Haaga-Helia
University of Applied Sciences

Ohjelmistokehityksen teknologioita - *S*eminaarityö

TypeScriptin opettelua

4. ES6, Node.js, Mongo

Julia Söderlund

14.4.2022

Sisältö

Tiivistelmä

Tavoitteenani oli opetella TypeScriptia, jotta voisin kehittyä JavaScript-ohjelmoijana. Ensin yritin tehdä työn erään open source projektin kehittämisestä, koska se olisi ollut kiinnostavaa, mutta se ei ihan onnistunut. Loppujen lopuksi päätin muuttaa vanhan React-projektin TypeScriptiin, ja onnistuin siinä kohtuullisen hyvin.

Opin paljon uutta, mutta opin myös, että aihe ei ollut niin helppo, kuin mitä oletin. Suurimpana ongelmana oli tiedon haku, sillä en tiennyt miten aihetta kannattaa opiskella.

1 Johdanto

1.1 Motivaatio

Olen jo jonkin aikaa miettinyt, että olisi kiva oppia TypeScriptia. JavaScript on aina ollut suosikkini, joten olisi hyvä opetella jotain mikä rakentuu sen päälle. Tämä halu konkretisoitui, kun kävin Junctionissa puhumassa eri yrityksille kyselemässä millaisia teknologioita heillä käytetään. TypeScript tuli mainittua aika monta kertaa. Koen, että TypeScriptin osaaminen voisi olla hyödyllistä työnhaussa.

Toisena motivaationa toimi myös eräs open source -projekti, jota halusin kehittää. Projekti on siis Discord-botti joka toimii yhdellä serverillä missä vietän aikaa (tämä ei siis liity mitenkään ohjelmistoprojekti 2:n työhön vaikka sielläkin oli aiheena Discord-botti). Minulla on pari tuttavaa ollut kehittämässä projektia, niin ajattelin sen olevan hauska tehtävä.

1.2 TypeScriptista

TypeScript rakentuu JavaScriptin päälle ja sen mahdollistaa muuttujien vahvan tyypityksen. Tämä on kuitenkin vapaaehtoista, sillä tyypittämisessä voi käytä tyyppiä *any*, joka ohittaa tarkistuksen tietyn muuttujan kohdalla. Vaikka any-tyypin käyttäminen tekee koodin kirjoittamisesta helpompaa, se myös poistaa kaikki TypeScriptin hyödyt kyseisen muuttujan kohdalla. Muuttujalla voi olla myös useampi vaihtoehto tyypille, jolloin ei tarvitse valita vain yhtä.

Tyyppien lisääminen helpottaa isompien ohjelmistojen ylläpitämistä, kun muuttujien tyypit on valmiiksi määritelty ja muuttujien tyyppejä tai funktioiden ulostuloa ei tarvitse lähteä ar-

vailemaan. Koodieditorit auttavat ymmärtämään koodia paremmin, kun ne voivat TypeScriptin avulla kertoa, millaisia tyypit ovat ja hyödyntää IntelliSenseä (älykäs koodin viimeistely) paremmin.

Tietenkin tämä tyypittäminen on alussa paljon vaivalloisempaa kuin pelkän JavaScriptin kirjoittaminen, mutta tarkoituksena on helpottaa ohjelmiston ylläpitämistä pitkässä juoksussa.

TypeScript transpiloidaan, eli käännetään JavaScript-koodiksi, jotta sen voisi suorittaa. TypeScript ei siis vaikuta ohjelman suoritukseen, vaan siitä on hyötyä ohjelmiston kehitysvaiheessa. Valmiin projektin muuttaminen TypeScriptiksi on siis aika hyödytöntä, koska sovellus ei itsessään muutu mitenkään, mutta se on hyvä tapa oppia aiheesta.

