Joni Kamppila

TOIMINNALLINEN NETTISIVU

Tietotekniikan koulutusohjelma

2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

TOIMINNALLINEN NETTISIVU

Kamppila, Joni
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Kesäkuu 2017
Sivumäärä: 31
Liitteitä: 1

Asiasanat: WWW-sivut, verkko-ohjelmointi, WWW

Opinnäytetyön aiheena oli toiminnallinen nettisivu. Opinnäytetyössä opastettiin, miten tehdään toiminnallinen nettisivu aina webhotellin hankkimisesta sen julkaisuun saakka, sekä mitä tulee ottaa huomioon, kun alkaa suunnittelemaan sivustoa. Opinnäytetyössä tehtiin nettisivut osoitteeseen www.jkamp.arkku.net/index.php.

Teoriaosuudessa käytiin lävitse kaikkien ohjelmointikielien, joita nettisivustoa luodessa käytettiin, keskeisimmät merkitykset nettisivuston kannalta. Opinnäytetyö tehtiin, jotta kaikki, keitä kiinnostaa nettisivujen tekeminen taikka ylipäätään se, miten nettisivut luodaan, saisivat käsityksen kyseisestä aiheesta.

Opinnäytetyössä käsiteltiin myös nykyaikaiset julkaisu- ja sisällönhallintajärjestelmät, joita ovat Joomla sekä WordPress.

Opinnäytetyössä käytettiin paljon esimerkkejä nettisivuston lähdekoodista, jotta asian hahmottaminen olisi lukijalle helpompaa ja selvempää. Kommentoituja koodinpätkiä on löydettävissä enemmän nettisivulta.

Opinnäytetyössä käytiin myös lävitse yleisimmät tietoturvaongelmat, jotka liittyvät nettisivuihin. Tietoturvaongelmista annettiin esimerkkejä sekä ratkaisuja.

FUNCTIONAL WEB SITE

Kamppila, Joni
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
June 2017
Number of pages: 31
Appendices: 1

Keywords: Web pages, web-programming, WWW

This thesis subject was functional web site. This thesis guides, how to make a functional web site from the beginning to the end. This thesis also gives advices, what should be considered before starting to make a web site. In this thesis there was made a web site, which can be found at www.jkamp.arkku.net/index.php.

This thesis introduces every programming languages, which was used to build the web site. The purpose of this thesis was to make a guide that everybody who is interested in making web sites or who is interested in how web sites are build, can get an impression of that subject.

The thesis also contains the most popular content management systems Joomla and WordPress.

In this thesis there was used plenty of web site's source code as an example to make all the codes and programming techniques clearer and easier to understand. More commented code examples can be found at the web site.

This thesis also includes the most common information security problems that are in various web sites. There was given examples and solutions from these information security problems.

LYHENTEET JA KÄSITTEET

HTML	Standardoitu kuvauskieli verkkosivujen luomiseen (Hypertext Markup Language).
CSS	Tyylikieli, jolla hallitaan HTML – dokumenttien tyylimäärittelyitä (Cascading Style Sheets).
JavaScript	Skriptikieli, jota käytetään muun muassa webbikoodaamisessa.
Joomla	Avoimen lähdekoodin julkaisujärjestelmä.
jQuery	Yksi JavaScript lukuisista kirjastoista.
MySQL	Yksi monista erilaisista tietokannoista.
PHP	Ohjelmointikieli, jota käytetään esimerkiksi palvelimien ohjelmoinnissa (Hypertext Preprocessor).
Responsiivisuus	Nettisivujen skaalautuvuus erikokoisille päätelaitteille.
SQL-injektio	Tietoturvahaavoittuvuus, jolla on mahdollista esimerkiksi sisäänkirjautuminen ilman oikeanlaisia tunnuksia.
Tagi	HTML – elementtien luonnissa käytetään tageja, esimerkiksi <table> on tagi.
WordPress	Avoimen lähdekoodin sisällönhallintajärjestelmä.
XSS-aukko	Tietoturvaongelma, jossa hyökkääjä upottaa omaa lähdekoodia sivuston syötteen kautta.

SISÄLLYS

1 JOHDANTO

Nettisivujen olemassa olo on nykypäivänä erityisen tärkeää niin yrityksille, kuin jopa yksityisille ihmisille. Ensivaikutelma uudesta tai muuten tuntemattomasta yrityksestä pohjautuu pitkälti yrityksen nettisivujen ulkoasuun. Myös web-sovellukset ovat nykypäivänä erittäin suuressa huudossa, johtuen sen tuomasta laiteriippumattomuudesta.

Tämä opinnäytetyö kertoo nettisivun tekemisen vaiheet alusta loppuun saakka. Opinnäytetyössä käydään läpi, miten nettisivut saadaan skaalautumaan erikokoisille päätelaitteille, miten rekisteröityminen ja sisäänkirjautuminen toteutetaan sivustolle sekä miten erilaisia toimintoja tehdään nettisivulle.

Nykyään tietoturvaongelmat ovat tulleet yhä suuremmaksi puheenaiheeksi, minkä takia kerron myös SQL – injektiosta sekä XSS – aukosta, jotka ovat yleisimpiä tietoturvaongelmia nettisivuilla.

Tähän opinnäytetyöhön pohjautuva nettisivu on katseltavissa osoitteessa www.jkamp.arkku.net/index.php.

2 WEBHOTELLI

2.1 Webhotelli

Webhotelli on palvelu, jonka avulla asiakas voi julkaista nettisivuja. Palveluntarjoaja tarjoaa kiintolevytilaa, jonne asiakas voi omat kotisivunsa laittaa. Käyttöönsä asiakas saa www-sivut sekä oheispalveluita, kuten sähköpostiosoitteen. Myös verkkotunnus (domain) kuuluu webhotelliin. (Haglund 2014)

2.1.1 Webhotellin hankinta

Palveluntarjoajia, jotka tarjoavat webhotelleja, on nykyään todella paljon. Tarjontaa on siis paljon, joten kannattaakin miettiä, kannattaako ottaa halvin palveluntarjoaja, vai tutkia eri vaihtoehtoja ja valita hyvämaineinen palveluntarjoaja. (Järvilehto 2013, 183–184)

Ennen kuin ostaa webhotellin, on syytä ottaa huomioon erilaisia asioita. Mikäli nettisivusto käyttää MySQL-tietokantaa sekä www-ohjelmointikielenä PHP:ta, kannattaa valita webhotelli, joka käyttää Linux-käyttöjärjestelmää. Aluksi kannattaa myös hankkia webhotelli, jossa levytila ja sallittu liikennemäärä on suhteellisen pieni ja sen jälkeen ruveta seuraamaan nettisivuston liikennemäärää. Mikäli näyttää siltä, että sallittu liikennemäärä kuukaudessa tulee ylittymään, on sallitun liikennemäärän nostaminen mahdollista. Maksullisiin webhotelleihin kuuluu pääsääntöisesti myös sähköpostilaatikko, joten kannattaa webhotellia valitessa laskea, kuinka monta sähköpostilaatikkoa tarvitsee, sillä sähköpostilaatikoiden määrä vaikuttaa suoraan webhotellin kuukausihintaan. Myös tietokantojen määrä kannattaa tarkastaa. Yleensä ottaen jokaiseen webhotelliin kuuluu vähintään yksi tietokanta, mutta mikäli koetaan tarpeelliseksi saada useampi tietokanta, on syytä hankkia toinen webhotelli, joka tässä tapauksessa on hieman kalliimpi. (Haglund 2014)

Suurin osa webhotelleista myy samalla domaineja. Domainilla (verkkotunnus) tarkoitetaan nettisivun nimeä. Verkkotunnuksella on yleensä kiinteä vuosihinta, toisin kuin webhotellilla.

