Miro Haapaniemi

JAVASCRIPT-KOMPONENTTIEN TESTAUS JA HAVANNOINTI

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Insinööri (AMK), Tieto- ja viestintätekniikka
Kesäkuu 2020

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Maaliskuu 2020	**Tekijä/tekijät** Miro Haapaniemi
Koulutusohjelma Tieto- ja viestintäteknologia, Insinööri (AMK)		
Työn nimi JAVASCRIPT-KOMPONENTTIEN TESTAUS JA HAVANNOINTI		
Työn ohjaaja Jari Isohanni		**Sivumäärä** 31
Työelämäohjaaja Jani Valjus		

Javascript on Internetin käytetyin ohjelmointikieli, ja sen päälle on rakennettu useita eri kirjastoja, joilla laajentaa Javascriptin käyttöliittymää. Tämä opinnäytetyö käsittelee Javascriptia ja React-kirjastoa, ja niiden hyödyntämistä verkkosivupalvelun käyttöliittymän suunnittelussa. Tämän opinnäytetyön teoriaosuus käsittelee Javascriptin toimintatapaa ja sen roolia verkkosivun ohjelmoinnissa, sekä React-kirjastoa ja sen perusteita. Lisäksi teoriaosuudessa käsitellään dokumentin oliomallia ja sen ominaisuuksia, Reactin tilallisia ja tilattomia komponentteja sekä Front End-verkkokehitystä.

Työn käytännönosuudessa käsitellään Reactin ominaisuuksia, ja yksinkertaista käyttöliittymäesimerkkiä hyödyntämällä kehitetään asianmukainen verkkokauppasovellus. Tämän työn tarkoituksena on luoda helppokäyttöinen verkkosivusto, josta löytyy kolme eri Reactilla kehitettyä toimintoa, sekä näyttää esimerkki yksinkertaisesta, mutta silti monipuolisestä käyttöliittymästä. Käytännönosuus on kehitetty Notepad++-ohjelmalla, ja sillä on kehitetty HTML, CSS ja JS-tiedostoja.

Asiasanat
front end development, html, javascript, reactjs, user interface, verkkosivu

ABSTRACT

Centria University of Applied Sciences	Date March 2020	Author Miro Haapaniemi
Degree programme Information Technology		
Name of thesis THE TESTING AND PERCEPTION OF JAVASCRIPT COMPONENTS		
Instructor Jari Isohanni		**Pages** 31
Supervisor Jani Valjus		

Javascript is the most used programming language in Internet, and several libraries have been built on top of it, that help to extend Javascript's User Interface. This work processes Javascript and React-library, and how they can be utilized in the user interface of a web service. This work contains a theory section, that talks about Javascript's mode of operation and it's role in web development, as well as React-library and it's basic operations. The theory section also handles Document Object Model and it's features, React's stateless and stateful components as well as Front End development.

This work also contains a practice section, where a proper online store program is developed by taking advantage of React's features and a simple user interface examples. This work's purpose is to create an user-friendly website, where three different React-developed functions can be found, as well as show example of a simple yet diverse user interface. Practice section has been developed by using Notepad++ software, and it has been used to program HTML, CSS and JS-files.

Key words
front end development, html, javascript, reactjs, user interface, webpage

KÄSITTEIDEN MÄÄRITTELY

Arvo – Javascriptin yksittäinen tieto, se voi olla numero

API – Ohjelmointirajapinta.

DOM – Document Object Model, Dokumentin oliomalli

Funktio – Joukko Javascript-lauseita, joilla on tietty tehtävä. Sitä voidaan kutsua tarvittaessa skriptin muista osista.

Muuttuja – Javascriptin osio, joka sisältää arvoja

Operaattori – Muuttujan kanssa käytettävät symbolit, samat kuin yleisessä matematiikassa.

Syöttö - Input, Käyttäjän tai ohjelman lisäämä data lähdekoodissa

Tägi - Lähdekoodissa sijaitseva tunniste, joka luo aineiston rakenteen.

User Interface - Käyttöliittymä, käyttäjän hallitsema ohjelma.

TIIVISTELMÄ

ABSTRACT

KÄSITTEIDEN MÄÄRITTELY

SISÄLLYS

1. JOHDANTO

Javascript on maailman käytetyin ohjelmointikieli, johon pohjautuu lähes 95% kaikista maailman verkkosivuista. Javascriptia käytetään kaikissa isoimmissa ja suosituimmissa verkkosivuissa, mutta aloittelevalle Internet-sivustojen kehittäjälle nämä Javascriptia käyttävät sivut voivat vaikuttaa monimutkaisilta tai ylivoimaisilta, tarkoittaen sitä, että verkkosivussa on liian paljon vaihtoehtoja. Tämän työn tavoitteena on toteuttaa verkkosivu, joka olisi ulkopuolelta yksinkertainen ja helppokäyttöinen. Mutta koska puhdas Javascript on useasti liian työläs modernin käyttöliittymän ylläpitämiseen, verkkosivun ohjelmoinnissa tullaan käyttämään Javascriptin päälle rakennettua kirjastoa. React-kirjasto on tässä työssä pääroolissa näyttämässä, kuinka sillä voidaan vaikuttaa HTML-verkkosivuun rakenteellisesti ja ohjelmallisesti. React on yksi tunnetuimmista Javascript-kirjastoista, ja sen päätarkoitus on käyttäjän ja palvelun välisen käyttöliittymän helpottaminen. Tässä työssä suunniteltu palvelu tulee toimimaan verkkokauppayritykselle, joka myy tuotteita internetissä.

Tässä työssä perehdytään Javascriptin ja Reactin komponentteihin sekä lisäosiin, ja kuinka niiden kautta voidaan parantaa käyttöliittymää. Käyttöliittymä on kokonaisuus, jossa yhdistetään ulkoasu, kieli, monipuolisuus ja käyttäjän vaatimukset mahdollisimman tehokkaasti. Käyttöliittymällä on tärkeä rooli verkkosivun ensivaikutelman kehittämisessä. Tämän työn tarkoituksena on esittää yksinkertaistettuja versioita todellisten verkkokauppojen käyttöliittymistä. Työssä toteutettujen versioiden kehittämisessä on käytetty HTML-kieltä ja Reactia.

Opinnäytetyön teoriaosuus käsittelee Javascriptin ja Reactin perusteita. Javascriptin osalta tullaan käsittelemään DOM-mekaniikkaa, jota käytetään solmujen lisäämisessä ja muokkauksessa. Työn tarkoituksena on näyttää, että Javascript-komponentit ovat äärimmäisen hyödyllisiä verkkosivun ohjelmoinnissa ja vaikeasta lähdekoodiopista huolimatta lopputulos on kehittäjälle yksinkertaista ja tehokasta.

