Karhuryhmä

Antti Rantapelkonen
Kristian Hansson
Niklas Lillqvist
Henri Karhu

JavaScript - Tyyppiturvallisuuden tavoittelua

Johdanto

Tyypittelemättömänä kielenä Javascript on herkkä vaikeasti löydettäville ohjelmointivirheille, koska kääntäjä tai tulkki ei välttämättä tee virheilmoituksia vääränlaisesta syötteestä. Tällöin esimerkiksi käyttäjä saattaa syöttää pyydetyn kokonaisluvun tilalla merkkijonon, joka sisältää SQL-käskyn, joka voi myöhemmin johtaa vakaviin ongelmiin, koska muuttuja on hyväksynyt merkkijonon.
Javascript sisältää seuraavanlaiset alkeismuuttujatyypit:
- number, joka käsittää niin liukuluvut kuin kokonaisluvut.
- boolean, joka käsittää totuusarvot true tai false.
- String, joka käsittää merkkijonot.

Näiden lisäksi on myös:
- Array, eli taulukko, joka voi sisältää mitä tahansa alkeismuuttujia tai olioita.
- Object, eli olio

Muuttujalle voi antaa arvoksi myös funktion, joka toimii omana muuttujatyyppinä.

Javascriptin standardikirjasto sisältää valmiiksi omia tyyppitarkastusfunktioita ja vertailuoperaatioita:

- typeof: Yksinkertaisesti palauttaa muuttujan tyypin. Saattaa tosin antaa huonoja tuloksia. Esimerkiksi typeof Null = “object” ja NaN = “number.”
- isNaN: Palauttaa totuusarvon true, jos parametrina annettu muuttuja on “Not a Number”.
- isFinite: Palauttaa arvon true, jos parametrina annettu muuttuja on vähemmän kuin number, jonka arvo on Infinity. Palauttaa muuten palauttaa false. Myös silloinkin kun syötteenä on esim. merkkijono tai NaN.
- instanceof: esimerkiksi x instanceof Array. Palauttaa true, jos x on taulukko. Instanceof saattaa kuitenkin palauttaa vääriä arvoja, jos sitä kutsutaan eri ikkunoiden - kuten popuppien - yli.
Näitä valmiita kirjastofunktioita ei kuitenkaan kannata käyttää sellaisenaan, vaan on suotavaa määritellä varmempia aliohjelmia tyyppitarkastukseen.
Aliohjelmaehdotuksia tarkempaan tyyppitarkasteluun

isNumber(x), tarkistaa onko *x* (kokonaisuudessaan) numero

```
function isNumber(n) {
      return !isNaN(n) && isFinite(n)
}
```

isInteger(x), -''- kokonaisluku

```
function isInt(n) {
      return typeof n === 'number' && n % 1 == 0;
}
```

isNumericArray(x), tarkistaa onko *x* Array jossa kaikki arvot ovat numeroita

```
function isNumericArray(n) {
    if (!(n instanceof Array)) {
        return false
    }
    for (var i = 0; i < n.length; i++) {
        if (!isNumber(n[i])) {
            return false
        }
    }
    return true
}
```

isString(x), tarkistaa onko *x* tyyppiä String

```
function isString(n) {
    return ((typeof n) === 'string')
}
```

Javascriptin vertailuoperaatiot:

== : ovatko “yhtäsuuret” (esim 0 == '')
=== : ovatko yhtäsuuret ja onko tyyppi sama
!= : ovatko erisuuret
!== : ovatko erisuuret ja onko tyyppi eri

esimerkkejä:

```
"3" == 3 // true
"3" === 3 // false
1 == true // true
1 === true // false
"1" == true // true
"1" === true // false
3 === parseInt(300000000000000000000) // true
9999999999999999 === 10000000000000000 // true
```

esimerkkejä tyyppiturvallisuutta tavoittelevasta ohjelmointityylistä funktioitanne käyttäen.

Esimerkki 1: Web-sovelluksen käyttäjää on pyydetty antamaan ikänsä numerona. Ikä syötetään ponnahdusikkunaan tai HTML-kaavioon. Tällöin määrittelemämme funktio “isInt()” tarkistaa onko kyseessä varmasti kokonaisluku eikä esimerkiksi merkkijono tai liukuluku.

Esimerkki 2: Olkoon olemassa web-sovellus, joka tarjoaa helpon mahdollisuuden laskea todistuksen numeroiden keskiarvon. Käyttäjä syöttää taulukkoon numeron yksi kerrallaan. Funktio isNumericArray(n) tarkistaa taulukon alkiot, jotta ne varmasti ovat numeroita. Tällöin ysiluokkalainen Jonne Trollaaja ei voi syöttää matikan arvosanakseen merkkijonoa “ope on pöljä”.

Linkkejä:
http://tobyho.com/2011/01/28/checking-types-in-javascript/
http://wtfjs.com/
http://stackoverflow.com/questions/1094531/when-should-you-use-vs-vs-etc-in-javascript?lq=1