2 Ensimmäinen aihe

2.1 Aiheen valinta

Tiesin, että tämä olisi hieman riski aihe, koska minulla ei ollut mitään takuuta siitä, että minulle löytyisi 20 h edestä tekemistä. Mutta halusin kuitenkin lähteä yrittämään sillä aihe oli mielenkiintoinen. Toinen riski oli, että kaikki tehtävät hommat olisivat liian vaikeita. Muilla projektin kehittäjillä on paljon enemmän kokemusta kuin minulla, niin oletin, että he mieluummin tekisivät helpot muutokset itse kuin ohjeistaisivat minua tekemään niitä. Opin kuitenkin, että heillä kyllä oli halua opastaa minua omien aikataulujen puitteissa. Joku muu olisi tehnyt muutokset nopeammin, mutta heille oli tärkeämpää innostaa lisää ihmisiä mukaan projektiin kuin säästää aikaa.

Vaikka tämä ei ole seminaarityön pääaiheena, päätin silti sisällyttää sen, koska käytin siihen aikaa ja se oli mielestäni mielenkiintoinen kokemus.

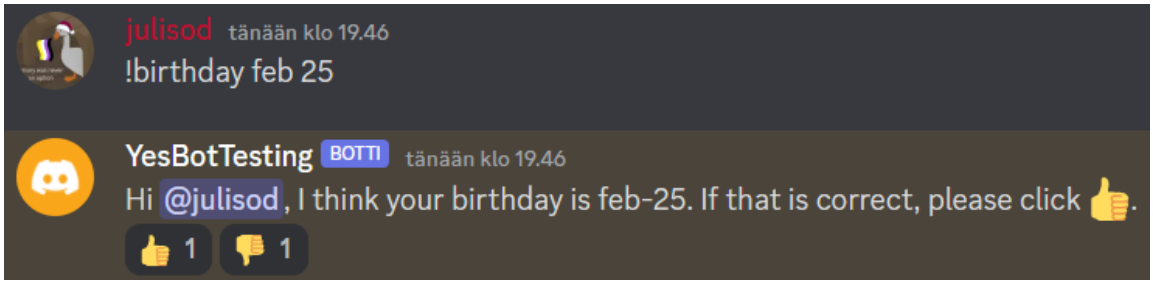
2.2 Tekemäni muutokset

Pääsin mukaan projektiin ja sain alustettua kaikki ohjelmistot, mutta projektin ylläpitäjällä on ollut nyt niin kova kiire, ettei hänellä ole löytynyt aikaa keksiä minulle tehtävää, vaikka se oli alun perin suunnitelmissa. Hän kuitenkin keksi pienen muutoksen, jolla pääsisin alkuun, alla osa hänen lähettämästä viestistä:

> As for something easy you could do: There is one thing that has been bothering me for a bit recently. We use BirthdayBot to handle birthdays (announcing the birthdays and handing out the roles) and for that YesBot generates two slash commands when someone adds their birthday which we copy and paste to set the birthday in BB. Currently those slash commands contain a userping @Username#1234 which doesn't always work if the user is not cached in the client so you have to go back and find that

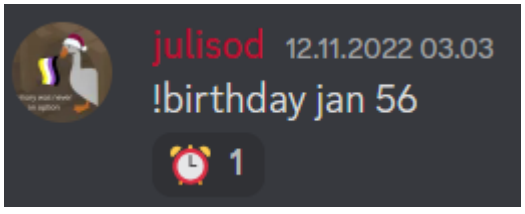
> user to make sure it's cached and only after that you can paste the command. That's super annoying 👊 What I found recently was that replacing that userping with just the userId works just as well and doesn't require having the user cached.
>
> So, if you want, you can track down where the two commands is generated and change it so that instead of the tag, the id is used. The change is fairly small but maybe that helps get a feeling for the files and how things are structured. The birthday command is arguably one of the more chaotic files, I believe 😅 If you have any questions about the setup, the code or anything else, let me know!
>
> The issues are a bit of a mess indeed; I really need to find some time again to clean those up buuuuuuut free time is an issue atm 😄

