# Java Script

**Cz.3**

# Wyrażenia regularne

**stanowią doskonały sposób na badanie i modyfikowanie tekstu. Dzięki swej olbrzymiej elastyczności pozwalają w łatwy sposób pobierać pasujące fragmenty tekstu.**

**Przykładowy wzorzec może mieć np postać:**

**/^[a-zA-Z]{2,}\s[a-zA-Z]{2,}$/**

**Może się on wydawać bardzo skomplikowanym zapisem, jednak w praktyce tak nie jest!**

Aby w Javascript korzystać z wyrażeń regularnych, musimy utworzyć obiekt **RegExp**(wyrażenie, flaga), który przyjmuje 2 argumenty: wyrażenie, którym będziemy testować, oraz dodatkowe flagi, które poznamy w tym rozdziale.

```
var reg = new RegExp("pani?" , "gi")
```

lub

```
var reg = /pani?/gi
```

**Którą metodę wybrać?**

Każda z nich ma swoje wady i zalety. Pierwsza wymaga poprzedzania specjalnych znaków np ? (które zaraz poznamy) podwójnym ukośnikiem //. W drugiej metodzie specjalne znaki poprzedzamy jednym ukośnikiem, ale całe wyrażenie musimy objąć parą ukośników. Większość deweloperów wybiera drugą metodę, jest to jednak kwestia gustu.

# Metaznaki

**Każdy wzorzec składa się z meta znaków, czyli specjalnych znaków, które opisują jak mają wyglądać wyszukiwane fragmenty tekstu.**

**Przykłaowe metaznaki:**

**^** **początek wzorca**

**Przykład:** **^za**

**Dobre będą:** **zapałka, zadra, zapłon, zarazek**

**Złe będą:** **kazanie,poza, bazar**

**$** **koniec wzorca**

**Przykład:** **az$**

**Dobre będą:** **uraz, pokaz**

**Złe będą:** **barka, warka** **azymut, pokazy**

. **dowolny pojedynczy znak**

**Przykład:** **.an.a**

**Dobre:** **panda, Wanda, panna, kania**

**Złe:** **rana, konia**

**+** **jeden lub więcej poprzedzających znaków lub elementów; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)**

**Przykład:** **[0-9]+[abc]**

**Dobre:** **10a, 1b, 003c, 42334b**

**Złe:** **a, b, c, z, 14, 03**

**Więcej meta znaków w sieci…**

# Flagi

**Poza wymienionymi meta znakami istnieją specjalne parametry (flagi), które oddziałują na wyszukiwanie wzorców.:**

```
var reg = /[a-z]*/mg
var reg = new RegExp("[a-z]*","g")
```

| 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 liniowym. W trybie tym znak początku i końca wzorca (^$) jest wstawiany przed i po znaku nowej linii (\n). |

# Zastosowanie metody test()

**Metoda test() służy do sprawdzania, czy dane wyrażenie znajduje się w tekście:**

```
var text = "cat dog";
var reg = /cat/;
reg.test(text) == true          //bo cat znajduje się w tekście

var reg2 = /^cat$/;
alert(reg2.test(tekst));        //false - bo wzorzec zaczyna się z początkiem i kończy
                                z końcem tekstu (znaki ^ i $) - jedyny pasujący tekst to
                                "cat"
```

# PRZYKŁAD

**Aby wyszukać w tekście kod pocztowy użyjemy wyrażenia:**

```
var codeReg = /[0-9]{2}-[0-9]{3}/g;
```

**//lub**

```
var codeReg = /[\d]{2}-[\d]{3}/g;
```

**[0-9]{2} - powinny znaleźć się 2 cyfry**

**- - po 2 cyfrach powinien znaleźć się znak -**

**[0-9]{3} - po którym powinny znaleźć się 3 cyfry**

**g - mają być zwrócone wszystkie wyszukane pasujące ciągi**

# Weryfikacja Imienia i Nazwiska

**Wzorzec opisujący poprawność tych danych ma postać:**

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

**/ - od tego znaku muszą się zaczynać i kończyć wszystkie wzorce w JavaScripcie**

**^ - Wzorzec ma się zaczynać z początkiem tekstu**

**[a-zA-Z]{3,} - Ciąg musi zawierać przynajmniej 4 litery (imię)**

**\s+ - Po których znajdą się spacje lub tabulatory (min jeden)**

**[a-zA-Z]{3,} - Po których znajdą się znowu przynajmniej 3 litery (nazwisko)**

**$ - Wzorzec ma się kończyć z końcem tekstu**

**KONIEC**