ZAJĘCIA SPECJALISTYCZNE

JavaScript

TEMAT 2-5: Obiekt „RegExp”.

Autor dokumentu: Wojciech Galiński | piątek, 26 września 2014 r. | 312[01]/T,SP/MENiS/2004.06.14

ŹRÓDŁA WIEDZY: http://pl.wikipedia.org/, http://webmaster.helion.pl/index.php/kurs-javascript, http://krook.org/jsdom/, http://www.dynamicdrive.com/, http://www.w3schools.com.

Zagadnienia obowiązkowe

1. **Obiekt „RegExp”** – służy do obsługi wyrażeń regularnych. Tworzy się go jedną z instrukcji:

   `/wyrażenie/flagi;` albo **`new RegExp("wyrażenie", "flagi");`**

   PRZYKŁADY: `wyr1 = /Nysa/i;` `wyr2 = new RegExp("Nysa", "i");`

   gdzie:

   ➔ **WYRAŻENIE** – to sformatowany ciąg znaków, który ma być potem wyszukiwany w tekście. Znaki specjalne w tym ciągu to następujące znaki:

   `. $ ^ { [ ( | ) ] } * + ? \`

   Poniższa tabela opisuje kombinacje powyższych znaków specjalnych ze zwykłymi znakami:

Wyrażenie	Znaczenie wyrażenia	Przykłady
	N A W I A S Y	
tekst	zawiera dosłownie podany tekst (nie może zawierać znaków specjalnych)	`/tekst/.test('To jest tekst.');`
`[aeiouy]`	zawiera jeden z podanych znaków	`/[ąćęłńóśźż]/.exec('Zażółć gęślą jaźń');`
`[^aeiouy]`	NIE zawiera żadnego z podanych znaków (**^** - zaprzeczenie)	`/[^ąćęłńóśźż]/.exec('Zażółć gęślą jaźń');`
`[0-9]` `[A-Za-z]`	zawiera podane przedziały znaków (cyfry, duże i małe litery alfabetu łacińskiego)	`/[01]/.exec('Liczba: 11010101');` `/[0-9a-fx]/.exec('HEX: 0x5a');`
`(wt|so|ni)`	Zawiera jeden z podanych tekstów	`/TAK|NIE/.exec('Jak nie, to NIE!');`
	M E T A Z N A K I	
`\d \D`	cyfra **/** nie cyfra	`/\d{2}-\d{3}/.exec('48-300 NYSA');`
`\s \S`	biały znak (' ', '\n', '\t') **/** nie biały znak	`/\s/.test('Jan\tLem\nTel. 678-123-456');`
`\b \B`	początek słowa **/** koniec słowa	`/ra\B/.exec('Kura raduje się z gratów');`
`\n`	koniec wiersza	`/\n/.test('Jan\tLem\nTel. 678-123-456');`
`\t`	tabulator	`/\t/.test('Jan\tLem\nTel. 678-123-456');`
`\0`	znak „null”	`/\0/.test('tekst\0');`
`\ddd`	kod znaku w postaci: ósemkowej	`/\101/.test('A');` `/\101/i.test('a');`
`\x``dd`	kod znaku w postaci szesnastkowej	`/\x41/.test('A');` `/\x41/i.test('a');`
`\u``dddd`	kod znaku w postaci Unicode	`/\u0041/.test('A');` `/\u0041/i.test('a');`
	K W A N T Y F I K A T O R Y	
`.`	dowolny 1 znak (oprócz '\n' – znaku końca wiersza)	`/ a.a /.exec(' abbba abba aba ');`
`{n}`	dokładnie n razy	`/ [a-z]{4} /.exec(' abc defgh ijkl ');`
`{n,}`	co najmniej n razy	`/ [a-z]{2,} /.exec(' a bcd ef ');`
`{n,m}`	co najmniej n razy, ale nie więcej, niż m razy	`/ [a-z]{2,3} /.exec(' a bcde fgh ');`
?	{0,1}, tzn. pojedynczy element opcjonalny	`/a?/.test('bcd');` `/a?/.exec('bcd');`
*****	{0,}, tzn. może występować, może się także powtarzać	`/a*/.test('bcd');` `/a*/.exec('bcd');`
+	{1,}, tzn. musi występować, może się powtarzać	`/a+/.exec('cccbbbaaa');`
`^n / n$`	ciąg **n** jest **na początku / na końcu** tekstu lub wiersza	`s = 'xyz\nayz';` `/^.y/.exec(s);` `/^a/.test(s);` `/^a/m.exec(s);` `/.yz$/.exec(s);` `/.yz$/m.exec(s);`
`(?=``n)`	sprawdza, czy ciąg **n** występuje po wcześniej szukanym ciągu	`/\d[a-z]{4}(?=abc)/.exec('0testabc1testxyz');`
`(?!``n)`	sprawdza, czy ciąg **n** NIE występuje po wcześniej szukanym ciągu	`/\d[a-z]{4}(?!abc)/.exec('0testabc1testxyz');`