Sain kuitenkin kaksi pientä committia tehtyä loppujen lopuksi. Toisen keksin itse, mutta isompaa hommaa en saanut keksittyä. Syntymäpäivän syöttämisen alku näyttää tältä:



Kuva 1. Komennon käyttäminen

Ennen muutoksiani, jos ei reagoinut tietyn ajan (15sek.) kuluessa, botti lakkaa kuuntelemasta vastauksia, mutta ei ilmoita tätä käyttäjälle ollenkaan. Eli jää vain tähän näkymään. Lisäsin pari riviä, ja nyt jos ei tule vastausta, botti poistaa oman viestin ja reagoi käyttäjän alkuperäiseen viestiin emojilla, joka viestii, että komento on aikakatkaistu. Tämä saa käyttäjän kirjoittamaan komennon uudestaan sen sijasta, että jää miettimään miksei reaktion painaminen toimi.



Kuva 2. Komennon aikakatkaisu

2.3 Mitä opin

Ehdin kuitenkin oppia kuinka pull requesteja tehdään, ja kuinka isompia projekteja pystyy kehittää ilman että on mahdollisuutta vahingossa rikkoa mitään. Projektin kehittämistä varten pii luoda oma kopio botista ja serveristä, ja tietokanta pyörii dockerin avulla paikallisesti.

Projektin koodiin tutustuminen vei oman aikansa, koska siellä oli jo valmiina erittäin paljon koodia, mutta koodin lukemista ei vaikeuttanut se, että se oli TypeScriptia, koska joku oli jo hoitanut tyyppien määrittelyn ennen minua.

Tämä kokemus myös lisäsi itsevarmuutta ohjelmoijana, sillä opin, että minun ei tarvitse osata kaikkea valmiiksi jotta voisin osallistua vastaavanlaisiin projekteihin.

3 Toinen aihe

3.1 Idea

Tarvitsin kuitenkin jonkun uuden aiheen, jossa voisin osoittaa teknistä osaamistani. Minulla ei ollut ideaa uudelle sovellukselle, joten päätin valita helpon ja turvallisen vaihtoehdon (ei se ollutkaan loppujen lopuksi hirveän helppo), eli muuttaa olemassa olevan projektin JavaScriptistä TypeScriptiin. Valitsin front end -kurssilla tehdyn React-sovelluksen.

3.2 Toteutus

En tehnyt mitään muutoksia sovelluksen toiminnallisuuksiin tai ulkoasuun, ainoa mitä tein oli, että muunsin tiedostoja TypeScript-tiedostoiksi parhaani mukaan.

3.2.1 Alkukankeus

En ole tottunut palaamaan vanhoihin projekteihin, sillä koulussa tehdään aina uusia projekteja. Koska olin koskenut projektiin viimeksi vuosi sitten, suuri osa npm-paketeista olivat vanhentuneita eikä sovellus lähtenyt käyntiin, joten minun piti lähteä päivittelemään niitä. Siinäkin oli oma oppimisensa keksiä miten ne saisi päivitettyä järkevästi ilman että kävisi jokaista yksitellen läpi tai että rikkoisi sovellusta. Jotenkin minä sain sitten lopulta ohjelman käyntiin muokkaamalla skriptejä package.json:ssa.

Alussa oli myös vaikea muistaa mikä funktio tekee mitäkin ja missä muodossa tietyt muuttujat on (tähän olisi saattanut auttaa se, että koodi oltaisiin alun perin kirjoitettu TypeSriptillä). Mutta koska kirjoitan (ainakin omasta mielestäni) järkevää koodia, tajusin taas koodin toiminnan nopeasti.

3.2.2 Itse työskentely

Ensimmäisenä piti luoda tiedosto nimeltä tsconfig.json, joka määrittelee sen miten TypeScriptin käännösohjelma toimii. Sinne pystyy erikseen määrittelemään vaikka ja mitä, mutta

kun suoritin projektin ensimmäisen kerran niin jokin lisäsi sinne automaattisesti kaikki tarvitsemani tiedot, kuten mihin ECMAScriptin versioon koodi käännetään tai että salliiko se projektissa olevan JavaScript-tiedostoja.