Opinnäytetyön käytännön osuudessa toteutetaan kaksi eri verkkosivua, yksinkertaisessa opetusmielessä kehitetty sivu sekä monimutkaisempi verkkosivukokonaisuus, johon sisältyy enemmän yleisen verkkokaupan peruselementtejä Reactilla ohjelmoituna. Nämä peruselementit ovat hinnasto ja hinnaston reaaliaikainen valuutan vaihtaminen, uutissivusto sekä yleinen palautelomakesivusto.

Tässä työssä esiteltyjen elementtien avulla on tarkoitus kehittää perusta yksinkertaisemmasta käyttöliittymästä, joka toteuttaa aloittelevien verkkokäyttäjien vaatimuksia vaivatta. Tästä perustasta voidaan myöhemmässä vaiheessa kasvattaa isompi kokonaisuus, jossa yhdistetään useampi Javascript-kirjasto ja ohjelmointikieli.

2. JAVASCRIPTIN PERUSTEET

HTML on ollut jo pitkään verkkosivun tekijöiden ykköskieli, joka alkoi hyvin yksinkertaisena ohjelmointikielenä. Vuosien saatossa kuitenkin verkkosivunsuunnittelijat halusivat tehdä verkkosivuista enemmän kattavampia ja näyttävämpiä, ja muun muassa integroida verkkosivuihin interaktiivisia komponentteja, minkä seurauksena HTML kehittyi monimutkaisemmaksi. Näiden vaatimusten täyttämiseksi HTML ei pian ainoastaan riittänyt, joten Netscape kehitti Javascriptin pitkälle kehittyneempää esittämistä varten. Se julkaistiin osana Navigator 2.0 nimistä selainta. Javascriptin nykyinen versio on osana Mozilla Firefox-selaimen avointa lähdekoodia. (Peltomäki 2004, 2-3.)

Verkkosivujen interaktiivisuus perustuu Javascriptiin. Javascript ei tuota visuaalista sisältöä, ellei se ole yhdistettynä HTML-sivustoon, joko alkuperäiskoodiin liitettynä tai erillisenä tiedostona. Javascript-koodi asetetaan HTML-koodissa <script>-tagien väliin. Tägi ei fyysisesti ilmaannu HTML-sivussa, mutta se auttaa selaimen havaitsemaan Javascript-koodin ja integroimaan sen osaksi nettisivua. <script> -osa voidaan joko sijoittaa HTML-sivuston <head>-osaan tai <body>-osaan. Tapauksissa, joissa hyödynnetään tekstin näyttämistä Javascript-muodossa, on kannattavampaa laittaa <script>-tägi <body>-osaan. (KUVA 1) (Smith, & Negrino, 2006. 2.)

```
<!DOCTYPE html>
<html>
<head>
    <title>Testing</title>
</head>
<body bgcolor="#FFFFFF">
<h1>
<script language="Javascript" type="text/javascript">
document.write("Hello World")</script>
</h1>
</body>
</html>
```

KUVA 1. Perusesimerkki Javascriptin lisäämisestä HTML-sivustoon

1.1. Javascriptin ominaisuudet

Javascriptiä kutsutaan niin sanotusti oliosuuntautuneeksi kieleksi. Oliot ovat tietosisällön omaavia yksiköitä, joihin kuuluvat selaimista löytyvät ikkunat, näppäimet, lomakkeet ja asetukset. Jokaisella oliolla on omat ominaisuutensa ja tehtävänsä eli metodinsa. Näillä ominaisuuksilla voi olla omat olionsa ja metodinsa, ja niilläkin voi olla ominaisuutensa. Näitä kutsutaan aliolioiksi. Kaikkia olioita, ominaisuuksia, metodeja ja aliolioita voidaan yhdistää, jonka seurauksena prosessia tai syntyvää oliota voidaan kuvata yksityiskohtaisesti. Osat voidaan erotella pisteiden avulla, esimerkiksi tietokone.usbohjain.hiiri.näppäin(). Tätä kutsutaan pistesyntaksiksi. (Smith, & Negrino 2006, 11-12.)

HTML-koodi, johon Javascript lisätään, pitää olla rakennettu skriptiystävälliseksi. Tämä tarkoittaa XHTML-variaatiokoodin ja CSS-kielen integroitumista HTML-sivustoon. CSS on tarkoitettu verkkosivujen ulkoasun ja ulkomuodon kuvailuun. XHTML-ohjelmistossa on kaksi tägiä, joilla on tärkeä rooli Javascriptin integroinnissa ja . on lohkotason elementti, mikä merkitsee fyysistä eroa edeltävillä ja seuraavilla elementeillä. sen sijaan on rivinsisäinen elementti. Ne jakavat sisällön semanttisiin lohkoihin, eli lohkoihin, joilla on samankaltainen merkitys. Näiden kahden tägin käyttäminen on tärkeää ja toistuvaa Javascriptiä hyödyntävässä sivussa. (Smith, & Negrino 2006, 17-18.)

Class ja id-attribuutit ovat tärkeitä sen kannalta, että XHTML:n sisällön muotoa ja niitten käyttäytymistä voidaan muuttaa. CSS voi muokata sivun ulkoasua käyttämällä näitä attribuutteja, ja Javascript-tiedosto voi käyttää samoja attribuutteja koodissa, jotka muuntavat sivun elementtien käyttäytymistä. Classilla merkittyä tägiä kutsutaan css-koodissa asettamalla # classin nimen eteen, id:llä merkittyä tägiä sen sijaan kutsutaan laittamalla piste eteen. (Smith & Negrino 2006, 18.)

Ohessa on esimerkki class ja id-attribuuttien erosta css-kielessä.

```
<head>
<style>
.center {
  text-align: center;
  color: red;
}
#center {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p id="center">Blue and center-aligned paragraph.</p>

</body>
```

Ulkoinen skripti on olennaista Javascriptin integroinnissa nettisivustoihin, koska sisäistä skriptiä ei voida siirtää sivujen välillä. Javascriptin js-loppuiset tiedostot sisältävät vain ja ainoastaan Javascript-koodia, ja ne voidaan lisätä sivustoon komennolla:

```
<script language=”Javascript” type=”text/javascript” src=”script.js”>
```

Tämä helpottaa sivun ylläpitoa, jos muutoksia pitää tehdä, ne tehdään .js-tiedostossa. Oheisissa kuvissa (KUVAT 2 JA 3) nähdään kuinka erillinen js-tiedosto kutsutaan yksinkertaisessa HTML-sivustossa.

```
<!DOCTYPE HTML>
<html>
    <head>
    <title>Test 2</title>
    <script language = "Javascript" type="text/javascript" src="script.js"></script>
<body>
<h1 id="helloMessage"></h1>
</body>
</html>
```

KUVA 2. js-tiedoston kutsuminen HTML-koodiin käyttämällä script-komentoa

```
window.onload = writeMessage;
function writeMessage() {
document.getElementById("helloMessage").innerHTML = "Hello, world";
}
```

KUVA 3. js-tiedosto joka kutsutaan kuvassa 2.

