Math

Sommige methoden van Math kunnen van pas komen in alledaagse situaties, zoals:

- `Math.round()` die afrondt naar het dichtstbijzijnde gehele getal
- `Math.max()` en `Math.min()` die het maximum of minimum van twee of meer getallen afleveren.

Zie w3schools.com - math en w3schools.com – obj_math. Bestudeer ook pagina 789 t/m 800 van Flanagan voor de referentie over Math.

Reguliere expressies

Reguliere expressies gebruikt u om in een tekst een bepaald teken of een bepaald patroon te vinden. Vooral bij communicatie met de gebruiker is het vaak essentieel de invoer goed te controleren. Bestaat het ingevoerde telefoonnummer wel uit tien cijfers? Is de datum correct ingevuld: 17-10-2017?

Een reguliere expressie kan helpen om in een tekst (een string) bepaalde gegevens te zoeken. Zo'n expressie kan uit een enkel teken bestaan, of veel ingewikkelder zijn.

```
var s =
"Ik ben geboren uit zonnegloren. En een zucht van de ziedende zee.";
```

Om de volgende voorbeelden makkelijker te kunnen volgen, schrijven we onder elk symbool zijn rangnummer (index) in de string:

```
Ik ben geboren uit zonnegloren. En een zucht van de ziedende zee.
0123456789112345678921234567893123456789412345678951234567896 1234
```

De vetgedrukte cijfers geven 0, 10, 20 et cetera aan. De nummering begint bij 0, de string heeft 65 tekens.

De methode search()

Laten we eens zoeken naar de lettercombinatie `en` in de string uit de vorige paragraaf. Als zo'n combinatie voorkomt, heet dat een *match*. Het zoeken kan met behulp van de methode `search()` en de reguliere expressie `/en/`.

```
<p id="demo"></p>
<script>
var s =
"Ik ben geboren uit zonnegloren. En een zucht van de ziedende zee.";

var n = s.search( /en/ );
document.getElementById("demo").innerHTML = n;
</script>
```

Dit levert de uitvoer 4.
De methode search() stopt als hij de eerste match heeft gevonden en levert -1 af als er geen match is.

Vervangen kan met replace(). De methode levert de gewijzigde string af.

```
var res = s.replace( /en/, "xx" );
```

Het resultaat is:

```
Ik bxx geboren uit zonnegloren. En een zucht van de ziedende zee.
```

De methode replace() stopt na de eerste vervanging. Door de modifier `g` (global) achter de reguliere expressie te zetten, wordt elke match vervangen:

```
var res = s.replace( /en/g, "xx" );
```

Het resultaat is nu:

```
Ik bxx geborxx uit zonneglorxx. En exx zucht van de ziedxxde zee.
```

Merk op dat het woord *En* niet vervangen is. Het zoeken en vervangen is case sensitive. Als u dat niet wilt, gebruikt u de modifier `i` (ignore case):

```
var res = s.replace( /en/gi, "xx" );
```

Dit levert:

```
Ik bxx geborxx uit zonneglorxx. xx exx zucht van de ziedxxde zee.
```

Met de methode `match()` kunnen alle matches worden gevonden, mits u een globale zoekopdracht geeft. De matches komen in een array terecht. De lengte van de array is dus het aantal matches. Als er geen matches zijn, levert de methode de waarde `null`.

```
var a = s.match( /en/gi );
```

Het resultaat is (de inhoud van `a`):

```
en,en,en,En,en,en
```

En `a.length` heeft dus de waarde 6.

De methode split()

De methode `split()` splitst een string in meerdere strings op grond van een of meer scheidingstekens die u als argument van de methode meegeeft en bergt de onderdelen van de gesplitste string op in een array.

Een voorbeeld:

```
<p id="demo"></p>
<script>
  function schrijfArray( a ) {
    var uitvoer = "";
    for( var i = 0; i < a.length; i++ ) {
      uitvoer += a[i] + "<br>";
    }
    return uitvoer;
  }
  var str = "Hoe gaat het?";
  var res = str.split(/ /);            // scheidingsteken is spatie
  document.getElementById("demo").innerHTML = schrijfArray(res);
</script>
```

In dit voorbeeld wordt de string "Hoe gaat het?" gesplitst. Het scheidingsteken is een spatie, die als reguliere expressie in de aanroep staat: `str.split(/ /);`
De uitvoer is:
Hoe
gaat
het?

Als er in de string meer spaties (bijvoorbeeld drie) staan tussen de woorden, zoals in: "Hoe   gaat het?", dan levert hetzelfde script deze uitvoer:
Hoe
{lege regel}
{lege regel}
gaat
het?

Tussen de drie spaties staan twee lege strings en deze komen in de uitvoer. Dit is niet altijd gewenst. Met een reguliere expressie is te voorkomen dat dit gebeurt, en wel met het plusteken. Het plusteken betekent: een of meer van de voorgaande. Dus `str.split(/ +/);` scheidt op een of meer spaties.
De uitvoer wordt dan:
Hoe
gaat
het?

