Roope Kivioja

Puheen ongelmista kärsiville tarkoitetun kommunikointisovelluksen toteuttaminen Ionic-kehyksellä

Tietotekniikan pro gradu -tutkielma

9. helmikuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Roope Kivioja

Yhteystiedot: roope.kivioja@gmail.com

Ohjaajat: Jukka-Pekka Santanen ja Jonne Itkonen

Työn nimi: Puheen ongelmista kärsiville tarkoitetun kommunikointisovelluksen toteuttaminen Ionic-kehyksellä

Title in English: Puheen ongelmista kärsiville tarkoitetun kommunikointisovelluksen toteuttaminen Ionic-kehyksellä

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 17+1

Tiivistelmä: Tutkielman tiivistelmä on tyypillisesti lyhyt esitys, jossa kerrotaan tutkielman taustoista, tavoitteesta, tutkimusmenetelmistä, saavutetuista tuloksista, tulosten tulkinnasta ja johtopäätöksistä. Tiivistelmän tulee olla niin lyhyt, että se, englanninkielinen abstrakti ja muut metatiedot mahtuvat kaikki samalle sivulle.

Avainsanat: Ionic, AAC, WWW-sovellukset, käytettävyys

Abstract: Tiivistelmä englanniksi.

Keywords: Ionic, AAC, web applications, usability

Termiluettelo

Ionic	Ohjelmistokehys.

Sisältö

1 Johdanto

- TODO: Tähän tulee johdanto.

2 Kommunikaatiossa avustavan sovelluksen rakentamiseen tarvittavia taustatietoja ja menetelmiä

- TODO: Lyhyt siltaava kappale johdannosta taustatietoihin

2.1 Avusteinen kommunikaatio

- TODO: Kerrotaan yleisesti taustoista (mihin avustavaa tietotekniikkaa tarvitaan, miten sitä käytetään, mitä ongelmia jne.)

On olemassa useita eri sairauksia ja kehityshäiriöitä, joiden johdosta henkilön puheen tuottamisen kyky voi hetkellisesti tai pysyvästi heikentyä. Puheen tuottamisen ongelmista kärsivä henkilö saattaa joutua turvautumaan arkipäivän kommunikaatiossa erilaisiin apuvälineisiin kommunikoidakseen muiden ihmisten kanssa. Yleisesti näitä viestintämenetelmiä kuvaamaan tarkoitettu termi on **puhetta tukeva ja korvaava kommunikaatio** (engl. *Augmentative and Alternative Communication, lyh. AAC*).

Puhetta tukeva ja korvaava kommunikaatio voidaan jakaa kahtia: avustamattomaan ja avusteiseen. **Avustamattomalla puhetta tukevalla ja korvaavalla kommunikaatiolla** tarkoitetaan kommunikaatiota, jossa ei tarvita apuvälineitä. Viittomankieli on yksi esimerkki avustamattomasta puhetta tukevasta ja korvaavaasta kommunikaatiosta, mutta myös ihmisen elekieltä voidaan pitää avustamattomana puhetta tukevana ja korvaavana kommunikaationa.

Avustettu puhetta tukevaa ja korvaava kommunikaatio tarkoittaa puolestaan kommunikaatiota, jossa käytetään jotain apuvälinettä. Apuvälineenä voi olla esimerkiksi valokuvia, kommunikaatiotaulu tai elektroninen laite. Tämän perusteella avustettu puhetta tukevaa ja korvaava kommunikaatio voidaan jakaa vielä eteenpäin kahdeksi ryhmäksi: matalan teknologian ja korkean teknologian puhetta tukevaan ja korvaavaan kommunikaatioon. Käyttäjä ei välttämättä käytä vain yhtä edellä mainituista tyypeistä. (Sigafoos ja Drasgow 2001)

Puhetta tukevaa ja korvaavaa kommunikaatiota tarjotaan puheen tuottamisen ongelmista kärsiville myös tietoteknisten sovellusten avulla, joiden kautta käyttäjä kirjoittaa joko suoraan

tekstiä tai kommunikoi valitsemalla symboleja. Esimerkiksi autistisilla käyttäjillä on tyypillisesti hyvin henkilö- ja yksityiskohtaisia tarpeita puhetta tukevalle ja korvaavalle kommunikaatiolle, joten tietoteknisten sovellusten suhteellisen helppo muokattavuus puoltaa niiden käyttöä perinteisempien puhetta ja kommunikointia korvaavien apuvälineiden sijaan.