Lähdin muokkaamaan tiedostoja yksi kerrallaan ja tarvittaessa lisäsin uusia tyyppejä tiedostoon nimeltä types.d.ts. Yritin välttää any-tyypin käyttöä, mutta koodissa oli paljon kohtia joihin en vain keksinyt oikeaa vastausta.

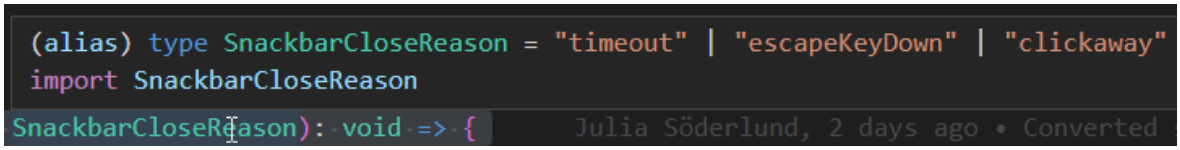
Minulla riitti hyvin aikaa muokata kaikki tiedostot mitkä osasin, mutta jos olisin halunnut poistaa kaikki any-tyypit, olisi se tarvinnut enemmän kuin 20 h ja hieman syvällisempää perehtymistä.

3.3 Esimerkki hyödystä

Minulla on tämän näköinen funktio MUI:n snackbarille:

```
const handleErrorClose = (event: React.SyntheticEvent | Event, reason:
SnackbarCloseReason): void => {
    if (reason !== 'clickaway') {
        setErrorOpen(false);
    }
};
```

Ja jos laitan hiiren reason-parametrin tyypin päälle, antaa VSCode minulle seuraavan vihjeen:



Kuva 3. Valmiit tyypit

Olisin voinut määritellä tyypiksi string, se ei olisi ollut sinällään väärin, mutta onhan siitä paljon enemmän apua, jos tietää kaikki mahdolliset arvot mitä voi parametrina saada, ilman, että lähtee erikseen googlettamaan MUI:n dokumentaatiota.

4 Ongelmat ja vaikeudet

Olen aina kuullut kuinka TypeScriptin olevan erittäin helppo kieli opetella ja jos osaa JavaScriptiä niin opettelu on tosi nopeaa. Ajattelin myös, että koska olen Javalla tottunut aina

lisäämään tyypit, niin tämä olisi erittäin helppoa, koska TypeScript on hieman dynaamisempi. Ja vaikka TypeScriptin opettelu on minulle helpompaa kuin jonkun ihan uuden ohjelmointikielen, on siinäkin omat haasteensa.

4.1 Ulkoiset kirjastot

Jos loi itse alusta alkaen jotain muuttujia tai funktiota, oli niiden tyypittäminen helppoa. Mutta koska projektissa on mahdollisuuksien mukaan hyödynnetty ulkoisia kirjastoja, menee homma paljon monimutkaisemmaksi. Vaikka lähes kaikki npm-paketit olivat valmiiksi tyypitetty, on se silti vaikeaa keksiä mitä tyyppiä voi käyttää missä ja mitä tietty syntaksi edes tarkoittaa. Varsinkin AG Gridin kanssa minun oli tosi vaikea keksiä millaista tietoa syötetään parametreinä ja mistä se edes tulee.

4.2 Aiheesta oppiminen

Vaikka yleensä itseopiskelu tutoriaalien avulla on ollut minulle aika helppoa ja luontevaa, TypeScriptin kohdalla se tuntui haasteelliselta. Suurin osa videoista mitä löysin selittivät perusidean, mutta ei sen enempää. Jos yritin etsiä vastausta johonkin spesifiin kysymykseen, en usein löytänyt vastausta, mikä oli yllättävää sillä TypeSriptilla on paljon käyttäjiä. Kysyin joitakin kysymyksiä ihmisiltä netissä, joihin sain ihan hyvät vastaukset, mutta joskus en tiennyt edes mitä halusin kysyä.

