**Введение: шаблоны и флаги**

Регулярные выражения – мощное средство поиска и замены в строке.

В JavaScript регулярные выражения реализованы отдельным объектом **RegExp** и интегрированы в методы строк.

**Регулярные выражения**

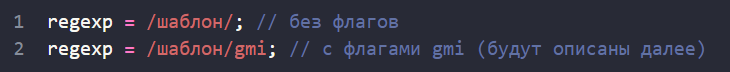
Регулярное выражение (оно же «регэксп», «регулярка» или просто «рег»), состоит из шаблона (также говорят «**паттерн**») и необязательных флагов.

Существует два синтаксиса для создания регулярного выражения.

«Длинный» синтаксис:



…И короткий синтаксис, использующий слеши "/":

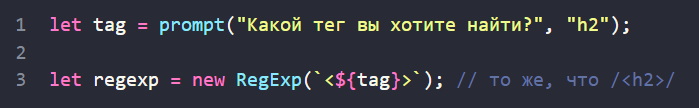


Слеши **/.../** говорят JavaScript о том, что это регулярное выражение. Они играют здесь ту же роль, что и кавычки для обозначения строк.

Регулярное выражение regexp в обоих случаях является объектом встроенного класса RegExp.

Основная разница между этими двумя способами создания заключается в том, что слеши /.../ не допускают никаких вставок переменных (наподобие возможных в строках через ${...}). Они полностью статичны.

Слеши используются, когда мы на момент написания кода точно знаем, каким будет регулярное выражение – и это большинство ситуаций. А new RegExp – когда мы хотим создать регулярное выражение «на лету» из динамически сгенерированной строки, например:



**Флаги**

Регулярные выражения могут иметь флаги, которые влияют на поиск.

В JavaScript их всего шесть:

* **i** – с этим флагом поиск не зависит от регистра: нет разницы между A и a
* **g** – с этим флагом поиск ищет все совпадения, без него – только первое.
* **m** – многострочный режим
* **s** – включает режим «dotall», при котором точка . может соответствовать символу перевода строки \n
* **u** – включает полную поддержку Юникода
* **y** – режим поиска на конкретной позиции в тексте

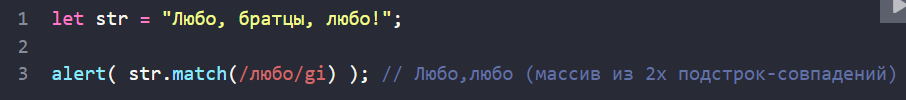
**Поиск: str.match**

Как уже говорилось, использование регулярных выражений интегрировано в методы строк.

Метод str.match(regexp) для строки str возвращает совпадения с регулярным выражением regexp.

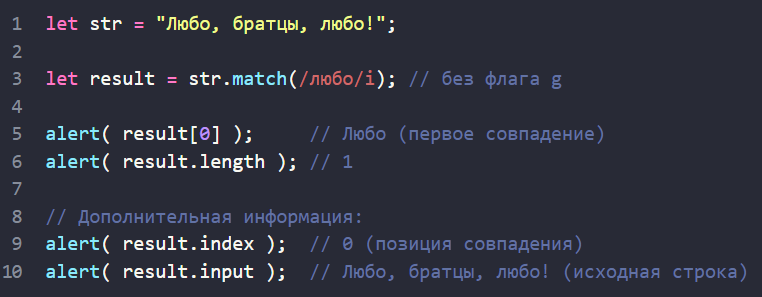
У него есть три режима работы:

Если у регулярного выражения есть флаг g, то он возвращает массив всех совпадений:



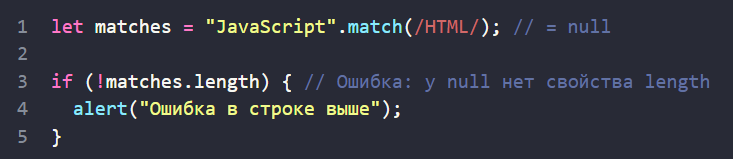
Обратите внимание: найдены и Любо и любо, благодаря флагу i, который делает регулярное выражение регистронезависимым.

