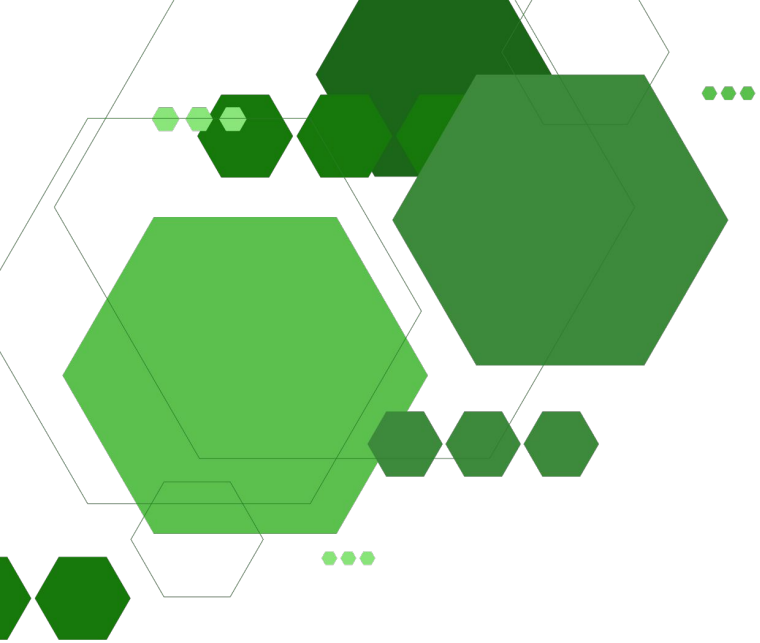
3 * null → **0**

25 - “15” → **10**

10 + +”20” → **30**

2 + “one” → **“2one”**

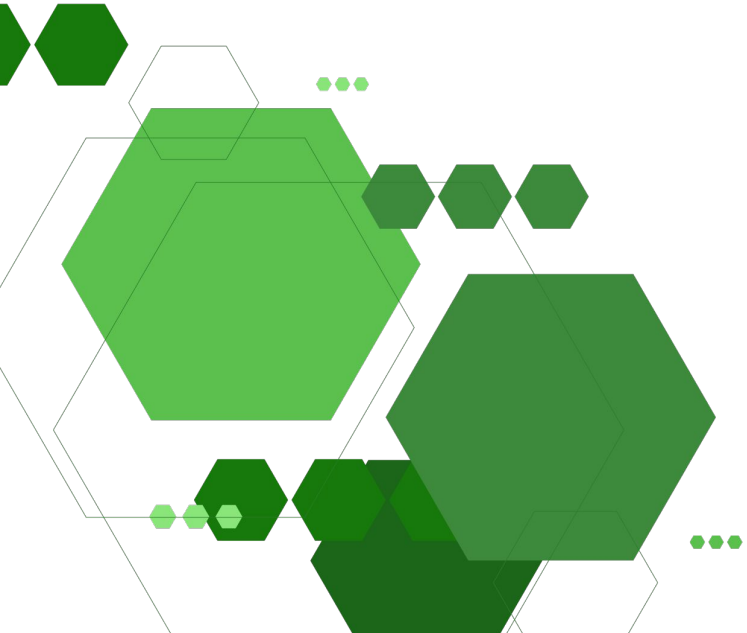
2 - “one” → **NaN**



Время для ваших вопросов

🤔❓

Логические операторы



Операторы сравнения

Операторы сравнения предназначены для сравнивания двух значений. Возвращают значение логического типа (true/false).

Оператор	Название	Пример
>	Больше	3 > 1 -> true 3 > 5 -> false
<	Меньше	6 < 9 -> true 6 > 15 -> false
>=	Больше или равно	3 >= 3 -> true 3 >= 10 -> false
<=	Меньше или равно	4 <= 4 -> true 4 <= 1 -> false