1.1.1. Javascriptin muuttujat

Javascriptissa on muuttujia, jotka ovat muistipaikkoja, jotka auttavat tietokoneen sisäisessä muistinhallinnassa. Kun ohjelma aloitetaan, RAM-muistista varataan tarvittava määrä muistia ja muistitilaan yhdistetään annettu muuttujan nimi. Muuttujaan sijoitettu arvo menee edellä varattuun muistipaikkaan. Muuttujan tietotyyppi muuttuu annetusta arvosta riippuen. Kokonaisluku tai desimaalityyppi numeroille, kirjaimille ja sanoille erilliset tietotyypit. Tärkein muuttujissa käytettävä operaattori on sijoitusoperaattori. (Peltomäki 2004, 18.)

Javascriptin muuttujien data on jatkuvasti uudelleen käytettävissä. Muuttuja luodaan var-komennolla. Muuttujalle annetaan nimi, joka myöhemmin kutsutaan lähdekoodissa. Näistä nimistä käytetään termiä tunniste (*identifier*). Tunnisteen nimeämisen sääntöihin kuuluu, että sitä ei saa nimetä jonkin valmiiksi käytetyn termin, kuten avainsanan mukaan. Sen lisäksi tunnisteen alussa pitää olla joko kirjain, alaviiva (_) tai dollarimerkki ($). (Agarwal, 2018)

Javascriptin muuttujalle voi antaa minkä tahansa arvon siitä huolimatta, onko kyseessä kirjain tai numeraalinen arvo. ES2015-päivityksen jälkeen Javascript on saanut kaksi uutta muuttujamuotoa, const ja let. Let-muuttujassa on Laajuusrajoitukset. Toisin kuin var:issa, laajuusrajoitukset kehitettiin koska var:in laajuuden rajattomuus johti useihin viallisiin koodeihin. Const-muuttujan arvo ei muutu tai pysty muuttumaan skriptin aikana. Tätä muuttujaa tullaan käyttämään useasti React-koodissa. (Agarwal 2018.)

