Przydatne strony

http://kursjs.pl/kurs/regular.html

http://www.poradnik-webmastera.com/kursy/javascript/wyrazenia_regularne.php

http://kursjs.pl/kurs/array.html

https://developer.mozilla.org/en-US/docs/Web/JavaS

Walidacja

Zasadniczo walidację podzielić można na:

- pomocną użytkownikowi (po stronie klienta - JavaScript)
- ochraniającą nasze interesy (po stronie serwera - [tu] PHP)

Sama walidacja po stronie serwera wystarcza.

Walidacja tylko po stronie klienta to decydowanie za mało.

Jednak walidacja na poziomie skryptu PHP ma pewną wadę: użytkownik musi poczekać, aż strona internetowa się przeładuje, zanim się dowie, że w polu nazwisko nic nie wpisał. Dużo wygodniej jest wprowadzić walidację poprzez dodanie prostego skryptu w języku JavaScript, który jeszcze przed wysłaniem formularza „z grubsza” sprawdzi, czy wszystko jest w porządku (np. sprawdzi, czy wszystkie obowiązkowe pola zostały wypełnione).

Wyrażenia regularne

Wyrażenia regularne (po angielsku Regular Expressions) są sposobem na zapisanie wzorca, z którym mogą być porównywane ciągi znaków (stringi). Mogą one być używane do sprawdzania czy dany ciąg znaków pasuje do podanego wzorca (np. czy kod pocztowy składa się kolejno z dwóch cyfr, myślnika i trzech cyfr), do wyszukiwania podciągów i ich ewentualnej zamiany na inny ciąg znaków. Proste reguły pozwalają na tworzenie wyrażeń regularnych opisujących reguły budowy nawet bardziej złożonych ciągów znaków. Dzięki temu wyrażenia regularne zdobyły popularność i są szeroko stosowane - nie tylko w JavaScript, ale też w większości obecnie stosowanych języków programowania.

Walidacja po stronie klienta- wykorzystanie wyrażeń regularnych

Wyrażenie regularne w JS może zostać utworzone na dwa sposoby:

1. wyrażenie regularne umieszczamy pomiędzy dwoma znakami slash ("/"), np.:

```
var rx1 = new RegExp('^[0-9]+[a-z]+$');
```

2. Poprzez formalne wywołanie konstruktora obiektu RegExp. np.:

```
var rx2 = /^[0-9]+[a-z]+$/;
```

Powyższe dwa przykłady są sobie równoważne

Walidacja po stronie klienta- wykorzystanie wyrażeń regularnych

Przykład wyrażenia regularnego-sprawdzenie poprawności formatu wpisanego kodu pocztowego

```
var wzorzec =  /^[0-9]{2}-[0-9]{3}$/;
```

- **wyrażenia regularne piszemy w ukośnikach / /**
- **znak początku tekstu (^) i końca tekstu ($)**
- **[0-9]{2} dokładnie dwie cyfry**
- **- po których jest znak minus**
- **[0-9]{3} a następnie dokładnie trzy cyfry**

lub inny, równoważny zapis:

```
var wzorzec =  /^[\d]{2}-[\d]{3}$/;
```

gdzie \d – dowolna cyfra: [0-9]

Walidacja-przykład wykorzystania wyrażenia regularnego -sprawdzenie poprawności kodu pocztowego


```
<body>
<h1>Podaj dane</h1>
<form>
    Wpisz kod (format: 00-000) <br><input type="text" id="inputKod" /><label id="info"
    ></label>
    <br>
    <input type="button"  onclick="sprawdzKod()"    value="zatwierdź" />
    <h3>Proszę sprawdzić działanie skryptu, w przypadku braku danych oraz w przypadku
    wprowadzenia błędnych danych  </h3>
</form>
<script>
function sprawdzKod() {
var wzorzec =  /^[0-9]{2}-[0-9]{3}$/;
//var wzorzec =  /^[\d]{2}-[\d]{3}$/;-zapis równoważny
    var kod = document.getElementById("inputKod").value;
    if (kod == null || kod == "") { document.getElementById("info").innerHTML = "*Brak
    kodu"; }
    else if (wzorzec.test(kod)) { document.getElementById("info").innerHTML = "*Kod
    poprawny"; }
    else { document.getElementById("info").innerHTML = "*Kod niepoprawny"; };
}
</script>
</body>
```

wzorzec.test(kod)-funkcja test zwraca wartość logiczną true, jeżeli kod pasuje do wyrażenia regularnego, zawartego w zmiennej „wzorzec”

