Тема F. Регулярные выражения.

Регулярное выражение — шаблон, кодирующий ожидаемый формат строки. С помощью регулярного выражения можно проверить, содержатся ли в некоей строке подстроки, соответствующие шаблону, и получить из этих подстрок какие-либо осмысленные части.

Регулярное выражение задаётся с указанием **шаблона** и **ключей**.

Регулярное выражение представляет собой объект класса RegExp.

Создать объект класса RegExp можно двумя способами, с указанием литерала шаблона:

```
var переменная=/шаблон/ключи;
```

или с указанием строкового выражения шаблона:

```
var переменная=new RegExp(шаблон,ключи);
```

Второй вариант синтаксиса удобен, когда регулярное выражение заранее точно не известно и нужно его скомпоновать динамически, программно.

Ключи могут содержать следующие символы:

Символ	Описание
i	При проверке соответствия строки шаблону не различать строчные и прописные буквы (с русскими буквами работает корректно).
g	Искать в строке все подходящие под шаблон подстроки, а не только самую первую.
m	Многострочный поиск, т.е. трактовать проверяемую строку как содержащую несколько строк, разделённых символами перевода строки.

Шаблон может содержать различные виды символов — символы содержания, квантификаторы, якорные символы.

Символы содержания указывают, что в данном месте в строке ожидается некий символ. Символ содержания может быть задан буквально (т.е. обычный символ обозначает самого себя), либо специальной escape-последовательностью, начинающейся с обратного слеша \, либо другими способами.

Некоторые escape-последовательности в символах содержания:

Символ	Описание
\d	Соответствует цифре.
\D	Соответствует нецифровому символу.
\n	Соответствует символу перевода строки.
\r	Соответствует символу возврата каретки.
\s	Соответствует пробельному символу.
\S	Соответствует любому непробельному символу.
\t	Соответствует символу табуляции.
\w	Соответствует латинской букве, цифре или подчеркиванию.
\W	Соответствует любому символу, кроме латинской буквы, цифры или подчеркивания.

Примеры:

Регулярное выражение	Какие строки соответствуют
/JavaScript/	содержащие подстроку "JavaScript"
/JavaScript/i	содержащие подстроку "JavaScript" в любом регистре
/A\d/	в которых указан размер бумаги — латинская буква A и цифра, например строки "размер A4" или "лист A1 или менее"

Другие символы содержания:

Символ	Описание
.	Соответствует любому символу (кроме символа новой строки).
[*xyz*]	Соответствует любому символу из заключенных в квадратные скобки. [0-9] эквивалентно \d, [\f\n\r\t\v] эквивалентно \s, [A-Za-z0-9_] эквивалентно \w Символы в квадратных скобках называются «классами символов».
[^*xyz*]	Соответствует любому символу, кроме заключенных в квадратные скобки. [^0-9] эквивалентно \D, [^ \f\n\r\t\v] эквивалентно \S, [^A-Za-z0-9_] эквивалентно \W
[*a-z*]	Соответствует любому символу в указанном диапазоне.
[^*a-z*]	Соответствует любому символу, кроме лежащих в указанном диапазоне.
x|*y*	Соответствует *x* или *y*. Более левый вариант всегда приоритетнее — если полное регулярное выражение нашлось для варианта *x*, любые варианты для *y* будут игнорироваться.
(*str*)	Соответствует строке *str* и запоминает найденное как один из результатов поиска.
(?:*str*)	Соответствует строке *str*, но не запоминает найденное. Используется для группировки частей шаблона.
N	Соответствует N-ному ранее запомненному результату (N — число от 1 до 99). Номера отсчитываются слева направо, по левым скобкам.
\	Означает, что следующий символ должен пониматься буквально (означать сам себя), даже если следующий символ является специальным, т.е. символ \ отменяет специальное действие следующего символа.

Примеры:

Регулярное выражение	Какие строки соответствуют
/\s.\s/	содержащие два пробельных символа, между которыми ровно один (любой) символ
/[Jj]ava[Ss]cript/	содержащие, например, "JavaScript" или "javascript"
/[А-Я]/	содержащие любую большую русскую букву
/[\da-f]/i	содержащие цифру либо букву от A до F в любом регистре
/A[1-8]/	в которых указан стандартный размер бумаги — латинская буква A и цифра от 1 до 8, например строка "лист A1 или менее"
/(Java|VB)Script/i	содержащие подстроку "JavaScript" либо "VBScript" в любом регистре
/(['"])(red|green)\1/	содержащие слово red или green, взятое в одиночные или двойные кавычки; кавычки до и после слова должны быть одинаковыми
/\d\d\d\-\d\d\-\d\d/	содержащие номер телефона в формате 999-99-99
/\\/	содержащие обратный слеш

Квантификаторы указывают, сколько раз может повториться символ содержания, указанный перед квантификатором.

