Søg og erstat med RegExp

Begrebet RegExp præsenteres her, og du får et lille program så du selv kan eksperimentere med RegExp.

Skrevet den **07. Feb 2009** af **per1291** I kategorien **Programmering / JavaScript** | ★★★☆☆

Vha. RegExp kan du forkorte din kode og samtidig gøre den hurtigere.
RegExp tillader dig at skrive en tekstgenkendelseskode på en enkelt linje, hvor du før skulle bruge mange linjer.

RegExp er et af de mere "langhårede" (avancerede) emner inden for Javascript.
Med denne lille guide vil jeg gerne videregive mine foreløbige erfaringer med denne teknik.
En stor del af denne artikel bygger på råd og anvisninger, som jeg har hentet fra eksperten.dk.
Tak til de rigtig mange eksperter, som har bidraget gennem svar på mine og andre brugeres spørgsmål.
Jeg har valgt at udelade eksempler, som kræver indsigt i andre "langhårede" emner: anonyme funktioner, funktioner med navngivne returværdier og den slags; disse emner fortjener selvstændig behandling.

RegExp kan bruges til både VALIDERING, SØGNING og REDIGERING.
En søgemaskine, som kan læse en RegExp, er hurtigere at bruge end en traditionel søgemaskine.
Søgeteksten "kompet[ae]nce" vil således erstatte 2 forskellige søgninger, nemlig søgning efter "kompetance" og "kompetence".

Det lille program i bunden af artiklen demonstrerer, hvordan du vha. RegExp kan FINDE et søgeord og ERSTATTE det med et andet.

> SØGEANVISNINGER: **i**, **g** og **m**

Sådan finder du ordet "mand" i teksten "Men så kom manden hjem!":

```
function findTekst()
{
var inp="Men så kom manden hjem!";
var reg=new RegExp("mand");
var erstat="MATCH";
var out=inp.replace(reg,erstat);
alert(out); /* Resultat: out="Men så kom MATCHen hjem!" */
}
```

Sådan finder du flere forekomster af ordet mand i en tekst:

```
function findTekst()
{
var inp="Manden arbejder som murerarbejdsmand.";
var reg=new RegExp("mand","ig");
var erstat="MATCH";
var out=inp.replace(reg,erstat);
alert(out); /* Resultat: out="MATCHen arbejder som murerarbejdsMATCH." */
```

}

De 2 ekstra søgeanvisninger i og g sørger for, at vi finder begge matches; uden dem ville vi ikke finde et eneste match!

i betyder "ignorecase" (Ignorer uppercase og lowercase);
g betyder "global" (Find alle matches og ikke blot det første).

Der findes en tredje søgeanvisning:
m betyder "multiline" (Medregn linjeskift i din søgning).
Søgeanvisningen m vil i mange tilfælde være uden betydning, idet hele teksten alligevel behandles hvis g er brugt.

Sådan MARKERER du hver enkelt linje i en tekst:
Kør testprogrammet i bunden af artiklen.
Indtast følgende i feltet Inputtekst, idet du trykker på enter-tasten efter første og anden linje:

Manden arbejder
hver dag
som murerarbejdsmand.

Indtast i feltet Søgestreng: ^(.+?)$
Indtast i feltet Erstat_med: [$1]
Klik på "SØG og ERSTAT".
Prøv igen, idet du fjerner fluebenet i m.

Sådan ISOLERER du tekstens enkelte linjer i et array:

```
var Linjer;
function isolerLinjer(Inputtekst)
{
var reg=new RegExp("^(.+?)$",gm);
Linjer=Inputtekst.match(reg);
for (var i=0; i<Linjer.length; i++) alert(Linjer);
}
```

- - -

Udtrykket "new RegExp" kan forkortes.
Således kan linjen
var reg=new RegExp("mand","igm");
også skrives sådan her:
var reg=/mand/igm;

Jeg vil bruge denne korte skrivemåde i resten af artiklen.