Если такого флага нет, то возвращает только первое совпадение в виде массива, в котором по индексу 0 находится совпадение, и есть свойства с дополнительной информацией о нём:



И, наконец, если совпадений нет, то, вне зависимости от наличия флага g, возвращается **null**.

Это очень важный нюанс. При отсутствии совпадений возвращается не пустой массив, а именно **null**. Если об этом забыть, можно легко допустить ошибку, например:



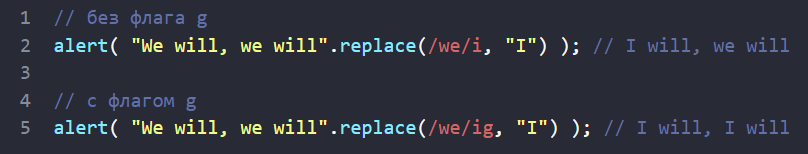
Если хочется, чтобы результатом всегда был массив, можно написать так:



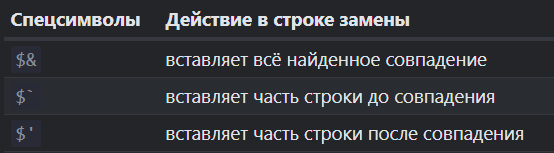
**Замена: str.replace**

Метод **str.replace(regexp, replacement)** заменяет совпадения с regexp в строке str на replacement (все, если есть флаг g, иначе только первое).

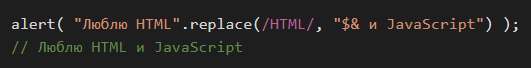
Например:



В строке замены replacement мы можем использовать специальные комбинации символов для вставки фрагментов совпадения:

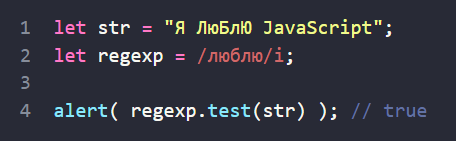


Пример с $&:



**Проверка: regexp.test**

Метод **regexp.test(str)** проверяет, есть ли хоть одно совпадение, если да, то возвращает true, иначе false.



**Символьные классы**

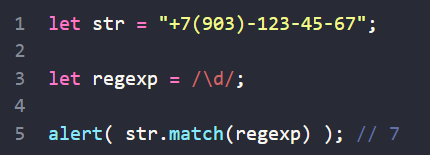
Рассмотрим практическую задачу – у нас есть номер телефона вида "+7(903)-123-45-67", и нам нужно превратить его в строку только из чисел: 79031234567.

Для этого мы можем найти и удалить все, что не является числом. С этим нам помогут символьные классы.

**Символьный класс** – это специальное обозначение, которое соответствует любому символу из определённого набора.

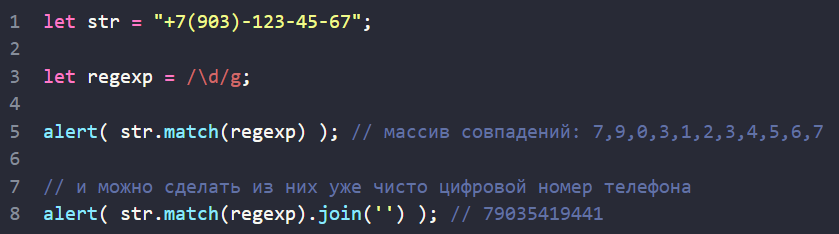
Для начала давайте рассмотрим класс «цифра». Он обозначается как **\d** и в регулярном выражении соответствует «любой одной цифре».

Например, давайте найдём первую цифру в номере телефона:



Без флага g регулярное выражение ищет только первое совпадение, то есть первую цифру \d.

Давайте добавим флаг g, чтобы найти все цифры:



Это был символьный класс для цифр. Есть и другие символьные классы.

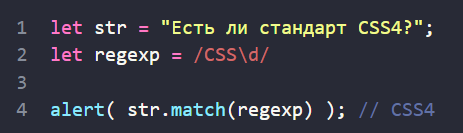
Наиболее используемые:

* **\d** («d» от английского «digit» означает «цифра») Цифра: символ от 0 до 9.
* **\s** («s»: от английского «space» – «пробел») Пробельные символы: включает в себя символ пробела, табуляции \t, перевода строки \n и некоторые другие редкие пробельные символы, обозначаемые как \v, \f и \r.
* **\w** («w»: от английского «word» – «слово») Символ «слова», а точнее – буква латинского алфавита или цифра, или подчёркивание \_. Нелатинские буквы не являются частью класса \w, то есть буква русского алфавита не подходит.

