Regular Expressions in JavaScript

- Besondere Zeichen
- Optional, Min, Max,…
- Zeichenklasse
- Flags
  - Unicode
- Methoden (test, exec,…)
  - `test` outputs true or false
  - `exec` outputs a weird array or null if nothing found:
  - `exec` outputs first index first
  - `exec` use sticky flag to set lastIndex manually
  - `match` (do not forget to use `g` flag)
- Beispiele
  - Zeichenklassen & Wiederholungen
  - Wordboundary
  - UTF8/16/…
  - Miscellaneous
    - Zeichenklassen
    - Oder
    - Within single qutoes
    - n’te Gruppe (naive Annahme: Zwischen Quotes)
    - n’te Gruppe (besser)
    - Benannte Gruppen `(?<name>X)` ⋯ `\k<name>`
    - Keine Gruppe trotz `(  )` → `(?:` ⋯ `)`

Besondere Zeichen

```
^ $ . , - * + ? = ! : | ( ) [ ] { } \

\f \n                   Seitenvorschub, Zeilenumbruch
\r \t \v                Wagenrücklauf, H-Tabulator, V-Tabulator

\d \D                   digit, non digit
\s \S                   white space, non white space
\w \W                   word, non word
\b \B                   word boundary, non word boundary

\cI                     horizontal tab
\cH                     STRG + H
\p{L}                   any kind of letter from any language
```

Optional, Min, Max,...

X is a placeholder for any regular expression (own notation)

```
X?                      optional
X*                      0 oder mehr
X+                      1 oder mehr
X{n}                    n mal
X{n,}                   mindestens n mal
X{m,n}                  mindestens m mal max n mal
```

Zeichenklasse

```
[abc]                   Zeichenklasse: eines davon
[^abc]                  Keines davon
```

Flags

```
i       ignoreCase
m       multiline     ^ und $ &rarr; Anfang/Ende der Zeile
s       dotAll        . steht nun auch für \n
u       unicode
g       global
y       sticky        beginnt bei lastIndex
```

Unicode

```
L                     Buchstabe
Lu                    Großbuchstabe
Ll                    Kleinbuchstabe
Nd                    Dezimalzahl
P                     Satzzeichen
S                     Symbol
White_Space           identisch mit \s
Emoji                 Emoji & Co.
```

Methoden (test, exec,...)

`test` outputs true or false

```
const res = /[0-9]+/.test('Bond 007 is active!');

console.log(res); // true
```

`exec` outputs a weird array or null if nothing found:

```
const res = /[0-9]+/.exec('Bond 007 is active!');
```

```
[
  '007',
  index: 5,
  input: 'Bond 007 is active!',
  groups: undefined
]
```

```
console.log(res?.[0]); // 007
console.log(res?.index); // 5
console.log(res?.input); // Bond 007 is active!
console.log(res?.groups); // undefined
```

`exec` outputs first index first

To find all results use `g` flag.

```
const someRegExp = /[0-9]+/g;
const someText = 'Agents 007 and 009 active!';
let res: RegExpExecArray | null;
res = someRegExp.exec(someText);
console.log(res);
res = someRegExp.exec(someText);
console.log(res);
res = someRegExp.exec(someText);
console.log(res);
```

outputs:

```
[
  '007',
  index: 7,
  input: 'Agents 007 and 009 active!',
  groups: undefined
]
[
  '009',
  index: 15,
  input: 'Agents 007 and 009 active!',
  groups: undefined
]
null
```

`exec` use sticky flag to set lastIndex manually

```
const someRegExp = /[0-9]+/y;
someRegExp.lastIndex = 15;
const someText = 'Agents 007 and 009 active!';
let res: RegExpExecArray | null;
res = someRegExp.exec(someText);
console.log(res);

/** outputs:
[
  '009',
  index: 15,
  input: 'Agents 007 and 009 active!',
  groups: undefined
]
*/
```

`match` (do not forget to use `g` flag)

Without `g` -flag similar result like `exec`.

Important: The order is changed: `Text.match(regExp)`

```
const someRegExp = /[0-9]+/g;
const someText = 'Agents 007 and 009 active!';
let res;
res = someText.match(someRegExp);
console.log(res);

/** outputs:
[ '007', '009' ]
*/
```

Beispiele

Zeichenklassen & Wiederholungen

```
let a: RegExp;

a = /[^0-9]/; // Keine Ziffern
a = /[^0-9]+$/; // Am Ende keine Ziffern
a = /[0-9]{4,6}/; // Vier bis sechs Stellen
```

Wordboundary

```
e\b                    jedes e vor einem "Wordboundary".
```

> Schul**e** Haus Hos**e** Matratz**e**

UTF8/16/...

```
\u{1f310}
```

Miscellaneous

Zeichenklassen

```
[^0-9]+$            Keine Ziffern am Ende
[1-9][0-9]*         Zahl ohne führende Null
```

Oder

```
http|ftp            http oder ftp
```

Within single qutoes

```
'([^']*)'
```

> Was heißt “hot”, ist es **‘heiß’** oder **‘kalt’**?

n’te Gruppe (naive Annahme: Zwischen Quotes)

Alles zwischen double- or single quotes. Der Fehler: Text dazwischen wird auch ausgewählt:

```
(['"]).*\1
```

> Was heißt **“hot”**, ist es **‘heiß’ oder ‘kalt’**?

n’te Gruppe (besser)

Alles zwischen double- or single quotes. (Korrigierte Version des obigen Beispiels)

```
(['"])[^'"]*\1
```

> Was heißt **“hot”**, ist es **‘heiß’** oder **‘kalt’**?

Benannte Gruppen `(?<name>X)` … `\k<name>`

```
(?<q>['"])[^'"]*\k<q>              Gruppe heißt q (same above)
```

> Was heißt **“hot”**, ist es **‘heiß’** oder **‘kalt’**?

Keine Gruppe trotz `(  )` → `(?: … )`

```
/(?:http|ftp):\/\/(.*)/gm       // wurde maskiert mit \/\/
```

http://example.com

https://example.com

ftp://example.com

dns://something.wrong