User Interfaces: JavaScript

Reguliere expressies

KdG Karel de Grote Hogeschool

Reguliere expressies

- Vaak nodig om het formaat te valideren:
  - Is het een juist gevormd e-mail adres?
  - Is het een correct telefoonnummer
  - Bevat de postcode enkel cijfers?
  - …
- Je kan hiervoor het **`RegExp`**-object gebruiken

Overzicht: http://regexlib.com/CheatSheet.aspx

Tutorial: http://www.regular-expressions.info

Testen: https://regex101.com/#javascript

RegExp

```
let pattern = /\d{3}|[a-z]{4}/;
let patroon = new RegExp("\d{3}|[a-z]{4}");
```

Exact 3 cijfers of 4 kleine letters

- Object dat een patroon van karakters beschrijft
- Handig om te controleren of een string een bepaalde vorm heeft
- Aanmaak:
  - via speciale literal vorm (zie voorbeeld)
  - via de constructor functie **`RegExp(…)`**

RegExp: algemeen

- `/java/` alle strings die "java" bevatten
- `/s$/` alle strings die eindigen op `s`
- Speciale karakters: `\0`,`\t`,`\n`,…
- Karakters met speciale betekenis:

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

  Heb je die toch letterlijk nodig: `\` ervoor

RegExp: character classes

Character	Matches
`[...]`	Any one character between the brackets.
`[^...]`	Any one character not between the brackets.
`.`	Any character except newline or another Unicode line terminator.
`\w`	Any ASCII word character. Equivalent to `[a-zA-Z0-9_]`.
`\W`	Any character that is not an ASCII word character. Equivalent to `[^a-zA-Z0-9_]`.
`\s`	Any Unicode whitespace character.
`\S`	Any character that is not Unicode whitespace. Note that `\w` and `\S` are not the same thing.
`\d`	Any ASCII digit. Equivalent to `[0-9]`.
`\D`	Any character other than an ASCII digit. Equivalent to `[^0-9]`.
`[\b]`	A literal backspace (special case).

RegExp: herhalen

Character	Meaning
{ *n* , *m* }	Match the previous item at least *n* times but no more than *m* times.
{ *n* ,}	Match the previous item *n* or more times.
{ *n* }	Match exactly *n* occurrences of the previous item.
?	Match zero or one occurrences of the previous item. That is, the previous item is optional. Equivalent to {0,1}.
+	Match one or more occurrences of the previous item. Equivalent to {1,}.
*	Match zero or more occurrences of the previous item. Equivalent to {0,}.

Voorbeelden:

`/\d{2,4}/` → 2 tot en met 4 cijfers

`/\w{3}\d?/` → exact 3 alfanumerieke tekens, daarna eventueel een cijfer

`/\s+java\s+/` → **`java`** met 1 of meerdere spaties voor of na

`/[^(]*/` → 0 of meerdere karakters, geen open haakje!

Alternatieven en groeperen

`|`	→ alternatieven
`/ab|cd|ef/`	→ **ab** of **cd** of **ef**
`/\d{3}|[a-z]{4}/`	→ 3 cijfers of 4 kleine letters
`()`	→ groeperen
`/java(script)?/`	→ **java** optioneel gevolgd door "**script**"
`/(ab|cd)+|ef/`	→ 1 of meerdere keren (**ab** of **cd**) ofwel **ef**

Groeperen kan ook gebruikt worden om er daarna naar te 'refereren'

Refereren

- Stel: twee delen van een string moeten hetzelfde zijn, bv:
  - ofwel tussen single quotes, ofwel tussen double quotes
- Poging: `/['"][^'"]*['"]/`
- Maar: 'appel" is dan ook correct
- Beter: met referentie: `/(['"])[^'"]*\1/`
  → nu moet het laatste karakter hetzelfde zijn als het eerste…

`\1` refereert naar het subpattern `(['"])`

\1 - it means the first capturing group in the matched expression.

\n would be the nth capturing group.

(a|b)\1
→ aa or bb

(1|2)(3|4)\1\2
→ 1313 or
1414 or
2323 or
2424

Match positie

- `^` begin van de string
- `$` einde van de string

- `\b` word boundary
- `\B` not word boundary

Flags

Character	Meaning
i	Perform case-insensitive matching.
g	Perform a global match—that is, find all matches rather than stopping after the first match.
m	Multiline mode. ^ matches beginning of line or beginning of string, and $ matches end of line or end of string.

```
"JavaScript".search(/script/i);
text.replace(/javascript/gi,"JavaScript");
```

String methods

```
1  var str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
2  var regexp = /[A-E]/gi;
3  var matches_array = str.match(regexp);
4
5  console.log(matches_array);
6  // ['A', 'B', 'C', 'D', 'E', 'a', 'b', 'c', 'd', 'e']
```

```
"JavaScript".search(/script/i);
```

--> geeft positie of -1 terug

```
text.replace(/jscript/gi,"JavaScript");
```

```
text.replace(/"([^"]*)"/g,'"$1"');
```

```
text.match(/\d+/g);
```
→ geeft array terug

```
var str = 'zie ook Hst 3.4.5.1 en Hst 3.6';
console.log(str.match(/hst \d+(\.\d)*/gi));
```

```
Array [ "Hst 3.4.5.1", "Hst 3.6" ]
```

```
text.split(/\s*,\s*/);
```
→ geeft array terug

```
var text = 'een, twee ,  drie  ,    vier';
console.log(text.split(/\s*,\s*/));
```

```
Array [ "een", "twee", "drie", "vier" ]
```

RegEx methods

- Functie die test op exact 4 cijfers in de range 1000..9999 (als **object** een textfield is):

```
function vierCijfers(object) {
    let inhoud = object.value;
    const regEx = /^[1-9]\d{3}$/;
    return regEx.test(inhoud);
}
```

- Alternatief (minder gebruikt):

```
function vierCijfers(object) {
    let inhoud = object.value;
    const regEx = /^[1-9]\d{3}$/;
    return regEx.exec(inhoud) != null;
}
```

Oefening regex 1

- Maak gebruik van regular expressions om het volgende te valideren (voorzie een formulier met de nodige invoervelden):
  - een niet-gepersonaliseerde Belgische nummerplaat (vanaf 3 letters + 3 cijfers, over 3 cijfers + 3 letters tot "1"gevolgd door 3 letters en 3 cijfers) Geldig: ABC-123 of 123-ABC of 1-ABC-123
  - een e-mailadres van de Karel de Grote Hogeschool, eindigt dus op @student.kdg.be of @kdg.be
  - een Belgisch rekeningnummer (IBAN)
    - Geldige nummers: BE15-7878-0144-5741 of BE15001470142457
  - een Belgisch telefoonnummer (vast toestel)
    - Geldige nummers: 014/25.57.47 09/548.14.15
    - Alleen de zones 2, 3, 4, en 9 hebben zeven cijfers na de slash

Oefening regex 1

- Het invoerformulier zou er bijvoorbeeld zo kunnen uitzien (nog aan te vullen met het formaat dat de gebruiker moet/mag gebruiken):

Valideren met Regular Expressions

Telefoon:

IBAN nummer:

Nummerplaat:

E-mail:

Verzend

Tutorial: https://regexone.com

Tip voor testen regular expressions, dit kan via: https://regex101.com/#javascript