JavaScriptin staattinen tyypittäminen

LuK -tutkielma
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
2017
Oskari Noppa

Sisältö

Luku 1

Johdanto

Kaikenlaisen ohjelmoinnin keskiössä on data ja yksi tärkeimmistä datan ominaisuuksista on sen tyyppi. Muuttuja “nimi” voi olla datatyypiltään teksti ja muuttuja “ikä” voi olla numero, eikä näitä kahta voi huolettomasti sekoittaa. Ohjelman tila ei ole järkevä jos se sanoo henkilön iän olevan “Matti”. Ohjelmointikielet voidaan hyvin karkeasti jakaa kahteen sen mukaan miten ja missä vaiheessa niissä käsitellään muuttujien tyyppejä.

Staattisesti tyypitetyiksi kutsutaan kieliä jotka vaativat että ohjelman käsittelemien tietorakenteiden tyypit on tulkittavissa käännös aikana, eli jo ennen ohjelman suorittamista. Lähes aina ohjelmointikieli saa tämän tiedon tietotyypeistä vaatimalla koodiin erityisiä tyyppimäärittelyitä. Ohjelman koodissa voitaisiin esimerkiksi määrittää että henkilön ikä on aina numero. Tällöin ohjelmointikielen kääntäjään rakennetut tarkistukset voivat vahtia jo ennen koodin suorittamista ettei ohjelmoija virheellisesti yritä asettaa henkilön iäksi tekstiä tai mitään muutakaan ei-numeerista arvoa. Dynaamisesti tyypitetyissä kielissä sen sijaan vastuu oikeiden tietotyyppien käyttämisestä jätetään ohjelmoijalle, kieli ei vaadi kehittäjältä tyyppien eksplisiittistä määrittämistä eikä niiden oikeellisuutta tarkasteta ainakaan ennen ohjelman suorittamista.

Kyseessä on kielen suunnittelullinen valinta. Kielen dynaaminen tyypittäminen vaatii yleensä vähemmän varsinaisen koodin kirjoittamista saman tuloksen saavuttamiseksi, sillä erillisiä tyyppimäärittelyitä ei tarvitse kirjoittaa. Toisaalta staattinen tyypitys mah-

dollistaa monia hyödyllisiä kehitystyökaluja ja auttaa poistamaan väärien tietotyyppien käyttämisestä johtuvan bugien osajoukon. Kielen suunnittelijan on löydettävä haluamansa tasapaino kielen ominaisuuksien välillä ja arvioitava mikä on parhaaksi niihin käyttötarkoituksiin joihin kielen on tarkoitus hyvin soveltua.

Luku 2

Peruskäsitteitä

2.1 Määritelmiä

On hyvä huomioida ero termien välillä kun puhutaan staattisesta ja dynaamisesta, tai toisaalta vahvasta ja heikosta tyypittämisestä. Myös dynaamisesti tyypitetty kieli voi nostaa virheen väärien tietotyyppien käyttämisestä ohjelman suorittamisen aikana, etenkin jos se on vahvasti tyypitetty kieli. Tässä tutkielmassa keskitytään kuitenkin lähinnä staattiseen ja dynaamiseen tyypitykseen, eli siihen miten tietotyyppeihin liittyviä virheitä käsitellään käännös aikana, jo ennen ohjelman suorittamista.

Luku 3

Työkalun käyttöönotto

3.1 Tyyppiannotaatiot

Syntaksi

3.2 Ohjelman kääntäminen ennen suorittamista

Luku 4

Virheiden havaitseminen

Luku 5

Ohjelman optimointi käännösvaiheessa staattisen analyysin perusteella

Luku 6

Tyyppimäärittelyt dokumentaationa

Luku 7

Ongelmat JavaScriptin staattisessa tyypittämisessä

7.1 Vaikeasti analysoitavat suunnittelumallit

Luku 8

Yhteenveto