➔ **FLAGI** – to opcje w postaci pojedynczych znaków (ich występowanie i kolejność są dowolne):
- ✔ **i** – ignorowanie wielkości liter w wyrażeniu regularnym,
  PRZYKŁAD: `alert(/d/.test('AbcD')); alert(/d/i.test('AbcD'));`
- ✔ **g** – wyszukiwanie wszystkich wystąpień zgodnych z wyrażeniem regularnym (bez tej flagi wyszukiwane jest tylko pierwsze zgodne wystąpienie),
  PRZYKŁAD:
```
var str = "Abrakadabra\nAlibaba je kebaba", regexp = /ab/gi;
if (regexp.global)
  while((tab = regexp.exec(str)) !== null)
  { for(id in tab) document.write('tab['+id+'] = ' +tab[id]+ '<br />');
    document.write("<hr />"); }
else if ((tab = regexp.exec(str)) !== null)
{ for(id in tab) document.write('tab[' +id+ '] = ' +tab[id]+ '<br />');
  document.write("<hr />"); }
```
- ✔ **m** – przetwarzany tekst składa się z wielu wierszy (istotne w przypadku użycia kwantyfikatorów: „**^**” lub „**$**”).
  PRZYKŁAD: `s='ba\nab'; alert(/^a/.test(s)); alert(/^a/m.test(s));`

2. **Składowe obiektu „RegExp”** – wyróżniamy:

➔ właściwości:
- ✔ **`lastIndex`** – zawiera indeks ostatniego dopasowania do wzorca;
- ✔ **`multiline`**, **`global`**, **`ignoreCase`** – zmienne logiczne przechowujące informacje, czy użyto w wyrażeniu regularnym flag: **m**, **g**, **i**;
- ✔ **`source`** – zawiera treść wyrażenia regularnego.

PRZYKŁAD:
```
var tekst = 'Abrakadabra', wyr = /abra/;   wyr.exec(tekst);
document.write(wyr.lastIndex + ' ' + wyr.source + ' ' +
               wyr.global +' '+ wyr.multiline +' '+ wyr.ignoreCase);
```

➔ metody:
- ✔ **`toString()`** – zwraca treść wyrażenia regularnego jako tekst, np.
  `wyr = /abc/i;   document.write(wyr.toString().toUpperCase());`
- ✔ **`test(tekst)`** – sprawdza, czy w ciągu „**tekst**” znajduje się co najmniej jedno dopasowanie zgodne z wyrażeniem regularnym.
  PRZYKŁADY:
```
alert(/a/.test('aba'));
var tekst = "Alibaba je 50-calowego kebaba", regexp = /al/;
regexp.test(tekst);
```
- ✔ **`exec(tekst)`** – przeszukuje ciąg „**tekst**”, by znaleźć tekst zgodny z wyrażeniem regularnym. Metoda „**exec**” zwraca tablicę złożoną z następujących elementów:
  - **tablica[0]** – znaleziony tekst,
  - **tablica[index]** – indeks pierwszego znaku w znalezionym tekście,
  - **tablica[input]** – przeszukiwany tekst.

  PRZYKŁAD:
```
var tekst = "Alibaba je 50-calowego kebaba", regexp = /ab/i;
alert(regexp.exec(tekst));
```

3. **Inne witryny poświęcone wyrażeniom regularnym** – miejsca warte odwiedzenia to m. in.:
   ➔ http://www.poradnik-webmastera.com/kursy/javascript/wyrazenia_regularne.php,
   ➔ http://www.w3schools.com/jsref/jsref_obj_regexp.asp.

Zadania

1. W tekście podanym z klawiatury znajdź: numer telefonu, identyfikator PESEL, identyfikator NIP, polską odpowiedź na pytanie logiczne z punktu widzenia matematyki, kod pocztowy, polski znak, adres WWW, jeden z 8 kolorów podstawowych (biały, czarny, czerwony, zielony, niebieski, żółty, różowy, szary), adres IP, tagi HTML, znaki interpunkcyjne z klawiatury, cyfry binarne / oktalne / szesnastkowe.
2. Znajdź wyrażenia odpowiadające znakom specjalnym „**%**” i „**_**” w wyrażeniu „LIKE” w języku SQL.