ANTAL FOREKOMSTER af et match

{2,5} betyder "mindst 2 og højst 5 forekomster".
{6} betyder "præcis 6 forekomster".
* er en forkortelse for {0,} ("nul eller flere forekomster")
+ er en forkortelse for {1,} ("1 eller flere forekomster")
? er en forkortelse for {0,1} ("nul eller 1 forekomst")

Sådan VALIDERER du, om en indtastet tekst er et 6-cifret hexadecimalt tal (eksempelvis #00ffCC):

```
function validerHex(Inputtekst)
{
var reg = /#[0-9a-f]{6}/i;
return (Inputtekst == Inputtekst.match(reg);
}
```

Forklaring:
Funktionen tester, om en indtastning består af tegnet # efterfulgt af 6 forekomster af tegn fra en nærmere defineret tegnmængde. En tegnmængde defineres vha. de firkantede parenteser ([og]). Disse tegn skal enten være cifre (0-9) eller bogstaver mellem a og f (a-f). Det kan dog også være bogstaver mellem A og F, hvilket søgeangivelsen "i" fortæller.

Sådan VALIDERER du, om en indtastet tekst er et heltal:

```
function validerHeltal(Inputtekst)
{
var reg=/[0-9]+/;
return (Inputtekst == Inputtekst.match(reg));
}
```

Forklaring:
Funktionen tester, om en indtastning består af 1 eller flere cifre.

GRÅDIGHED og DOVENSKAB

Sådan finder du samtlige HTML-tags i en tekst:

```
var found;
function findHTMLtags(Inputtekst)
{
var reg=/<.*?>/g;
found=Inputtekst.match(reg);
for (var i=0; i<found.length; i++) {alert(found[i])};
}
```

Lad os sige at Inputtekst="Det er <b>her[/b] du skal klikke";
Funktionen vil så sætte found lig med et ARRAY af samtlige matches, nemlig found[0]='<b>' og found[1]='[/b]'.

Forklaring:
Det regulære udtryk <.*?> vil matche enhver tegnfølge, som består af et < efterfulgt af 0 eller flere tegn afsluttet af det først forekommende >.

Men hvorfor skal vi egentlig have spørgsmålstegnet med?

Forklaringen er, at * i sig selv vil give "greedy matching"; maskinen vil finde det længst mulige match.
Prøv at udelade spørgsmålstegnet og i stedet skrive:
var reg=/<.*>/g;
så vil funktionen sætte found[0]='<b>her[/b]';

Tegnkombinationen *? vil derimod give "lazy matching"; maskinen vil finde det kortest mulige match.

Operatorerne + samt de krøllede parenteser er "grådige" ligesom * - men kan gøres "dovne" vha. spørgsmålstegnet.

META-TEGN og SPECIALTEGN

Disse "meta-tegn" har speciel betydning i en RegExp:

[(firkantet startparentes) bruges til at angive en tegnmængde; udtrykket afsluttes med].
] (firkantet slutparentes) - se forrige linje.
\ (backslash) bruges til at angive specialtegn og "normalisere" meta-tegn.
^ (caret) matcher positionen "linjestart"; men inde i firkantede parenteser betyder det "IKKE".
$ (dollartegn) matcher positionen "linjeslut".
. (punktum) matcher ethvert tegn.
| (lodret streg) betyder "eller".
? (spørgsmålstegn) betyder "0 eller 1 forekomst".
* (gangetegn) betyder "0 eller flere forekomster".
+ (plustegn) betyder "1 eller flere forekomster".
((rund startparentes) bruges til at sammenbinde deludtryk; senere kan man bruge disse deludtryk ved "back-reference"; udtrykket afsluttes med).
) (rund slutparentes) - se forrige linje.

(OBS: Mine eksperimenter viser, at procenttegnet % tilsyneladende også har en særlig betydning - men hvilken?)

Somme tider har man brug for i en RegExp at referere til disse tegn som TEGN (og altså IKKE bruge dem som "meta-tegn"). Så skal det pågældende tegn "escapes" eller "normaliseres" ved at man sætter en backslash foran det: [skrives \[osv.

