RegExp

Regular Expressions / Reguläre Ausdrücke

Was sind reguläre Ausdrücke?

- Erlauben es Zeichenketten zu vergleichen
- bspw. ob die Syntax einer E-Mail-Adresse korrekt geschrieben wurde oder
- ob eine Telefonnummer die korrekte Formatierung hat oder
- ob ein Passwort gültige Zeichen beinhaltet und die korrekte Länge besitzt
- Infos findet man unter www.google.de mit dem Suchbegriff „reguläre +ausdrücke +tutorial“

Reguläre Ausdrücke in JavaScript

- JavaScript bietet die Möglichkeit Objekte mit dem Konstruktor „`RegExp`“ zu erzeugen.
- Dieses Objekt kann Operationen mit regulären Ausdrücken überwachen
- und speichert Informationen über zuletzt durchgeführte Operationen.
- Methoden des String-Objektes, welche reguläre Ausdrücke verwenden:
  - `match()`
  - `replace()`
  - `search()`

Aufbau von regulären Ausdrücken

- reguläre Ausdrücke sind Muster (Patterns), die auf Zeichenketten angewendet werden.
- gewöhnliches Muster besteht aus drei Teilen:
  - dem Pattern
  - den Flags
  - den Begrenzern (Delimiter - kennzeichnen den Anfang und das Ende des Musters)

```
/Pattern/Flags
```

Instanzen erzeugen

- Variante 1:

```
var suche1 = /sel/;
```

- Variante 2:

```
var suche2 = new RegExp('sel');
```

Flags

- `g` – globale Suche: alle Treffer werden gefunden
- `i` – lässt die Groß- und Kleinschreibung unberücksichtigt

Metazeichen (Auszug)

Metazeichen	Findet…	Beispiel
`\b`	Eine Wortgrenze	`/\bthe/` findet "thematisch", "theatralisch". `/the\b/` findet "Agathe", "Mathe". `/\bthe\b/` findet nur "the".
`\B`	Eine Nicht-Wortgrenze, also nicht am Beginn oder Ende eines Wortes	`/\Ber\B/i` findet "Wert", aber nicht "Erde" oder "Heer".
`\d`	Eine Ziffer von 0 bis 9	`/\d\d/` findet eine zweistellige Ziffer wie "42".
`\D`	Eine Nicht-Ziffer	`/\D\D/` findet alles außer Zahlen.
`\s`	Ein Leerzeichen, Tabulator, Zeilenumbruch	`/E-Mail\sAdresse/` findet "E-Mail Adresse" aber nicht "E-Mail-Adresse"
`\S`	Ein Nicht-Leerzeichen	`/E-Mail\SAdresse/` findet "E-Mail-Adresse" aber nicht "E-Mail Adresse"
`(...)`	eine Zeichenkette und legt sie in der Variablen $1 ab	`/(\d{1,},\d{2}))€./` findet "Es kostet 49,90€." und legt "49,90€." in `$1` ab.

Metazeichen (Auszug)

Metazeichen	Findet…	Beispiel
`\w`	einen Buchstaben, eine Ziffer oder einen Unterstrich	`/\w1\w/` findet "A1A, "_12", "Z1_".
`\W`	keinen Buchstaben, keine Ziffer oder keinen Unterstrich	`/\W1\W/` findet " 1%", aber nicht "11%".
`.`	jedes Zeichen außer einem Zeilen-umbruch	`/../` findet zwei zusammenhängende Zeichen, wie z.B. "Z3".
`^`	den Beginn einer Zeichenkette	`/^Frau/` findet "Frau Klößer…", aber nicht "Ich bin Frau Klößer."
`$`	das Ende einer Zeichenkette	`/sie\.$/` findet "Alle schätzen sie."
`[…]`	irgendein Zeichen, das in der Klammer aufgelistet ist	`/W[eoa]rt/` findet "Wert", "Wort", "Wart", aber nicht "Wirt".
`[^…]`	keines der in Klammern angegebenen Zeichen	`/W[^au]rt/` findet "Wirt", "Wert", "Wort", aber nicht "Wart" und "Wurt".

Metazeichen für die Häufigkeit des Suchbegriffs

Metazeichen	Findet…	Beispiel
`+`	ein- oder mehrmals	`/\d+/` findet "3", "5678".
`?`	kein- oder einmal	`/\d?/` findet " ", "3" oder "9" aber nicht "11".
`*`	kein-, ein- oder mehrmals	`/\d*/` findet " " und alle zusammenhängenden Zahlen.
`{n}`	genau n-mal	`/\d{3}/` findet dreistellige Zahlen, wie "345".
`{n,}`	n- oder mehrmals	`/\d{2,}/` findet zwei- und mehrstellige Zahlen
`{n,m}`	mindestens n-, maximal m-mal	`/\d{3,5}/` findet drei-, vier- und fünf-stellige Zahlen.

Zeichen maskieren

- die Zeichen `.  [  ]  *  ?  {  }  (  )  ^  $` müssen mit einem Backslash `\` maskiert werden (auch der Backslash selbst)
- Beispiel:

```
var suche = /\D:\\Daten\\Grafiken\\Apfel\.gif/i;
```

Methoden des RegExp-Objektes

- `exec(Zeichenkette)`:
  - führt Suche nach dem Pattern in der Zeichenkette durch
  - liefert Array mit den Eigenschaften `index` und `input`
  - `index`: Position des ersten Zeichens des gefundenen Musters
  - `input`: die durchsuchte Zeichenkette
  - weiterhin die Teilstrings der gefundenen Treffer
  - wird nichts gefunden liefert `exec null`
- `test(Zeichenkette)`:
  - liefert `true` oder `false`, je nachdem ob die Suche erfolgreich war oder nicht