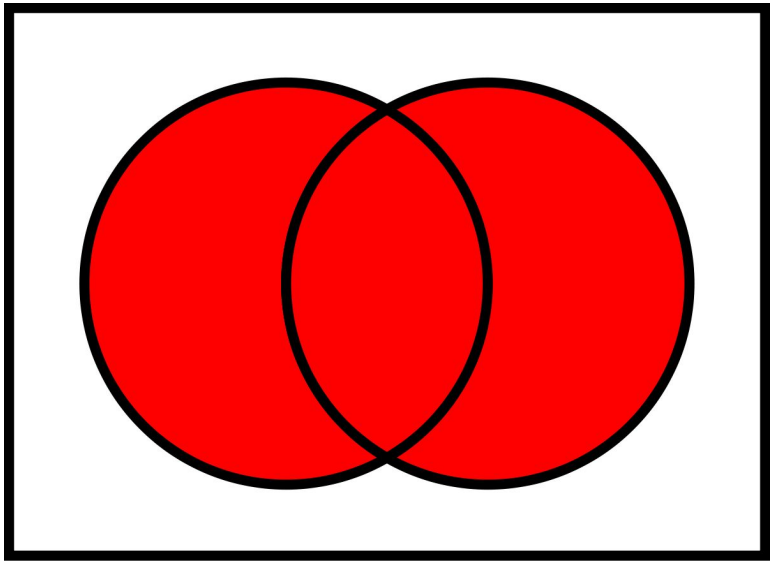
Операторы сравнения

Оператор	Название	Пример
==	Равенство	10 == 10 -> true 10 == “10” -> true 4 == 6 -> false
===	Строгое равенство	10 === “10” -> false 3 === 3 -> true
!=	Неравенство	4 != 5 -> true 6 != 6 -> false 6 != “6” -> false
!==	Строгое неравенство	3 !== 8 -> true 3 !== 3 -> false 3 !== “3” -> true

Логическое ИЛИ

Логическое ИЛИ (||) вернёт true, если хотя бы одно из значений в логическом выражении равно true

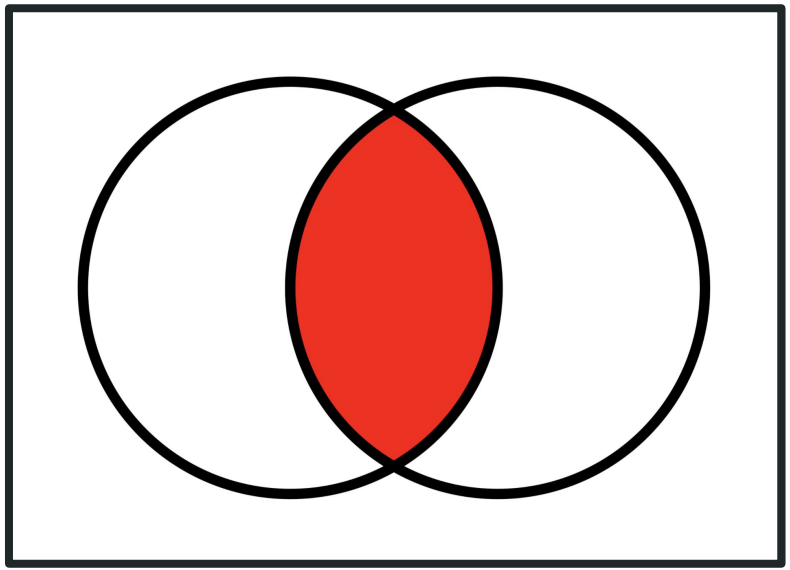
true || true -> true
false || true -> true
true || false -> true
false || false -> false



Логическое И

Логическое И (&&) вернет true, только если все значения в логическом выражении равны true

true && true -> true
false && true -> false
true && false -> false
false && false -> false



Логическое НЕ

Логическое НЕ (!) применяется только с одним аргументом и возвращает противоположное ему значение

!true -> false
!false -> true

Особенности логических операторов

1. Приоритет оператора && больше, чем ||, поэтому он выполняется раньше. Чтобы изменить порядок выполнения операторов, можно использовать скобки:

```
a && (b || c) && d
```

2. Логические операторы можно использовать с разными типами данных (не только true/false). В таком случае значение будет сначала преобразовано в логический тип

```
0 || 1 // true
“string” && null // false
```

Практика

Что получится в результате выполнения данных логических выражений?

