Javascript - Olio-ohjelmointi ja periminen

- Tämä vaihe ei ole helppo! Luennoillakaan ei ole ehditty asiassa kovin pitkälle. Hyvä työ siis edellyttää itsenäistä ja kriittistä perehtymistä verkosta (ja kirjoista?) löytyvään materiaaliin.
- Tunnetusti (*"I hope"! ;-)*) olio-ohjelmoinnin periytymistekniikan kaksi tärkeää tavoitetta ovat: a) ratkaistavan ohjelmointiongelman käsitteiden luonteva mallintaminen ja siten siis ongelman ratkaisijan ajattelun selkeyttäminen ja helpottaminen, sekä toisaalta b) koodin turhan kopioimisen välttäminen, koodin *uudelleenkäyttö*.
- Pyrkikää hahmottamaan ja löytämään, millaiset JavaScriptin ohjelmointitekniikat mahdollisimman hyvin palvelisivat näitä tavoitteita. Perusteluja tarvitaan!
- Varsinkin tämä kohta varmaan täydentyy ja kypsyy lopulliseen dokumenttiin!

Karhuryhmä
Antti Rantapelkonen
Kristian Hansson
Niklas Lillqvist
Henri Karhu

Javascriptin olio-ohjelmointi on melko helppoa, tuntuu kuin olioihin voisi viitata ihan miten sattuu. Välillä tuntuukin, että rakenne on melko hauras ja sekava. Toisaalta koodia siistimällä ja järkeilemällä saa aikaan melko selkeää ja turvallisen oloista olio-ohjelmoinnille tyypillistä koodia.

Javascriptissä ei ole luokkia Javan tapaan. Javascriptin olio on **assosiaatiotaulukko**, joukko avain-arvo -pareja. Arvo voi olla tietokenttä, funktio, toinen olio, jne. Olioita luodaan toisten olioiden perusteella käyttämällä niitä prototyyppeinä. Täten muodostuu prototyyppiketju, jonka lopussa on Object-luokka.

```
var olio = {
    avain1: 1,
    avain2: 2
    avain3: function () {
        return this.avain1
    }
}
```

Olion luonti olioliteraalilla, josta assosiaatiotaulukko-luonne paljastuu

Olioihin voi dynaamisesti lisätä ja poistaa kenttiä, eli niitä ei tarvitse määritellä oliota tehdessä. Esim:

```
olio.avain3 = 3 // siellä on nyt avain3-avain
delete olio.avain1 // ja, tadaa, ei avain1:stä
```

Olioista:

```
function objekti(name) {
    this.nimi = name;
}

function scheisse(elain, koko) {
    this.elukka = elain;
    this.size = koko;
    this.mjono = "Tämän elukan laji on: " + this.elukka + " Ja sen koko   on:
" + this.size;
}

function maatila(nimi) {
    this.name = nimi;
    this.barn = new Array();
    this.amount = 0;

}

function lisaaElukka(tila, elain) {
    this.farm = tila
    tila.barn[tila.amount] = elain;
    document.writeln("Lisätty: " + elain.elukka);
    tila.amount++;
}

var b = new objekti("tyhjis");
var c = new scheisse("lehmä", "400kg");
var d = new scheisse("sika", "500kg");
var hevonen = new scheisse("hevonen", "1000kg");
var tila = new maatila("Piippolan vaarila");
lisaaElukka(tila, c);
lisaaElukka(tila, d);
lisaaElukka(tila, hevonen);
document.writeln(tila.name);
var arr = tila.barn;
document.writeln(arr.length);
document.writeln(arr[2].elukka);
```

Perintä

Javascriptin perintä poikkeaa esim. Javasta.

```
function Yli() {
    this.name = "Väinö";
```

```
}

function Ali() {
    this.ika = 11;
    //this.name = nimi;
}
// Ali perii Ylin
Ali.prototype = new Yli();

var ali = new Ali();
document.writeln(ali.name);
document.writeln(ali.ika);

function Uiva() {
    this.osaauida = "Osaan uida";
}

function Koira() {
    this.kerroTaidot = "Osaanko uida? " + this.osaauida;
}
Koira.prototype = new Uiva();

function Lammas() {
    var villatpaalla = true;
    this.villatpaalla = true;
    this.kerinta = keritse;
    if (villatpaalla) {
        this.kerroTaidot = "MäÄä en osaa uida!";
    } else {
        this.kerroTaidot = this.osaauida;
    }
}

function keritse() {
    document.writeln("Keritään lammas");
    this.villatpaalla = false;
}

Lammas.prototype = new Uiva();

var Late = new Lammas();
var Gromit = new Koira();
document.writeln("Late Lammas: " + Late.kerroTaidot);
document.writeln("Gromit-koira: " + Gromit.kerroTaidot);
Late.kerinta();
document.writeln("Late Lammas: " + Late.kerroTaidot);
```