1.1.2. Javascriptin tapahtumat

Javascriptillä voidaan luoda myös ikkunatapahtumia, jotka tapahtuvat, kun koko selainikkunan toiminta muuttuu käyttäjän tekemän komennon vaikutuksesta. Ikkunatapahtumiin kuuluvat tapahtumankäsittelijät, jotka laukaistaan ikkunan sulkeutumisessa tai piilottamisessa. Tapahtumankäsittelijät kannattaa sijoittaa ulkoiseen skriptiin sisäisen HTML-skriptin sijaan. Tällä ratkaisulla erotetaan Javascript-koodi HTML-koodista, ja näin on helpompi muokata Javascript-koodia. Onload-tapahtuma on tärkeä osa ikkunatapahtumia. Se laukaistaan, kun käyttäjä siirtyy sivulle ja sivun elementit ovat ladattu. Onloadia käytetään muun muassa käyttäjien vihaamissa ponnahdusikkunamainoksissa. Oheisessa esimerkkiskriptissä näytetään, miten sivun eri osiot voidaan ladata käyttämällä ikkunatapahtumia. (KUVA 4) (Peltomäki, 2004.)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Testing</title>
    <script type="text/javascript" language="Javascript" src="Script.js"></script>
</head>
<body id="pageBody">
<h1>Welcome to the test website!</h1>
</body>
</html>
```

KUVA 4. Esimerkkisivujen eri osioiden lataamisesta ikkunatapahtumien avulla

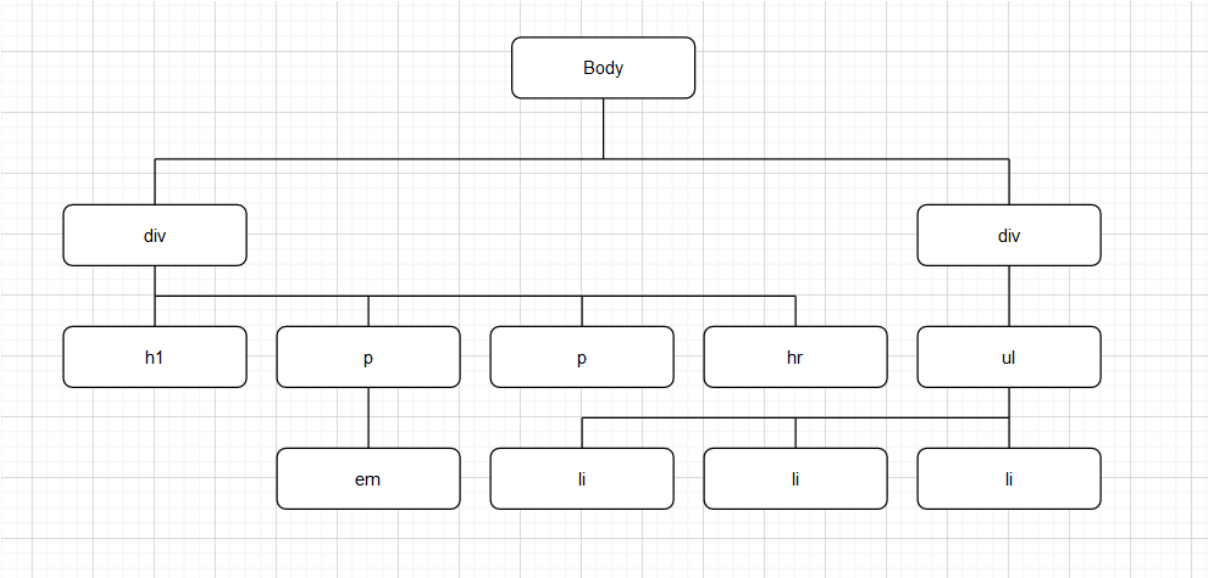
Syy, miksi skriptissä ei ole kolme window.onload-lausetta, johtuu siitä että lausekkeet tallentavat tietonsa toistensa päälle. Sen sijaan addOnload()-funktio käsittelee onload-tapahtumankäsittelijän toimintoja. Jokaisen kutsun myötä yksi lauseke välitetään funktiolle. (Smith, & Negrino 2006. 222.)

1.2. DOM

World Wide Web Consortium-järjestö joka tunnetaan myös nimellä W3C, kehitti standardit CSS-määrityksille, joiden avulla sivujen kehittäjät kykenivät suunnittelemaan sivujen elementtien ulkoasun ja tyypin. Sivuston ulkonäköä voidaan myös muuttaa tyylisivujen määritysten avulla, ja muutokset tulevat voimaan automaattisesti kaikilla sivuilla. W3C on myös julkaissut määrityksiä dokumentin oliomallien, eli DOMin, käsittelemisestä. DOM määrittelee miten elementit välittävät tietoa toisilleen ja kuinka viitata HTML-elementteihin, sekä skriptien suhteet ja viittaukset elementteihin sekä CSS-tyylisivuun. W3C DOM tai WHATWG DOM on asennettu useihin moderneihin verkkoselaimiin. (Introduction to the DOM 2020.)

DOM:iin liittyvät solmut, ne ovat olioita, jotka voivat olla dokumentissa oleva elementti, attribuutti tai jokin muu sallittu osa. DOM mahdollistaa solmujen lisäämisen, muokkaamisen ja poistamisen. Solmujen manipulointi on W3C:n suosittelema tapa tukea dynaamisia UI-komponentteihin perustuvia verkkosivuja. Sivut eivät ole yhteydessä palvelimeen, mutta sivujen päivittäminen suoriutuu skriptien avulla. (Introduction to the DOM 2020.)

Solmuista voidaan muodostaa dokumenttipuu, joka visualisoi oliohierarkiaa. (KUVA 5) Dokumenttiolio on oliohierarkian ylin taso. Sen alapuolella on juurisolmu, joka on dokumentin juurielementti. Jokaisen dokumenttipuun solmulla paitsi juurisolmulla voi olla lapsisolmuja tai äitisolmuja. Jokainen solmu on numeroitu sen sijainnin mukaan, mutta sen voi nimetä. Näin DOM tarjoaa mahdollisuuden viitata dokumenttipuun olioihin tarvittaessa. Jokaisella HTML-elementillä on lapsisolmuista muodostettu taulukko, sekä viittaus äitisolmuun. Ohjelma pystyy muokkaamaan koko dokumenttipuuta, tai sitten poimia tietyn elementtiryhmän tägien perusteella. Lisäksi id-ominaisuuden kautta voidaan etsiä tiettyä solmua, jota sitten kutsua koodissa. (Introduction to the DOM, 2020)



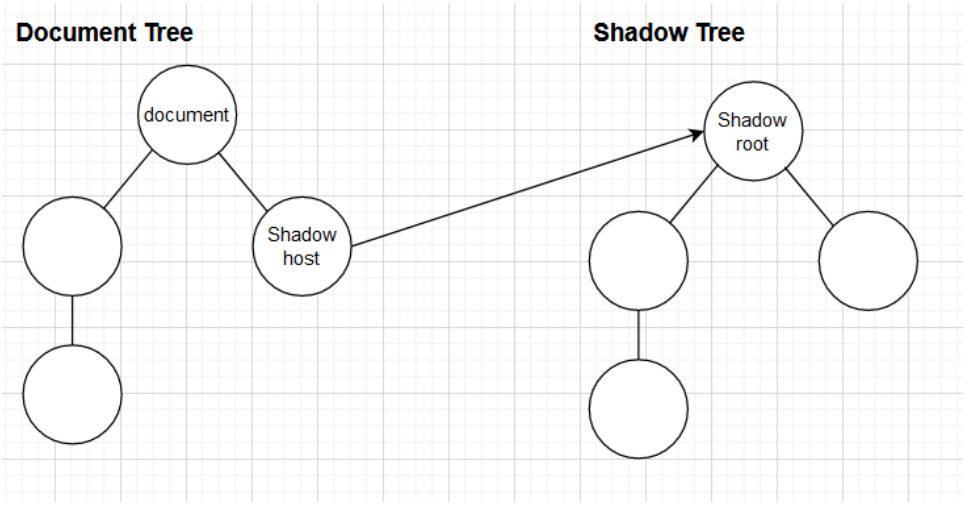
KUVA 5. Dokumenttipuu (Kuvaillen Selectutorial)

DOM-malli tukee XML- ja HTML-kieliä, vaikka kielet tarvitsevat eri rajapintoja. XML tarvitsee vain DOM:in ydinrajapinnan, kun taas HTML tarvitsee sekä ydinrajapinnan sekä HTML-rajapinnan. Tämän takia HTML-kielen DOM-tuki on XML-kielen DOM-tukea raskaampi. (Bui 2019.)

Alun perin DOM ja Javascript olivat kietoutuneet yhteen, mutta ajan myötä ne kehittyivät omiksi ominaisuuksikseen. Sivuston sisältöä säilytetään DOM:issa, ja sitä voidaan seurata ja muokata Javascriptin avulla. HTML tai XML-sivustot ovat rajapintoja, jotka DOM ja Javascript muodostavat. DOM ei ole ohjelmointikieli, mutta ilman sitä Javascript ei pääse muokkaamaan HTML tai XML-sivustoja ja niitten elementtejä. (Bui 2019.)

1.2.1. Varjo DOM

Joskus ohjelmoija voi kohdata ongelmia sivujen rakentamisessa. Näihin kuuluu muun muassa tyylien yliajaminen, jossa tyylisivulla määritetyt tyylit saattavat ylittää toisensa HTML-sivulla. Myös skriptin muuntaminen, jossa Javascript muuntaa osiota itsenäisesti, voi olla joskus vaikeaa. Ongelmia voi aiheuttaa myös id-päällekäisyys, jossa id-ominaisuuksien valinnanmäärä lisääntyy, mikä johtaa valintavaikeuksiin. Varjo-DOM korjaa CSS:n ja DOM:in tuomalla verkkosivulle scoped-tyylin. Varjo-DOM voidaan asettaa tavallisen DOM:in elementtitägeihin. Näihin kuuluvat muun muassa header, footer, div, nav, p ja span. Varjo-DOM sijaitsee yhdessä dokumenttipuun “oksassa”, jossa muodostaa oman, pienemmän puunsa, nimeltä varjopuu (KUVA 6). (Bui, 2019)



KUVA 6. Varjo-Dom (mukaillen Singh 2019)

Varjo-DOM kutsutaan attachShadow()-komennolla. Tämä komento palauttaa varjojuuren, jota voidaan muokata HTML-elementtinä. Tällä tavalla muun muassa kaksi eri h1-elementtiä saavat erilaiset CSS-tyylit. (MDN Web Docs.)

1.3. React

React on sosiaalisen median jättiläisen Facebookin kehittämä Javascript-kirjasto, jonka tarkoitus on tuoda lisää interaktiivisuutta UI-komponenteille. Reactin perusta on render()-komento, minkä avulla Javascriptissä voidaan helpommin tuottaa visuaalista sisältöä lukemalla syötettyä dataa eli input dataa. React toimii XML-komponentilla nimeltä JSX, mutta sitä ei tarvitse ladata erikseen. Sen sijaan on tärkeää, että React-koodi kutsutaan Babel-nimisen kääntäjän avulla, jonka Facebook on myös kehittänyt. Ilman sitä, React-skripti ei toimi HTML-koodissa. Input data luetaan this.props komennolla, kun taas sisäisen tilan data kutsutaan komennolla this.state. Kun komponentin sisäinen data muuttuu, render-komento päivittää skriptin. (Goalkicker.com, sivut 1-2)

React kutsutaan html-koodissa sulkevaa </body>-tägiä edeltävillä riveillä.