2.1.2 SSL

SSL (Secure Sockets Layer) on suojausprotokolla, jota käytetään henkilökohtaisten sekä sensitiivisten tietojen suojaamiseen silloin, kun tietoja, kuten luottokorttitietoja, salasanoja ja käyttäjätunnuksia käytetään Internetissä. Nettisivut, joiden URL – osoitteessa on HTTPS – alku, käyttävät SSL – suojausta. (SSL 2017)

3 VERKKOSIVUN TOTEUTUS

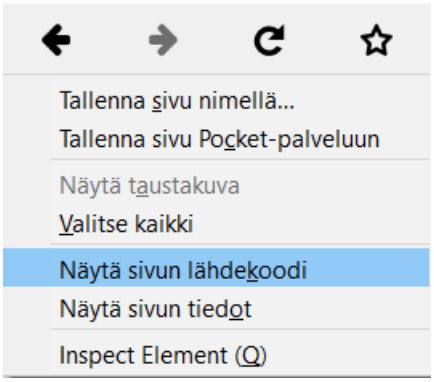
3.1 HTML

HTML on standardoitu hypertekstin merkintäkieli, jolla luodaan nettisivuja. Nettisivun HTML -koodista käy selville sivuston rakenne, joka luodaan erilaisilla HTML elementeillä. Elementit tehdään erilaisilla tageilla, esimerkiksi taulukko -elementin aloitustagi on <table> ja lopetustagi on </table>. Kyseiset tagit eivät kuitenkaan näy Internet – selaimessa kyseisessä muodossa, vaan selain kykenee muuttamaan HTML-tagit erilaisiksi kokonaisuuksiksi, kuten yllä mainittu <table> tagi muuttuu selaimessa taulukoksi. (W3Schools 2017a)

HTML:stä on tällä hetkellä olemassa 6 versiota. Ensimmäinen versio HTML:stä on julkaistu jo vuonna 1991 ja viimeisin, HTML5, on julkaistu vuonna 2014. (W3School 2017a)

HTML elementtiin voidaan lisätä attribuutteja, kuten class, id ja name. Näitä attribuutteja kannattaa lisätä silloin, jos haluaa juuri siihen elementtiin viitata CSS -tiedostossa tai jos haluaa kyseiselle elementille tehdä muutoksia JavaScript – tiedostossa.

Kaikkien nettisivujen HTML -koodi on nähtävillä selaimella. Esimerkiksi Mozilla Firefox – selaimella painamalla sivustoon hiiren oikealla näppäimellä, tulee näkyviin erilaisia toimintoja (Kuva 1). Näytä sivun lähdekoodi -kohdasta tulee näkyviin kyseisen nettisivun HTML -koodi (Kuva 2).



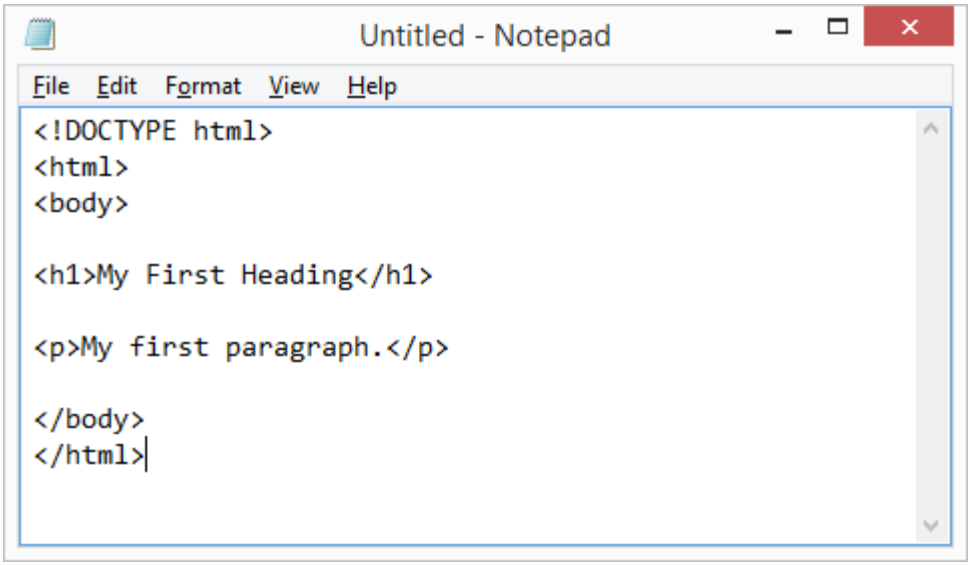
Kuva 1. Hiiren oikeanpuoleisella näppäimellä esiin tulevat toiminnot

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <link type="text/css" rel="stylesheet" href="Content/style.css" />
    <script type="text/javascript" src="Content/jquery-3.1.0.js"></script>
    <script type="text/javascript" src="Content/script.js"></script>
    <title> Opinnäytetyö - Joni Kamppila </title>
```

Kuva 2. Näytä sivun lähdekoodi -kohdan esimerkki

3.1.1 HTML:n käyttötarkoitus

Kuvassa 3 on esillä HTML:n tärkeimmät tagit, joita ilman ei nettisivun tekoa voi aloittaa. Alussa määritetään versiotyyppi selainta varten <!DOCTYPE html> -merkinnällä. Kyseinen merkintä ei kuitenkaan ole HTML -tagi. Vasta tämän merkinnän jälkeen aloitetaan varsinaisilla HTML -tageilla. Kaikki, mitä nettisivu käyttää hyväkseen, kuuluu laittaa <html> -tagin sisälle. Kuvasta puuttuu <html> -tagin jälkeen laitettava <head> -tagi, jossa esitellään mahdollisen CSS – tyylitiedoston sijainti sekä JavaScript -tiedoston sijainti. <Head> -tagin sisään laitetaan myös <title> -tagi, jonka sisältö näkyy Internet selaimen välilehden otsikkona. <Body> -tagin sisäpuolelle laitetaan kaikki, minkä halutaan näkyvän nettisivuilla. Kuvassa näkyvä h1 -elementti on otsikko elementti ja p -elementti on lause elementti. Pääsääntöisesti on erittäin suositeltavaa, että normaali tekstisisältö olisi sijoitettuna edes jonkun elementin sisälle, vaikka teksti tuleekin esille ilman, että se olisi elementin sisällä.



Kuva 3. HTML:n tärkeimmät tagit (W3Schools 2017b)

3.2 Responsiivisuus ja tyylimäärittelyt

Nettisivujen responsiivisuudella tarkoitetaan nettisivun mukautumista eri kokosille näyttöpäätteille. Jos nettisivu on responsiivinen, se skaalautuu hyvin mobiililla tai tabletilla sitä katsoessa. Perinteisesti skaalautuvuuteen on käytetty kahta tapaa, ensimmäinen tapa on, että luodaan nettisivustosta kokonaan eri sivusto, joka toimii mobiilisivustona. Tämä johtaa siihen, että ylläpidettäviä sivuja on vähintään kaksi yhden sijasta. Toinen tapa on luoda jokaiselle mobiilialustalle oma sovellus. Tämä tapa on erittäin työläs. Kokonaisuuden muuttuessa pitää kaikille alustoille tehdä päivitetyt sovellukset.
Nämä tavat kuitenkin korvaa CSS3:n media query. (Leiniö 2012)

Kun aloittaa responsiivisten sivujen tekemistä, kannattaa miettiä etukäteen yleisimmät laitteiden resoluutiot. Tällä hetkellä kyseiset resoluutiot ovat 320px, 480px, 600px, 768px ja 1020px. Kuvassa 4 on esitetty CSS3 media query, jolla määritetään miltä HTML -elementit näyttävät alle 480 pikselisellä näytöllä.

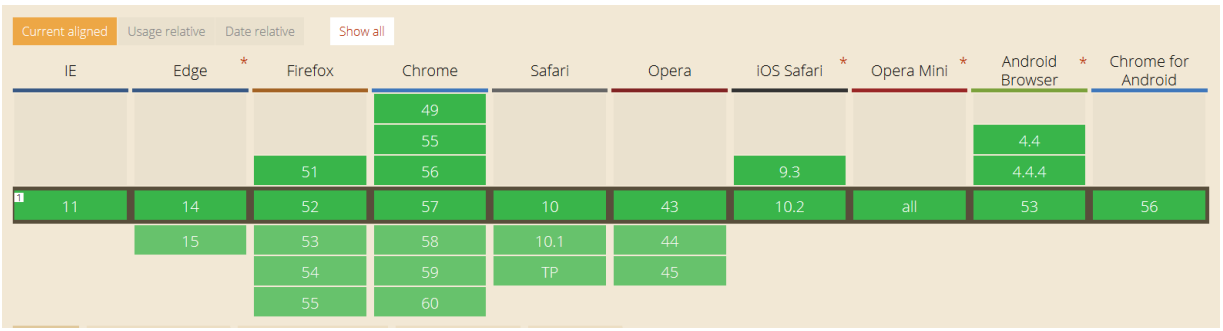
```
/* for 480px or less */
@media screen and (max-width: 480px) {