Кванти-фикатор	Описание
* *?	Соответствует повторению предыдущего символа ноль или более раз. В первом случае — жадный режим («*» заберёт как можно больше символов), во втором — ленивый режим («*?» заберёт как можно меньше символов)
+ +?	Соответствует повторению предыдущего символа один или более раз. Жадный и ленивый режим соответственно.
? ??	Соответствует повторению предыдущего символа ноль или один раз. Жадный и ленивый режим соответственно.
{*n*}	Соответствует ровно n вхождениям предыдущего символа.
{*n*,} {*n*,}?	Соответствует n или более вхождениям предыдущего символа. Жадный и ленивый режим соответственно.
{*n*,*m*} {*n*,*m*}?	Соответствует не менее чем n и не более чем m вхождениям предыдущего символа. Жадный и ленивый режим соответственно.

Примеры:

Регулярное выражение	Какие строки соответствуют
/Java ?Script/	содержащие подстроку "JavaScript" либо "Java Script"
/BEGIN.+END/	содержащие подстроку "BEGIN", потом один или несколько любых символов, и потом подстроку "END"
/\d{2}[abc]/	содержащие двузначное число, за которым следует буква "a", "b" или "c"
/\s.*\s/	содержащие два пробельных символа, даже если они идут подряд
/.+\s.+\s.+/	содержащие два пробельных символа, идущих не подряд, находящихся не в начале и не в конце строки
/\d.+\d/	содержащие две любые цифры (не подряд)
/(\d).+\1/	содержащие одну и ту же цифру в двух местах, но не подряд
/(['"])([^'"]+)\1/	содержащие любой непустой текст в одинарных или двойных кавычках
/\d*/	любые (т.к. в любой строке найдётся 0 цифр)

Якорные символы не соответствуют никаким символам строки, они соответствуют неким позициям «между символами», для которых выполняется определённое условие.

Символ	Описание
^	Соответствует началу строки, а в многострочном режиме — началу каждой строки.
$	Соответствует концу строки, а в многострочном режиме — концу каждой строки.
\b	Соответствует границе слова, т. е. позиции между словом и пробелом или переводом строки. Русские буквы не считаются частью слов.
\B	Соответствует любой позиции, кроме границы слова.
(?=*str*)	Опережающая проверка — соответствует позиции, после которой строка совпадает с регулярным выражением *str*.
(?!*str*)	Негативная опережающая проверка — соответствует позиции, после которой строка **не** совпадает с регулярным выражением *str*.

Примеры:

Регулярное выражение	Какие строки соответствуют
/^JavaScript/	начинающиеся с подстроки "JavaScript"
/JavaScript$/	заканчивающиеся на подстроку "JavaScript"
/\bscript\b/	содержащие подстроку "script" в виде отдельного слова, не как часть более длинного слова, например "my script", но не "my javascript"

Проверка, соответствует ли строка регулярному выражению:

регулярное_выражение.test(*строка*)

Метод test возвращает true либо false, в зависимости от того, *строка* соответствует *регулярному выражению* или нет.

```
console.log( /JavaScript/.test("Мы изучаем JavaScript") );
true
console.log( /JavaScript/.test("Мы изучаем Java Script") );
false
console.log( /Java ?Script/.test("Мы изучаем Java Script") );
true
```

Проверка, соответствует ли строка регулярному выражению, а также получение всех запомненных подстрок:

регулярное_выражение.exec(*строка*)

Метод exec возвращает объект с результатами поиска (или null, если *строка* не соответствует *регулярному выражению*).