```
<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>
```

Edellämainittu Babel asetetaan <head>-tägin loppuun.

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

JSX on tärkeä osa React-kirjastoa, sillä suurin osa UI-ohjelmoinnista perustuu siihen. React hyödyntää JSX:ää koska UI-logiikka on yhdistetty hahmottamislogiikkaan, siinä miten funktioita käsitellään, miten tila muuttuu ajan myötä ja miten dataa valmistetaan. React erottaa huolenaiheita (separation of concerns) komponentteihin, jotka erottavat funktioiden logiikan ja datan. JSX myös pystyy näyttämään varoitus- ja virheviestit tarkemmin. (Masiello 2017.)

JSX:llä voidaan hyötyä const-muuntujasta. Esimerkiksi constille voidaan antaa arvoksi name, minkä voi sitten kutsua pistämällä sen aaltosulkeitten väliin. (KUVA 7) Aaltosulkeisiin voi myös laittaa minkä tahansa Javascript-lausekkeen. (Masiello 2017.)

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

KUVA 7. Esimerkki Reactin render-elementistä

JSX:stä tulee kutsumisen jälkeen osa tavallista Javascript-koodia. Sitä voidaan käyttää if ja for-lausekkeissa, viedä se arvoihin ja palauttaa se funktioista. (React – A JavaScript library for building user interfaces 2020.)

Babel tuo XML:n perustuksiinsa React.createElement-komennon avulla. Tämän avulla pystytään luomaan yksinkertaisia virhevapaita arvokoodeja. Sillä pystytään luomaan React-elementtejä, joita voidaan pitää kuvauksina siitä, mitä käyttäjä näkee visuaalisesti ruudulla. React rakentaa hahmotettavan sisällön näillä elementeillä luettuaan ne. Render-komennon avulla elementtejä voidaan kutsua. (React – A JavaScript library for building user interfaces, 2020.)

Reactissa on omat komponenttiluokat, jotka luodaan Class-komennolla. Luokat käyttävät parametreja, joita kutsutaan props-komennolla, ja nämä jälkeenpäin palauttavat parametrien kuvauksen return-komennolla. (React – A JavaScript library for building user interfaces, 2020.)

1.3.1. Tilattomat komponentit

Tilattomat komponentit (stateless components) toimivat funktio-ohjelmoinnin avulla. Tämä perustuu siihen, että funktio aina palauttaa saman arvon minkä sille on syötetty. Esimerkiksi

```
const statelessSum =(a, b)=> a + b;

let a =0;

const statefulSum =()=> a++;
```

StatelessSum aina palauttaa a:n ja b:n arvot. Sen sijaan StatefulSum ei palauta samoja annettuja arvoja. Funktio, jonka käyttäytyminen vaikuttaa komponentin ulkopuolelle tunnetaan sivuvaikutuksena (side-effect). Tämän takia tilattomia komponentteja kannattaa käyttää enemmän, koska niillä ei ole sivuvaikutuksia. Tämä myös estää ylläpidettävän sivuston saamasta vaihtelevaa tilaa. React-komponenttilla ei yleisesti ole tilaa. React-komponentit, jotka ovat puhtaasti funktioita eivätkä tarvitse sisäistä tilan muuntamista voidaan kirjoittaa yksinkertaisina Javascript-komentoina. Nämä ovat tilattomia funktionaalisia komponentteja, koska ne vain toimivat props-komentojen kautta, ilman mitään tilaa, jota seurata. Tämä johtaa siihen, että Reactissa on tilaa optisointeihin. (Goalkicker.com 4.)

Kun React-tiedostoihin pitää tehdä UI-päivityksiä, kannattavinta on käyttää setState-funktiota. Tämä funktio tekee niin kutsutun matalan sulautumisen edellisen tilan ja uuden käyttäjän antaman tilan välillä, ja näin uudelleen luo komponentit edeltäjineen. (KUVA 8 & 9) SetState tarvitsee toimiakseen updater-parametrin, ja se voi olla joko olio tai funktio. (Goalkicker.com 12.)

```
class ObjectSetState extends React.Component {
    constructor(props) {
        super(props);
        this.click = this.click.bind(this);
        this.state = {
            object: 'This is a sentence'
        };
    }
    click(e) {
        this.setState({
            object: 'This is an altered sentence'
        });
    }
    render() {
        return(
            <div>
                <p>{this.state.object}</p>
                <button onClick={this.click}>Click me</button>
            </div>
        );
    }
}
```

KUVA 8. Esimerkki setsStaten muutoksesta