    #header {
        height: auto;
    }
    h1 {
        font-size: 24px;
    }
    #sidebar {
        display: none;
    }

}
```

Kuva 4. CSS3 media query (Web Designer Wall 2011)

Tällä hetkellä, kun erilaisia selaimia on olemassa lukematon määrä, voidaan todeta, ettei kyseinen CSS3 media query toimi kaikilla mahdollisilla selaimilla. Tähän ongelmaan on kuitenkin ratkaisuja, kuten Wouter van der Graafin tekemää JavaScript – kirjastoa hyväksi käyttäen. Kyseinen kirjasto mahdollistaa media queryn käytön myös Internet Explorer 5-8 selaimilla, Mozilla Firefox 1-3 selaimilla sekä Safari 2-3 selaimilla. (Leiniö 2012) Kuvassa 5 on esitettynä taulukko, jossa on näkyvillä tällä hetkellä CSS3 media queryä tukevat selaimet ja selainten versiot.



Kuva 5. CSS3 media queryä tukevat selaimet (Can I use 2017)

3.2.1 CSS

CSS on yksinkertainen mekanismi lisätä tyylimäärittelyjä HTML -dokumentteihin. Tyylimäärittelyjä ovat esimerkiksi fontin väri sekä koko. (W3C 2017)

CSS:llä voidaan viitata monella eri tavalla HTML -elementtiin. Kuvassa 6 on esiteltynä kolme eri tapaa viitata HTML -elementtiin. Ensimmäisenä viitataan elementtiin, jonka id on menuicon. Id:seen viittaaminen tapahtuu # - merkillä. Kun halutaan viitata kaikkiin samoihin elementteihin, kirjoitetaan CSS:ssä pelkkä elementin nimi viittaukseen, kuten kuvassa 6 oleva a, jolla viitataan kaikkiin a – elementteihin. CSS:llä jos halutaan viitata HTML -elementin luokkaan, käytetään viittauksessa pistettä, kuten kuvassa viitataan luokkaan nimeltä dropdownBtn.

```
#menuicon {
    visibility: visible;
    position: absolute;
    right: 0;
    display: inline-block;
    margin: auto;
    margin-top: 20px;
}
a {
    color: #FFF;
    text-decoration: none;
    font-weight: bold;
}
.dropdownBtn {
    position: relative;
    font-size: 30px;
    color: white;
    border: none;
    cursor: pointer;
    background-color: transparent;
}
```

Kuva 6. CSS viittaukset HTML – elementteihin

3.2.2 Miksi responsiiviset nettisivut

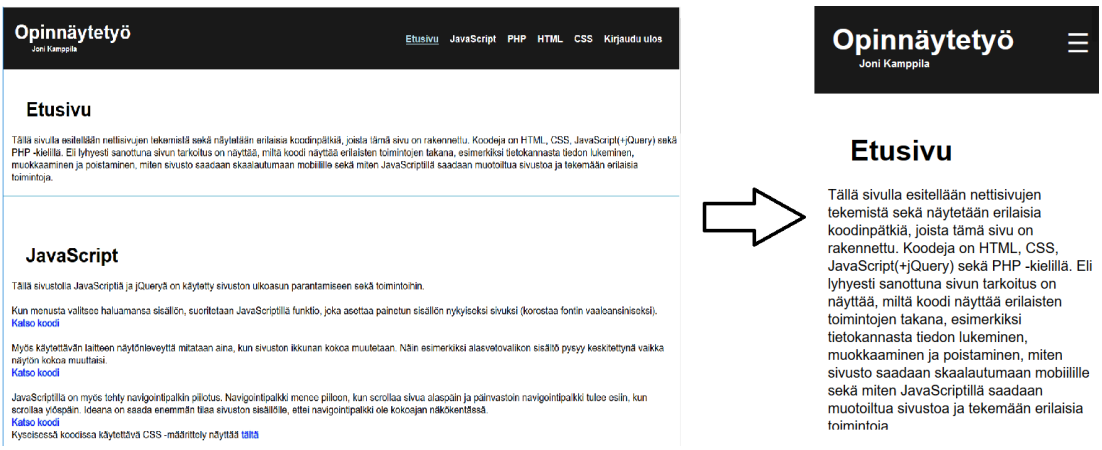
Vaikka nettisivujen responsiivisuus vaikuttaisi mitättömältä asialta yrityksen kannalta, sitä se ei kuitenkaan ole asiakkaan mielestä. Asiakkaan käyttäessä skaalautumatonta nettisivua mobiililla, pilaa se hänen käyttökokemuksensa, eikä anna yrityksestä hyvää kuvaa. Nykyisin 80 % netin käyttäjistä käyttää surffaukseen älypuhelinta, joten luulisi sen jo olevan erittäin pätevä syy hankkia yritykselle responsiiviset nettisivut. (Vilperi 2015)

Asiakkaan näkökulmasta responsiivisien nettisivujen hyödyt ovat selvät. Käydessään nettisivuilla, on mukavampi selailla hyvin skaalautuvia sivuja, kuin alkaa zoomailemaan normaaleja sivuja niin, että niistä saisi jotain selvää. Myös ostopäätöksen teko helpottuu asiakkaalla, kun kaikki tieto on heti näkyvillä, eikä

tarvitse useasta paikasta zoomailla tietoa. Yrityksen kannalta responsiivisuus nostatattaa myyntiä ja Googlen arvostaessa responsiivisia sivuja, sijoitut korkeammalle hakutuloksissa kuin ei-responsiiviset nettisivut. (Vilperi 2015)

Responsiivisuuden ansiosta yritys tarvitsee vain yhdet nettisivut, eikä erillisiä nettisivuja mobiililaitteille. Myös nettisivujen korvaajaksi nousseet mobiilisovellukset eri alustoille ovat jäämässä unholaan responsiivisuuden takia. Mobiilisovelluksien hankkiminen on erittäin kallista, koska jokaiselle alustalle täytyy suunnitella ja tehdä oma sovelluksensa. Suurista mediataloista Suomessa esimerkiksi YLE Uutiset on siirtynyt responsiivisiin nettisivuihin mobiilisovelluksen sijasta. Verkkoanalytiikkaa on myös helpompi seurata, kun yrityksellä on vain yhdet sivut sekä sivuston ylläpitäminen ja päivittäminen on huomattavasti helpompaa ja nopeampaa. (Olander 2014)

Kuvassa 7 on nähtävissä tätä opinnäytetyötä varten tehdystä nettisivusta normaalinäkymä ja mobiilinäkymä. Sivuston responsiivisuus on toteutettu CSS media queryllä.



Kuva 7. Nettisivun normaalinäkymästä mobiilinäkymään

3.2.3 JavaScript ja jQuery

JavaScript on ohjelmointikieli, jonka on kehittänyt Netscape. Modernit selaimet tukevat yleisesti JavaScriptiä, mikäli selaimen asetuksista on JavaScript päällä. JavaScript on käyttöjärjestelmäriippumaton ja kevyt, mikä on JavaScriptin suurin

vahvuus. JavaScript -tulkki suorittaa koodin, minkä vuoksi JavaScript ei tarvitse erillistä kääntäjää, joka kääntäisi koodin ohjelmaksi, ennen kuin se voitaisiin suorittaa. JavaScript ei myöskään tarvitse erillistä ohjelmaa, jolla JavaScriptiä voitaisiin kirjoittaa, vaan melkeinpä mikä tahansa editori ajaa asiansa, riittää kun muistaa vain tallentaa tiedoston .js –muotoon. (Saarikumpu 2017)

JQuery on yksi JavaScriptin monista kirjastoista. JQueryn kaikki metodit koostuvat puhtaasta JavaScript -koodista ja sen tarkoitus on lyhentää JavaScriptin komentoja. Esimerkiksi JavaScriptillä viitattaessa HTML -elementtiin id:n avulla tapahtuu viittaus näin: document.getElementById(id), kun taas jQueryllä samanlaiseen elementtiin viittaaminen on lyhennetty muotoon $(’#id’).

JavaScriptillä ja jQueryllä voi tehdä lukuisia asioita nettisivustolla. Niillä pystyy myös tekemään uusia HTML -elementtejä sekä tehdä CSS – tyylimäärittelyjä. Nykypäivän nettisivustoissa melkein jokainen toiminto tehdään JavaScriptillä. Tätä opinnäytetyötä varten tehdyssä nettisivustossa JavaScriptiä on käytetty seuraaviin asioihin:

- Yläpalkki (header) piilotetaan alaspäin liikkuessa, ja se palautetaan, kun liikutaan takaisin ylöspäin.
- Kun klikataan jokin haluttu sivu auki, muuttuu valitun sivun fontti eriväriseksi.
- Kun sivuston kokoa muutetaan, sivun leveys haetaan JavaScriptin avulla.
- Painaessa menu -painiketta mobiilinäkymässä, tulee alasvetovalikko näkyviin.

Kuvassa 8 oleva JavaScript -koodi asettaa yläpalkista valitun otsikon aktiiviseksi sivuksi. Koodissa luodaan funktio currentPage, jolle annetaan parametriksi id. Kun HTML -elementtiin on annettu attribuutti onclick ja annettu sille arvo currentPage ja sille halutun sisällön id parametriksi, siirtyy nettisivu suorittamaan kyseistä funktiota, kun HTML -elementtiä painetaan. CurrentPage - funktiossa haetaan jQueryä hyväksi käyttäen HTML -elementti, jonka luokka on current ja sijoitetaan se muuttujan curPage arvoksi. Sen jälkeen haetaan puhdasta JavaScriptiä käyttäen HTML -elementti id:n avulla. Kyseinen id on siis sama, kuin painetun HTML -elementin onclick attribuutin arvossa mainittu parametri. Tässä tapauksessa id on siis sama kuin javascr. Nyt kun on selvillä valittu sivu, poistetaan silloisesta aktiivisesta sivusta luokkamääre current ja asetetaan valittu sivu currentiksi. Aktiivisen sivun fontin väri vaihdetaan CSS -tiedostossa, jossa on tyylimäärittely current -luokalle.

