Arrays og deres slægtskab med objects

Grundlæggende beskrivelse af arrays, deres oprettelse og brug. For den lidt mere øvede er der også en snak om associative arrays og deres lighed med objects. Der behandles kun endimensionale arrays!

Skrevet den **03. Feb 2009** af **olebole** I kategorien **Programmering / JavaScript** | ★★★☆☆

I JavaScript (herefter benævnt 'JS') er vi blevet begavet med en temmelig god implementering af arrays - et herligt værktøj til transport og opbevaring af data.

I denne lille artikel vil jeg forsøge at gøre rede for forskellige metoder til at oprette arrays, samt - for den mere erfarne JS'er - belyse slægtsskabet mellem objekter og arrays i JS.
Efter en gennemgang for begynderen skriver jeg nogle små kodeeksempler og her vil det være en god idé at skrive eksemplerne ind i en HTML-fil og afprøve dem ... 'hands-on' er altid en god måde at lære på ;o)

Hvis vi lige tager begrebet 'arrays' helt fra Adam og Eva, så den grønne JavaScripter også kan være med, så vil jeg lige beskrive et array med et billede, der intet har med en PC at gøre:
Vi tager fire skotøjs-æsker og binder en stump snor imellem dem. I den ene ende binder vi en længere snor i, så vi kan trække den efter os, som et andet 'fut-tog' - hvilket kunne se sådan ud:

```
-------|__|--|__|--|__|--|__|
```

Hvis vi så forstiller os et sæt script-tags som et cykelværksted, kan vi se på JS-funktionerne som mekanikere, der hver har deres specielle funktion på værkstedet.

Nu, da vi har lavet vores lille fut-tog, kan vi hjælpe til på værkstedet ved at trække toget rundt fra mekaniker til mekaniker. Så kan Ib lægge en ringeklokke ned i skotøjs-æske nummer et. Vi kan derefter trække toget over til Bo, som lægger en seddel til Bent ned i æske nummer tre, osv, osv.

På den måde bidrager vi til infrastrukturen på værkstedet. Man kan lægge beskeder og cykelstumper til hinanden ned i toget, lade det stå som opbevaring, eller trække det rundt til de andre ansatte - og alle ansatte kan tage noget op af toget og/eller lægge noget nyt i det. Samtidig har man mulighed for at binde ekstra vogne på toget - og klippe nogle af, hvis man får brug for det.

I JS findes et sådan tog, der kan transportere data rundt mellem funktioner og/eller lagre dem til senere brug. Det hedder et array (en liste) og de enkelte 'vogne' benævnes 'elementer'.

Constructor'en for et array ser sådan ud:

```
var myArray = new Array();
```

og man fylder data i arrayet på denne måde:

```
```

```
myArray[0] = "noget";
myArray[1] = "noget andet";
myArray[2] = "noget tredie";
```

Læg mærke til, at det første element har nummer 'nul' (sådan nummereres arrays i JS og de fleste andre sprog). Ethvert element kan nu tilgåes via dets indeks på denne måde:

```
alert( myArray[1] );
```

hvilket vil returnere en alert-boks med teksten "noget andet".

Vi kan også fylde data i arrayet samtidig med, vi opretter det. Dette eksempel gør præcis det samme, som det foregående:

```
var myArray = new Array( "noget", "noget andet", "noget tredie" );
alert( myArray[1] );
```

Et lille eksempel på, hvordan vi kan bruge et array mellem funktioner (start med at læse fra funktions-kaldet i bunden - og op):

```
function tellMe(a) {
    alert( a[0] );  // Returnerer: "noget nyt"
}

function myFuncTwo(a) {
    a[0] = "noget nyt";  // Fyld nogle nye data i første element
    tellMe(a);
}

function myFuncOne() {
    var myArray = new Array( "noget", "noget andet", "noget tredie" );
    myFuncTwo(myArray);
}

myFuncOne();
```