Vi kan efterprøve, at hver af disse escape-strenge fortolkes som almindelige tegn, ved at køre programmet og f.eks. sætte Inputtekst="[" og Søgetekst="\[".

- - -

Disse "special-tegn" er defineret i browserens RegExp-maskine:

\A (start of string) matcher positionen før tekstens allerførste tegn.
\b (word boundary) matcher en position imellem et "word char" og et "ikke-word char".
\B (NOT word boundary) matcher enhver position hvor \b ikke matcher.
\c (control char) - desværre kan jeg ikke finde eksempler.
\d (digit) matcher ethvert ciffer - er en forkortelse for tegnmængden [0-9].
\f (formfeed) matcher formfeed-tegnet.
\n (newline) matcher ethvert linjeskift.
\r (return) bruges af Windows sammen med \n, idet linjeskift hedder \r\n.
\s (whitespace) matcher [\n\t\v].
\t (tabulator) matcher det almindelige tabulatortegn.
\v (vertical tab) matcher vertikalt tabulatortegn.
\w (word char) matcher ethvert tegn, som tilhører tegnmængden [a-zA-Z0-9_]
\W (NOT word char) matcher ethvert tegn, som IKKE tilhører tegnmængden [a-zA-Z0-9_]
\0 (octal char) - f.eks. er \40 den oktale betegnelse for mellemrumstegnet.

Vi kan efterprøve, at hvert af disse tegn fortolkes som specialtegn, ved at køre programmet og f.eks. sætte Inputtekst="abc" og Søgetekst="\b".

Andre RegExp-maskiner (f.eks. Perl) har desuden disse specialtegn:
\a (alarm/bell), \e (escape), \E (end lowercase|uppercase), \l (lowercase nextchar), \L (lowercase till \E), \N (named char), \Q (quote meta till \E), \u (uppercase nextchar), \U (uppercase till \E), \x (hex char) og \Z (end of string).

Vi kan efterprøve, at ingen af disse tegn fortolkes som specialtegn, ved at køre programmet og f.eks. sætte Inputtekst="abc" og Søgetekst="\a".

(OBS: Måske er der andre "special-tegn" end de nævnte. I så fald har jeg ikke fundet dem!)

BACK-REFERENCE

Runde parenteser tjener 2 formål: For det første binder man nogle anvisninger sammen i en logisk helhed, for det andet kan man senere referere til de fundne matches.

Den første parentes i en RegExp kaldes $1, den anden parentes kaldes $2 osv op til $9.

Sådan ERSTATTER du alle forekomster af <b> og [/b] med og :

```
function changeBtoStrong(Inputtekst)
{
var reg=/<(/?)b>/ig;
var erstat="<$1strong>";
return Inputtekst.replace(reg,erstat);
}
```

OBJEKTET **RegExp**

Objektet RegExp er en global variabel, som på ethvert tidspunkt vil indeholde oplysninger om det sidst benyttede (korrekte) regulære udtryk.
Regulære udtryk bruges i forbindelse med String-metoder som match, exec, replace, split, test, search og indexOf.
Der er ikke plads til at forklare alle disse metoder her.
RegExp indeholder de fundne matches - som du derefter kan referere til.
Næste gang du bruger en af disse String-metoder, vil RegExp blive overskrevet, og nu er det de nye matches du kan referere til.

Sådan FINDER du det første HTML-tag i en tekst:

```
function findHTMLtag(Inputtekst)
{
var reg = /(<.*?>)/;
reg.exec(Inputtekst);
alert (RegExp.$1);
}
```

FLERE PRAKTISKE EKSEMPLER

Sådan SLETTER du alle forekomster af <b>, [/b], <i>, , og :

```
function deleteFormtags(Inputtekst)
{
var reg = /</?(b|i|u)>/ig;
var erstat="";
return Inputtekst.replace(reg,erstat);
}
```

Sådan SLETTER du alle HTMLtags:

```
function deleteTags(Inputtekst)
{
var reg = /<[^>]+>/g; /* Husk at ^ inde i firkantede parenteser betyder IKKE */
var erstat="";
return Inputtekst.replace(reg,erstat);
}
```