Puhetta tukevat ja korvaavat kommunikointisovellukset käyttävät yleisimmin symboleja. Autisteille tyypillistä on vahva visuaalis-avaruudellinen hahmotuskyky, joten tutkimuksen mukaan piirroksiin ja valokuviin liitetyt merkitykset ovat tälle ryhmälle luontevin tapa kommunikoida. Vaughnin ja Hornerin tutkimuksessa (Vaughn ja Horner 1995) Karl-nimisen autistisen koehenkilön haastava käytös ja aggressio vähenivät kun pelkästään verbaalisesti annettujen ruokavaihtoehtojen rinnalle tuotiin kuvat ruoka-annoksista. Symbolipohjaiselle puhetta tukevalle ja korvaavalle kommunikaatiolle on siis sekä tutkimuspohjaista näyttöä että käytännön kokemuksiin perustuvaa kannustetta.

Puhetta tukevat ja korvaavat kommunikointisovellukseen voidaan liittää myös symboleita lukeva ääniominaisuus. Ääniominaisuus mahdollistaa kommunikoinnin näköyhteyden ulkopuolelle, vähentää symbolien tulkitsijan läsnäolon pakollisuutta ja helpottaa pidempien viestien rakentamista puhetta tukevassa ja korvaavassa kommunikointisovelluksessa. (Nunes 2008)

2.2 Progressiiviset WWW-sovellukset

- TODO: Kerrotaan yleisesti PWA-sovelluksista (mihin pohjautuu, miksi käytetään, miksi juuri tähän projektiin)

Progressiiviset sovellukset ovat selaimessa ajettavia WWW-sovelluksia, joiden ulkoasu on optimoitu laitekohtaisesti niin, että niiden ulkoasu ei eroa laitteen natiiveista sovelluksista. Progressiivisella sovelluksella tarkoitetaan sovellusta, joka pystyy käyttämään tarjolla olevan sovellusympäristön resursseja joustavasti sen sijaan, että se itse määrittäisi vaatimukset. Termi on verrattain tuore ja vakiintumaton, sillä esimerkiksi Ionic-kehyksen dokumentaatiossa on käytetty aiemmin myös termiä **hybridisovellus**.

Tällä hetkellä erityisesti Google panostaa progressiivisten sovellusten kehittämiseen Chrome-

selaimen kehitysympäristön yhteydessä. Googlen mukaan progressiiviset sovellukset tarjoavat perinteisiin WWW-sovelluksiin verrattuna enemmän luotettavuutta, käytettävyyttä ja monipuolisempaa sisältöä. Luotettavuus paranee, sillä progressiiviset sovellukset pystyvät tarjoamaan sisältöä myös ilman verkkoyhteyttä. Käytettävyyttä parantaa nopeammin käyttäjän komentoihin vastaava käyttöliittymä. Sovellusmaisuus puolestaan parantaa käyttäjän immersiota. (Google 2018) - TODO: tähän omaa kriittistä pohdintaa (Microsoft ja Edge-selaimen siirtyminen Chromium-pohjaiseksi?)