Wyrażenia regularne -podstawy

Metaznak	Znaczenie	Przykład wyrażenia	Zgodne ciągi z wyrażeniem	Niezgodne ciągi z wyrażeniem
^	początek wzorca	^za	zapałka, zadra, zapłon, zarazek	kazanie, poza, bazar
$	koniec wzorca	az$ ^.arka$	uraz, pokaz barka, warka	azymut, pokazy parkan
.	dowolny pojedynczy znak	.an.a	panda, Wanda, panna, kania	rana, konia
[...]	dowolny z wymienionych znaków; możemy podawać kolejne znaki lub wpisywać zakres - na przykład [a-z] oznacza wszystkie małe litery. Wymieniając specjale znaki z końca tej tabeli nie musimy poprzedzać znakiem \	[a-z]an[nd]a [a-z][a-zA-Z0-9.-] [pus]	pana, panda, wanna pas, mAs, p2p, m3u, b-s, z.u	Wanda, kania Bas, bal, balu, mp3
[^...]	dowolny z niewymienionych znaków	kre[^st]	krew, krem	kres, kret
|	dowolny z rozdzielonych znakiem ciągów	[nz]a|pod|przed trzynasty|13-ty|13	na, za, pod, przed trzynasty, 13-ty, 13	
(...)	zawężenie zasięgu	g(ż|rz)eg(ż|rz)(u|ó)łka (ósmy|8-my|8)(maj|maja)	gżegżółka, gżegrzółka, gżegrzułka, grzegrzułka ósmy maja, 8-my maj, 8 maja	

Wyrażenia regularne -podstawy

Metaznak	Znaczenie	Przykład wyrażenia	Zgodne ciągi z wyrażeniem	Niezgodne ciągi z wyrażeniem
?	zero lub jeden poprzedzający znak lub element; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	ro?uter (ósmy|8(-my)?)maja?	router, ruter ósmy maja, ósmy maj, 8-mymaja, 8-my maj, 8 maja, 8 maj	
+	jeden lub więcej poprzedzających znaków lub elementów; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	[0-9]+[abc] pan+a (tam)+	10a, 1b, 003c, 42334b pana, panna, pannnna tam, tamtam, tamtamtam	a, b, c, z, 14, 03, 12d, 1231z paa, panda, ta, tamta, mat
*	zero lub więcej poprzedzających znaków lub elementów; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	[0-9]*[abc] pora*n*a*	10a, 1b, 003c, 42334b, a, b, c por, poa, poranna, poraannnaa, pornnna	k, 2335, porada, panna
{4}	dokładnie 4 poprzedzające znaki lub elementy	[0-9]{4}	8765, 8273, 2635	12345, 234, 2123456
{4,}	4 lub więcej poprzedzających znaków lub elementów	[ah]{4,}	haha, haaaaahaha, ahaaa	haa, ha, hehe, aha
{2,4}	od 2 do 4 poprzedzających znaków lub elementów	p.{2,4}a	piana, pola, polana	psa, poranna

Wyrażenia regularne -podstawy

Metaznak	Znaczenie	Przykład wyrażenia	Zgodne ciągi z wyrażeniem	Niezgodne ciągi z wyrażeniem
\.	znak kropki	[0-9]{,3}\.[0-9]{,3}\.[0-9]{,3}	128.0.0.2	128-0-0-2
*	znak *	*.+	*nic	nic*, nic
\/	znak /	^\/\/$	//	
\?	znak ?	^.+\?$	Czy to jest kot?	Czy to jest kot
\:	znak :	^.+\:$	Oto one:	:nic
\.	znak .	\.+	.....	
\^	znak ^	.*\^	To jest ^	To jest &
\+	znak +	[0-9]+\+[0-9]+	928374+29832	23873-32787 238738278
\\	znak \	c\:\\	c:\	
\=	znak =	[0-9]+\+[0-9]+\=[0-9]+	11+12=23	11+12+23
|	znak |	x || y	x || y	

Wyrażenia regularne -podstawy

Flagi

specjalne parametry (flagi), które oddziałują na wyszukiwanie wzorców.:

```
1  var Wyrazenie = /[a-z]*/mg
2  var Wyrazenie = new RegExp("[a-z]*","g")
3
```

znak Flagi	znaczenie
i	powoduje niebranie pod uwagę wielkości liter
g	powoduje zwracanie wszystkich psujących fragmentów, a nie tylko pierwszego
m	powoduje wyszukiwanie w tekście kilku linijkowym. W trybie tym znak początku i końca wzorca (^$) jest wstawiany przed i po znaku nowej linii (\n).