Sådan SLETTER du alle forekomster af (enkelt eller dobbelt) citationstegn:

```
function deleteQuotes(Inputtekst)
{
var reg=/["']/g;
var erstat="";
return Inputtekst.replace(reg,erstat);
}
```

Sådan SLETTER du indledende og afsluttende "whitespace" fra teksten:

```
function removeSurroundingspace(Inputtekst)
{
var reg=/(^\s+|\s+$)/g;
var erstat="";
return Inputtekst.replace(reg,erstat);
}
```

Sådan ERSTATTER du overflødigt "whitespace" med enkelte space-tegn:

```
function replaceWhitespace(Inputtekst)
{
var reg=/\s+/g;
var erstat=" ";
return Inputtekst.replace(reg,erstat);
}
```

Sådan VALIDERER du, om en indtastet tekst er en mulig emailadresse:

```
function validEmail(Inputtekst)
{
var reg=/.*@\w[\w\.-]*\.[a-z]{2,6}/i;
return (Inputtekst == Inputtekst.match(reg));
}
```

SLUT PÅ ARTIKEL - lidt om programmet:

Ved udskrift af en tekst (fx på skærmen) vil browseren FORTOLKE tegnene, hvilket ikke altid er hensigtsmæssigt.
For at undgå dette formaterer jeg mine tekststrenge før udskrift.

Der kommer to alertbokse når du kører programmet.
I den første alertboks er Inputtekst formateret således at alle whitespace-tegn (f.eks. blanke og linjeskift) er erstattet af en firkant. Bemærk at i Windows indeholder et linjeskift 2 tegn, nemlig \r\n.
I den anden alertboks er Outputtekst formateret på samme måde.

Du kan selvfølgelig fjerne de to alerts, hvis du ikke har brug for dem.

En RegExp-streng skal i sig selv være skrevet korrekt, ellers får man en kørselsfejl når man bruger den.
I mit program under artiklen vil funktionen handleError blive igangsat hvis der sker en kørselsfejl.
Denne funktion går ud fra, at en indtastet RegExp-streng ikke er korrekt.
Vær opmærksom på, at hvis du ændrer i programmet bør du SLETTE linjen
window.onerror = handleError;
... indtil du er sikker på at dit program i øvrigt fungerer.

Du kan sikkert godt få programmet til at lave noget uhensigtsmæssigt, hvis du virkelig anstrenger dig.
Men til husbehov fungerer det.

OG HER ER PROGRAMMET SÅ:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Søg og erstat med RegExp</title>

<script language="Javascript" type="text/javascript">

function killTags(inpString)
{
inpString=inpString.replace(/(<|&)/g,"$1<q></q>"); /* så htmltags og tegnkoder ikke fortolkes */
inpString=inpString.replace(/ /g," "); /* så blanke ikke udelades på skærmen */
return inpString;
}

function formatOutput(outString) /* så strengen ser rigtig ud på skærmen */
{
outString=killTags(outString);
return outString.replace(/\r?\n/,"<br>");
}

function formatReplacetekst(changetekst)
{
changetekst=killTags(changetekst);
return changetekst.replace("\"","\\\"");
}

function formatAlert(inpString)
{
```

```
return inpString.replace(/\s/g,"\177");
}

function clearResults()
{
document.getElementById("javascriptTekst").innerHTML="";
document.getElementById("outputTekst").innerHTML="";
}

function erstatTegn()
{
var Inputtekst=document.getElementById("inputTekst").value;
clearResults();
alert("Inputtekst (længde="+Inputtekst.length+"):\n"+formatAlert(Inputtekst));
var regExpression=document.getElementById("searchTekst").value;
var searchType="";
if (document.getElementById("ignorecase").checked) {searchType+="i"};
if (document.getElementById("global").checked) {searchType+="g"};
if (document.getElementById("multiline").checked) {searchType+="m"};
var showReg=new RegExp(killTags(regExpression),searchType);
var regExpression=new RegExp(regExpression,searchType);
var changeTekst=document.getElementById("replaceTekst").value;
var scriptTekst="var reg="+showReg+";<br>";
scriptTekst+="var erstat=\""+formatReplacetekst(changeTekst)+"\";<br>";
scriptTekst+="Outputtekst=Inputtekst.replace(reg,erstat);";
document.getElementById("javascriptTekst").innerHTML=scriptTekst;
var Outputtekst=Inputtekst.replace(regExpression,changeTekst);
alert("Outputtekst (længde="+Outputtekst.length+"):\n"+formatAlert(Outputtekst));
document.getElementById("outputTekst").innerHTML=formatOutput(Outputtekst);
}

