Digitaalisen median tekniikat
JavaScript

JavaScript

- ent. LiveScript (Netscape), muunnelma JScript (Microsoft)
  - yhteensopivat yksinkertaisissa asioissa, aiemmin yhteensopimattomat hiemankin edistyneemmissä
- nyk. ECMAScript (standardi)
- selaimessa toimiva tapahtumaperustainen skriptikieli
- ei mitään tekemistä Javan kanssa paitsi, että molempien lähtökohtana on C-kielen syntaksi

JavaScript

- Tulkattava kieli
- Toimii selainympäristössä (client side) – pääsy vain dokumentin dataan
- Ohjelmat ovat enimmäkseen erilaisiin tapahtumiin (käyttäjätoimintoihin tai dokumentin käsittelyvaiheisiin) liittyviä käsittelijöitä, joilla toteutetaan
  - tarkistuksia
  - esittämisen ohjausta
  - dynaamista HTML:ää,
    - dokumentin näkyvään muotoon voidaan tehdä muutoksia sen jälkeen kun dokumentti on ladattu selaimeen – dokumentti itsessään ei kuitenkaan muutu
    - sivuja tai niiden osia voidaan tuottaa ohjelmallisesti (js_esim1.html)
  - vuorovaikutusta

JavaScript

- JavaScriptillä ei voi
  - tuottaa grafiikka
  - käsitellä paikallisia tiedostoja
  - tehdä verkko-operaatioita

JavaScript syntaksi

- perusrakenteeltaan C:n ja Javan kaltainen
- case sensitive
- sijoitusoperaatio (=) esim. a=b;
- lauseet erotetaan toisistaan puolipisteellä, mutta rivin loppu toimii myös erottimena

```
function sample( ) {
var a = 1;
var b = 2;
var c = 3;
return
a + b + c;
}
```


JavaScript syntaksi

- perusrakenteeltaan C:n ja Javan kaltainen
- case sensitive
- sijoitusoperaatio (=) esim. a=b;
- lauseet erotetaan toisitaan puolipisteellä, mutta rivin loppu toimii myös erottimena

```
function sample( ) {
  var a = 1;
  var b = 2;
  var c = 3;
  return
    a + b + c;
}
```

=

```
function sample( ) {
  var a = 1;
  var b = 2;
  var c = 3;
  return;
  a + b + c;
}
```

JavaScript syntaksi

- Rivin loppumisen toimiminen erottimena vaikuttaa myös siihen miten lausekkeet on kirjoitettava:

```
teksti= 'ensimmäinen rivi '
      + 'toinen rivi'
      + 'kolmas rivi';
```

ei toimi

```
teksti= 'ensimmäinen rivi ' +
          'toinen rivi' +
          'kolmas rivi';
```

toimii

JavaScript syntaksi

- Kommentit:
  /* ……monirivinen …..*/
  // yksirivinen
- <!-- tunnistetaan yksirivisen kommentin alkumerkiksi, mutta --> ei tunnistu loppumerkiksi, siksi koodin piilotus vanhoilta selaimilta:

```
<script type="text/javascript">
<!--
  JavaScriptiä tähän väliin
// -->
</script>
```


JavaScript syntaksi

- Merkkijono ei voi jakautua monelle riville

```
teksti= 'ensimmäinen rivi
  toinen rivi
  kolmas rivi';
```

väärin

JavaScript syntaksi

- Vakiot:
  - Numeeriset vakiot: 123
  - Merkkijonovakiot jonkinlaisten lainausmerkkien sisässä kunhan kummallakin puolella samanlainen
  - ”merkkijono”, ’merkkijono’, `merkkijono´

JavaScript tietotyypit

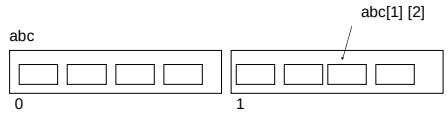
- luvut
  - kokonaisluvut,
  - liukuluvut 3.14, 1.2e3 = $1.2*10^3$ =1200,
  - erikoisarvo: NaN = not a number
- merkkijonot
  - erikoismerkit kuten Javassa \b (backspace),\t (tab),\n (rivinvaihto),\”, \’, \\,
  - \x99 (ascii merkki hexana), \u9999 (unicode)
- totuusarvot:
  - true/false (laskennassa 1/0)