```
<li><a href="#js-content" onclick="currentPage('javascr')" id="javascr">JavaScript</a></li>
```

```
function currentPage(id) {
  var curPage = $('.current');
    var page = document.getElementById(id);
    page.classList.toggle("current");
    curPage.removeClass('current');
}
```

Kuva 8. HTML -elementti ja sitä painettaessa seuraava JavaScript -funktio.

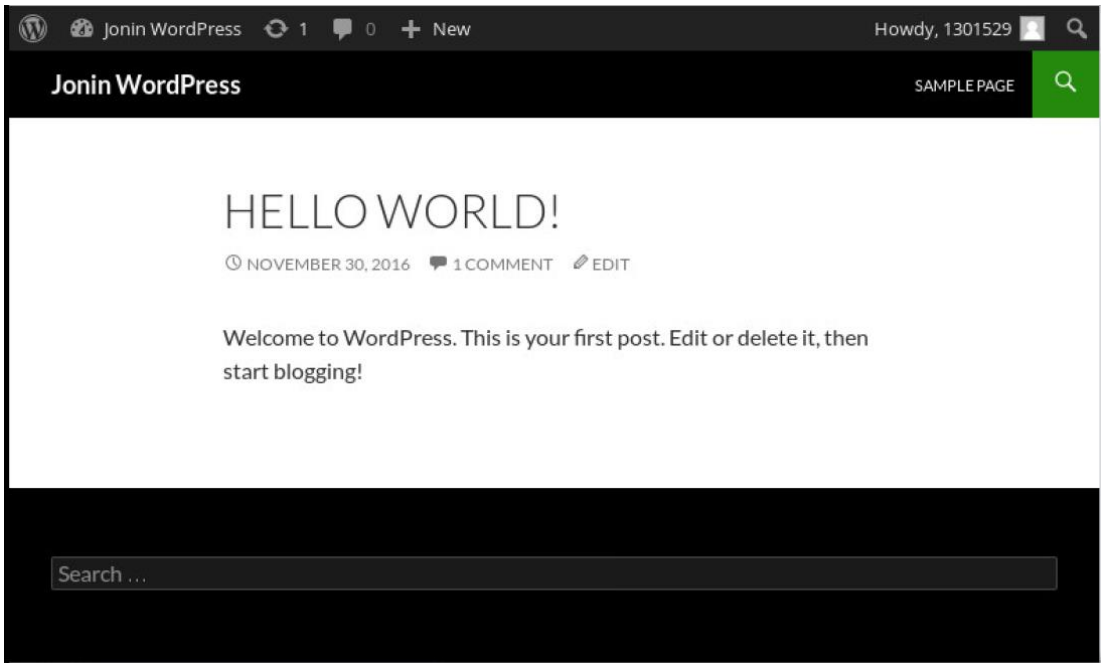
4 SISÄLLÖNHALLINTA- JA JULKAISUJÄRJESTELMÄT

Verkkosivulle asennettavaa ohjelmistoa, jonka avulla voidaan ilman ohjelmointitaitoja hallita nettisivuston sisältöä, kutsutaan julkaisujärjestelmäksi. Parhaimmilla julkaisujärjestelmillä voidaan muokata todella monipuolisesti verkkosivustoja. Uusien toimintojen asentaminen sekä valikoiden muokkaus on tehty erittäin helpoksi julkaisujärjestelmien avulla. Julkaisujärjestelmät takaavat kotisivujen helpon ylläpidon itse, ilman, että tarvitsisit yhtään ohjelmointitaitoja. Avoimen lähdekoodin julkaisujärjestelmiä ovat muun muassa WordPress ja Joomla, jotka ovat avoimen lähdekoodin vuoksi täysin lisenssimaksuttomia. Sivustojen päivittäminen onnistuu suoraan Internet selaimella, kunhan sinulla on järjestelmään annetut tunnukset. (Ubinet 2017)

4.1 WordPress

WordPress on vuonna 2003 julkaistu alusta, joka on nykyisin isoin itse päivitettävä blogi-alusta. WordPress on rakennettu PHP:ta ja MySQL:ää hyväksi käyttäen. Sitä käyttää miljoonat sivustot ja sillä on jokapäiväisiä käyttäjiä kymmeniä miljoonia. WordPress on avoimen lähdekoodin projekti, minkä vuoksi tänäkin päivänä sadoittain ihmisiä parantelee ja työskentelee sen parissa. Vaikka WordPress julkaistiin aluksi blogi-alustaksi, on se tänä päivänä niin kehittynyt, että sillä voidaan julkaista melkeinpä mitä vain, esimerkiksi nettisivuja. WordPressissä on käytössä tuhansittain lisäosia sekä valmiita teemoja. Ainut vaatimus WordPressin asennukseen on sellaisen webhotellin hankkiminen, jonka minimi laitevaatimukset tukevat WordPressiä. (WordPress 2017)

WordPress on heti valmiina käyttöön, kun sen on asentanut palvelimelle (kuva 9, liite 1).



Kuva 9. Vasta-asennettu WordPress -sivusto

4.2 Joomla

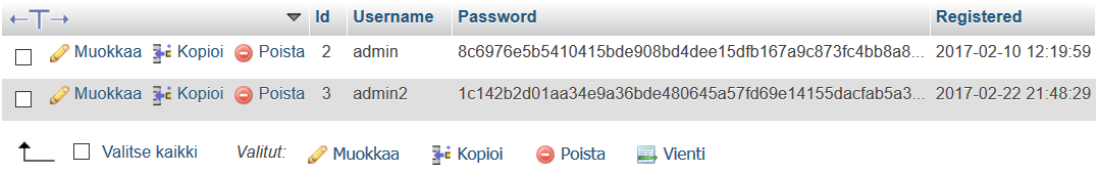
Joomla on maailman suosituimpiin kuuluva avoimeen lähdekoodiin perustuva julkaisjärjestelmä, jolla voidaan hallita nettisivuja. Joomlassa on tuhansittain lisäosia ja sivupohjia. Tietoturva-asioita Joomlaan päivitetään jatkuvasti laajan kehittäjäyhteisön toimesta. Joomlaa pystyy käyttämään niin aloitteleva, kuin kokeneempikin ohjelmoija. (Joomla 2017)

Kuten WordPressikin, myös Joomla skaalautuu automaattisesti päätelaitteen ruudun mukaan.

5 TIETOKANTA

Tietokanta on erilaisesta datasta koostuva kokoelma, joka on organisoitu siten, että siihen on helppo päästä käsiksi, helppo hallita sekä helppo päivittää. Data on järjestetty tauluihin, riveihin sekä sarakkeisiin. Kaikki data on indeksoitu, jotta juuri haluttuun dataan olisi helpompi viitata. Ensimmäinen tietokanta kehitettiin 1960 – luvulla. Nykyään on olemassa erilaisia tietokantoja, kuten relaatiotietokanta, pilvipalvelussa toimiva tietokanta sekä NoSQL (Not only SQL) -tietokanta. (Rouse 2017)

Kuvassa 10 on nähtävillä, miltä dataa sisältävä taulu näyttää MySQL -tietokannassa. Kyseisessä kuvassa on nähtävillä id, joka on aina yksilöllinen jokaisella datalla, käyttäjänimi, suojattu salasana sekä aikaleima, koska käyttäjä on rekisteröitynyt.



Kuva 10. Users -taulu tietokannassa

5.1 PHP

PHP on laajasti käytetty avoimen lähdekoodin skriptikieli, joka on tarkoitettu erityisesti webkehittämiseen ja sitä voidaan myös upottaa keskelle HTML – koodia. PHP – koodi alkaa aina <?php -aloituksella, ja päättyy ?> -lopetukseen. PHP suoritetaan aina palvelimella, mikä erottaa PHP -kielen esimerkiksi JavaScriptistä. PHP:lla on mahdollista luoda koko nettisivu kirjoittamalla kaikki HTML -merkinnät PHP:lla, ja suorittaa ne sitten palvelimelta, jolloin sivuston lähdekoodia on mahdotonta ulkopuolisen nähdä. Parasta PHP:ssa on, että se on yksinkertaista uusillekin koodareille. (PHP 2017)

Tähän opinnäytetyöhön pohjautuvaan nettisivuun PHP:lla on tehty seuraavat asiat:

- rekisteröityminen,
- sisäänkirjautuminen,

- tietokantaan lisääminen, muokkaaminen, poistaminen ja tietokannasta hakeminen.

5.1.1 Tietokantaan yhdistäminen

PHP 5 versiosta alkaen, tietokantaan on pystynyt yhdistämään joko MySQLi:n tai PDO:n avulla. On kuitenkin huomioitava, että yhdistäessä MySQLi:n avulla, pitää tietokannan olla MySQL -tietokanta, toisin kuin PDO:lla, joka toimii 12 eri tietokannassa. PHP:n aikaisemmilla versioilla yhdistäminen tapahtui MySQL:n avulla. Tämä tapa kuitenkin todettiin vanhentuneeksi vuonna 2012. (W3Schools 2017c)

Kuvassa 11 on esimerkki tietokantaan yhdistämisestä MySQLi:ä käyttäen. Tietokantaan yhdistäessä, on määritettävä palvelimen osoite, tietokannan käyttäjätunnus ja salasana sekä tietokannan nimi, mihin halutaan yhdistää. Yhdistäminen kannattaa tallentaa muuttujaan, jolloin yhteyden onnistuminen on helppo tarkistaa. Yhteys muodostetaan new mysqli funktiolla, jolle annetaan edellä mainitut palvelimen osoite, tietokannan käyttäjätunnus ja salasana sekä tietokannan nimi parametreina. Jos yhdistäminen ei onnistu, annetaan virheilmoitus die funktiota käyttäen.

