JavaScript ja virtuaalikoneet

Ville Lahdenvuo

Referaatti
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 18. syyskuuta 2015

JavaScript on ohjelmointikieli, joka suunniteltiin ensisijaisesti verkkosivujen tekijöille, joilla ei välttämättä ole paljon ohjelmointikokemusta. Sen idea oli täydentää Java-ohjelmointikieltä ja HTML-merkkauskieltä. Se mahdollistaa monimutkaisten Internet-selaimissa suoritettavien sovellusten kirjoittamisen ilman selainlaajennuksia [Pao95].

Selainten suosio sovellusalustana on kasvanut ja samalla JavaScriptin käyttö on lisääntynyt räjähdysmäisesti. Se ei ole yllättävää, sillä kaikissa kuluttajatietokoneissa on jokin selain. Lisäksi sovellusten jakaminen on helppoa, koska siihen riittää pelkkä URL-osoite.

Selaimet ovat kehittyneet ja niiden käyttämiä teknologioita on standardoitu. Tämä on auttanut tekemään JavaScriptistä varteenotettavan vaihtoehdon moderniin sovelluskehitykseen. JavaScriptin käyttö ei kuitenkaan rajoitu vain selaimiin, vaan sitä käytetään myös palvelinsovelluksissa sekä muissa ilman selainta toimivissa sovelluksissa, kuten esimerkiksi Atom-tekstieditorissa [Git15].

JavaScript on dynaaminen oliopohjainen kieli, mutta se tukee myös imperatiivista ja funktionaalista ohjelmointityyliä. JavaScript siis tarjoaa monta tapaa toteuttaa samoja asioita [ECM15, Osio 4.2.1.]. Muuttujat JavaScriptissä ovat dynaamisesti tyypitettyjä ja tämä helpottaa ohjelmointia tekemällä muuttujien käytöstä vapaampaa. Tästä dynaamisuudesta seuraa kuitenkin myös ongelmia, sillä virtuaalikoneiden on vaikea ennustaa dynaamisia muutoksia ja tehdä järkeviä optimointeja, joilla suorituskykyä saisi parannettua [ACS+14].

Modernit JavaScript-virtuaalikoneet ovat kehittyneet paljon viime vuosina ja niihin on toteutettu monimutkaisia ja kekseliäitä menetelmiä suorituskyvyn parantamiseksi. Esimerkkejä tälläisistä ovat muun muassa piiloluokat (hidden classes) ja erilaiset ajonaikaiset optimoivat kääntäjät.

Suuri osa virtuaalikoneiden parannuksista perustuvat oletukseen, että vaikka kieli itsessään on dynaaminen, ohjelmat kuitenkin käyttäytyvät

yleensä melko staattisesti. Keräämällä tyyppitietoa suorituksen aikana, virtuaalikoneet pystyvät esimerkiksi luomaan optimoitua konekoodia osalle lähdekoodista. Tämä edellyttää tietenkin, että JavaScript-ohjelmoija tietää miten asiat kannattaa toteuttaa.

Kieltä kuitenkin kehitetään jatkuvasti ja siihen ollaan tuomassa muun muassa luokka- ja moduulijärjestelmät [ECM15, Osiot 14.5. ja 15.2.], jotka pyrkivät yhtenäistämään toteutustapoja ja mahdollistavat tällä tavoin paremman ennustettavuuden ja suorituskyvyn.

On selvää, että JavaScriptin standardoiminen, sen käytön lisääntyminen ja selainvalmistajien keskinäinen kilpailu suorituskyvystä on parantanut kielen asemaa ja mainetta. JavaScriptin rooli on muuttunut skriptikielestä yleiskäyttöiseksi ohjelmointikieleksi [ECM15, Osio 4.]. Kielen tulevaisuus näyttää lupaavalta. Siihen on tulossa paljon ohjelmointia helpottavia ominaisuuksia ja tapoja välttää yleisiä sudenkuoppia, joihin varsinkin aloittelevat ohjelmoijat usein törmäävät.

Lähteet

ACS+14 Ahn, Wonsun, Choi, Jiho, Shull, Thomas, Garzarán, María J. ja Torrellas, Josep: *Improving JavaScript Performance by Deconstructing the Type System.* SIGPLAN Not., 49(6):496–507, kesäkuu 2014, ISSN 0362-1340. `http://doi.acm.org/10.1145/2666356.2594332`.

ECM15 ECMA International: *ECMAScript® 2015 Language Specification*, kuudes painos, Kesäkuu 2015. `http://www.ecma-international.org/ecma-262/6.0/`.

Git15 GitHub: *Atom - A hackable text editor for the 21st Century*, 2015. `https://atom.io/`, vierailtu 2015-09-16 .

Pao95 Paolini, G: *Netscape and Sun Announce JavaScript, The Open Cross-Platform Object Scripting language for Enterprise Networks and the Internet.* Press Release. Sun Microsystems, Inc, 1995.