```
this.setState(function(previousState, currentProps) {
    return {
        counter: previousState.counter + 1
    };
});
```

KUVA 9. Erillinen esimerkki setStaten muutoksesta

Kun monta SetState-kutsua luodaan koodissa, funktion käyttäminen updater-argumenttina voi olla turvallisempaa kuin olion käyttäminen, sillä React voi aktivoida kaikki kutsut samaan aikaan. Nämä kutsut voidaan yhdistää assign-komennolla.

1.3.2. Komponenttien luonti

Komponentit ovat erittäin tärkeitä Reactille. Komponentti on uudelleenkäytettävä elementti, joka näyttää dataa ja muuttuu koko ajan. Koko käyttöliittymä koostuu pienistä yhteen liitetyistä komponenteista, komponentit voivat jakaa käyttöliittymän pieniin osiin sekä myös rakentaa ja suunnitella käyttöliittymän monipuolisesti. Komponentit toimivat samalla tavalla kuin javascript-funktiot, paitsi erillisessä ympäristössä ja eri lähestymistavoilla. (Naimul, 2017 16-18.)

Reactin komponentti kirjoitetaan myös niin kuin Javascript-funktio, kutsuttujen komponenttien määrä vaihtelee luokan mukaan. Jos kutsuntatapa on ES5 (kuten React.createClass) niin kutsutaan viittä käyttäjän luomaa funktiota, kun taas ES6 (kuten class Komponentti extends React.Component) vaatii vain kolmea. Funktionaaliset komponentit voivat myös olla tilattomia. GetDefaultProps on ensimmäinen metodi, joka kutsutaan, tosin vain ES5-kutsuntatavassa. Tämän metodin palauttamat arvot toimivat oletusarvoina, jos niitä ei aseteta, kun komponentti asennetaan. GetInitialState kutsutaan myös vain ES5-kutsuntatavassa. Se mahdollistaa komponentin nykyisen tilan tallentamisen this.state-komentona. ComponentWillMount on kolmannes metodi, joka kutsutaan ja ES6-tavassa ensimmäinen. Tällä metodilla tehdään lopulliset muutokset komponenttiin ennen kuin se lisätään DOMiin. Render on neljännes kutsuttava metodi, se palauttaa yhden elementin, joka edustaa komponenttia. ComponentDidMount on viides kutsuttava metodi, sillä voidaan päästä asennetun komponentin DOM solmuihin. Komponentti poistetaan componentWillUnmount-komennolla, jolla voidaan myös poistaa ComponentDidMount-komennolla muutetut solmut.

1.3.3. Props

Propseilla voidaan viedä tietoa Reactissa äitikomponentista lapsikomponenttiin ja niillä voi olla mikä tahansa tyyppi, esim. funktio. On tärkeää, että kaikille propseille ja niitten tyypeille annetaan oletusarvo. Propseilla voidaan luoda uudelleenkäytettäviä komponentteja. Esimerkkinä voidaan antaa luokka Parentissa luodaan Child-osiot div-tägien sisälle, jotka myöhemmin kutsutaan omassa luokassaan erillisten tägien sisällä valmiiksi määritettynä. (Goalkicker.com, 30.)

PropTypes pystyy kertomaan, mitä props-tyyppejä komponentti tarvitsee (KUVA 10). Komponentit voivat toimia ilman propTypesin määrittämistä, mutta propTypesin lisääminen auttaa tekemään komponentista selkeämmän. React varoittaa, jos kehittäjä yrittää asettaa propin joka on eri tyyppiä kuin sen määritelmä. (Goalkicker.com, 31.)

```
MyComponent.propTypes = {
    someObject: React.PropTypes.object,
    userID: React.PropTypes.number.isRequired,
    title: React.PropTypes.string
};

MyComponent.defaultProps = {
    someObject: {},
    title: 'My Default Title'
}
```

KUVA 10. Luettelo tarvittavista propseista, ja niitten vaatimusaste

KUVA 10:n esimerkissä nähdään, että someObject on vapaaehtoinen, mutta sen sijaan UserID on pakollinen. Jos komponentissa ei ole UserID;ta, React varoittaa, että tarvittua propia ei löydetty.

1.3.4. Lomakkeet Reactilla

Reactilla suunniteltujen lomakkeiden komponenttien kannattaa olla kontrolloituja, jotta komponentin tila ja syötteen arvo ovat jatkuvasti synkronoinnissa, vaikka jokin muu kuin käyttäjä muuntaisi arvoa. Kontrolloitavat lomakekomponentit määritellään value-omaisuudella. Käyttäjän syötteillä ei ole vaikutusta sulatettuun syöttöön, sen sijaan value-omaisuuteen syötetyt muutokset vaikuttavat lomakekoodiin. Value-omaisuus määrittelee nykyisen syötteen arvon, ja onChange-komento päivittää komponentin tilan käyttäjän syötteellä.

Kontrolloimattomilla lomakekomponenteilla ei ole value-omaisuutta, joten syötteen arvon ja komponentin tilan synkronointi jää ohjelman vastuulle. Komponentin tila päivitetään onChange-

komennolla niin kuin kontrolloiduissa komponenteissa, mutta value-omaisuuden sijaan käytetään defaultValueta. Tämä päättää mikä on sulatettavan syötteen arvo ensimmäisessä renderöinnissä, myöhemmät muutokset komponentin tilaan eivät vaikuta syötteen arvoon.

3 FRONT END-VERKKOKEHITYS

Verkkosivuja rakennetaan yleisesti asiakkaille, joilla on rajattu tekninen tietämys Internet-kehittämisestä. React ja Javascript edustavat Front End-verkkokehitystä. Se tarkoittaa käyttäjän nähtävissä olevien toimintojen ja elementtien rakentamista verkkosivulla. Front End-verkkokehityksellä on tärkeä rooli käyttäjän ja yrityksen välisen vuorovaikutuksen kanssa. Verkkosivu luo hyvän käyttäjäkokemuksen, mikäli se onnistuu täyttämään käyttäjän toiveet ilmaan hidasteita tai häiriötekijöitä. Käytettävyys määrittelee käyttöliittymän laadun, sillä katetaan palvelun käytön tehokkuus, helppous ja oppiminen. Verkkokehittäjän tulee testata sivustonsa yleisimmin käytetyillä selaimilla koko sivuston kehityskaaren ajan. Selainten väliset erot tulevat helpommin selville tällä tavalla. (Heinonen 2014.)

Ohjelma, jota pääosin käytetään tässä opinnäytetyössä, on Notepad++. Se on avoimeen lähdekoodiin perustuva C++-kielellä kirjoitettu ohjelma. Ohjelma tukee eri ohjelmointikieliä, mukaan lukien HTML, Javascript ja CSS, joita kaikkia käytetään myöhemmässä projektiosiossa. Notepad++ mahdollistaa muun muassa pitkän koodin piilottamisen. Notepad++;lla voidaan avata monta tiedostoa samaan aikaan yhtenä sessiona. Nämä sessiot voidaan tallentaa, jotta samat tiedostot voidaan avata kaikki samaan aikaan uudestaan myöhemmin. (Heinonen 2014.)

Ensimmäinen versio projektista on yksinkertaistettu ohjeversio, joka kehitettiin DUDE-projektille työharjoitteluna syyskuussa 2019. Tässä ohjeessa näytetään, kuinka React lisätään nettisivuun yksinkertaisen komponentin kautta. (KUVA 11). Tämä ohje luo nettisivun fiktiiviselle verkkokaupalle nimeltä Puustisen Kone, joka myy rakennuskoneita. Tätä sivua käytetään myös monimutkaisemmassa esimerkissä.

