a) Ohjelmistokehitys

- Tällä kurssilla syvennytään tutustumaan, mitä sisältää ohjelmistotuotanto. Ohjelmistotuotanto sisältää laajasti kaikkea, mitä tarvitaan onnistuneen ohjelmistotuotteen teossa aina ihan alusta aina siihen asti kun ohjelmiston tuki lakkaa. Tämän vuoksi tämän kurssin sisällysluettelossa on ohjelmistokehityksestä, algoritmeista, kehitysympäristöistä, ohjelmoinnista (tässä JavaScript), ohjelmistojen elinkaaresta ja lopuksi kehitysmenetelmistä.
- Ensimmäiseksi tutustumme, mitä ohjelmistokehitys pitää sisällään. Ohjelmistokehitys pitää sisällään melko laajasti erilaisia asioita, kuten kattavasta Wikipedia materiaalista käy selville (https://en.wikipedia.org/wiki/Software_development). Lyhyesti esitettynä ohjelmistokehityksessä tutkitaan millainen sovellus kannattaisi suunnitella ja toteuttaa. Kun uuden ohjelmiston tarpeellisuudesta ollaan päästy selville, voidaan aloittaa sen suunnittelu ja toteuttaminen (ohjelmointi). Usein samaan aikaan kannattaa myös tarkistaa, että tehty ohjelma myös toimii riittävällä tavalla, eli se kannattaa testata. Näin edetään kunnes sovellus on siinä vaiheessa, että se voitaisiin julkaista asiakkaalle tai asiakkaille. Tästä eteenpäin usein sovelluksesta julkaistaan uusia versioita, joissa tulee uusia ominaisuuksia sovellukseen. Työvaiheita on siis melko paljon. Lisäksi tarvitaan useita eri aloihin erikoistuneita, jotta voidaan nopeuttaa kehitystyötä. Tarvitaan vaatimusmäärittelyä, ohjelmointia, testausta ja dokumentointia ja nämä ovat vain muutamia tehtäviä, joita tarvitaan ohjelmiston teossa. On siis selvää, että tarvitaan systemaattista otetta kaikkeen tähän. Kyseessähän on usein monimutkainen ohjelmisto, joka lisäksi usein muuttuu matkan varrella. Lisäksi on useita henkilöitä tekemässä tätä kaikkea ja nämä mukana olevat henkilöt voivat hekin muuttua ohjelmiston kehityksen aikana. Tämän vuoksi tämän kurssin loppu puolella perehdymme ohjelmiston elinkaareen ja ohjelmistokehityksen menetelmiin.

b) Ohjelmointi

- Uutta sovellusta tehtäessä tai jo olemassa olevaa sovellusta päivitettäessä, tarvitaan tietysti ohjelmointia, jotta halutut ominaisuudet saadaan toteutettua. Tällä kurssilla perehdymme ohjelmointiin JavaScriptin avulla. JavaScript on syntaksin eli kirjoitettavan ohjelmakoodin puolesta hyvin samankaltainen C –kielen ja Java –kielen kanssa. Siinä mielessä muiden kielten oppiminen on ainakin tämän osalta ehkä helpompaa. JavaScript on kuitenkin kielenä toteutettu hyvin erilailla C –kieleen ja myös Java –kieleen verrattuna. Tavallaan konepellin alla on ihan erilainen moottori kuin muilla kielillä. Ensinnäkin JavaScript tulkkaa kirjoitetun ohjelmakoodin komento komennolta, jota nykypäivänä kaikki selaimet pystyvät ymmärtämään (eli ajamaan muutettuaan sen ensin konekieliseksi). Esimerkiksi C –kieli ja Java –kieli ovat tässä täysin erilaisia, niiden lähdekoodi käännetään kääntäjällä kerralla tiedostoon, jota vasta pystytään ajamaan tietokoneella. Tämä vaikuttaa koodaamiseen eli ohjelmointiin, jonka vuoksi meille on tärkeää tietää tulkattavan kielen ja käännettävän kielen erot.

- Tulkkeihin (interpreter) olet mahdollisesti tutustunut linux –käyttöjärjestelmän merkkipohjaisessa komentokehotteessa (shell:issä). Siinähän kirjoitetaan ensin komento ja sen jälkeen komentotulkki ottaa komennon heti käsittelyyn ja annetaan heti vastaus. Esimerkiksi jos Linux käyttöjärjestelmässä kirjoittaa komentokehotteeseen komennon `ls` eli listaa kansiossa olevat kaikki tiedostot ja alikansiot, suorittaa komentotulkki heti annetun komennon. MS-DOS:ssa, linux:ssa tämä on kätevä ja nopea tapa käyttää käyttöjärjestelmää. JavaScript on kuitenkin täysiverinen ohjelmointikieli, joten tarvitsemme asianmukaiset työkalut. Tarvitsemme tekstieditorin, jolla voimme kirjoittaa niin sanottua puhdasta ascii koodia ja tallentaa sen sitten tiedostoon. Esimerkiksi Word ei kelpaa, sillä se lisää kaikenlaisia ohjausmerkkejä sivutuksesta, fonttikoosta yms. tiedostoon. Tällainen tiedosto ei kelpaa lähdekoodia sisältäväksi tiedostoksi. Tarvitsemme editorin, jollainen on esimerkiksi Visual Studio Code, Sublime, Notepad++ jne. Nämä