```php
<?php
 error_reporting( ~E_ALL & ~E_DEPRECATED &  ~E_NOTICE );

 define('DBHOST', '127.0.0.1');      //määritetään DBHOST
 define('DBUSER', '[redacted]');          //määritetään DBU
 define('DBPASS', '[redacted]');      //määritetään DBP
 define('DBNAME', '350_oppari');          //määritetään DBN

 $connect = new mysqli(DBHOST,DBUSER,DBPASS,DBNAME);     /

 if(!$connect)       //jos yhteyden muodostaminen EI onnis
 {
     die("Yhdistäminen ei onnistunut!" . mysql_error());
 }
?>
```

Kuva 11. Tietokantaan yhdistäminen

5.1.2 Rekisteröityminen ja sisäänkirjautuminen

Tässä opinnäytetyössä rekisteröityminen ja sisäänkirjautuminen tehtiin HTML:llä ja PHP:lla. Itse nettisivulla näkyvät rekisteröitymis- ja sisäänkirjautumislaatikot tehtiin HTML:llä ja muotoilu CSS:llä. Kaikki palvelinpuolen toiminnot tehtiin PHP:lla. Rekisteröitymisessä tarvitaan vain nimimerkki sekä salasana. Kun on antanut nimimerkin ja salasanan ja yrittää sen jälkeen rekisteröityä, katsoo palvelinpuolen koodi, onko kyseisellä käyttäjätunnuksella jo rekisteröity. Jos käyttäjätunnus on käytössä, annetaan siitä asianmukainen virheilmoitus. Jos taas ongelmia ei synny, talletetaan käyttäjän antamat tiedot tietokantaan, josta ne voidaan myöhemmin lukea sisäänkirjautuessa. Tietojen tallentaminen tietokantaan tapahtuu SQL -lauseilla ja tässä tapauksessa, kun halutaan lisätä uusi käyttäjä tietokantaan, käytetään SQL käskyä insert into, ja sitten haluttu taulu, joka on tässä tapauksessa users (käyttäjät). (Kuva 12)

```
$uname = trim($_POST['uname']);
$pword = trim($_POST['pword']);

$uname = strip_tags($uname);
$pword = strip_tags($pword);

$query =  "INSERT INTO users(Username,Password) VALUES ('$uname', '$pword')";

if($uname != "" && $pword != ""){
    $que = "SELECT Username FROM users WHERE Username='".$uname."'";
    $res = mysqli_query($connect, $que);
    $count = mysqli_num_rows($res);
    if($count > 0) {
        $continue = false;
        $errorMSG = "Käyttäjätunnus on jo käytössä!";
    }
    else {
        $continue = true;
    }
    if($continue){
        $result = mysqli_query($connect,$query);
    }
}
```

Kuva 12. Rekisteröinnissä suoritettava käyttäjätunnuksen tarkistus sekä lisääminen tietokantaan

Sisäänkirjautuminen tapahtuu periaatteessa samalla ajatuksella, kuin rekisteröityminenkin. Ainoana erona on, että kirjautuessa ei lisätä dataa tietokantaan, vaan haetaan tietokannasta jo olemassa olevaa dataa.

Sisäänkirjautumisen yhteydessä kannattaa asettaa kyseisen istunnon käyttäjäksi kirjautuneen käyttäjän jokin data tietokannasta. Koska istunnossa halutaan tunnistaa, eli autentikoida käyttäjä, on kyseisen datan oltava sellainen, joka yksilöi käyttäjän, kuten id (Kuva 13).

```
$row = $result->fetch_assoc()){
$_SESSION["user"] = $row['Id'];
```

Kuva 13. Asetetaan istunnon käyttäjäksi kirjautuneen käyttäjän id

5.1.3 Tietokantaan lisääminen

Kun tietokanta ja taulu ovat luotuina, voidaan niihin ruveta lisäämään dataa. PHP:lla on tiettyjä syntaksisääntöjä, kun käsitellään SQL -lauseita. Säännöt ovat seuraavat:

- SQL -kysely pitää olla lainausmerkkien sisällä.
- Merkkijonot on oltava lainausmerkkien sisällä.
- Numeeriset arvot eivät saa olla lainausmerkkien sisällä.
- NULL arvoa ei myöskään merkitä lainausmerkkien sisälle.

(W3Schools 2017d)

SQL lauseketta INSERT INTO käytetään, kun halutaan lisätä jotain haluamaansa tietokantatauluun. Jos taulussa kuitenkin on arvoja, joille taulua luodessa on antanut määreen AUTO_INCREMENT tai TIMESTAMP, ei kyseisiä arvoja tarvitse koskaan itse lisätä, vaan MySQL hoitaa ne käyttäjän puolesta. Kyseisiä määreitä käytetään muun muassa id:ssä sekä rekisteröitymisen ajankohtana. (W3Schools 2017d; Tutorial Republic 2017)

Kun halutaan HTML lomakkeesta lähettää SQL -kyselyitä PHP:n avulla palvelimelle, annetaan lomakkeen parametrille action arvoksi kyseinen PHP tiedosto, jota halutaan käyttää. Lomakkeessa käytetään <input> elementtejä, jolle annetaan tyypiksi esimerkiksi text, password tai submit ja nimeksi haluttu sana tai merkkijono. PHP tiedostossa haetaan komennolla $_POST[’input_nimi’] lomakkeen vastaavan kohdan arvo. (Tutorial Republic 2017a)

Kuvassa 14 näytetään, miten lomakeen arvot haetaan käytännössä. HTML -koodissa tehdään lomake, jossa on kolme syöttöaluetta ja kaksi painiketta. Nimi syöttöalueen nimenä on itemName, joka talletetaan PHP -koodissa muuttujaan $name hakemalla se $_POST:in avulla. Sama homma tehdään myös hinnalle sekä vuosimallille. Lomakkeen tyhjennys tapahtuu Tyhjennä -painiketta painamalla. Tyhjentämiseen ei tarvita PHP -koodia, vaan riittää, että painikkeen tyypiksi ilmoitetaan reset. Lähetä –painike lähettää tietoa eteenpäin, jos tyypiksi on ilmoitettu submit. PHP -koodissa tarkistetaan, onko painiketta painettu komennolla isset($_POST[’inputin/buttonin_nimi’]).

HTML, index.php

