Mer JS

syntax

Agenda

- Variabler
- Metoder
- Array
- Logiska uttryck
- loop

Språklig uppbyggnad

- Språket byggs upp av variabler, funktioner och objekt.
- **Variabler** är en mekanism för att lagra/spara värden/data.
- **Funktioner** en mekanism att exekvera flera rader kod i ett stycke.
- **Objekt** – en mekanism som har inbyggd funktionalitet.

variabler

- En mekanism för att lagra/spara data...
- Vi sparar data i variabel för att kunna använda samma data i exempelvis beräkningar, flytta data, arbeta med innehåll.
  Datat vi sparar i en variabel kan vara ‘vad som helst’.
- Syntax för variabler;
  var ***namnPåVariabel = värde***;
  (jmfr css -> property: value; eller html attribut=”värde”)
- I js kan vi skapa variabler på 2 sätt;

```
namn1 = 10;
var namn2 = “mamma”;
```

- *lägg märke till att varje rad avslutas med ;*

variabler

- Det som sparas i variabler kallas värden;

```
var myVar1 = 100;
var myVar2 = “Arne Karlsson”;
var myVar3 = document.title;
```

- Exempel;

html

```
<img src=”hej.jpg” />
```

js

```
var img = document.images[0];
img.style.width = “100px”;
```

Aritmetiska operatorer

- I JavaScript kan man göra beräkningar med hjälp av **operatorer:**
- **<u>Aritmetiska:</u> +, -, *, /, %, ++,**

```
var num = 5 + 5;//vad blir num?
```

- ```
  var a = 10;
  var b = 10;
  var c = a - b;//vad blir c?
  ```

- ```
  var d = 50;
  var e = d * c;//vad blir e?
  ```

- ```
  var f = 1;
  f = f + 1;//vad blir f?

  var g = 1;
  g++;//vad blir g?
  ```
  ```

Tilldelnings-, jämförelse-, och logiska operatorer

- **Tilldelning:** **`=, +=, =, *=, /=`**

```
var a = 10;

a = a + 1;//vad blir a?
a += 1;//vad blir a?

a -= 2;//vad blir a?
```

- Jämförelse: `==, !=, >, <, >=, <=`

```
if(a >= b)
```

- Logiska: &&, ||, !

```
If(a > b || a == b)
```

Strings - strängar

- Strängar kallas de värden som innehåller text;

```
var m = “mamma”;
var p = “pappa”;
var b = “barn”;
```

- När man lägger ihop en e.l flera strängar – konkatenering. Görs med **+**.

```
var family = m + p + b;// family är
“mammapappabarn”

var family2 = m + “ “ + p + “ och “ + b;
//family2 är “mamma pappa och barn
```

Öppna developer console

- I Chrome:
  Ctrl + shift + j

- I övriga browser’s:

  http://webmasters.stackexchange.com/questions/8525/how-to-open-the-javascript-console-in-different-browsers

- Metoder

Methods - metoder

- Metoder låter oss 'kapsla' in kod som vi kan välja att exekvera vid ett visst tillfälle och hur många gånger som helst.

- Att exekvera en metod kallas för att ***anropa*** en metod (på eng. call).

Syntax metoder

- Syntax;

```
function namnMetod()
{
   ...kod som hör till metoden...
}
```

- **function:** talar om att det är en metod
- **namnMetod:** valfritt namn, parenteser ska vara med
- Allt mellan { och } är kod som hör till metoden.

Exempel metod och anrop