JavaScript tietotyypit

- oliot (object)
  - kokoelma nimettyjä ominaisuuksia
    - ominaisuudella nimi ja arvo
    - ominaisuuden arvoon viitataan joko pistenotaatiolla
      - olio.ominaisuus (esim. image.src)
    - tai assosiatiivisen taulukon tapaan
      - olio[’ominaisuus’] (esim. image[’src’])
      - mahdollistaa ominaisuuden nimen antamisen muuttujana
    - Olion ominaisuuden arvona voi olla olio. Sen ominaisuuden arvoon viitataan
      - olio.ominaisuus.ominaisuuden_ominaisuus
      - esim document.form1.action
  - olioilla voi olla metodeja esim document.write(’text’)
  - valmiiksi määritellyt oliot
  - olioiden sijoitus kuten Javassa, eli viite samaan olioon

JavaScript tietotyypit

- Taulukot (array)
  - taulukko on kokoelma alkioita, joihin voi viitata järjestysnumeron avulla
    - array [index], indeksointi alkaa nollasta
  - taulukon alkiona voi olla taulukko
    - array [outer_index] [inner_index]



JavaScript muuttujat

- Muuttujanimet:
  - alkavat kirjaimella tai _ tai $ -merkillä
  - muut merkit ascii-merkkejä, ei operaatiosymboleja
  - varatut sanat eivät käy muuttujina
- Muuttujat tyypittömiä, toisin kuin esim. Javassa
  - muuttujan tyyppiä ei määritellä esittelyssä
  - tyyppi määräytyy käytön mukaan, jos muuttujaan sijoitetaan luku muuttuja tulee tyypiltään lukuarvoiseksi
  - tyyppi voi muuttua

JavaScript muuttujat

- automaattiset tyyppimuunnokset

```
// set item1 to string '10'
   var item1 = '10';
// set item2 to string '20'
   var item2 = '20';
// item1 set equal to number 10
// by multiplication, a numbers only process
   item1 = item1 *1;
// item2 set equal to string '2010'
// item1 converted to string and concatenated
// concatenation wins over addition
   item2 = item2 + item1;
// item2 set equal to number 201
// division is a numbers only process
   item2 = item2 / item1;
```


JavaScript muuttujat

- Eksplisiittiset tyyppimuunnokset
- parseInt(string), parseFloat(string)
  - ottaavat merkkijonon alusta maksimipituisen kyseiseksi lukutyypiksi tulkittavissa olevan luvun

```
var string1 = '3.1417 is the value of Pi';
var string2 = 'The value of Pi is 3.1417';
Int_1 = parseInt(string1);      // value of Int_1 is 3
Int_2 = parseInt(string2);      // value of Int_2 is NaN
Int_3 = parseFloat(string1);    // value of Int_3 is 3.1417
Int_4 = parseFloat(string2);    // value of Int_4 is NaN
```


JavaScript muuttujat

- Muuttuja esitellään ***var*** avainsanan jälkeen
- Samassa ***var***-lauseessa voi esitellä monta muuttujaa
- Muuttujalle voidaan antaa esittelyn yhteydessä alkuarvo, ellei alkuarvoa ole annettu on muuttujan arvona ***undefined***
- On sallittua esitellä muuttuja toistuvasti (potentiaalinen virhelähde)
  - jos myöhemmässä esittelyssä asetetaan alkuarvo se korvaa aiemman arvon
  - jos myöhemmässä esittelyssä ei anneta alkuarvoa säilyy aiempi arvo

JavaScript muuttujat

```
// simple declaration
   var item1;
// declaration of multiple variables
   var item1, item2, item3;
// declaration with assignment
   var item1 = 7;
// multiple variables with assignment
   var item1 = 7, item2 = 'cat', item3 = 3.17;
```

JavaScript muuttujat

- paikalliset muuttujat
  - funktioiden sisällä määritellyt muuttujat ovat paikallisia eli voimassa vain funktion sisällä
  - funktioiden ulkopuolella määritellyt muuttujat ovat globaaleja
  - globaaleja muuttujia voi käyttää funktioissa ellei paikallisesti ole määritelty samannimistä muuttujaa, jolloin käytetään sitä

JavaScript muuttujat

```
var item1 = 'global';

function testTheScope( ) {
   item1 = 'local';
   document.write(item1);
}
testTheScope( );
document.write(item1);
```