soveltuvat nettisivujen tekemiseen, jossa on html, css ja JavaScript koodia. Juuri tarkalleen se mitä tarvitsemme editorilta tällä kurssilla.

- Seuraava tärkeä työkalu, jonka tarvitsemme on siis selain. Itseasiassa kaikki ne selaimet, joita käyttäjät käyttävät. Tällä hetkellä niitä ovat Chrome, Firefox, Opera, Safari (Apple maailmassa) ja Microsoftin Edge. Muitakin on, mutta ainakin nämä kannattaa ottaa huomioon. Tällä kurssilla keskitymme melko pitkälle Chromeen.

- Editori ja selain eivät vielä riitä. Tarvitaan myös Web-palvelin, jotta voimme ajaa tällä kurssilla tekemiämme JavaScript ohjelmia. Selaimilla voi kyllä testailla JavaScriptia ilman palvelinta ja niin teemmekin usein. Web –palvelin on kuitenkin oleellinen olla olemassa Web sovelluskehittäjällä. Koulullamme Web –palvelin on Apache Tomcat, joka on asennettu Linux –palvelimelle. Sen käyttäminen on tehty niin helpoksi kuin se on ollut mahdollista. Riittää kun etsii Windows koneelta File Explorer:illa M –aseman. Siirtyy sinne ja etsii sieltä kansion, jonka nimi on public_html. Kun tähän kansioon kopioi html tiedoston, joka sisältää JavaScriptiä, voi sitä ajaa selaimessa ja Apache Tomcat hoitaa palvelemisen. Sivut näkyvät tietysti myös Internetiin. Opettelemme käyttämään tätä Viopessa löytyvissä tehtävissä.

- Web sovelluskehittäjän tietokoneelta on siis löydyttävä ainakin yllä listatut työkalut. Hyvä olisi niiden lisäksi olla myös testaukseen sopiva kirjasto jne. Mutta näillä työkaluilla pääsee hyvään alkuun ja ne ovat ne joita tulemme tarvitsemaan tällä kurssilla.

c) HTML (Hypertext markup language)

- JavaScript osuus alkaa tällä kurssilla 4. viikon vaiheilla. Kuten edellä tuli jo mainittua, JavaScriptiä ei käännetä, vaan se tulkataan selaimessa. Sen vuoksi joudumme tekemään html kielisen tiedoston, jossa on viittaus JavaScript tiedostoon. Tämä on siis se perusasetelma, joka toistuu kaikissa JavaScript tehtävissä. Sen vuoksi on hyvä katsoa jo tässä vaiheessa niitä html kielisiä elementtejä, joita tarvitsemme.

  Ensimmäiseksi siis tarvitsemme tehdä vaikkapa Visual Studio Code:lla ascii tiedoston, jonka sisältönä on html kielen peruselementti eli html. Se kertoo selaimelle kyseessä on html sivu:

```
<html>
</html>
```

  Jos haluat kertoa selaimelle, että kyseessä on html5 version html sivu, niin lisää ensimmäiselle riville määrittely:

```
<!DOCTYPE html>
<html>
</html>
```

  Seuraavaksi voimme lisätä html sivun perusrakenteen, joka kaikissa web –sivuissa pitäisi olla:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

head elementtiin tulee nettisivun otsikkotason tiedot, kuten esimerkiksi title, joka näkyy selain ikkunan otsikko kentässä. Lisäksi head elementin sisälle voisi lisätä meta elementin avulla tiedon käytettävästä merkistöstä. Tällä hetkellä yllä oleva nettisivumme ei osaisi esimerkiksi ä,ö ja å merkkejämme. Me haluamme, että osaa, joten lisäämme elementin <meta charset=”utf-8”/>. utf-8 tarkoittaa kansainvälistä merkistöä, jossa tarvittavat merkkimme ovat.

```
<!DOCTYPE html>
<html>
    <head>
              <title>Kotisivulleni</title>
              <meta charset = "utf-8"/>
    </head>
    <body>
    </body>
</html>
```

Koska haluamme tehdä JavaScript ohjelman, tarvitsemme myös script elementin. Sen src (source) attribuutilla voimme kertoa mistä JavaScript tiedosto löytyy. Tätä tarvitsemme vasta 4 viikon vaiheilla, joten emme lisää vielä sitä nettisivullemme.

