
JavaScriptin staattinen tyypittäminen

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

Sisältö

1	Johdanto	1
2	Peruskäsitteitä	3
2.1	Määritelmiä	3
3	Työkalun käyttöönotto	4

Luku 1

Johdanto

Kaikenlaisen ohjelmoinnin keskiössä on data ja yksi tärkeimmistä datan ominaisuuksista on sen tyyppi. Muuttuja “nimi” voi olla datatyybiltään 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 tyypinmäärittelyitä. Ohjelman koodissa voitaisiin esimerkiksi määrittää että henkilön ikä on aina numero. Tällöin ohjelmointikielen kääntäjään rakennetut tarkistukset voivat vahvistaa jo ennen koodin suorittamista ettei ohjelmoija virheellisesti yritä asettaa henkilön iäksi tekstiä tai mitään muutaakaan ei-numeerista arvoa. Dynaamisesti tyypitetyissä kielissä sen sijaan vastuu oikeiden tietotyyppien käyttämisestä jätetään ohjelmoijalle, kieli ei vaadi kehittäjältä tyypin 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 tyypinmäärittelyitä ei tarvitse kirjoittaa. Toisaalta staattinen tyypitys mah-

dollistaa monia hyödyllisiä kehitystyökaluja ja auttaa poistamaan väärin 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 tyyppittamisestä. Myös dynaamisesti tyyppitetty kieli voi nostaa virheen väärin tietotyyppien käyttämisestä ohjelman suorittamisen aikana, etenkin jos se on vahvasti tyyppitetty kieli. Tässä tutkielmassa keskitytään kuitenkin lähinnä staattiseen ja dynaamiseen tyyppitykseen, eli siihen miten tietotyyppeihin liittyviä virheitä käsitellään käännös aikana, jo ennen ohjelman suorittamista.

Luku 3

Työkalun käyttöönotto