

# Reguläre Ausdrücke

# Reguläre Ausdrücke = intelligentes Suchen/Ersetzen

```
<table>
[...]  
<tr height="12">  
<td height="12" align="right">3245599</td>  
<td class="xl24" align="right">5.0</td>  
</tr>  
<tr height="12">  
<td height="12" align="right">3293599</td>  
<td class="xl24" align="right">3.3</td>  
</tr>  
[...]  
</table>
```

Aufgabe: Alle Vorkommen von height="..." ausgeben bzw. ersetzen.

# Reguläre Ausdrücke

## Wozu?

- Überprüfen auf ein bestimmtes Format
  - Gültige E-Mail-Adresse: baldes@wara.de
  - Datumsformat: 21.03.2021
- Finden von Informationen
  - Suche alle Links auf der HTML-Seite
  - Finde alle Maiers: Maier, Meier, Mayer, ...
- Suchen und Ersetzen
  - Ersetze Wörter mit dem Genderstern
  - Tausche die Spalten einer HTML-Tabelle

# Reguläre Ausdrücke

## Metazeichen

Meta	Beschreibung	Beispiel
<b>.</b>	Findet jedes Zeichen	"Schmi.." Schmitt Schmidt Schmi5x
<b> </b>	Oder	"Schmidt Schmiedt" Schmidt Schmiedt
<b>[abc]</b>	Findet a,b oder c	"Schmi[dt][dt] " Schmidd Schmitd Schmidt
<b>[a-z]</b>	Findet einmal alle Zeichen von a bis z	"[w-z]" w x y z
<b>[^abc]</b>	Findet ein Zeichen, das <b>nicht</b> a oder b oder c ist	"Schmi[dt][^d] " Schmidt Schmitt Schmidx
<b>()</b>	Gruppierung	"(wort1 wort2)" wort1 wort2
<b>\</b>	Folgendes Metazeichen wird als normales Zeichen (Literal) interpretiert.	"\". .
<b>^</b>	Zeilenanfang	"^Anfang" Anfang
<b>\$</b>	Zeilenende	"Ende\$" Dieser ... hat ein Ende

# Reguläre Ausdrücke

## Übung 1

"^wort" {'wort' at the start of a line}

...

"wort\$" {'wort' at the end of a line}

...

"^wort\$" {lines containing only 'wort'}

...

"\^s" {word '^s,, not only at the beginning, "\" escapes the ^}

...

"[Ww]ort" {search for ,Wort' or 'wort'}

...

"B[oO][bB]" {search for BOB, Bob, BOb or BoB }

...

"^\$" {search for blank lines}

...

"[0-9][0-9]" {search for pairs of numeric digits}

...

# Reguläre Ausdrücke

## Wiederholungsfaktoren

Faktor	Beschreibung	Beispiel
<b>*</b>	Der vorstehende Ausdruck darf <b>beliebig oft (auch keinmal)</b> vorkommen.	<b>0*[1-9]+</b> -> 05 011 0087 <b>00</b> 0002 12
<b>+</b>	Der voranstehende Ausdruck muss <b>mindestens einmal</b> vorkommen, darf aber auch mehrfach vorkommen.	<b>0+[1-9]+</b> -> 05 011 0087 0002
<b>*? bzw +?</b>	Non-greedy operator; Der Ausdruck matcht nur das erste Vorkommen, nicht das letzte.	<b>a.*?a</b> angewendet auf abbabba matcht abba  Wohingegen <b>a.*a</b> mit abbabba matcht.
<b>{min,max}</b>	Der voranstehende Ausdruck kommt <b>mindestens min-mal</b> und <b>höchstens max-mal</b> vor.	<b>0{0,1}[1-9]{0,1}</b> -> 05 0 5 <b>Eine Abkürzung für {0,1} ist ?.</b>

**Vorsicht:** \*.txt bei Dateien entspricht dem regulären Ausdruck **.\*\.**txt

# Reguläre Ausdrücke

## Vordefinierte Ausdrücke

<code>\d</code>	digit	eine Ziffer, also [0-9]
<code>\D</code>	no digit	ein Zeichen, das keine Ziffer ist, also [^\d]
<code>\w</code>	word	also [a-zA-Z_0-9]
<code>\W</code>	no word	weder Buchstabe noch Zahl, also [^\w]
<code>\s</code>	space	Leerzeiche
<code>\S</code>	non-space	kein Leerzeichen
<code>\b</code>	word boundry	Hier beginnt ein Word, z.B. <code>\b\w+\b</code>