Voimme jatkaa body elementiin, jonne tulee selaimessa näkyvät tekstit, kuvat, elementit jne. Esimerkiksi tekstin lisääminen on helppoa. Kirjoittaa haluamansa tekstin boby elementin sisälle:

```
<!DOCTYPE html>
<html>
    <head>
              <title>Kotisivulleni</title>
              <meta charset = "utf-8"/>
    </head>
    <body>
    Tervetuloa kotisivulleni

    </body>
</html>
```

Nyt ensimmäistä kertaa voisi kokeilla, miltä tiedoston sisältö näyttää nettiselaimessa. Tallenna html tiedosto johonkin sopivaan kansioon M –asemalle (tallentuu siellä omaan ”kotihakemistoosi”). Anna tiedostolle jokin sopiva nimi, jonka päätteenä on .html. Muutoin tekemäsi tiedosto ei toimi ihan niin kuin pitäisi selaimessa. Käynnistä seuraavaksi vaikkapa Chrome. Raahaa hiirellä File Explorer:sta (File Manager) nettisivusi sisältämä tiedosto ja tipauta se Chromeen. Sivusi pitäisi näkyä nyt selaimessa.

Nettisivujen tekemisestä on kurssin Moodle sivulla myös opetusvideoita. Niissä esitellään lisää elementtejä, joita voisit tarvita Viopesta löytävän html tehtävän tekemiseksi.

d) Kääntäjä (Compiler) ja tulkki (interpreter)

Tiedätkö mitä tarkoitetaan kääntäjällä (compiler) ohjelmoinnissa? Kääntäjä poikkeaa edellä esitellystä tulkista (interpreter) siten, että kääntäjälle kirjoitetaan valmis ohjelma koodi, joka käännetään kerralla kääntäjällä usein binäärikoodiksi. Binäärikoodia tarvitaan, että sovellutusta voidaan ajaa tietokoneessa. Tulkki puolestaan tulkkaa vaikkapa JavaScriptin ohjelma koodia rivi riviltä ja antaa heti tuloksen tulkkauksesta. Tosin tämän tekee tehtävissämme selain, joten useimmiten emme huomaa tätä vaihetta.

e) Ohjelmointi (programming)

Ohjelmoinnilla tarkoitetaan tässä ohjelmakoodin kirjoittamista, jotta saadaan tehtyä haluttu sovellus. Tämän kurssin 4. viikosta lähtien toteutamme pieniä sovelluksia JavaScriptillä web sivuille, joten tästä tulee myöhemmin runsaasti kokemusta.

f) Ohjelmistokehitys (Software Development)

Ohjelmistokehitys kattaa laajemmin asioita kuin ohjelmointi. Ohjelmistokehitys pitää sisällään vaatimusmäärittelyn, ohjelmiston suunnittelun, toteutuksen eli ohjelmoinnin, testauksen, julkaisun ja ylläpidon. Tässä ohjelmointi on siis yhtenä vaiheena mukana. Vaatimusmäärittelyssä puolestaan selvitetään asiakkaan tarpeet tulevalle sovellukselle. Tämän jälkeen voidaan suunnitella, miten tuleva sovellutus kannattaisi toteuttaa. Ohjelmointi tehdään siis tämän jälkeen. Tietysti edellyttäen, ettei huomata, että asiakkaan vaatimukset ovat muuttuneet, jolloin tilanne täytyy tietysti korjata. Ohjelmoinnin jälkeen testataan sen toiminta. Näin ohjelman määrittely ja ohjelmointi etenee ikäänkuin pala palalta eteen. Näitä paloja ovat usein sovellutuksen ominaisuudet. Kun sovellutusta on tehty riittävän kauan, valmistuu se julkaisu kuntoon. Asiakkaat voivat nyt tilata kyseistä ohjelmistotuotetta. Ylläpito pitää huolen, että mahdolliset ohjelmistovirheet korjataan ja tietysti kehitetään tuotetta eteenpäin uusin ominaisuuksin. Siten taataan, että ohjelmisto kiinnostaa käyttäjiä edelleen ja he ovat valmiita hankkimaan sen uusimman version.

g) Sovellutus (Software Application)

Esimerkkejä sovellutuksista voisivat olla esimerkiksi Microsoftin Word, jota käytetään tekstinkäsittelyyn. Muita esimerkkejä voisivat olla erilaiset peliohjelmat, jotka nekin ovat sovellutuksia. Sovellutus on siis ohjelmisto, joka on tehty jonkin tietyn asian tekemiseen. Jotkut sovellutukset voivat olla hyvin laajojakin ominaisuuksilta, kuten esimerkiksi Excel. Mutta voivat ne olla pieniäkin, kuten vaikkapa Paint piirto-ohjelma.