```
<h2>Lisää tuote</h2><br/>
<form method="POST">
    <p>Nimi</p>
    <input type="text" name="itemName">
    <p>Hinta</p>
    <input type="text" name="itemPrice">
    <p>Vuosimalli</p>
    <input type="text" name="itemAge"><br/>

    <button type="reset" name="reset" id="addFormReset">Tyhjennä</button>
    <button type="submit" name="addItem" value="Lisää" id="addFormAdd">Lisää</button>
</form>
```

PHP, additems.php

```
$name = mysqli_real_escape_string($connect,trim($_POST['itemName']));    //tal
$quantity = mysqli_real_escape_string($connect,trim($_POST['itemQuantity']));
$price = mysqli_real_escape_string($connect,trim($_POST['itemPrice'])); //tal
$year = mysqli_real_escape_string($connect,trim($_POST['itemAge']));    //tal
```

Kuva 14. Lomakkeen teko ja arvojen hakeminen PHP:lla

5.1.4 Tietokannasta muokkaaminen

Kun halutaan muokata olemassa olevaa dataa tietokannassa, käytetään siihen SQL lauseketta UPDATE. Muokkausta suorittaessa pitää huomioida, ettei WHERE -lauseketta kannata jättää käyttämättä, sillä muuten tietokantaan päivittyy kaikkiin samannimisiin sarakkeisiin sama arvo. Yleensä WHERE -lausekkeeseen siis kannattaakin sijoittaa esimerkiksi muokattavan datan id, joka ainakin pitäisi olla yksilöllinen. (W3Schools 2017e; Tutorial Republic 2017b)

5.1.5 Tietokannasta poistaminen

Jos haluaa poistaa kokonaan dataa tietokannan taulusta, tapahtuu se SQL lausekkeella DELETE FROM. Tätä lauseketta käyttäessä, tulee olla tietoinen seuraavista asioista:

- Jos SQL lauseketta WHERE ei ole määritelty, kaikki data poistuu kyseisestä taulusta, johon kysely suoritetaan.
- Minkä tahansa ehdon voi määritellä WHERE lausekkeella.
- Voit poistaa vain yhdestä taulusta kerrallaan dataa.

(Tutorialspoint 2017)

Kuten datan muokkaamisessa, myös datan poistattaessa kannattaa käyttää yksilöityä dataa, kuten id:tä. Kuvassa 15 näytetään, miten tietokannan taulusta poistaminen tapahtuu.

```
$id = mysqli_real_escape_string($connect,trim($_POST['poistatuote']));
//valmistellaan kysely, jolla poistetaan taulusta items haluttu tuote
//bind_paramilla kerrotaan mikä tietotyyppi on kyseessä, tässä tapauks
$query = $connect->prepare("DELETE FROM items WHERE Id=?");
$query->bind_param("i", $id);

if($id != NULL || $id != "")    //suoritetaan poisto vain jos poistett
{
    $res = $query->execute();   //suoritetaan valmisteltu kysely
    if($res) {  //jos kysely onnistui -> tulostetaan viesti siitä
        $msg3 = "Poistaminen onnistui!";
    }
    else {          //jos ei -> tulostetaan virheviesti siitä
        $errMsg3 = "Poistaminen ei onnistunut!";
    }
}
else {
    $errMsg3 = "Et ole valinnut tuotetta!";
}
```

Kuva 15. Tietokannasta poistaminen id:n avulla

5.1.6 Tietokannasta hakeminen

SQL lauseketta SELECT käytetään silloin, kun halutaan hakea tietokannasta dataa. Yksinkertainen esimerkki datan hakemisesta on SQL -kysely SELECT * FROM haluttu_taulu. Kyseessä oleva kysely palauttaa kaiken datan halutusta taulusta. Jos

haluaa taulusta vain tietyn sarakkeen, pitää sarakkeen nimi sijoittaa kyselyssä SELECT:in jälkeen (* valitsee kaikki sarakkeet).

Kuvassa 16 haetaan kaikki data items taulusta, jonka jälkeen luodaan PHP – koodilla HTML elementti table ja sille otsikot ID, Nimi, Määrä, Hinta (€) sekä Vuosi. Sen jälkeen while -loopissa tulostetaan kyseisiin sarakkeisiin arvoja $row muuttujan avulla, johon talletetaan jokainen items taulun sarakkeen nimi.

PHP:lla tehty SQL-kysely

```
$sql = mysqli_query($connect, "SELECT * FROM items");
echo "<table style='text-align: center; margin: auto'><tr><th>ID</th><th>Nimi</th><th>Hinta</th><th>Vuosi</th></tr>";
while($row = $sql->fetch_assoc()) {
    echo "<tr><td>".$row["Id"]."</td><td>".$row["Name"]."</td><td>".$row["Price"]."</td><td>".$row["Year"]."</td></tr>";
}
```

Nettisivulla näkyvä tulostus

ID	Nimi	Määrä	Hinta (€)	Vuosi
8	Esimerkki	10	29.00	1999
11	Esimerkki 2	20	30.50	2000
12	Esimerkki 3	30	46.00	2001

Kuva 16. SQL -kysely, jossa haetaan kaikki data items –taulusta

6 TIETOKANNAN TIETOTURVA

6.1 SQL -injektio

Jos tietokantakysely on tehty virheellisesti, pystytään kyselyyn lisäämään osia, jotka suoritetaan samalla, kun suoritetaan tekijän haluama tietokantakysely. Tätä kutsutaan SQL -injektioksi. Tällaisia hyökkäyksiä tehdään nykyään automatisoidusti ja sen onnistuminen edellyttää, että käyttäjältä tulevaa syötettä ei tarkisteta eikä putsata mitenkään, vaan tietokantakysely suoritetaan sellaisenaan. (Lahtinen 2012)

Mikäli SQL -injektiolta haluaa välttyä, kannattaa kaikki SQL -kyselyt valmistella ennen niiden suoritusta. Kun kyselyn valmistelee prepare funktiolla, asetetaan halutun tiedon kohdalle pelkkä kysymysmerkki. Sen jälkeen bind_param funktiolla kerrotaan, mikä tietotyyppi on kyseessä. Tässä tapauksessa kyseessä on merkkijono, eli string, jolloin lainausmerkkeihin sijoitetaan kaksi s -kirjainta, ja sen perään merkitään ne muuttujat, joihin kyseisillä tietotyypeillä viitataan (Kuva 17).