# Reguläre Ausdrücke

## Beispiele

- **"the .\* (who|that)"**
  - Findet Relativ-Sätze im englischen Text.
- **"([a-zA-Z]+)@([a-zA-Z]+)\.([a-zA-Z]{2,4})"**
  - Findet E-Mailadressen ohne Zahlen
- **"^[01]\*\$"**
  - Findet alle Zeilen, die nur aus Nullen und Einsen bestehen.



# Reguläre Ausdrücke

## Übung 2

"^From:" {list your mail}

...

"[a-zA-Z]" {any line with at least one letter}

...

"[^a-zA-Z0-9]" {anything not a letter or number}

...

"[0-9]{3,5}-[0-9]{4,10}" {0681-84321, like phone numbers}

...

"^.\$" {lines with exactly one character}

...

...

"\"\*wort\"\*" {'wort', with or without quotes}

...

"^\\. \*? \$" {any line that starts with a Period "."}

...

"^\\.[1-9]+[a-z]" {line start with "." atleast one number and a letter}

...

# Testumgebung C# (Match)

```
using System;
using System.Text.RegularExpressions;
class Reginput {
    static void Main(string[] args) {
        Regex pattern = new Regex("M(a|e)(y|i)er");
        string input = "Mayer, Maier, Meyer";
        Match match = pattern.Match(input);
        while(match.Success) {
            Console.WriteLine(match.Groups[0]);
            match = match.NextMatch();
        }
    }
}
```

//Bemerkung: match.Groups[1] = Ergebnis der ersten Gruppierung

# Testumgebung C# (MatchCollection)

```
using System;
using System.Text.RegularExpressions;
class Reginput {
    static void Main(string[] args) {
        Regex pattern = new Regex("M(a|e)(y|i)er");
        string input = "Mayer, Maier, Meyer";
        MatchCollection matches = pattern.Matches(input);
        foreach(Match match in matches)
        {
            Console.WriteLine(match.Groups[0]);
        }
    }
}
```

//Bemerkung: match.Groups[1] = Ergebnis der ersten Gruppierung

# Testumgebung Web

<http://regexpal.com/>

<http://regex101.com/>

<https://regexr.com/>

<http://www.freeformatter.com/regex-tester.html>

<http://debuggex.com>

# Matchen über mehrere Zeilen

- Problem: \n bzw. \r wird als Ende erkannt.
- Ausweg:
  - Ergänze Parameter `Pattern.MULTILINE`
  - Nutze `[\s\S]` anstatt `.`
  - Nutze Non-greedy-Operator `*?` anstatt `*`

```
Regex pattern = new Regex(@"a[\s\S]*?b", RegexOptions.Multiline);
string input = @"a
ab
abc";
Match match = pattern.Match(input);
while (match.Success) {
    Console.WriteLine(match.Groups[0]);
    match = match.NextMatch();
}
```

# Ersetzen von Ergebnissen

- **i-te Gruppe** wird durch **\$i** ersetzt
- **Beispiel Wörter tauschen: a-b -> b-a**  
**Suchausdruck: "(.)-(.)"** **Ersatzausdruck: "\$2-\$1"**

```
using System;
using System.Text.RegularExpressions;
class Reginput {
    static void Main(string[] args) {
        Regex pattern = new Regex("(.)-(.)");
        string replaceS = "$2-$1";
        string input = "a-b";
        Console.Write($"input:{input}");
        input = pattern.Replace(input, replaceS); //"b-a"
        Console.WriteLine($" Result:{input}");
    }
}
```

# Arbeitsauftrag

- Homepage Schule

- Überschriften: `<h1>....</h1>`, ..., `<h7>....</h7>`
- Links: `<a href=http://www.internet.de>Zum Internet</a>`
- Bilder `</img>`
  - Oder ``

- Tabellen Spalten tauschen

`<tr><td>...1...</td><td>...2...</td></tr>`

->

`<tr><td>...2...</td><td>...1...</td></tr>`

# Matchen in HTML-Tabellen