function handleError()
{
var errorTekst="ERROR:<br>Din søgestreng er ikke en korrekt RegExp -<br>og vil give fejl, hvis du
bruger den i et program.";
document.getElementById("javascriptTekst").innerHTML=errorTekst;
return true;
}

window.onerror = handleError;

</script>

<style type="text/css">
body { font-family:arial,verdana,helvetica,sans-serif; font-size:16px; }
h1 { font-size:20px; }
table { background-color:white; }
td { align:left; background-color:cyan; }
#BodyDiv { width:680px; }
#inputTekst,#searchTekst,#replaceTekst,#outputTekst { font-family:Courier, MS Courier New,
monospace; font-size:14px; }
#javascriptTekst { background-color:yellow; font-family:Courier, MS Courier New, monospace; font-
weight:bold; font-size:14px; }
</style>
</head>
```

```
<body bgcolor="cyan">
<center>
<div id="BodyDiv">
<h1>Søg og erstat med RexExp</h1>
<table border="1">
<tr><td><b>Inputtekst</b></td>
<td><textarea rows=2 cols=80 id="inputTekst"></textarea></td></tr>
<tr><td><b>Søgestreng (RegExp)</b></td>
<td><input type="text" size="80%" id="searchTekst"></td></tr>
<tr><td><b>Searchmode</b></td>
<td>
  <table width="100%" cellspacing="0"><tr>
  <td align="center">i = ignorecase <input type="checkbox" id="ignorecase" checked="true"></td>
  <td align="center">g = global <input type="checkbox" id="global" checked="true"></td>
  <td align="center">m = multiline <input type="checkbox" id="multiline" checked="true"></td>
  </tr></table>
</td>
</tr>
<tr><td><b>Erstat med</b></td>
<td><input type="text" size="80%" id="replaceTekst" value="MATCH"></td></tr>
<tr><td><b>Udfør[/b]</td>
<td align="center">
  <button onclick="erstatTegn()">SØG og ERSTAT</button>  
</td></tr>
<tr><td><b>Scriptkode[/b]</td>
<td id="javascriptTekst" width="100%" align="left"> </td></tr>
<tr><td><b>Outputtekst[/b]</td>
<td id="outputTekst" align="left"> </td></tr>
</table>
[/div]
</center>
</body>
</html>
```

Kommentar af roenving d. 04. Jan 2006 | 1

En rigtig god gennemgang !-)
-- og busschous bemærkning om at RegExp sluger kræfter er jeg også stødt ind i, men en umiddelbar vurdering vil være, at almindelige operationer er så lidt omkostningskrævende, at browseren har masser af kræfter, mens koden bliver adskilligt nemmere at skrive og gennemskue !o]

Kommentar af coderdk d. 04. Jan 2007 | 2

Lækkert! Lad os få udbredt kendskabet til regular expressions ;) Jeg fik hovedpine af din kode; indentér den! ;)

Kommentar af busschou d. 08. Sep 2005 | 3

Du starter med at skrive følgende linje - Citat start - Vha. RegExp kan du forkorte din kode og samtidig gøre den hurtigere. - Citat slut.
Jeg har hørt brugere herinde tale om at RegExp er en "sluger". Jeg har ingen dokumentation herom, men omvendt savner jeg dokumentation for din påstand om at RegExp kan gøre mine ting hurtigere

Kommentar af wicez (nedlagt brugerprofil) d. 07. Sep 2005 | 4

Rigtig god artikel!