true || false → **true**

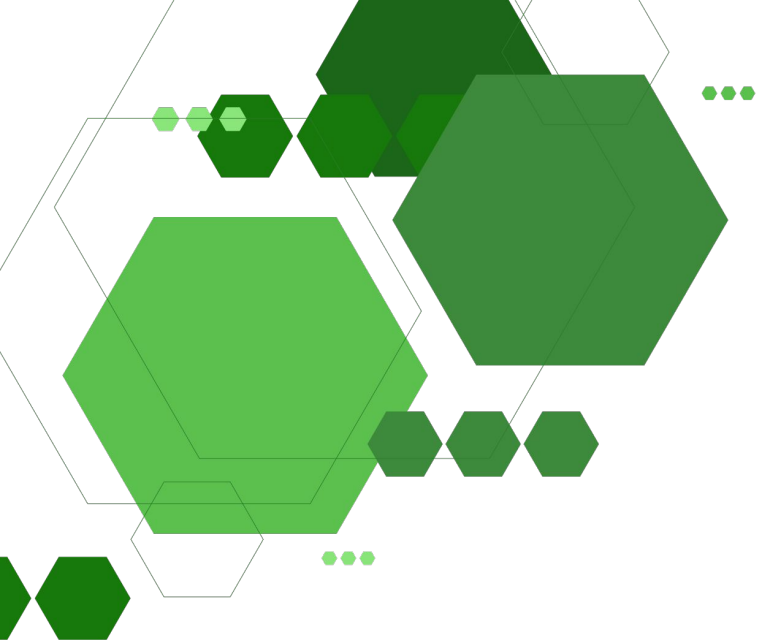
true && false → **false**

true || false && false → **true**

false && true && !false || true → **true**

true && (false || false) && true → **false**

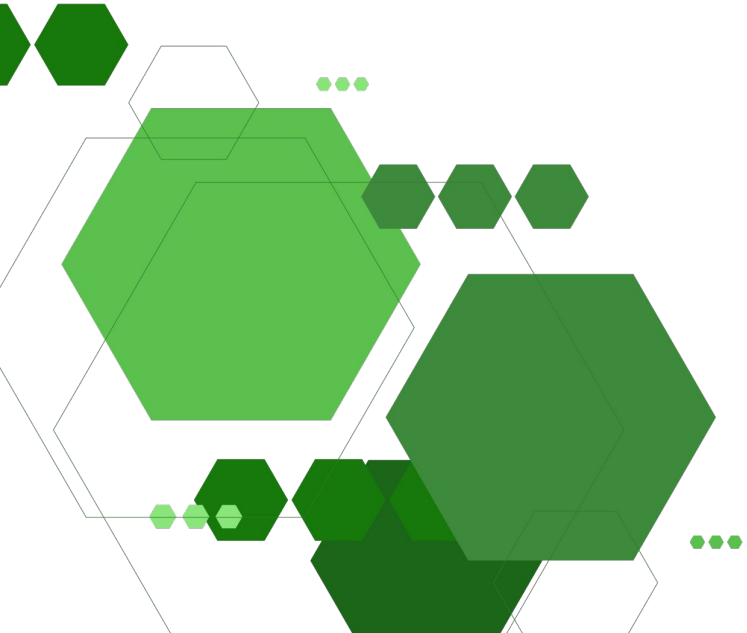
!(false || true || false) && false → **false**



Время для ваших вопросов

🤔❓

Условный оператор if

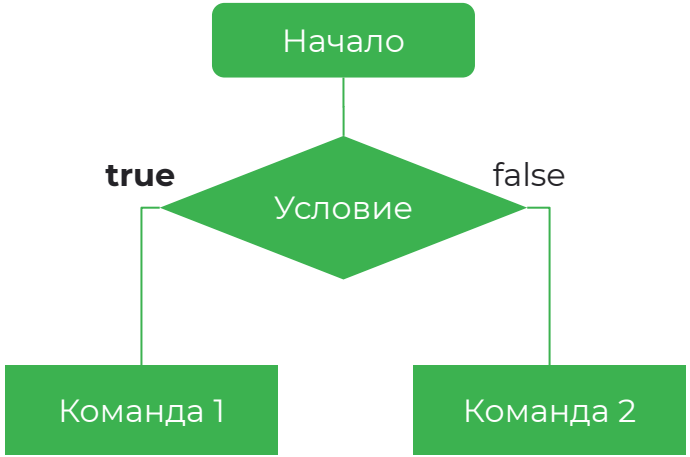


Основной синтаксис

Условный оператор if позволяет выполнить определенные инструкции в зависимости от некоторого условия

Блок команд в фигурных скобках выполнится, только если условие (выражение в круглых скобках) **равно true**