En vigtig egenskab ved denne slags arrays er array'ets 'length' - dets antal af elementer:

```
var myArray = new Array( "noget", "noget andet", "noget tredie" );
alert( myArray.length );  // Returnerer: 3
```

Den property kan vi f.eks. bruge, hvis vi vil bladre et array igennem:

```
var myArray = new Array( "noget", "noget andet", "noget tredie" );
for ( i=0; i<myArray.length; i++ ) {
    alert( myArray[i] );
}
```

eller hvis vi ønsker at indsætte et ekstra element i slutningen af arrayet:

```
myArray[ myArray.length ] = "en værdi";
```

NB: Der findes godt nok en speciel array-funktion, der gør netop dette, men man skal passe lidt på med at bruge den, da den ikke er ens implementeret i alle browsere (f.eks. stinker IE/MacOS):

```
myArray.push("en værdi");
```

Læs mere om indbyggede array-funktioner bag de links, der står i slutningen af artiklen.

Den type arrays, vi indtil nu har set på, kaldes tal-indekserede arrays. I JS har vi også associative arrays, som er indekseret med en tekststreng i stedet for et tal ... vi knytter en tekst-båret association til elementet:

```
var myAssocArray = new Array();

myAssocArray["et"] = "noget";
myAssocArray["to"] = "noget andet";
myAssocArray["tre"] = "noget tredie";
```

Her kan vi kalde et array-element med en tekst-streng:

```
alert( myAssocArray["to"] );  // Returnerer: "noget andet"
```

NB: Associative arrays' length-property er altid nul, hvorfor dette eksempel blot returnerer en alert-boks med '0' - uanset hvormange elementer, der ligger i array'et:

```
alert( myAssocArray.length );
```

Vi kan altså ikke bladre et associativt array i gennem med en alm. for-løkke, som vi gjorde det med det tal-indekserede array. Her må vi bruge en for/in-løkke:

```
for ( key in myAssocArray ) {
   alert( key );
   alert( myAssocArray[key] );
}
```

Nu har vi set på, hvordan vi kan oprette arrays og - ganske kort på - hvordan vi kan bruge dem. Der findes dog en anden constructor, det er værd at nævne ... short-hand constructor'en:

```
var myArray = [];
```

Dette udtryk opretter et array, og vi kan efterfølgende fylde elementer i det, som vi plejer.
Men vi kan også oprette arrayet med elementer i:

```
var myArray = [ "noget", "noget andet", "noget tredie" ];
```

Folk, der kommer fra andre sprog f.eks. PHP, spørger ofte: "Hvordan opretter jeg så et associativt array, der får tildelt elementer ved oprettelsen? I PHP kan jeg bare gøre sådan:"

```
<?
   $myAssocArray = array( "et"=>"noget", "to"=>"noget andet", "tre"=>"noget tredie" );
?>
```

Svaret lyder umiddelbart: "Det kan man ikke i JS." - men det er ikke helt korrekt :)
Godt nok findes der ikke en indbygget constructor i JS, der kan gøre det, men kikker vi lige ned under det lag, vi sidder og koder i, afsløres en yderst interessant ting!

Ser vi på, hvordan data optræder og behandles på en PC, kan vi tale om 'lag'. Det aller nederste lag er den såkaldte 'maskinkode', hvor alle data er repræsenteret ved ettaller og nuller. Herefter følger en række lag, hvor data-repræsentationen bliver mere og mere en sproglig repræsentation - og data bearbejdelsen bliver følgelig mere og mere en sproglig øvelse. Der, hvor vi befinder os med JS, er koden meget sproglig og meget abstraheret fra den oprindelige 'maskinkode'.

I laget lige under JS-laget er den data-konstruktion, vi kalder et 'array' - og den konstruktion, vi kalder et 'object' to repræsentationer for én og samme data-konstruktion.

Derfor beskæftiger vi os lige ganske kort på det data-objekt, der i JS hedder 'object'.