Idea natiivien sovellusten rakentamisesta selainalustoja hyödyntämällä ei ole uusi. Muutamia esimerkkejä idean käytöstä ovat Adobe AIR - ja Electron -sovellukset. Twitter, AliExpress ja Lancôme ottivat vuonna 2017 käyttöön progressiiviset sovellukset ja julkaistut tulokset ovat olleet positiivisia. Twitter onnistui uuden progressiivisen sovelluksensa ansiosta lisäämään sivupäivityksiä 65% per sessio, lisäämään lähetettyjen Twitter-viestien määrää 75% ja vähentämään käytön lopettamista 20%. Lisäksi uusi sovellus käytti vähemmän kuin 3% natiivin sovelluksen vaatimasta tilasta ja vähensi datan käyttöä 70%. Datan käytön määrän vähentyminen on erityisen merkittävää, koska Twitter arvioi, että vuonna 2017 45% sen sisällöstä ladattiin 2G-verkon läpi. (AppInstitute 2017)

AliExpress ja Lancôme ovat molemmat verkkokauppoja, joilla on ollut ongelmia mobiiliverkkokauppojensa tehokkuuden kanssa. Progressiiviseen sovellukseen siirtymällä AliExpress lisäsi uusien asiakkaiden myyntitapahtumien määrää 104%:lla ja Lancôme 17%:lla. Lisäksi AliExpress tuplasi sivunlatausten määrän sekä kasvatti sessioiden pituutta 74%:lla. Lancôme puolestaan onnistui lisäämään iOS-sessioiden määrää 53%:lla. (AppInstitute 2017) - TODO: tähän omaa kriittistä pohdintaa

Progressiivisten sovellusten käyttö ei kuitenkaan ole ongelmatonta. Ensinnäkin, koska kyse on uudesta suuntauksesta ohjelmistokehitysessä, vaaditaan kehittäjiltä paljon uusien asioiden opettelua sekä vakiintumattomien työkalujen sekä kehysten käyttämistä kehitystyössä. Toinen merkittävä haaste voi olla tiedon tallentaminen, sillä selaimen toimintaan pohjautuvalle sovellukselle ei ole varattu luontevaa tallennustilaa. Kolmas ongelma voi tulla eteen käytettävyydessä, sillä selaimessa toimivan sovelluksen vaatimat resurssit saattavat hidastaa sovelluksen toimintaa. (Divante 2018) - TODO: tähän omaa kriittistä pohdintaa

2.3 Ionic

Ionic on yksi suosituimpia progressiivisten sovellusten tuottamiseen suunnitelluista kehyksistä. Ionic on avointa lähdekoodia ja hyödyntää Apache Cordova -ympäristöä sekä Angular-ohjelmistokehystä. Ionic on MIT-lisenssin alainen. Angular-ohjelmistokehys perustuu TypeScript-ohjelmointikieleen. Kehyksenä Ionicin pääpaino on tarjota kehittäjälle oikealta näyttävä ja visuaalisesti toimiva sovellus. Sen ei ole tarkoitus korvata tyypillisiä JavaScript-kirjastoja vaan se toimii niiden tukena. (Ionic 2019)

- TODO: kaavio Ionicin hierarkiasta?

Ionic koostuu komponenteista. Ionicin komponentit ovat uudelleenkäytettäviä käyttöliittymäelementtejä, jotka toimivat sovelluksen käyttöliittymän rakennusosina. Niiden avulla sovellus näyttää yhtenäiseltä ja käyttöliittymän kehitystyö nopeutuu. Komponentit koostuvat HTML:stä, CSS:stä ja JavaScriptistä. Esimerkkejä tällaisista komponenteista ovat muunmuassa painikkeet, välilehdet ja erilaiset listat.

Ionicin ulkoasu perustuu teemoihin. Myös teemat lisäävät sovelluksen ulkoasun yhtenäisyyttä. Teemat myös mukautuvat eri alustojen ulkoasustandardien mukaisiksi kehittäjän niin halutessa. Esimerkiksi Android- ja iOS -mobiilikäyttöjärjestelmille suunnattujen sovellusten ulkoasu poikkeaa toisistaan, jos käytetään Ionicin oletusteemoja. Teemojen käyttäminen parantaa sovelluksen käytettävyyttä tekemällä sen ulkoasusta ennustettavamman ja tutumman loppukäyttäjälle.