De punt als scheidingsteken

In webadressen komt de punt als scheidingsteken voor. Deze kunt u in een reguliere expressie vinden met backslash gevolgd door een punt: `\.`
De punt heeft in een reguliere expressie de betekenis van een willekeurig teken (behalve newline), dus \. geeft aan dat u echt naar de punt wil zoeken. Alternatief is de punt niet als reguliere expressie, maar als string in te voeren:

```
var str = "www.loi.nl/test";
var res = str.split(/\./);    // of: str.split(".");
```

Dit levert:
www
loi
nl/test

De laatste string kunt u desgewenst splitsen met `str.split(/\//)` of met `str.split("/")`.
Ook hier geldt weer dat in de reguliere expressie een backslash voor de slash staat, omdat de slash in een reguliere expressie een speciale betekenis heeft.

Lees meer op w3schools.com - regexp.

Het RegExp-object

Zoals in JavaScript een string (een tekst tussen aanhalingstekens) een string-object is, zo is een rijtje tekens tussen `/` en `/` een RegExp-object. Deze objecten beschikken over een belangrijke methode die `test()` heet. Hiermee kunt u testen of een bepaalde string aan een het patroon voldoet van het betreffende RegExp-object. De methode `test()` levert `true` of `false`, afhankelijk van het feit of het patroon gevonden is of niet.

Een helder overzicht van de mogelijkheden waarmee u patronen kunt maken vindt u op w3schools.com – obj_regexp. Indien u in de tabellen op deze pagina op een van de items in de linkerkolom klikt, krijgt u over dat item meer informatie en voorbeelden.

Ook hier een paar voorbeelden:

RegExp-object `r`	Betekenis	`s1`	`r.test(s1)`	`s2`	`r.test(s2)`
`/d/`	Kijk of letter d voorkomt.	`"abcd"`	`true`	`"klm123"`	`false`
`/\d/`	Kijk of een cijfer (digit) voorkomt.	`"abcd"`	`false`	`"klm123"`	`true`
`/[a-z]/`	Kijk of een kleine letter voorkomt.	`"123"`	`false`	`"klm123"`	`true`
`/[a-zA-Z]/`	Kijk of een kleine of hoofdletter voorkomt.	`"123"`	`false`	`"KLM"`	`true`
`/a-c/`	Kijk of de combinatie a-c voorkomt.	`"123"`	`false`	`"12a-cM"`	`true`
`/\d{3}/`	Kijk of er drie cijfers achter elkaar voorkomen.	`"123"`	`true`	`"12a-cM"`	`false`
`/^\d{2}/`	Kijk of aan het begin van de string twee cijfers voorkomen.	`"a123"`	`false`	`"12a-cM"`	`true`
`/\d{2}$/`	Kijk of aan het einde van de string twee cijfers voorkomen.	`"a123"`	`true`	`"12a-cM"`	`false`

Over reguliere expressies valt nog veel meer te zeggen en ze kunnen knap ingewikkeld worden. Daar staat tegenover dat ze ook heel krachtig kunnen zijn. Zie hier voor een voorbeeld en hier nog een voorbeeld.

De methoden indexOf(), lastIndexOf() en substring()

Er zijn ook methoden die niet met reguliere expressies werken, zoals `indexOf()` en `lastIndexOf()`. Ze leveren respectievelijk de eerste index en laatste index van een te zoeken string.

```
var s =
"Ik ben geboren uit zonnegloren. En een zucht van de ziedende zee.";
var indexEerste = s.indexOf( "en" );
var indexLaatste = s.lastIndexOf( "en" );
```

Dit levert de waarden 4 en 56.

Andere nuttige methode is `substring()`. Deze methode kopieert een gedeelte uit een string. De methode heeft twee parameters: de index van het begin en de index van een voorbij het einde van het gedeelte dat uit de string gekopieerd moet worden.
Stel dat de string `geboren` uit de volgende string gekopieerd moet worden:

```
Ik ben geboren uit zonnegloren.
0123456789112345678921234567893
```

Het woord ‘geboren’ begint op index 7 en voorbij het einde van ‘geboren’ is index 14.
In het volgende fragment is dit toegepast:

```
<p id="demo"></p>
<script>
var s = "Ik ben geboren uit zonnegloren.";
var subs = s.substring( 7, 14 );
document.getElementById("demo").innerHTML = subs;
</script>
```

In een realistische situatie weet u bij een string met onbekende inhoud natuurlijk meestal niet waar een woord begint of eindigt. Als er echter wel iets over de string bekend is – zoals het feit dat woorden van elkaar gescheiden zijn door een spatie al dan niet voorafgegaan door een punt of komma – dan kan in principe met de methode `search()` de index van het begin en van het eind gevonden worden. Langere strings kunnen eventueel eerst met `split()` gesplitst worden, om dan de kortere strings te onderzoeken.