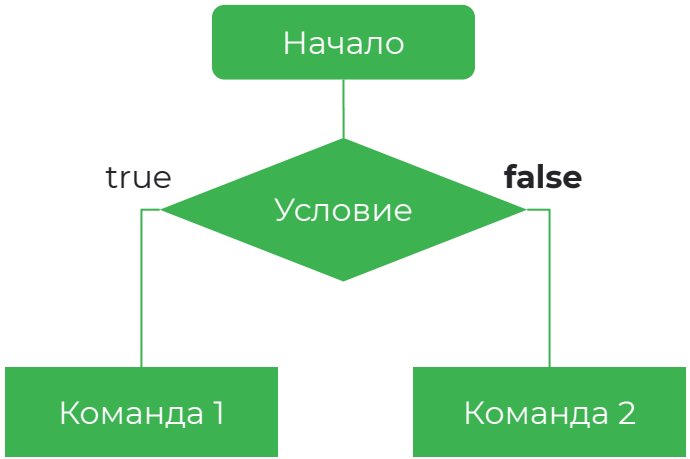
```
if (number > 3) {
    console.log("Введенное число больше 3");
}
```



Блок else

К логической конструкции также можно добавить **блок else**, команды в котором выполнятся в случае, если условие ложно, т.е. **равно false**

```
if (number > 3) {
    console.log("Введенное число больше 3");
} else {
    console.log("Введенное число меньше 3");
}
```



Блок else if

Блок **else if** позволяет проверить несколько вариантов условия

```
if (number > 3) {
    console.log(“Введенное число больше 3”);
} else if (number < 3) {
    console.log(“Введенное число меньше 3”);
} else {
    console.log(“Введенное число равно 3”);
}
```

Примечания по условной конструкции

1. Условные операторы могут вкладываться друг в друга:

```
if (typeof number === "number") {
  if (number > 0) {
    console.log("Число больше 0");
  } else {
    console.log("Число меньше 0");
  }
}
```

2. Блоки else и else if не являются обязательными

Тернарный оператор

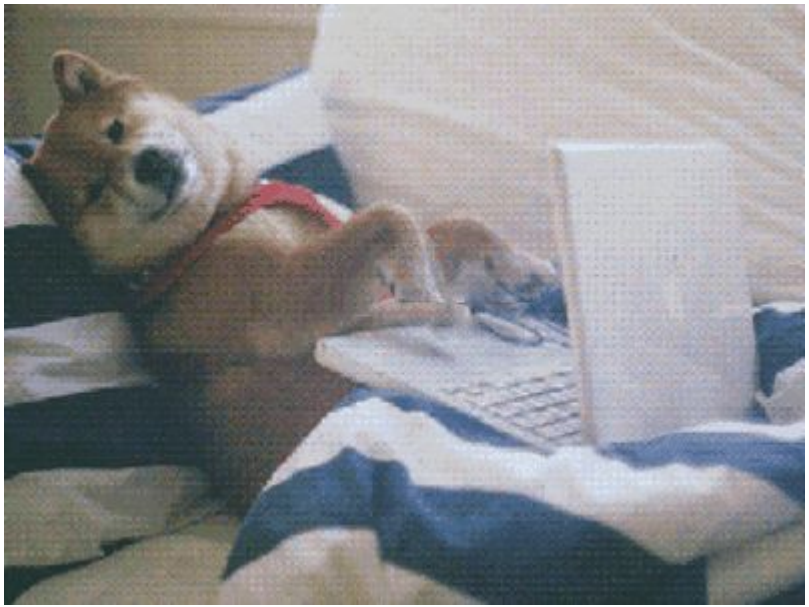
Тернарный оператор представляет собой сокращенную запись условного оператора if и имеет следующий синтаксис:

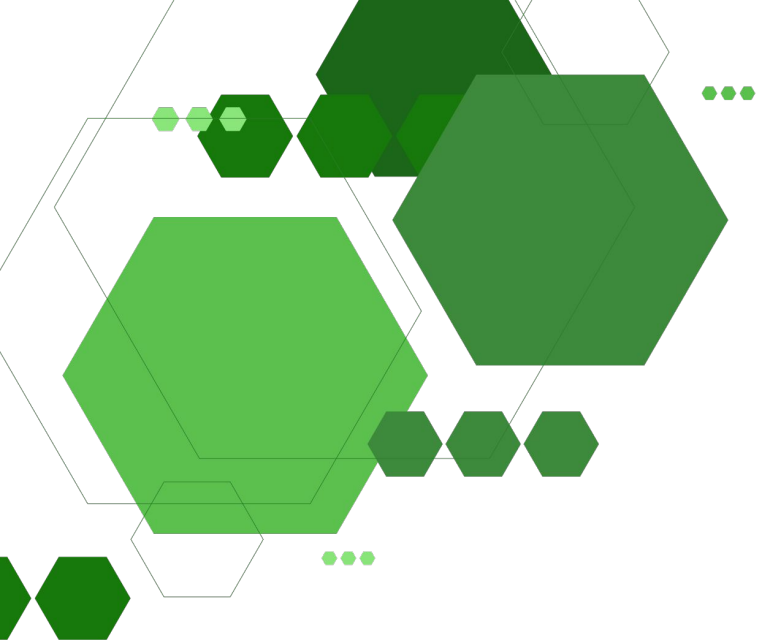
условие ? значение1 : значение2

Пример использования:

```
let isNumberNegative = num < 0 ? true : false;
let genderValue = gender === "f" ? "Женский" : "Мужской";
let userName = name ? name : "Аноним";
```