- TODO: lisää Ionicista

2.3.1 Apache Cordova

Apache Cordova (aikaisemmin PhoneGap) on alunperin Nitobin kehittämä sovelluskehitysympäristö mobiililaitteille. Adobe osti Nitobin vuonna 2011 ja myöhemmin uudelleenjulkaisi Apache Cordovan avoimena lähdekoodina. Apache Cordova toimii WWW-sovelluskehyksien ja natiivien sovelluskehyksien välisenä linkityskerroksena (Waranashiwar ja Ukey 2018). Apache Cordovan avulla Ionic ja muut Apache Cordovan päälle rakennetut ohjelmistokehykset voivat toimia syvemmällä tasolla kuin tyypillinen WWW-sovellus, sillä niille tarjou-

tuu rajapinta suoraan natiiveihin sovelluskehyksiin.

2.3.2 Angular

Angular on Googlen Angular Teamin ylläpitämä TypeScript-pohjainen käyttöliittymäkehys. Se on jatkoa aiemmin ilmestyneelle AngularJS-kehykselle. Alkuperäinen AngularJS-kehys ilmestyi vuonna 2010, ja se oli ensimmäinen valmis ratkaisu dynaamisten HTML-sivujen rakentamiseen mahdollistaen tehokkaamman yhden sivun WWW-sovellusten rakentamisen. Angular julkaistiin vuonna 2016, ja se on kokonaan uudelleenohjelmoitu versio AngularJS:stä.

AngularJS perustuu MVC-arkkitehtuuriin, kun taas uudempi Angular on komponenttipohjainen. Komponenttipohjainen arkkitehtuuri pyrkii tarjoamaan paremman uudelleenkäytettävyyden, luettavuuden, testattavuuden ja ylläpidettävyyden. Ohjelma jaetaan itsenäisiin komponentteihin, joita voidaan käyttää useasti ja niiden itsenäisyys helpottaa yksikkötestaamista. Itsenäiset komponentit ovat huomattavasti helpommin ymmärrettävissä ja niiden korvaaminen on joustavampaa, mikä parantaa ylläpidettävyyttä. (AltexSoft 2018) Angularin komponenttipohjainen rakenne perustuu kolmeen hiljattain ilmestyneeseen teknologiaan: Web-komponentteihin (engl. Web Components), JavaScriptin ES2015-standardiin ja TypeScript-ohjelmointikieleen.

Web-komponentit on laaja-alainen termi, jolla tarkoitetaan neljää WWW-selaimissa yleistyvää standardia: kustomoitavat elementit (custom elements), varjo-DOM (shadow DOM), mallit (templates) ja HTML-tuonti (HTML imports). Kustomoitavat HTML-elementit ovat HTML-standardielementtien ulkopuolisia elementtejä, joita voidaan käyttää standardielementtien seassa. Kustomoitava HTML-elementti irroittaa komponentin sivun muista osista, joten se mahdollistaa komponentin eristämisen. Varjo-DOM on piiloitettu osa sivua, jolla on oma eristetty ympäristö skripteille, CSS-tyylitiedostoille ja HTML-elementeille. Varjo-DOM:n elementit ja tyylit eivät vaikuta varjo-DOM:n ulkopuolisiin alueisiin ja vastavuoroisesti muut sivun elementit ja tyylit eivät vaikuta varjo-DOM:n alaisiin osiin. Komponentti voi täten käyttää tätä eristettyä aluetta renderöintiin. Mallit ovat HTML-palasia, jotka eivät lataudu välittömästi HTML-sivun auetessa vaan ne voidaan aktivoida JavaScriptillä myö-

hemmin. Malleista on useita toteutuksia eri kehyksissä, mutta Web-komponentit standardisoivat mallien rakentamisen ja tarjoavat suoran tuen niiden hyödyntämiselle selaimessa. Malleja käyttämällä varjo-DOMiin piiloitetusta sisällöstä voidaan tehdä dynaamista. Viimeinen web-komponenttien osa on HTML-tuonti. HTML-tuonnin avulla HTML-, CSS- ja JavaScript-tiedostoja voidaan ladata yhtenäisinä osina. Angular ei käytä HTML-tuontia vaan se käyttää JavaScriptin moduulilatausta. (Arora ja Hennessy 2018)

2.3.3 TypeScript

Angular-ohjelmointikehyksen avulla ohjelmointiin käytetään TypeScript-ohjelmointikieltä. TypeScript on avoimen lähdekoodin ohjelmointikieli, jota ylläpitää Microsoft. TypeScript kääntyy suoritusvaiheessa JavaScriptiksi, ja se on tarkoitettu tukemaan JavaScript-ohjelmien kehitystä lisäämällä siihen ominaisuuksia. TypeScript sisältää ES 2015 -standardin mukaiset ominaisuudet ja lisäksi se tarjoaa ohjelmoijan käyttöön tyypit ja koristelijat.