```
<html>
    <head>
        <link href="design.css" rel="stylesheet" type="text/css">
        <meta charset="UTF-8">
    </head>
    <body>
    <center>
        <div class="mainbody">
            <div class="header">
                Puustisen Kone
            </div>
            <div class="topmenu"></div>
            <div class="content">
                Tervetuloa Puustisen Koneen nettisivuille, meillä on jotain täällä.
            </div>
        </div>
    </center>
    </body>
</html>
```

KUVA 11. Esimerkki aloitetaan luomalla index.html ja sen css-tyylisivutiedosto

Tämän jälkeen sivustoon integroidaan Javascript-osio, Reactia käytetään yksinkertaisen katalogin luomiseen. Ennen sitä Babel ja Reactjs pitää kutsua, jotta sivun komponentti näkyisi. (KUVA 12) Jotkin internetissä olevat ohjeet suosittelevat JQueryn käyttämistä, mutta se ei esimerkkisivussa toiminut Reactin kanssa.

```
class Form extends React.Component {
  constructor(props) {
      super(props);
      this.state = {value: 'hammer'};

  this.handleChange = this.handleChange.bind(this);
  this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleChange(event) {
      this.setState({value: event.target.value});
  }


  handleSubmit(event) {
      alert('We have run out of product ' + this.state.value + '. Please try again later.');
      event.preventDefault();
  }


  render() {
      return (
          <form onSubmit={this.handleSubmit}>
              <label>
              Please choose an item.
              <select value={this.state.value} onChange={this.handleChange}>
                  <option value="hammer">Hammer</option>
                  <option value="drill">Drill</option>
                  <option value="saw">Saw</option>
                  <option value="wrench">Wrench Set</option>
                  <option value="toolbox">Toolbox</option>
              </select>
          </label>
      <input type="submit" value="Submit" />
      </form>
      );
  }
}

const rootElement = document.getElementById("root");
ReactDOM.render(<Form />, rootElement);
```

KUVA 12. Seuraavaksi laaditaan sivusto nimeltä form.js, joka sitten integroidaan osaksi index.html-sivustoa

Kun yksinkertainen sivusto on tehty, voidaan suunnitella monimutkaisempi sivusto. Monimutkaisemmassa sivustossa käsitellään React-elementtien lisäämistä nettisivustoon. Kolme lisättävää pääelementtiä ovat katalogi, jossa voidaan ostaa tuotteita, nähdä tuotteiden hinnat ja kääntää hinnat dollareihin tai muihin valuuttoihin arvon mukaan. (KUVA 13) Toiselta sivulta löytyy palautepalsta, ja kolmannelta löytyy dynaaminen Puustisen Koneeseen liittyvien uutisten selaus. Verrattuna alkuperäiseen yksinkertaiseen versioon, nettisivun HTML-rakenteessa on enemmän elementtejä, joista osa liittyy Javascriptiin, mutta nämä elementit eivät ole opinnäytetyön fokus, joten sivustoon lisätään vain oleelliset elementit, kuten selauspalkki.

```
class ProductCategoryRow extends React.Component {
  render() {
    const category = this.props.category;
    return (
      <tr>
        <th colSpan="2">
          {category}
        </th>
      </tr>
    );
  }
}

class ProductRow extends React.Component {
  render() {
    const product = this.props.product;
    const name = product.stocked ?
      product.name :
      <span style={{color: 'red'}}>
        {product.name}
      </span>;

    return (
      <tr>
        <td>{name}</td>
        <td>{product.price}</td>
      </tr>
    );
  }
}
```

KUVA 13. Tämä on hinnastokomponentin alkuosa, joka on otettu React-sivustolta otetusta esimerkistä

Hinnasto tehdään ensin. Luovutaan kokonaan alkuperäisestä komponentista, ja korvataan se pitkällä hinnastokomponentilla. Koodin alussa luodaan kaksi taulukkoa (table), näihin voidaan sijoittaa tuote ja niiden hinnat. (KUVA 14) Jos tuote on loppunut varastosta, se näkyy punaisella.

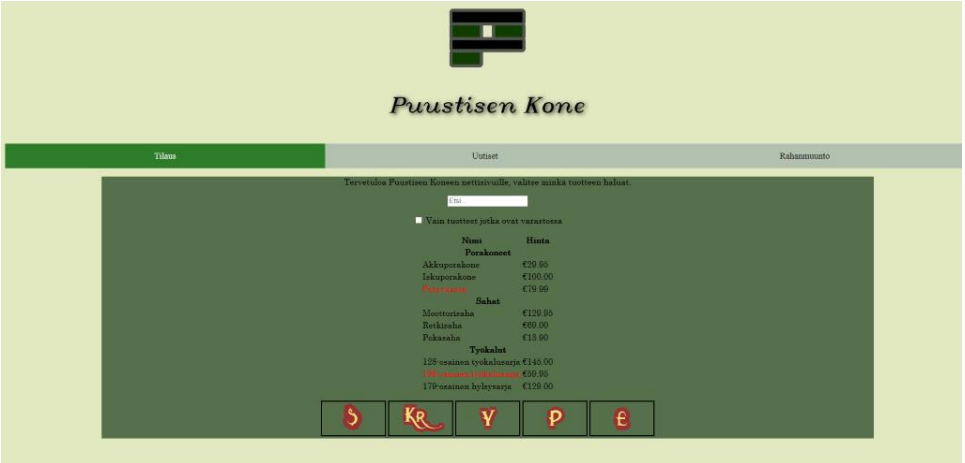
```
render() {
  return (
    <form>
      <input
        type="text"
        placeholder="Search..."
        value={this.props.filterText}
        onChange={this.handleFilterTextChange}
      />
      <p>
        <input
          type="checkbox"
          checked={this.props.inStockOnly}
          onChange={this.handleInStockChange}
        />
        {' '}
        Vain tuotteet jotka ovat varastossa
      </p>
    </form>
  );
  }
}
```

KUVA 14. Notepad++-sovelluksessa otettu kuva

Toinen olennainen osio koodista on hakukenttä, jota myös voidaan luoda Reactilla. Sen lisäksi mukana on mahdollisuus suodattaa tuotteet, jotka eivät ole varastossa. Kun saadaan koodi muutettua ja hinnasto laitettua, nettisivu näyttää tältä. (KUVA 15)

KUVA 15. Kuva sivustosta sen nykyisessä muodossa

Tämän jälkeen on tarkoitus lisätä komponentti, jonka avulla hinnat voidaan muuntaa eri valuutaksi, esim. Dollareiksi tai kruunaksi. Koska React-tiedoston const-arvoja ei pystytä muuttamaan koska hinnat ovat tarkoitettu lopullisiksi, koodiin pitää tehdä muutos HTML:n sisältä, tarkoitus on tehdä näppäin, joka muuntaa div-komponentin id-luokkaa mikä tuo esille toisen koodin jossa on erilaiset arvot. Se ei kuitenkaan tunnu toimivan. Voi olla, että sivusto ei pysty muuntamaan diviä johon on kiinnitetty skripti. Sen sijaan näppäintä painamalla sivusto lataa itsensä uudelleen eri arvoilla. Tässä vaiheessa myös tehostettiin sivun ulkomuotoa, lisäämällä logon, keskittää toimintapalkin ja hyötyä enemmän css-designista. (KUVA 16)



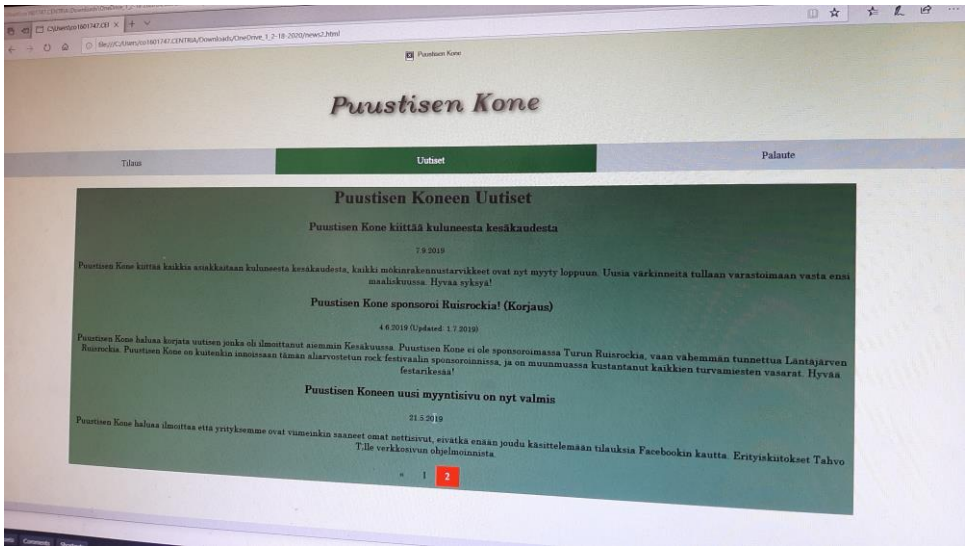
KUVA 16. Sivun ehostettu versio, joka tehtiin rahanmuuntamiskomponentin valmistuksen jälkeen.

Seuraavaksi luotiin sivustoon uutiset. Reactia käytetään jälleen luotettavan uutisrajapinnan luontiin. Tämä vaatii koodin, jossa on otsikko, päivämäärä ja itse artikkeli (KUVA 17). Sivussa on myös sivustonavigaattori ja viimeisimpien uutisten otsikkojen kertaus.