- ```
  function changeTitle()
  {
    document.title =”titel på min fina sida”;
  }
  ```
- changeTitle kan läggas var som helst i ett script-avsnitt.
  Vi kan använda metoden när som och var som helst*.
- Anrop – kallas det när man använder en metod i koden.
  Ett anrop changeTitle ovan;
  `changeTitle();`
- ```
  function a() {
   alert(“Kex och soppa”);
  }
  ```

  Vad gör ovanstående metod?
  Hur ser ett anrop ut?

- *) nja…. Funktionen måste ha lasts in innan den anropas….

Övning

- Gör en funktion som får något att hända i webbläsaren/på sidan.

Parameters - parametrar

- Med parametrar i en metod kan vi skicka värden till metoden. Dessa fungerar som variabler i metoden.
- ```
  function myMethod(a, b, c)
  {
    var sum = a + b + c;
  }
  ```
  Hur anropas metoden?

- 1) **`myMethod(10, 10, 10);`**

  2)
  ```
  var x = 10;
  var y = 10;
  var z = 9;
  myMethod(x, y, z);
  ```
  ```

Exempel parametrar

- ```
  function setTitle(title)
  {
        document.title = title;
  }
  ```

  hur anropar vi denna metod?

- ```
  //kan anropas:
  setTitle("Min fina titel");
  ```
- ```
  //eller t ex
  var str = "Min nya titel";
  setTitle(str);
  ```
  ```

Övning

- Skriv en funktion som tar 3 stycken parametrar.
  De första två ska adderas och den sista ska subtrahera föregående resultat.
  Visa resultatet med någonting på sidan.

Return - returvärde

- Metoder kan returnera värden (från t ex beräkningar etc) med **`return`**. Detta returvärde kan vi använda/spara.
- ```
  function calculate(a, b, z)
  {
   var sum = a + b - z;
   return sum;
  }
  ```
- Möjliga anrop:
  ```
  calculate(10, 15, 5);
  var nbr = calculate(10, 20, 10);//vad sparas i nbr?
  ```
- ```
  var sum=calculate(1, 1, 1) + 10 - calculate(1,1,1);
  //Vad sparas i sum?
  ```

Övning

- Gör en funktion som hämtar ut text (innehållet) från ett element/tagg och returnerar det värdet.

  Funktionen ska ta en parameter, vilken ska vara id:t på det element som innehållet ska hämtas ifrån.

- Metoder är ett kraftfullt sätt att återanvända kod;
  - Placera metoder i en egen .js fil – återanvänd där du behvöer.
  - alternativt placera metoderna bara i head.

- Placera med fördel metoder i externa script – kan återanvändas.

- Tips med metoder:
  - om du gör en sak mer än två gånger – gör en metod av det!

Övning

- Gör en metod som kan returnera strängen ”Hello”.
- Gör en till metod som anropar föregående metod och konkatenerar mottagen sträng med strängen ”world”. Resultatet av konkateneringen ska returneras.

- Gör en tredje metod som tar en parameter.
  Denna metod ska nu anropa ’den andra metoden’ och konkatenera resultatet med parametern. Resultatet ska presenteras med en alert().

- Anropa den tredje metoden med ett argument.

Scope { }

- Scope = åtkomst, tillgänglighet.
- Allting i JS har ett visst scope; variabler, metoder, objekt etc.
- Betrakta följande (testa dem);

```
function foo() {
  var number = 10;
}

alert(number);

//vad händer????
```

```
var number = 10;

function foo() {
  number = number + 10;
}

foo();
alert(number);

//vad händer????
```

```
x = 10;

function a() {
  var x = x + 90;
}
a();
alert(x);

//vad händer???
```

Övning

- Ta reda på skillnaden mellan att använda **var** och att *inte* använda **var** framför en variabel.

Array

Array [ ]

- En array kan, till skillnad från en variabel spara flera värden samtidigt.
- Varje värde indexeras med en siffra från 0 till n (där n är sista värdet).

- Syntax 1 – skapa tom array:
```
var arr = [];
```
- Syntax 2 – skapa populerad array:
```
var arr = [1, 2, 3, 4, 5, ”kalas”, 3, ”kaka”];
```

Array [ ]

- Alla värden kallas element.
  De indexeras med en siffra efter sin plats. Dessa siffor kallas index.

- Skriv följande kod:

```
var arr = [100, 200, ”kaka”];

alert(arr[2]);
```

- Varför blev det så?

Array - indexering

- `Var arr = [10, 20, 30, 40, 50];`
- För att komma åt element 20 skriver vi dess index -> `arr[1]`.
  För att spara det i en variabel;
  `var nbr = arr[1];`

- För att förändra ett element – använd dess index;
  `arr[3] = ”kaka”;`

- För att lägga till nya element – ange ett index som inte är upptaget;
  `arr[5] = 60;`

Övning

- Skapa en array där varje element består av ord i en mening.
- Skriv meningen med hjälp av elementen i arrayen.

- Byt ut orden till tal.
  Skriv ut talen.

Logiska uttryck – if, else, elseif

- För att göra jämförelser av variabler och värden kan vi använda speciella s.k logiska styrstrukturer. Detta tillsammans med de logiska operatorerna gör att vi kan göra finfördelade jämförelser.

- Vi kan t ex jämföra vilken eller vilka variabler som har högst/minst värde, om en sträng innehåller ett visst värde etc etc.

- Till vår hjälp har vi;
  If, else och elseif

syntax

- Där A och B är logiska uttryck.
- **If:**

```
if(A) {
  //kod som exekveras om if-satsen är sann.
}
```

- **If-else:**

```
if(A) {
  //kod som exekveras om if-satsen är sann.
}
else {
  //kod som exekveras om if-satsen är falsk.
}
```

If—elseif-else:

```
if(A) {
 //kod som exekveras om if-
satsen är sann.
}
elseif(B) {
  //kod som exekveras om if-
  //satsen är falsk.
}
else {
 //kod som exekveras om if-
 //satsen OCH elseif-satsen
 // är falska.
}
```

Exempel if

- ```
  var a = 5;
  var b = 10;
  if(a > b)
    alert(”hurra”);
  ```

- ```
  var a = 250;
  var b = 1000;
  if(a == b)
    alert(”jaha”);
  else
    alert(”mjau”);
  ```

- ```
  var a = 5;
  var b = 10;
  if(a == b)
    alert(”hurra”);
  elseif(a >= b)
    alert(”voff”);
  else
    alert(”knasigt”);
  ```

- ```
  var a = 250;
  var b = 1000;
  if(a == b)
    alert(”jaha”);
  else
    alert(”mjau”);
  ```

loopar

- I JS finns ett antal strukturer som kan användas för att repetera exekvering av kodstycken. Den här iterationen kan göras av något som kallas loopar. Vi ska titta på en sådan – for-loopen.

- Syntax:

```
for (inkrementator; villkor; uppräkningsvillkor)
  //kod som ska repeteras
```

Diskutera följande

- Testa följande;

```
for (var number = 0; number <= 12; number = number + 1)
  alert(number); //eller console.log(number);
```

- Testa nu att byta ut den sista 1:an mot talet 2.

```
for (var number = 0; number <= 12; number = number + 2)
   alert(number);//eller console.log(number);
```

När for-loop?

- Användbara för att iterera över data på vilken samma typ av operationer ska utföras.
- I samband med arrayer är for-loopen användbar.

- `var arr=[10,20,30,40,50,60,70,80,90,100,110,120,130,140];`
  Vi inser plötsligt att alla element är tio gånger för stora som var tänkt… vad göra?

- ```
  for(var i = 0; i < 14; i++)
    arr[i] = arr[i]/10;
  ```
- Eller bättre:
- ```
  for(var i = 0; i < arr.length; i++)
    arr[i] = arr[i]/10;
  ```

Vad efter det här…

- Codeacademy
- W3schools
- Bli varm i kläderna (detta är samtidigt en förberedelse inför programmeringskurserna)
- Kolla upp jQuery
- Kolla upp ramverk

- Nä, nu får det vara bra.