Et object kan oprettes med constructor'en:

```
var myObj = new Object();
```

På dette object kan vi nu 'klistre' forskellige properties (egenskaber) - ligesom vi gjorde det i det associative array:

```
var myObj = new Object();

myObj.et = "noget";
myObj.to = "noget andet";
myObj.tre = "noget tredie";

alert( myObj.to );  // Returnerer: "noget andet"
```

At man også kan 'klistre' metoder (funktioner) på et object vil vi i denne forbindelse springe over. Det agter jeg at diskutere i en kommende artikel om OOP (Objekt Orienteret Programmering) i JS :)

Forskellen på måden, vi kalder et element i et array og den måde, vi tilgår en property på et object på, ligger i notationen.
Den ene kaldes 'array-notation':

```
myAssocArray["et"]
```

- den anden 'dot-notation':

```
myObj.et
```

Da array og object som sagt er det samme underliggende data-objekt, kan vi endda kalde med array- eller dot-notation ved både associative arrays og objects:

```
myAssocArray["et"]
```

og:

```
myAssocArray.et
```

repræsenterer begge array'ets andet element.

```
```

```
    myObj["et"]
```

og:

```
    myObj.et
```

repræsenterer begge object'ets property med navnet 'et'.

Du kender det måske fra dokumentets forms-array:

```
    document.forms["myForm"].submit();
```

er det samme som:

```
    document.forms.myForm.submit();
```

Du kan f.eks. også bruge det på et HTML-elements style-object:

```
    HTML_ELEMENT.style.backgroundColor = "red";
```

og:

```
    HTML_ELEMENT.style["backgroundColor"] = "red";
```

gør det samme. Hvad det kan bruges til, skal jeg komme tilbage til.

Nu er det sådan, at også et object har en short-hand constructor:

```
    var myObj = { "et":"noget", "to":"noget andet", "tre":"noget tredie" };
    alert( myObj.to );  // Returnerer: "noget andet"
```

Her har vi vores PHP'ers redning - som vi andre så sandelig også kan benytte med udsøgt fornøjelse :)
Et associativt JS-array kan oprettes on-the-fly med object-constructoren:

```
    var myAssocArray = { "et":"noget", "to":"noget andet", "tre":"noget tredie" };
    alert( myAssocArray["to"] );  // Returnerer: "noget andet"
```

Vi går lige tilbage til HTML-elementet og dets style-object. Man ser ofte, folk skifter class på et HTML-element, når udseendet skal ændres. Det er som regel en temmelig skidt idé, da hele dokumentet derved

skal genberegnes og layout'es. Det er hurtigere at nøjes med at sætte de enkelte style-properties på elementet.

Vi kan oprette et associativt array med styles (om det skrives på én linie, eller som her på flere linier, gør ingen forskel):

```
var myStyles = {
   "color": "yellow",
   "backgroundColor": "red",
   "fontWeight": "bold",
   "fontStyle": "italic"
};
```

Derefter kan vi oprette en funktion med en for/in-løkke, der sætter de enkelte style-properties på et element:

```
function setStyles(elm) {
   for ( x in myStyles ) {
      elm.style[x] = myStyles[x];
   }
}
```

... hvilket bliver til:

```
<html>
<head>
<script type="text/JavaScript">
var myStyles = {
   "color": "yellow",
   "backgroundColor": "red",
   "fontWeight": "bold",
   "fontStyle": "italic"
};

function setStyles(elm) {
   for ( x in myStyles ) {
      elm.style[x] = myStyles[x];
   }
}
</script>
</head>
<body>

<div onclick="setStyles(this)" >... Klik her ...</div >

</body>
</html>
```

Den lidt mere erfarne scripter vil måske kunne få idéer i retning af brugerdefinerede temaer, der bestemmes udfra, hvilken JS-fil, der loades ... og meget andet andet godt.
Under alle omstændigheder er arrays et uundværligt og fantastisk effektivt værktøj i JavaScript ... god fornøjelse ;o)
/mvh

Relevante links:

Netscape (JavaScript):
Array: http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Literals#Array_Literals
Object: http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Literals#Object_Literals

Microsoft (JScript):
Array: http://msdn2.microsoft.com/en-us/library/k4h76zbx.aspx
Object: http://msdn2.microsoft.com/en-us/library/kb6te8d3.aspx

Rettet links d. 24. okt. 2007

Kommentar af whatever d. 17. Oct 2004 | 1

Kanon.