TypeScriptin on tarkoitus tarjota Microsoftin .NET-ympäristöön tottuneille ohjelmoijille oliolähtöisempi lähestymistapa JavaScript-kehitykseen (Maharry 2013). JavaScript on suosittu ohjelmointikieli, mutta varsinkin ohjelmiston lähdekoodin määrän kasvaessa sen heikkoudet alkavat tulla esiin. TypeScript pyrkii puuttumaan näihin ongelmiin tarjoamalla ohjelmoijalle moduulijärjestelmän, luokat, rajapinnat ja staattisen tyypityksen (Gavin Bierman ja Torgersen 2014).

Ohjelmointikielen modulaarisuudella tarkoitetaan sitä, että muuttujat, funktiot, luokat ja muut vastaavat ohjelmointikielen perusrakenteet ovat olemassa vain moduulien sisällä, ellei niitä erikseen esitellä muille moduuleille (Microsoft 2019c). - TODO: lisää pohdintaa modulaarisuuden edut?

Perinteisesti JavaScript on käyttänyt uudestikäytettävien komponenttien rakentamiseen funktioita ja prototyyppipohjaista perintää. Iso osa nykyohjelmoijista ei kuitenkaan ole tottunut käyttämään edellä mainittuja keinoja, vaan nykyisin suosituin lähestymistapa uudelleenkäytettävyyteen on luokkien käyttäminen. (Microsoft 2019a) - TODO: luokkien edut vrt. js?

Rajapintoja käytetään kuvaamaan luokkien ominaisuuksia. TypeScriptissä rajapintoja käytetään tyyppien nimeämiseen ja niiden avulla voidaan tehdä olioiden välisiä sopimuksia ohjel-

moijan itse laatiman ohjelmistokoodin sisällä sekä myös ohjelmoijan oman ohjelmistokoodin ja muiden tuottamien kirjastojen välillä. (Microsoft 2019b) Esimerkiksi Swagger-niminen työkalu generoi .NET-palvelinkoodista valmiita rajapintoja, joita TypeScript-selainpuolen on helpompi käyttää. - TODO: rajapintojen edut vrt. js?

TypeScript on staattisesti tyypitetty kieli toisin kuin JavaScript. Staattisen tyypityksen avulla ohjelmoija näkee suoraan, että ohjelman tieto on oikean tyyppistä sitä käsitellessä. - TODO: ducktyping - TODO: staattisen tyypityksen edut vrt. js?

2.4 Käytettävyyden perusteet

Kaikki ihmisen tuottamat esineet ja asiat täytyy suunnitella. Vaikka ihmiset ovat suorittaneet aktiivista suunnittelua esihistoriallisista ajoista saakka, on tietoisten suunnitteluprosessien tutkimus verrattain tuoretta. Nykyinen suunnittelun kenttä voidaan jakaa karkeasti kolmeen eri osaan: teollinen suunnittelu, käytettävyyssuunnittelu ja kokemussuunnittelu (Norman 2013). Kommunikaatiossa avustavan sovelluksen suunnittelussa tulisi olla erityisen kiinnostuneita käytettävyyssuunnitelusta.

Hyvin suunniteltu sovellus on miellyttävä käyttää ja ohjaa käyttäjää käyttämään sovellusta oikealla tavalla. Tämä on erityisen tärkeää sovelluksissa, joita on tarkoitus käyttää useasti päivittäin.