Wyrażenia regularne -podstawy

Flagi

specjalne parametry (flagi), które oddziałują na wyszukiwanie wzorców.:

```
1  var Wyrazenie = /[a-z]*/mg
2  var Wyrazenie = new RegExp("[a-z]*","g")
3
```

znak Flagi	znaczenie
i	powoduje niebranie pod uwagę wielkości liter
g	powoduje zwracanie wszystkich psujących fragmentów, a nie tylko pierwszego
m	powoduje wyszukiwanie w tekście kilku linijkowym. W trybie tym znak początku i końca wzorca (^$) jest wstawiany przed i po znaku nowej linii (\n).

Wyrażenia regularnepodstawy -

Podstawowymi składowymi wyrażeń regularnych są znaki. Każdy znak dopasowuje się do odpowiadającego mu znaku w ciągu znaków - tak jest dla wszystkich znaków z wyjątkiem znaków specjalnych, wymienionych poniżej:

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

Aby dopasować się do jednego z tych znaków, trzeba go w treści wyrażenia regularnego poprzedzić znakiem backslash ("\"). Ważna jest też wielkość znaków - domyślnie wielkość liter ma znaczenie (ale można to wyłączyć).

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any digit between the brackets
[^0-9]	Find any digit NOT between the brackets
(x|y)	Find any of the alternatives specified

Wyrażenia regularne -podstawy

poniższe wyrażenie regularne może dopasować się do imion Ala, Ola, Ula, Ela:

```
.la
```

Można też podać listę znaków które mogą pojawić się na danej pozycji, poprzez wymienienie ich wewnątrz nawiasów kwadratowych. Przykładowo w taki sposób można określić że po literze "a" ma się pojawić cyfra:

```
a[0123456789]
```

-lub wersja skrócona, pozwalająca na podanie wszystkich znaków z określonego zakresu. Stosując ją podaje się pierwszy i ostatni znak z zakresu, rozdzielając je myślnikiem. Poniższy przykład jest równoważny temu poprzedniemu:

```
a[0-9]
```

Wyrażenia regularne -podstawy

Wewnątrz nawiasów kwadratowych można równocześnie podać kilka różnych zakresów znaków oraz wymieniać pojedyncze znaki. Uważać należy jedynie na znaki minusa oraz daszek ("^"
- jeżeli potrzebujemy ich użyć, należy poprzedzić je znakiem backslash ("\"). Zazwyczaj też działa umieszczenie znaku minus jako ostatniego znaku w nawiasach kwadratowych, a daszka - gdzieś poza pierwszą pozycją.

[0-9a-fxyz^-] [0123a-ce-jyz\^\-]

Możliwe jest też zanegowanie listy, czyli określenie jakie znaki nie mogą się znaleźć na danej pozycji. Do tego celu służy znak daszka ("^"), który należy umieścić jako pierwszy w nawiasach kwadratowych. Przykładowo w taki sposób można dopasować się do dowolnego znaku który nie jest cyfrą ani literą:

[^0-9a-zA-Z]

Wyrażenia regularne -przykłady

[A-Z]{1} - jedna duża litera

[^\s]+ - plus oznacza jeden lub więcej znaków, a [^\s] oznacza znak nie będący spacją - czyli razem: jeden lub więcej znaków nie będących spacją

Czy liczba

Chcemy sprawdzić, czy użytkownik wpisał numer.
Javascript udostępnia nam klasę **\d**, która oznacza dowolą cyfrę:

```
1  var zmienna = "909384758699";
2  var wzor = /^\d+$/
3  if (wzor.test(zmienna)) {
4      document.write("To jest liczba")
5  } else {
6      document.write("To nie jest liczba...")
7  }
```