```
$mysql_query = $connect->prepare("SELECT * FROM users WHERE Username=? AND Password=?");
$mysql_query->bind_param("ss", $uname, $paWord);
```

Kuva 17. Prepare funktiolla valmisteltu SQL -kysely

Mikäli jättää valmistelun tekemättä, onnistuu esimerkiksi sisäänkirjautuminen antamalla käyttäjätunnukseksi ’-0||’ ja salasanaksi vaikkapa 1. Opinnäytetyöhön perustuvalla sivulla on käytössä kaksi kirjautumisikkunaa, joista toisessa on kyseinen haavoittuvuus käytössä.

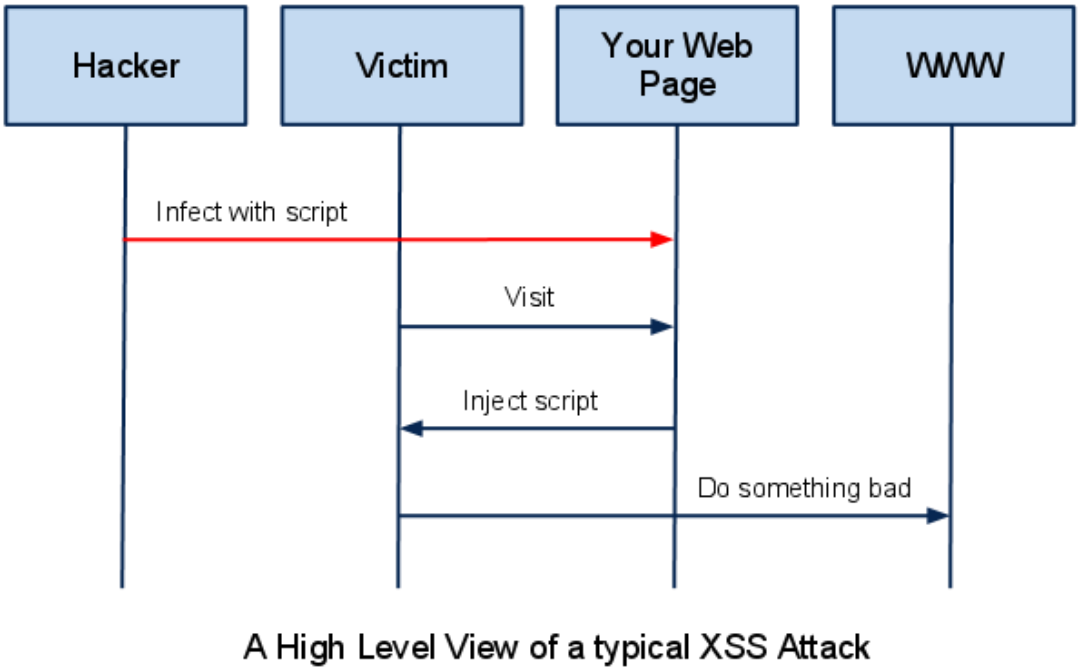
6.2 XSS -aukko

XSS (Cross-site Scripting) on eräänlainen injektio, jossa käyttäjä pystyy kirjoittamaan omaa JavaScript – koodia HTML -dokumenttiin. Selaimen suorittaessa hyökkääjän kirjoittaman JavaScript -koodin, ei selain pysty tunnistamaan koodin lähdettä, vaan suorittaa sen, niin kuin se olisi jo alkuperäisessä ohjelmakoodissa valmiina. XSS -aukon avulla voidaan muun muassa muokata sivuja sekä käsitellä evästeitä. XSS -aukkoja on sekä pysyviä, että ei-pysyviä. Pysyvässä XSS-hyökkäyksessä hyökkääjä

pyrkii ujuttamaan omaa ohjelmakoodia web-sovelluksen tietokantaan, jolloin kuka tahansa sovelluksen käyttäjä, joka lataa sovellussivun, joutuu niin sanotusti uhriksi. Toisin kuin pysyvässä XSS-hyökkäyksessä, niin ei-pysyvässä XSS-hyökkäyksessä tietokantaan ei tallennu mitään hyökkääjän ohjelmakoodia. Tässä tapauksessa hyökkääjä pyrkii pääsemään käsiksi sellaiseen kohtaan alkuperäistä ohjelmakoodia, jonka käyttötarkoitus liittyy sivujen generointiin. (Tampereen teknillinen yliopisto 2015)

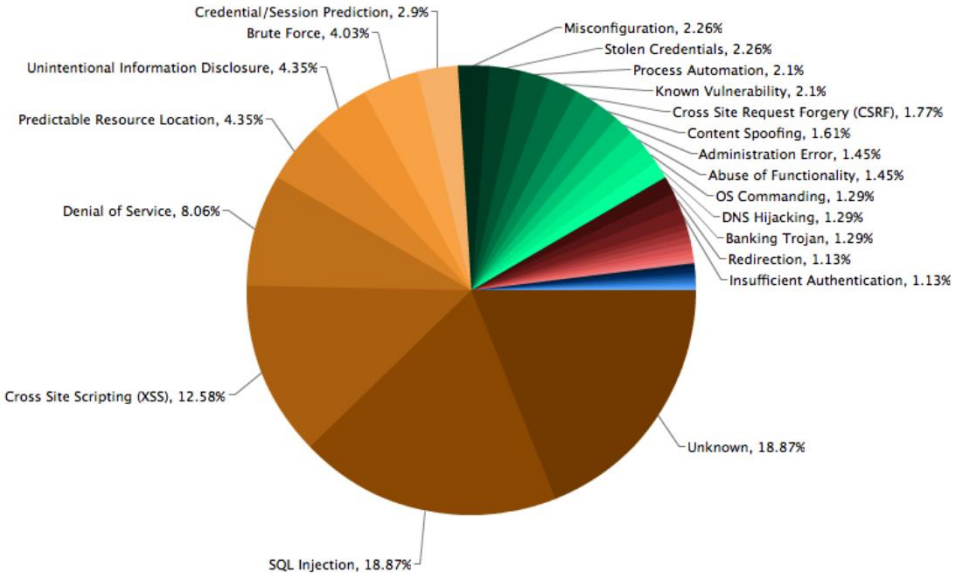
Hyvä ja toimiva keino ehkäistä XSS-aukkoja on käyttää jokaisessa kohdassa, jossa käyttäjä voi syöttää tietoja sivulle sekä sellaisiin kohtiin, jossa tulostetaan jokin käyttäjän antama tieto nettisivulle, PHP -koodin funktiota htmlspecialchars(). Funktio muuttaa syötteeseen syötetyt erikoismerkit, kuten esimerkiksi & -merkin ja ” -merkin eri muotoon, jolloin käyttäjän on vaikeampi saada omaa ohjelmakoodia ajettua HTML -koodin seassa. (Laaksonen 2011; W3Schools 2017)

Kuvassa 18 on esitettynä XSS-aukko yksinkertaisuudessaan. Kuvassa hyökkääjä saastuttaa sivut, jossa normaali käyttäjä käydessään muuttuu uhriksi, koska käyttäjä saa haavoittunutta skriptiä. Esimerkkinä voidaan pitää tilannetta, jossa hyökkääjä kirjoittaa JavaScript -koodin, jossa tulostetaan jatkuvasti popup -ikkunoita, jolloin käyttäjällä ei ole muuta mahdollisuutta kuin sulkea koko selain.



Kuva 18. XSS-aukko yksinkertaistettuna (Acunetix 2017)

Kuvassa 19 olevassa ympyrädiagrammissa on nähtävillä, miten eri haavoittuvuuksien kautta tehdyt hyökkäykset ovat jakautuneet. SQL-injektio sekä XSS-aukko ovat yleisimpiä haavoittuvuuden aiheuttajia.



Kuva 19. Eri tekniikoilla tehtyjen hyökkäyksien jakauma (Acunetix 2017)

7 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli luoda nettisivut, johon opinnäytetyön teoriaosuus pohjautuu. Nettisivut tehtiin alusta loppuun asti itse koodaten, eikä muita valmiita kirjastoja ollut käytössä kuin JavaScript kirjasto jQuery. Nettisivut olisi voinut tehdä helpommin ja vähemmällä vaivalla, käyttäen esimerkiksi bootstrap -kirjastoa, jossa on lukuisia valmiita CSS -määrittelyjä. Myös WordPressiä tai Joomlaa olisi voinut käyttää hyväkseen, jolloin itse ei olisi tarvinnut miettiä nettisivujen skaalautuvuutta eri näyttöpäätteillä.

Nettisivuja olisi voinut kehittää lisäämällä Ajaxin käytettyihin tekniikkoihin, jolloin työskentely tietokannan kanssa olisi keventynyt merkittävästi, eikä sivu päivittyisi jokaisen tietokanta muokkauksen myötä. Mutta mielestäni Ajaxin lisäämisen myötä olisi käsiteltävien ohjelmointitekniikoiden kirjo kasvanut jo liian suureksi.

Opinnäytetyön tekeminen kehitti minua näkemään pienten tekojen merkityksen isohkossa koodikokonaisuudessa. Opinnäytetyön tekeminen on ollut mielekästä ja opettavaa aikaa.

LÄHTEET

Acunetix. 2017. Cross Site Scripting Attack. Viitattu 4.4.2017. https://www.acunetix.cz/websitesecurity/cross-site-scripting/

Can I use. 2017. CSS3 Media Queries. Viitattu 16.3.2017. http://caniuse.com/#feat=css-mediaqueries

