Regular Expressions

escape: \ (backslash)

RegExp oggetto

comodo quando è statica ≈ utile quando la Regular Expression è dinamica.

literal Syntax

/ s$ / g

s$ → Pattern
g → flag

RegExp() constructor

new RegExp("s$", g)

---

default: sono GREEDY (max)
con ? diventano LAZY (min)

GREEDY
"aabab".match(/a.*b/) ⇝ "aabab"
"aabab".match(/a.*?b/) ⇝ "aab"
LAZY — no!

NB → la prima che incontra

() :
→ scopo 1 raggruppare per considerare come una singola unità (ab)+
→ scopo 2 estrarre valori (ab) gruppo 1 e poi \1 si riferisce al valore del gruppo 1

LI = LastIndex (quindi ATTENZIONE!)

METODI

Sulle Stringhe

"...".search(regexp)

"...".replace(regexp, replacement)
→ stringa o Funzione o $1 gruppi

"...".match(regexp) LI
g ⇝ [...] array di tutti i match
g ⇝ [matched String, gruppi catturati] +named properties
"y ⇝ attenzione al lastIndex

"...".matchAll(regexp) g flag set
torna un iteratore di matchedString

"...".split(regexp)
occhio se la regexp definisce gruppi!

Sull'oggetto RegExp

reg-exp.test(stringa) LI true o false
occhio che se uso g o y dipende dal valore del LastIndex

reg-exp.exec(stringa) LI
• [matched String, gruppi catturati]
• occhio che ritorna sempre UN solo match, non importa se ha g o no.
NB • Il match che torna dipende dal valore del LastIndex! Quindi occhio se utilizzo la stessa regexp in un ciclo!

CONCLUSIONE
Il metodo più comodo e sicuro è il "...".matchAll(...)