|  |  |
| --- | --- |
| **Символ** | **Значение** |
| \ | Для обычных символов - делает их специальными. Например, выражение /s/ ищет просто символ 's'. А если поставить \ перед s, то /\s/ уже обозначает пробельный символ.И наоборот, если символ специальный, например \*, то \ сделает его просто обычным символом "звездочка". Например, /a\*/ ищет 0 или больше подряд идущих символов 'a'. Чтобы найти а со звездочкой 'a\*' - поставим \ перед спец. символом: /a\\*/. |
| ^ | Обозначает начало входных данных. Если установлен флаг многострочного поиска ("m"), то также сработает при начале новой строки.Например, /^A/ не найдет 'A' в "an A", но найдет первое 'A' в "An A." |
| $ | Обозначает конец входных данных. Если установлен флаг многострочного поиска, то также сработает в конце строки.Например, /t$/ не найдет 't' в "eater", но найдет - в "eat". |
| \* | Обозначает повторение 0 или более раз. Например, /bo\*/ найдет 'boooo' в "A ghost booooed" и 'b' в "A bird warbled", но ничего не найдет в "A goat grunted". |
| + | Обозначает повторение 1 или более раз. Эквивалентно {1,}. Например, /a+/ найдет 'a' в "candy" и все 'a' в "caaaaaaandy". |
| ? | Обозначает, что элемент может как присутствовать, так и отсутствовать. Например, /e?le?/ найдет 'el' в "angel" и 'le' в "angle."Если используется сразу после одного изквантификаторов \*, +, ?, или {}, то задает "нежадный" поиск (повторение минимально возможное количество раз, до ближайшего следующего элемента паттерна), в противоположность "жадному" режиму по умолчанию, при котором количество повторений максимально, даже если следующий элемент паттерна тоже подходит.Кроме того, ? используется в предпросмотре, который описан в таблице под (?=), (?!), и (?: ). |
| . | (Десятичная точка) обозначает любой символ, кроме перевода строки: \n \r \u2028 or \u2029. (можно использовать [\s\S] для поиска любого символа, включая переводы строк). Например, /.n/ найдет 'an' и 'on' в "nay, an apple is on the tree", но не 'nay'. |
| (x) | Находит x и запоминает. Это называется "запоминающие скобки". Например, /(foo)/ найдет и запомнит 'foo' в "foo bar." Найденная подстрока хранится в массиве-результате поиска или в предопределенных свойствах объекта RegExp: $1, ..., $9.Кроме того, скобки объединяют то, что в них находится, в единый элемент паттерна. Например, (abc)\* - повторение abc 0 и более раз. |
| (?:x) | Находит x, но не запоминает найденное. Это называется "незапоминающие скобки". Найденная подстрока не сохраняется в массиве результатов и свойствах RegExp.Как и все скобки, объединяют находящееся в них в единый подпаттерн. |
| x(?=y) | Находит x, только если за x следует y. Например, /Jack(?=Sprat)/ найдет 'Jack', только если за ним следует 'Sprat'./Jack(?=Sprat|Frost)/ найдет 'Jack', только если за ним следует 'Sprat' или 'Frost'. Однако, ни 'Sprat' ни 'Frost' не войдут в результат поиска. |
| x(?!y) | Находит x, только если за x не следует y. Например, /\d+(?!\.)/ найдет число, только если за ним не следует десятичная точка. /\d+(?!\.)/.exec("3.141") найдет 141, но не 3.141. |
| x|y | Находит x или y. Например, /green|red/ найдет 'green' в "green apple" и 'red' в "red apple." |
| {n} | Где n - положительное целое число. Находит ровно n повторений предшествующего элемента. Например, /a{2}/ не найдет 'a' в "candy," но найдет оба a в "caandy," и первые два a в "caaandy." |
| {n,} | Где n - положительное целое число. Находит n и более повторений элемента. Например, /a{2,} не найдет 'a' в "candy", но найдет все 'a' в "caandy" и в "caaaaaaandy." |
| {n,m} | Где n и m - положительные целые числа. Находят от n до m повторений элемента. |
| [xyz] | Набор символов. Находит любой из перечисленных символов. Вы можете указать промежуток, используя тире. Например, [abcd] - то же самое, что [a-d]. Найдет 'b' в "brisket" и 'c' в "ache". |
| [^xyz] | Любой символ, кроме указанных в наборе. Вы также можете указать промежуток. Например, [^abc] - то же самое, что [^a-c]. Найдет 'r' в "brisket" и 'h' в "chop." |
| [\b] | Находит символ backspace. (Не путать с \b.) |
| \b | Находит границу слов (латинских), например пробел. (Не путать с[\b]). Например, /\bn\w/ найдет 'no' в "noonday"; /\wy\b/найдет 'ly' в "possibly yesterday." |
| \B | Обозначает не границу слов. Например, /\w\Bn/ найдет 'on' в "noonday", а /y\B\w/ найдет 'ye' в "possibly yesterday." |
| \cX | Где X - буква от A до Z. Обозначает контрольный символ в строке. Например, /\cM/ обозначает символ Ctrl-M. |
| \d | находит цифру из любого алфавита (у нас же юникод). Используйте[0-9], чтобы найти только обычные цифры. Например,/\d/ или /[0-9]/ найдет '2' в "B2 is the suite number." |
| \D | Найдет нецифровой символ (все алфавиты). [^0-9] - эквивалент для обычных цифр. Например, /\D/ или /[^0-9]/ найдет 'B' в "B2 is the suite number." |
| \f,\r,\n | Соответствующие спецсимволы form-feed, line-feed, перевод строки. |
| \s | Найдет любой пробельный символ, включая пробел, табуляцию, переводы строки и другие юникодные пробельные символы. Например, /\s\w\*/ найдет ' bar' в "foo bar." |
| \S | Найдет любой символ, кроме пробельного. Например, /\S\w\*/найдет 'foo' в "foo bar." |
| \t | Символ табуляции. |
| \v | Символ вертикальной табуляции. |
| \w | Найдет любой словесный (латинский алфавит) символ, включая буквы, цифры и знак подчеркивания. Эквивалентно [A-Za-z0-9\_]. Например, /\w/ найдет 'a' в "apple," '5' в "$5.28," и '3' в "3D." |
| \W | Найдет любой не-(лат.)словесный символ. Эквивалентно [^A-Za-z0-9\_]. Например, /\W/ и /[^$A-Za-z0-9\_]/ одинаково найдут '%' в "50%." |
| \n | где n - целое число. Обратная ссылка на n-ю запомненную скобками подстроку. Например, /apple(,)\sorange\1/ найдет 'apple, orange,' в "apple, orange, cherry, peach.". За таблицей есть более полный пример. |
| \0 | Найдет символ NUL. Не добавляйте в конец другие цифры. |
| \xhh | Найдет символ с кодом hh (2 шестнадцатиричных цифры) |
| \uhhhh | Найдет символ с кодом hhhh (4 шестнадцатиричных цифры). |