Время попрактиковаться :)



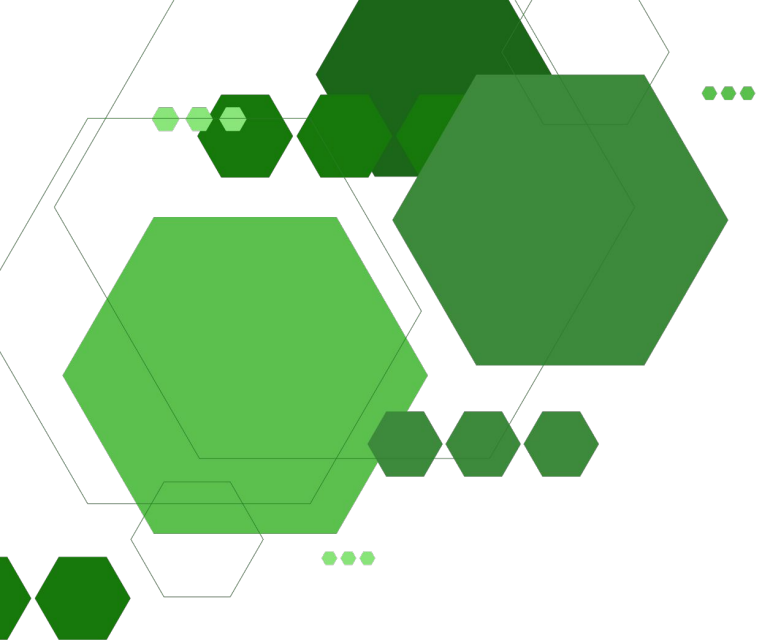


Время для ваших вопросов

🤔❓

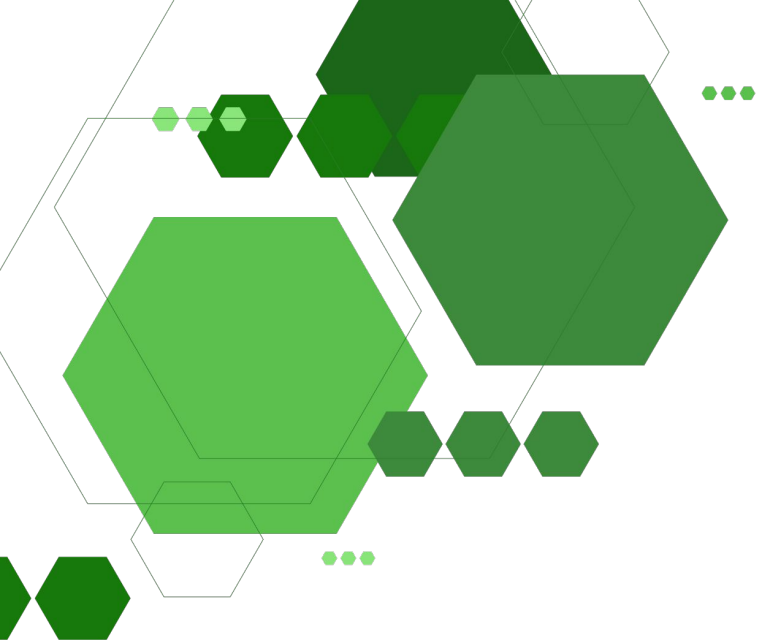
Полезные ссылки:

1. Документация Javascript (MDN)
2. Онлайн-учебник по Javascript
3. Список зарезервированных слов в Javascript
4. Книга Этан Браун “Изучаем JavaScript. Руководство по созданию современных веб-сайтов”
5. Книга Дэвид Флэнаган “JavaScript. Подробное руководство”
6. Гарвардский курс по основам программирования (CS50)



Напоминалка: пожалуйста, поделитесь вашими впечатлениями о вебинаре в форме обратной связи. Так вы поможете мне улучшить качество вебинаров 😊

Ссылку на форму можно найти в анонсе вебинара в Slack



Спасибо за внимание!
Успехов ❤️