- tulostaa: local local

JavaScript muuttujat

```
var item1 = 'global';

function testTheScope( ) {
   var item1 = 'local';
   document.write(item1);
}
testTheScope( );
document.write(item1);
```

- tulostaa: local global

JavaScript operaattorit

- JavaScriptin matemaattiset, vertailu- ja loogiset operaattorit ovat pääasiassa samat kuin Javassa
- merkittävimpiä eroja:
  - merkkijonojen arvoja verrataan samoilla operaattoreilla kuin lukuja (==, !=,>,…)
  - tavallisten vertailujen yhteydessä tehdään tyyppimuunnos, jos verrattavat ovat eri tyyppiä
  - 1==”1”, 1==true
  - ’11’<’3’ mutta ’11’>3
  - tiukkojen operaatioiden === ja !== yhteydessä ei tyyppimuunnosta ts 1!==”1”
  - null==undefined , null==null

JavaScript operaattorit

- binäärinen + on kuormitettu
  - yhteenlasku, katenaatio
  - jos molemmat osapuolet lukuja niin yhteenlasku muuten katenaatio
- typeof(muuttuja) antaa muuttujan tyypin

JavaScript lauseet

- ehtolauseet

```
if (ehto) { lauselohko }

if (ehto) { lauselohko }
else {lauselohko}

if (ehto1) { lauselohko }
else if (ehto2) {lauselohko}
…

switch (lauseke) {
  case arvo: lauselohko
  case …
  default: lauselohko
}
```

break kuten Javassa

JavaScript lauseet

```
switch (parseInt(xyz)) {
// each of these three cases
// will produce the same statement
  case NaN:
  case 0:
  case 10:
     window.alert(xyz +   ' is not a value I can work with!');
     break;
  default:
     window.alert(xyz +   ' is ready for processing!');
     abc = someFunc(xyz);
     break;
}
```


JavaScript lauseet

- Toistolauseet while, do ja for kuten Javassa
- Lisäksi:
  - for (muuttuja in objekti) {lauselohko}
  - käy läpi taulukon alkiot tai olion ominaisuudet

```
for (elName in navigator) {
  document.write(elname);
  document.write(" = ");
  document.write(navigator[elName]);
  document.write("<br />");
}
```


JavaScript funktiot

- JavaScriptissä voidaan määritellä funktioita.
- Toisin kuin Javassa:
  - voidaan määritellä myös irrallisia funktioita, jotka eivät ole minkään olion metodeja
  - funktioiden paluuarvon tyyppiä ei voi määritellä
  - funktion esittelyssä ja kutsussa voi olla eri määrä argumentteja (puuttuvilla arvo ***undefined***, ylimääräisiin pääsee käsiksi taulukon ***arguments*** kautta)
  - return:n perässä voidaan antaa paluuarvo, ellei anneta palautetaan ***undefined***
  - taulukot ja objektit viiteparametreja, muut argumentit arvoparametreja

```
function functionName(arguments) {
  statements;
  return;
}
```


JavaScript taulukot

- Taulukon luonti
  - var taulu= new Array();
    - alkioiden määrää ei ole annettu
    - taulukot ovat dynaamisia, joten alkioita voidaan sijoittaa ylärajan ulkopuolellekin
    - taulu[0]=1;
    - taulukkoa voi käyttää myös assosiatiivisena (vrt hashtable)
      - **taulu['uusialkio']=2; (jos taulukossa oli aiemmin yksi alkio (jonka indeksi siis 0), niin uusialkio assosioidaan arvoon 1.**
  - var taulu1= new Array(6); // 6 alkiota, undefined
  - var taulu2= new Array(1,20,'abc',200);
    - tauluko jossa 4 alkiota
  - var taulu3 = [1,20,'abc',200];
  - ominaisuus length ilmoittaa taulukon koon, taulu3.length==4

JavaScript taulukot

- Taulukon koko määräytyy sen todellisen koon mukaan ei esittelyn

```
var pikkutaulu= Array(5);
pikkutaulu[200]=1;
pikkutaulu.length==201
```

- Taulukkometodeja, esim:

  taulukko.concat(taulukko1) liittää taulukon loppuun toisen taulukon alkiot

  taulukko.sort() järjestää alkiot