Для примера, **\d\s\w** обозначает «цифру», за которой идёт пробельный символ, а затем символ слова, например, 1 a.

Регулярное выражение может содержать как обычные символы, так и символьные классы.

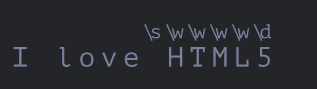
Например, CSS\d соответствует строке CSS с цифрой после неё:



Также мы можем использовать несколько символьных классов:



Соответствие (каждому символьному классу соответствует один символ результата):



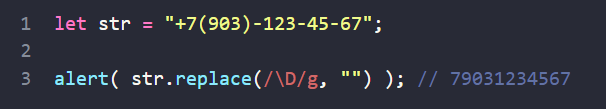
**Обратные символьные классы**

Для каждого символьного класса существует «обратный класс», обозначаемый той же буквой, но в верхнем регистре.

«Обратный» означает, что он соответствует всем другим символам, например:

* **\D** – Не цифра: любой символ, кроме \d, например буква.
* **\S** – Не пробел: любой символ, кроме \s, например буква.
* **\W** – Любой символ, кроме \w, то есть не буквы из латиницы, не знак подчёркивания и не цифра. В частности, русские буквы принадлежат этому классу.

Мы уже видели, как сделать чисто цифровой номер из строки вида +7(903)-123-45-67: найти все цифры и соединить их. Альтернативный, более короткий путь – найти нецифровые символы **\D** и удалить их из строки:



**Точка – это любой символ**

Точка . – это специальный символьный класс, который соответствует «любому символу, кроме новой строки».



Обратите внимание, что точка означает «любой символ», но не «отсутствие символа». Там должен быть какой-либо символ, чтобы соответствовать условию поиска:



**Точка как буквально любой символ, с флагом «s»**

Обычно точка не соответствует символу новой строки \n.

То есть, регулярное выражение A.B будет искать символ A и затем B, с любым символом между ними, кроме перевода строки **\n**:



Но во многих ситуациях точкой мы хотим обозначить действительно «любой символ», включая перевод строки.

Как раз для этого нужен флаг s. Если регулярное выражение имеет его, то точка . соответствует буквально любому символу:



Итого, существуют следующие символьные классы:

\d – цифры.

\D – не цифры.

\s – пробельные символы, табы, новые строки.

\S – все, кроме \s.

\w – латиница, цифры, подчёркивание '\_'.

\W – все, кроме \w.

. – любой символ, если с флагом регулярного выражения s, в противном случае любой символ, кроме перевода строки \n.

Это лишь те, что мы будем использовать, осталньое на <https://learn.javascript.ru/regular-expressions>

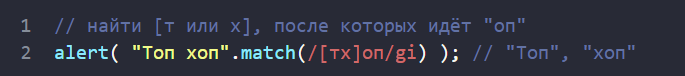
**Наборы и диапазоны [...]**

Несколько символов или символьных классов в квадратных скобках […] означают «искать любой символ из заданных».

**Наборы**

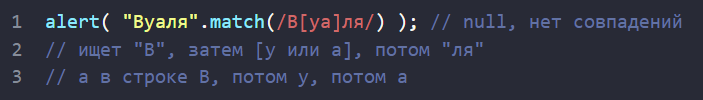
Для примера, [eao] означает любой из 3-х символов: 'a', 'e' или 'o'. Это называется набором.

Наборы могут использоваться в регулярных выражениях вместе с обычными символами, например:



Обратите внимание, что в наборе несколько символов, но в результате он соответствует ровно одному символу.

Так что этот пример не даёт совпадений:



Шаблон ищет:

1. В,
2. затем один из символов [уа],
3. потом ля.

В этом случае совпадениями могут быть Вуля или Валя.

**Диапазоны**

Ещё квадратные скобки могут содержать диапазоны символов.

К примеру, [a-z] соответствует символу в диапазоне от a до z, или [0-5] – цифра от 0 до 5.