Haglund, J. 2014. Webhotelli kannattaa valita huolella. Viitattu 12.3.2017. http://www.jonhaglund.fi/valitse-webhotelli-huolella

Joomla. 2017. Mikä on Joomla? Viitattu 16.3.2017. https://www.joomla.fi/

Järvilehto, L. 2013. Upeaa työtä!: näin teet itsellesi unelmien työpaikan. Helsinki: Tammi

Laakkonen, A. 2011. Oppaat: PHP-ohjelmointi: Osa 13 – Tietoturva. Viitattu 4.4.2017. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_13

Lahtinen, T. 2012. Mikä on SQL – injektio? Ja kuinka siltä suojaudutaan? Viitattu 27.3.2017. https://www.gelo.fi/mika-on-sql-injektio-ja-kuinka-silta-suojaudutaan/

Leiniö, T. 2012. Mitä on responsiivinen design? Viitattu 16.3.2017. https://www.sofokus.com/blogi/mita-on-responsiivinen-design/

Olander, I. 2014. Responsiiviset sivut – verkon minimistandardi 2014. Viitattu 16.3.2017. http://sometek.fi/responsiiviset-verkkosivut-minimistandardi-2014/

PHP. 2017. What is PHP? Viitattu 27.3.2017. http://php.net/manual/en/intro-whatis.php

Rouse, M. 2017. Database (DB). Viitattu 27.3.2017. http://searchsqlserver.tech-target.com/definition/database

Saarikumpu, O. 2017. JavaScript-kielen alkeet – osa 1. Viitattu 16.3.2017. http://weppipakki.com/js/opas/alkeet1.htm#mojs

SSL. 2017. What is SSL? Viitattu 12.3.2017. https://www.ssl.com/

Tampereen teknillinen yliopisto. 2015. Tietoturva. Viitattu 4.4.2017. http://www.cs.tut.fi/~seitti/2015/kalvot/tietoturva/all.html

Tutorial Republic. 2017a. PHP MySQL INSERT Query. Viitattu 27.3.2017. http://www.tutorialrepublic.com/php-tutorial/php-mysql-insert-query.php

Tutorial Republic. 2017b. PHP MySQL UPDATE Query. Viitattu 27.3.2017. http://www.tutorialrepublic.com/php-tutorial/php-mysql-update-query.php

Tutorialspoint. 2017. MySQL DELETE Query.Viitattu 27.3.2017. https://www.tutorialspoint.com/mysql/mysql-delete-query.htm

Ubinet. 2017. Julkaisujärjestelmät. Viitattu 16.3.2017. http://www.ubinet.fi/www-suunnittelu/julkaisujärjestelmä

Vilperi. 2015. Miksi tarvitset responsiiviset kotisivut? Viitattu 16.3.2017. http://www.vilperi.fi/ajankohtaista/digivinkit/miksi-tarvitset-responsiiviset-kotisivut.html

W3C. 2017. What is CSS? Viitattu 16.3.2017. https://www.w3.org/Style/CSS/

W3Schools. 2017a. HTML Introduction. Viitattu 16.3.2017. https://www.w3schools.com/html/html_intro.asp

W3Schools. 2017b. HTML Editors. Viitattu 16.3.2017. https://www.w3schools.com/html/html_editors.asp

W3Schools. 2017c. PHP Connect to MySQL. Viitattu 27.3.2017. https://www.w3schools.com/php/php_mysql_connect.asp

W3Schools. 2017d. PHP Insert Data Into MySQL. Viitattu 27.3.2017. https://www.w3schools.com/php/php_mysql_insert.asp

W3Schools. 2017e. PHP Update Data In MySQL. Viitattu 27.3.2017. https://www.w3schools.com/php/php_mysql_update.asp

W3Schools. 2017f. PHP htmlspecialchars() Function. Viitattu 4.4.2017. https://www.w3schools.com/php/func_string_htmlspecialchars.asp

Web Designer Wall. 2011. Responsive Design in 3 Steps. Viitattu 16.3.2017. http://webdesignerwall.com/tutorials/responsive-design-in-3-steps

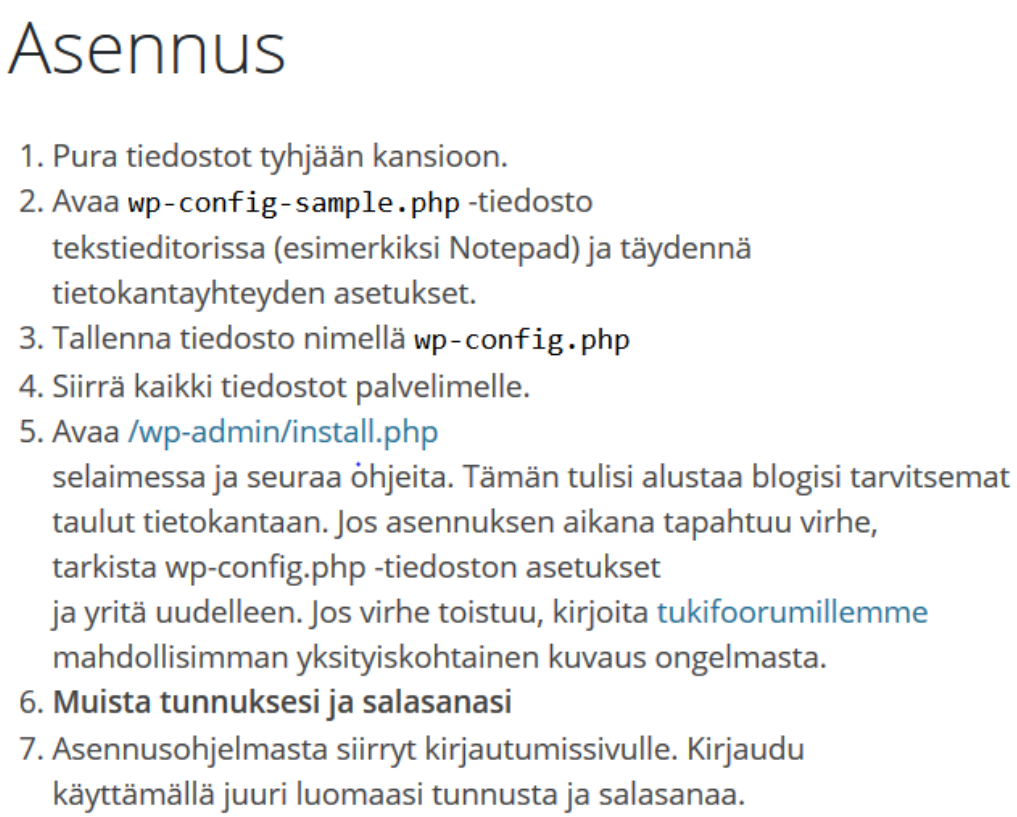
WordPress. 2017. About WordPress. Viitattu 16.3.2017. https://wordpress.org/about/

Liite 1

WordPressin asennus

Lataa viimeisin WordPress linkistä: https://fi.wordpress.org/latest-fi.zip

Sen jälkeen toimi kuvan ohjeiden mukaisesti.



Asennus

1. Pura tiedostot tyhjään kansioon.
2. Avaa `wp-config-sample.php` -tiedosto tekstieditorissa (esimerkiksi Notepad) ja täydennä tietokantayhteyden asetukset.
3. Tallenna tiedosto nimellä `wp-config.php`
4. Siirrä kaikki tiedostot palvelimelle.
5. Avaa /wp-admin/install.php selaimessa ja seuraa ohjeita. Tämän tulisi alustaa blogisi tarvitsemat taulut tietokantaan. Jos asennuksen aikana tapahtuu virhe, tarkista wp-config.php -tiedoston asetukset ja yritä uudelleen. Jos virhe toistuu, kirjoita tukifoorumillemme mahdollisimman yksityiskohtainen kuvaus ongelmasta.
6. **Muista tunnuksesi ja salasanasi**
7. Asennusohjelmasta siirryt kirjautumissivulle. Kirjaudu käyttämällä juuri luomaasi tunnusta ja salasanaa.