- TODO: tyypillisimmät käytettävyystekijät DoET:sta tai muusta perusteoksesta käyttäen muitakin lähteitä

2.4.1 Käytettävyys ja autismi

Autismi vaikuttaa monella tapaa ihmisen kykyyn havainnoida ympäristöä ja sen myötä kykyyn käyttää sovelluksia. Iso-Britannian The National Autistic Society listaa verkkosivuillaan autistisille ihmisille sopivien verkkosivujen visuaalisesta toteutuksen päävaatimukset (Society 2018). Listauksessa painotetaan erityisesti visuaalisen ilmeen selkeyttä, staattisuutta ja yksiselitteisyyttä. Lisäksi koekäyttäjien roolia korostetaan.

Symbolien käyttöä tulisi välttää - TODO: lähde

Yksi varsin vähän tutkittu seikka on autismin vaikutus ihmisen suosikkiväreihin. Grandgeorgen ja Masatakan (Grandgeorge ja Masataka 2016) mukaan autistiset lapset pitävät erityisesti vihreästä väristä. Keltaista ja ruskeaa tulisi välttää. Edellä mainitun tutkimuksen otoskoko oli kuitenkin pieni, joten tuloksia ei voi yleistää. Lapset pitävät erityisesti pääväreistä.

- TODO: etsi lisää lähteitä, .txt-tiedostossa oli jotain

3 Kommunikaatiossa avustavan sovelluksen suunnittelu ja toteutus

- TODO: Siltaava kappale, jossa alustetaan aliotsikot.

3.1 Ohjelmistotekninen näkökulma

- TODO: Tekninen kuvaus ohjelmasta.

Sovellus koostuu valikkorakenteesta, korttien luontinäkymistä ja kommunikaationäkymästä.

3.1.1 Valikot ja navigaatio

Valikot Ionicin standardielementeistä luotu.

3.1.2 Korttien luontinäkymä

Korttien luontinäkymä Voidaan tehdä erikokoisia kortteja. Käytetään Papu.net-sivuston vapaata kuvapankkia.

3.1.3 Kommunikaationäkymä

Kommunikaationäkymä Valitaan kortti. Kirjataan viesti symboleja painamalla.

3.2 Käytettävyysnäkökulma

- TODO: Kerrotaan käytettävyyden näkökulmasta. Erityisesti erikoisryhmien käytettävyys nostetaan esille.

Tutkimuksessa toteutettu sovellus on tarkoitettu erityisesti autistisille käyttäjille.

4 Toteutuneen sovelluksen arviointi

Kuvien käsittely ja tallentaminen haastavaa. Progressiiviset sovellukset kohtalaisen hyviä tähän käyttötarkoitukseen eli yksinkertaisiin mobiilikeskeisiin sovelluksiin.

5 Pohdinta

– Mitä aineistoanalyysi kertoo tutkimushypoteesista? Tukevatko tulokset teorialuvussa esitettyjä näkemyksiä? Vastasiko tutkimusmetodi odotuksia? Voiko tapaustutkimuksen pohjalta tehdä yleistyksiä?

Pohdintaluvussa harjoitetaan itsenäistä ajattelua. Sen voi periaatteessa kirjoittaa ilman lähdekirjallisuutta, sillä tarkastelun kohteena on aiemmin kirjoitettamasi materiaali. Nyt tutkimuksen tuloksia on aika arvioida kriittisesti. Niitä verrataan aiempaan tutkimukseen ja pohditaan, mitä uutta tutkimus paljastaa aiheesta ja miten tulokset suhteutuvat aiempaan tutkimukseen – tukevatko vai ovatko ristiriidassa? Jos analyysivaiheessa on ollut vaikeuksia, niitä ei pidä lakaista maton alle; niiden puntaroiminen kertoo tutkimuksen läpinäkyvyydestä ja tulosten luotettavuudesta. –

6 Yhteenveto

– Tutkielman viimeinen luku on Yhteenveto. Sen on hyvä olla lyhyt; siinä todetaan, mitä tutkielmassa esitetyn nojalla voidaan sanoa johdannon väitteen totuudesta tai tutkimuskysymyksen vastauksesta. Yhteenvedossa tuodaan myös esille tutkielman heikkoudet (erityisesti tekijät, jotka heikentävät tutkielman tulosten luotettavuutta), ellei niitä ole jo aiemmin tuotu esiin esimerkiksi Pohdinta-luvussa. Tässä luvussa voidaan myös tuoda esille, mitä tutkimusta olisi tämän tutkielman tulosten valossa syytä tehdä seuraavaksi.