В приведённом ниже примере мы ищем "x", за которым следуют две цифры или буквы от A до F:



Здесь в [0-9A-F] сразу два диапазона: ищется символ, который либо цифра от 0 до 9, либо буква от A до F.

Если мы хотим найти буквы и в верхнем и в нижнем регистре, то мы можем добавить ещё диапазон a-f: [0-9A-Fa-f]. Или поставить у регулярного выражения флаг i.

Также мы можем использовать символьные классы внутри […].

Например, если мы хотим найти «символ слова» \w или дефис -, то набор будет: [\w-].

Можем использовать и несколько классов вместе, например, [\s\d] означает «пробельный символ или цифра».

**Исключающие диапазоны**

Помимо обычных диапазонов, есть «исключающие» диапазоны, которые выглядят как [^…].

Они обозначаются символом каретки ^ в начале диапазона и соответствуют любому символу за исключением заданных.

Например:

* [^aeyo] – любой символ, за исключением 'a', 'e', 'y' или 'o'.
* [^0-9] – любой символ, за исключением цифры, то же, что и \D.
* [^\s] – любой непробельный символ, то же, что и \S.

Пример ниже ищет любые символы, кроме латинских букв, цифр и пробелов:



**Квантификаторы +, \*, ? и {n}**

Давайте возьмём строку вида +7(903)-123-45-67 и найдём все числа в ней. Но теперь нас интересуют не цифры по отдельности, а именно числа: 7, 903, 123, 45, 67.

Число — это последовательность из 1 или более цифр \d. Чтобы указать количество повторений, нам нужно добавить квантификатор.

Самый простой квантификатор — это число в фигурных скобках: **{n}**.

Он добавляется к символу (или символьному классу, или набору [...] и т.д.) и указывает, сколько их нам нужно. Можно по-разному указать количество, например:

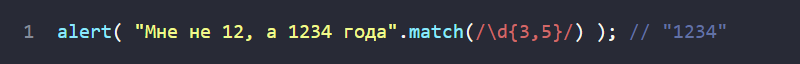
**Точное количество: {5}**

Шаблон \d{5} обозначает ровно 5 цифр, он эквивалентен \d\d\d\d\d. Следующий пример находит пятизначное число:



**Диапазон: {3,5}, от 3 до 5**

Для того, чтобы найти числа от 3 до 5 цифр, мы можем указать границы в фигурных скобках: \d{3,5}



Верхнюю границу можно не указывать.

Тогда шаблон \d{3,} найдёт последовательность чисел длиной 3 и более цифр

**Короткие обозначения**

Для самых востребованных квантификаторов есть сокращённые формы записи:

+ - означает «один или более». То же самое, что и {1,}.

Например, \d+ находит числа (из одной или более цифр):

? - означает «ноль или один». То же самое, что и {0,1}. По сути, делает символ необязательным.

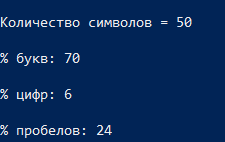
Например, шаблон ou?r найдёт o после которого, возможно, следует u, а затем r.

Поэтому шаблон colou?r найдёт два варианта: color и colour:

В этой практическом мы рассмотрели лишь примеры того, что будем использовать, более подробную информацию о регулярных выражениях можно найти по адресу <https://learn.javascript.ru/regular-expressions>.

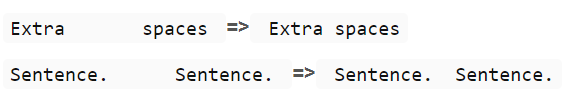
**Задачи:**

1. Проверьте начинается ли строка с заглавной буквы.
2. Найти количество гласных в строке.
3. В строке, состоящей из нескольких чисел (прим. «1, 11, 111, 1111») найти лишь те у которых количество цифр больше 3.
4. Подсчитать количество пробелов, цифр, латинских букв в введенной пользователем строке, отобразить статистику вида:



**Задачи с доп. оценкой:**

1. Убрать повторяющиеся пробелы и знаки табуляции, оставить по одному пробелу между словами и по два между предложениями.



1. Написать регулярное выражение, проверяющее является ли строка адресом эл. почты.