Jäi sellainen fiilis, että minulta puuttui jokin olennainen tieto mikä olisi helpottanut työskentelyä, mutta en vain tiennyt mitä tietoa tarvitsen. Minun olisi varmaankin pitänyt rohkeammin kysyä miten jokin kannattaa kirjoittaa juuri minun projektissa, sillä vaikka tiesin perusidean, soveltaminen omaan koodiin oli vaikeaa.

Yritin tehdä tehtäviä nimeltä TypeScript Exercises. Ensimmäiset tehtävät olivat sopivan vaikeita ja hyödyllisiä, mutta jossakin kohtaa jäin jumiin, sillä sivusto ei opettanut itse asiaa, ja dokumentaatiota on välillä vaikea lukea jos on vain aloittelija.

5 Pohdinta

Open source projektiin tutustuminen oli erittäin mielenkiintoista ja mielekästä, sääli ettei siitä saanut tarpeeksi koko seminaarityön aiheeksi. Oman projektin muuttaminen TypeScriptiksi on hyödyllistä, opin kuitenkin paljon ja nyt minulla on jotain mitä esitellä omassa Githubissa.

Mutta rehellisesti sanottuna, sen tekeminen ei ollut yhtä mielekästä kuin mitä olisin halunnut. Vaikka sain paljon aikaiseksi, minusta tuntui etten tiennyt mitä olin tekemässä enkä

myöskään tiennyt teinkö jotain väärin kun kukaan ei ollut tarkistamassa. Vaikka sain kaikki virheilmoitukset poistettua, huomasin että joissakin kohdissa olisin voinut laittaa aivan väärän tyypin, mutta siitä ei olisi tullut mitään ilmoitusta. Ja koska koodi kääntyy JavaScript-koodiksi suorittaessa, en olisi huomannut virheitä valmiin sovelluksen toiminnassa.

Kun lähden oppimaan tästä aiheesta lisää, palaan ehkä enemmän siihen teoriapuoleen, eli luen miten TypeScriptia on mahdollista käyttää ja miten se syntaksi toimii sen sijasta, että lähtisin heti työstämään jotain ilman hyvää pohjatuntemusta (vaikka se on muissa projekteissa toiminut ihan hyvin). Olisi kiva löytää joku valmis kurssi tai tutoriaali niin ei turhautuisi niin helposti siihen, ettei tiedä mitä pitäisi tehdä.

Vielä kertauksena linkit ensimmäiseen ja toiseen projektiin:

https://github.com/Yes-Theory-Fam/yesbot-ts/commits?author=julisod

https://github.com/julisod/personaltrainer

6 Video

https://youtu.be/L0GGLfW3HOM

Lähteet

Työskentelyssä käytetyt lähteet

Create React App. Adding TypeScript. Luettavissa: https://create-react-app.dev/docs/adding-typescript/#installation. Luettu 28.11.2022.

Sitepoint 2020. How to Migrate a React App to TypeScript. Luettavissa: https://www.sitepoint.com/how-to-migrate-a-react-app-to-typescript/. Luettu 29.11.2022.

Ben Awad 2019. React Typescript Tutorial. Katsottavissa: https://youtu.be/Z5iWr6Srsj8.

Joy of Code 2022. TypeScript Fundamentals - #13 Type Narrowing Using Type Guards and Type Predicates. Katsottavissa: https://youtu.be/rewRQA9rBvs.

Zhengheo. A Complete Guide To TypeScript's Never Type Luettavissa: https://www.zhenghao.io/posts/ts-never. Luettu 6.12.2022

Raportin kirjoittaessa käytetyt lähteet

Wikipedia 2022. TypeScript. Luettavissa: https://en.wikipedia.org/wiki/TypeScript. Luettu 5.12.2022.

Symfony Finland 2016. Mikä on TypeScript? Luettavissa: https://symfony.fi/artikkeli/mika-on-typescript. Luettu 5.12.2022