Jos Yhteenveto alkaa pitkittyä, se kannattaa jakaa kahtia niin, että tulosten tulkinta otetaan omaksi Pohdinta-luvukseen, jolloin Yhteenvedosta tulee varsin lyhyt ja lakoninen.

Yhteenvedon jälkeen tulee \printbibliography-komennolla laadittu lähdeluettelo ja sen jälkeen mahdolliset liitteet. –

Lähteet

AltexSoft. 2018. “The Good and the Bad of Angular Development”. Viitattu 29. joulukuuta 2018. https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/.

AppInstitute. 2017. “A Beginner’s Guide to Progressive Web Apps”. Viitattu 3. heinäkuuta 2018. https://appinstitute.com/a-beginners-guide-to-progressive-web-apps/.

Arora, Chandermani, ja Kevin Hennessy. 2018. *Angular 6 by Example.* 3. painos. Livery Place, 35 Livery Street, Birmingham, UK: Packt Publishing.

Divante. 2018. “PWA Design Challenges”. Viitattu 30. kesäkuuta 2018. https://divante.co/blog/pwa-design-challenges/.

Gavin Bierman, Martin Abadi, ja Mads Torgersen. 2014. “Understanding TypeScript”. *Lecture Notes in Computer Science* 8586:257–281. doi:978-3-662-44202-9_11.

Google. 2018. “Progressive Web Apps”. Viitattu 26. kesäkuuta 2018. https://developers.google.com/web/progressive-web-apps/.

Grandgeorge, Marine, ja Nobuo Masataka. 2016. “Atypical Color Preference in Children with Autism Spectrum Disorder”. *Frontiers in Psychology* 7. doi:10.3389/fpsyg.2016.01976.

Ionic. 2019. “Ionic Documentation”. Viitattu 4. tammikuuta 2019. https://ionicframework.com/docs.

Maharry, Dan. 2013. *TypeScript Revealed.* 1. painos. New York City, NY: Apress.

Microsoft. 2019a. “TypeScript Documentation - Classes”. Viitattu 17. tammikuuta 2019. https://www.typescriptlang.org/docs/handbook/classes.html.

———. 2019b. “TypeScript Documentation - Interfaces”. Viitattu 20. tammikuuta 2019. https://www.typescriptlang.org/docs/handbook/interfaces.html.

———. 2019c. “TypeScript Documentation - Modules”. Viitattu 17. tammikuuta 2019. https://www.typescriptlang.org/docs/handbook/modules.html.

Norman, Don. 2013. *The Design of Everyday Things.* 3. painos. New York City, NY: Basic Books.

Nunes, Débora R. P. 2008. “AAC Interventions for Autism: a Research Summary”. *International Journal of Special Education* 23 (2): 17–26.

Sigafoos, Jeff, ja Erik Drasgow. 2001. “Conditional Use of Aided and Unaided AAC: A Review and Clinical Case Demonstration”. *Focus On Autism And Other Developmental Disabilities* 16 (3): 152–161.

Society, The National Autistic. 2018. “Designing Autism-friendly Websites”. Viitattu 28. kesäkuuta 2018. https://www.autism.org.uk/professionals/others/website-design.aspx.

Waranashiwar, Juilee, ja Manda Ukey. 2018. “Ionic Framework with Angular for Hybrid App Development”. *International Journal of New Technology and Research* 4 (5): 1–2.

Vaughn, Bobbie, ja Robert Horner. 1995. “Effects of Concrete Versus Verbal Choice Systems on Problem Behavior”. *Augmentative and Alternative Communication* 11 (2): 89–92. doi:10.1080/07434619512331277179.

Liitteet