Kommentar af horsmark d. 13. Oct 2005 | 2

olebole er min helt (igen) skriver rigtigt godt og forklarende og går i dybden uden at tabe begynderne :-)

Kommentar af mik789 d. 15. Mar 2004 | 3

olebole er altid god for en kyndig JavaScript forklaring. Jeg ser frem til den næste artikel. Bare ærgeligt at han springer det med de multidiminsionale arrays over. For en logik-spasser som mig er det altid dem der giver de største vanskeligheder. Giv dem gerne en artikel for sig selv med eksempler. Sjovt det med short-hand objekt syntaksen anvendt som css: det ser jo ud som om css syntaksen faktisk er taget direkte fra denne. Noget andet: jeg mener da ikke man behøver anførselstegn om det assoc. arrays keys, kun om deres alues.

Kommentar af roenving d. 21. Oct 2004 | 4

Hov, den har jeg vist aldrig fået ratet, så den skal da lige have en top-karakter !-)

Kommentar af phoenixv (nedlagt brugerprofil) d. 07. Apr 2004 | 5

Som sædvanlig lykkes det Olebole at lave et grundigt stykke arbejde. Lang og god artikel, som dog nok spænder lidt for vidt på den - trods alt - lille plads. Men med de mange eksempler og særligt afsnittet om sammenhængen mellem arrays og objects er den alligevel til topkarakter.

Kommentar af skovenborg d. 17. Mar 2004 | 6

Dette er virkelig en god og meget fyldesgørende artikel - håber du vil lave flere af den slags :-)

Kommentar af johan.o d. 31. Jan 2006 | 7

Velskrevet, informativ og lærerig - tak.....men jeg savner lidt gnu'er og næbdyr :) /johan.o

Kommentar af wicez (nedlagt brugerprofil) d. 22. Apr 2005 | 8

Rigtig god og velskrevet artikel.

Kommentar af saucer d. 23. May 2004 | 9

nice artikel

Kommentar af el_barto (nedlagt brugerprofil) d. 06. Jan 2005 | 10

Kanon artikel, især tricket med shorthand på object-constructoren

c",)

Kommentar af softspot d. 11. Feb 2007 | 11

Super artikel! Som sædvanlig leverer olebole top-kvalitet!
Jeg kunne specielt godt lide den sidste pointe med en generisk funktion til at sætte styles på elementer - fed idé!

Kommentar af psykochicken d. 30. Nov 2006 | 12

Kanon artikel. Den sætter lige nogle ting på plads for en hjemmebygger som mig - thumbs up ;o)

Kommentar af per1291 d. 24. Jul 2005 | 13

Jeg var desværre nødt til at stå af på halvvejen, fordi der var for mange nye ting for mig. Men vil kigge på den igen i morgen. Meget pædagogisk.

Kommentar af slst d. 13. Jan 2006 | 14

Du' så smart olebole!

Kommentar af mclemens d. 28. May 2006 | 15

Den object contructor er super :)

Kommentar af jhe-ting d. 26. Nov 2006 | 16

Giver god inspiration at læse en velskrevet artikkel.
Det er eksemplarisk grundigt forklaret, dog forældes link's så her er et par nye:

http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Literals#Array_Literals

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/785c5acd-b8b3-4152-af9a-dd42ecdd75ba.asp?frame=true

der sikkert også hurtigt 'fordamper' ;)