Возвращённый объект с результатами поиска содержит свойства:

- index — позиция найденной подстроки, соответствующей регулярному выражению;
- 0 — подстрока, соответствующая регулярному выражению;
- length — общее количество запомненных результатов (результаты для запоминания в шаблоне выделяются скобками);
- 1, 2, ... — запомненные результаты.

```
var A="Мы купили 10 кг яблок по цене 523 рубля за кг";
console.log( /(\d+).+(\d+)/.exec(A) );
{0:'10 кг яблок по цене 523', 1:'10', 2:'3', index:10}
console.log( /(\d+).+?(\d+)/.exec(A) );
{0:'10 кг яблок по цене 523', 1:'10', 2:'523', index:10}
console.log( /\D(\d+)\D.*\D(\d+)\D/.exec(A) );
{0:' 10 кг яблок по цене 523 ', 1:'10', 2:'523', index:9}
var A2="10 кг яблок по цене 523";
console.log( /\D(\d+)\D.*\D(\d+)\D/.exec(A2) );
null
console.log( /\b(\d+)\b.*\b(\d+)\b/.exec(A2) );
{0:'10 кг яблок по цене 523', 1:'10', 2:'523', index:0}
```

Если в регулярном выражении есть флаг g, метод exec возвращает информацию только о первом найденном с совпадении с регулярным выражением. Этот метод можно вызывать повторно для той же строки и **того же** (не такого же) регулярного выражения для получения информации о следующем совпадении:

```
var A="Мы пошли на рынок и купили 10 кг яблок по цене 523 рубля за кг";
var RE=/(\d+)\s+(\S+)/g;
console.log( RE.exec(A) );
{0:'10 кг', 1:'10', 2:'кг', index:27}
console.log( RE.exec(A) );
{0:'523 рубля', 1:'523', 2:'рубля', index:47}
console.log( RE.exec(A) );
null
```

В объекте типа RegExp есть свойство lastIndex, которое и используется для продолжения поиска; можно установить его в 0, если требуется начать поиск с начала строки:

```
RE.lastIndex=0;
```

Есть метод в классе String, который позволяет находить в строке соответствие регулярному выражению:

строка.search(*регулярное_выражение*)

Метод search ищет в *строке* подстроку, соответствующие *регулярному_выражению*, и возвращает её позицию, либо -1, если такое соответствие не найдено. Флаг g в регулярном выражении игнорируется; если совпадающих с регулярным выражением подстрок несколько, вернётся позиция первой из них.

```
var A="Мы пошли на рынок и купили 10 кг яблок по цене 523 рубля за кг";
console.log( A.search(/\d+/) );
```
27

Есть метод в классе String, который позволяет заменять найденные в строке соответствия регулярному выражению на какую-либо другую подстроку:

строка.replace(*регулярное_выражение*,*строка_замены*)

Метод replace ищет в *строке* подстроки (или первую подстроку, если флага g нет), соответствующие *регулярному_выражению*, заменяет их на *строку_замены* и возвращает получившуюся строку. *Строка_замены* может содержать ссылки на запомненные при сопоставлении с регулярным выражением результаты в виде "$1", "$2" и т.д., а также другие специальные последовательности символов.

```
var A="Мы купили 10 кг яблок по цене 523 рубля за кг";
console.log( A.replace(/\b(\d+)\b.*\b(\d+)\b/,"$1 $2") );
```
Мы купили 10 523 рубля за кг
```
console.log( A.replace(/^.*\b(\d+)\b.*\b(\d+)\b.*$/,
  "Стоимость: $2 руб/кг, масса: $1 кг") );
```
Стоимость: 523 руб/кг, масса: 10 кг

Метод match() в классе String в случае регулярного выражения без флага g работает аналогично методу exec регулярного выражения:

```
var A="Мы пошли на рынок и купили 10 кг яблок по цене 523 рубля за кг";
console.log( A.match(/\d+/) );
```
{0:'10', index:27}
```
console.log( "93458".match(/[0-7]+/) );
```
{0:'345', index:1}

В случае регулярного выражения с флагом g метод match() возвращает массив совпавших с регулярным выражением подстрок:

```
var A="Мы пошли на рынок и купили 10 кг яблок по цене 523 рубля за кг";
console.log( A.match(/\d+/g) );
```
['10', '523']

Метод split() в классе String может в качестве разделителя получать регулярное выражение. В этом случае, в строке отыскиваются все подстроки, соответствующие регулярному выражению, и они являются местами разбиения строки.

```
var Txt="Мы не строим, мы не пашем!!! мы - бездельники?";
console.log( Txt.split(/[ .,!?:;-]+/) );
```
['Мы', 'не', 'строим', 'мы', 'не', 'пашем', 'мы', 'бездельники', '']

Методы класса String всегда начинают работу с обнуления свойства lastIndex регулярного выражения, т.е. всегда ищут соответствие с начала строки.