Wyrażenia regularne -przykłady

Weryfikacja Imienia i Nazwiska

Wzorzec opisujący poprawność tych danych ma postać:

```
1  var WzorNazwiska = /^[a-zA-Z]{2,}\s+[a-zA-Z]{2,}$/;
2  //lub
3  var WzorNazwiska = /^[\D]{2,}\s+[\D]{2,}$/;
```

/	- od tego znaku muszą się zaczynać i kończyć wszystkie wzorce w JavaScripcie
^	- Wzorzec ma się zaczynać z początkiem tekstu
[a-zA-Z]{2,}	- Ciąg musi zawierać przynajmniej 2 litery (imie)
\s+	- Po których znajdą się spacje lub tabulatory (min jeden)
[a-zA-Z]{2,}	- Po których znajdą się znowu przynajmniej 2 litery (nazwisko)
$	- Wzorzec ma się kończyć z końcem tekstu

Teraz możemy nasz wzór wykorzystać w skrypcie za pomocą metody **test**:

```
1  var imieINazw = "Marcin Domanski";
2  WzorNazwiska = /^[a-zA-Z]{2,}\s[a-zA-Z]{2,}$/;
3  if (WzorNazwiska.test(imieINazw)) document.write('Imie i Nazwisko jest OK!')
```

Wyrażenia regularne -przykłady

Weryfikacja E-Maila

Jak wiemy, aby adres **email** był prawidłowy musi spełniać kilka zasad:

- musi posiadać nazwę konta (składającą się z kiku znaków - poza znakami **?,:*/** i spacji)
- po dokumencie musi znaleźć się znak @
- po małpie powinna znaleźć się przynajmniej jedna **.** i to pomiędzy domeną a końcówką adresu
- końcówka adresu powinna składać się z przynajmniej 2 liter (pl, com itp.)

Wzór który by opisywał adres mail będzie miał postać:

```
1 var wzorMaila = /^[0-9a-zA-Z_.-]+@[0-9a-zA-Z.-]+\.[a-zA-Z]{2,3}$/
```

/	- od tego znaku muszą się zaczynać i kończyć wszystkie wzorce w JavaScripcie
^	- Wzorzec ma się zaczynać z początkiem tekstu
[0-9a-zA-Z_.-]+	- Następnie badamy nazwę konta, która może składać się z dowolnych znaków (cyfry, litery, .-_)
@	- Potem sprawdzamy wystąpienia znaku @
[0-9a-zA-Z_.-]+	- Po znaku @ sprawdzamy domenę, która może składać się z takich samych znaków co nazwa konta oprócz znaku _
\.	- Po dokumencie musi wystąpić kropka
[a-zA-Z]{2,3}	- Po kropce musi wystąpić końcówka domeny, która może się składać wyłącznie z liter i jej długość musi być od 2 do 3 znaków
$	- Wzorzec ma się kończyć z końcem tekstu

Zamiast za każdym razem wpisywać a-zA-Z, możemy uprościć formularz. Na końcu za znakiem / należy wpisać i, co sprawi, że wielkość liter nie będzie brana pod uwagę. Dzięki temu nasze wyrażenie nieco się uprości:

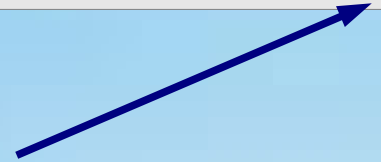
```
1 var wzorMaila = /^[0-9a-z_.-]+@[0-9a-z.-]+\.[a-z]{2,3}$/i
```

Sprawdźmy nasz wzór w przykładowym polu tekstowym (za pomocą metody **test**):

```
1 var mail = "mar-dom@wp.pl";
2 var wzorMaila = /^[0-9a-z_.-]+@[0-9a-z.-]+\.[a-z]{2,3}$/i
3 if (wzorMaila.test(mail)) document.write('Mail OK')
```

Wyrażenia regularne -przykłady

```
var wzorzecR =  /^[-]?[0-9]+[.]?[0-9]*$/;
```



Wzorzec liczby rzeczywistej:

[-]? - występuje co najwyżej raz znak minus

[0-9]+ po którym występuje co najmniej jedna cyfra

[.]? po której może wystąpić znak kropka

[0-9]* po której mogą wystąpić cyfry

```
var wzorzecC =  /^[-]?[0-9]+$/;
```

Wzorzec liczby całkowitej