```
function Blog(props) {
  const latest = (
    <ul>
      {props.posts.map((post) =>
        <li key={post.id}>
          {post.title}
        </li>
      )}
    </ul>
  );
  const content = props.posts.map((post) =>
    <div key={post.id}>
      <h3>{post.title}</h3>
      <small>{post.date}</small>
      <p>{post.content}</p>
    </div>
  );
  return (
    <div>
      <p>Viimeisimmät Uutiset</p>
      {latest}
      <hr />
      {content}
    </div>
  );
}
```

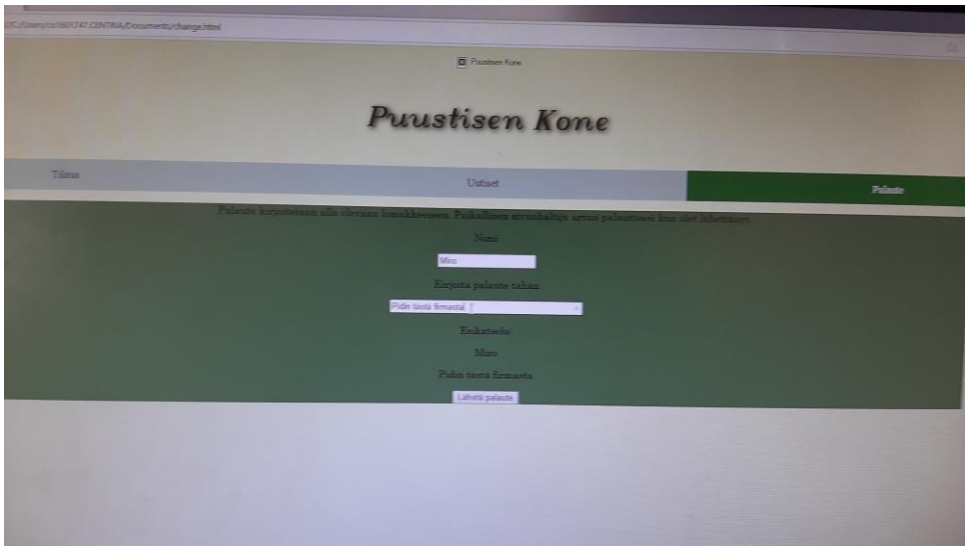
KUVA 17. Uutissivuston koodia

Seuraavaksi sivustoon lisätään osittain realistisia ja hieman humoristia uutisanekdootteja Puustisen Koneen historiasta. Viimeisimpien uutisten lisäksi sivustoon lisätään erillinen sivunvaihtokomponentti. (KUVA 18) Puustisen Koneen sivuihin tulee lyhyet uutisparret, jotka käyttävät ylemmällä kuvattua koodia. Toiselta sivulta edellisten uutisten otsikoita ei löydy.



KUVA 18. Puustisen Koneen uutissivuston toinen sivu

Viimeisenä oli tarkoitus suunnitella palautesivusto. Tekstilaatikon tekeminen Reactilla on jokseenkin hankalaa, koska React ei pysty käsittelemään isoja tekstilaatikoita, joihin yleensä syötetään pidempiä artikkeleita. Tämän takia tekstilaatikot ovat noin yhden rivin pituisia. Vaikka on mahdollista tehdä tekstilaatikosta paksumpi style-määritelmillä, mutta jostain syystä teksti on aina keskittynyt, sen sijaan että se olisi ylimmällä rivillä niin kuin muissa tekstilaatikkokomponenteissä. (KUVA 19) Tätä varten vain itse palautelaatikon leveyttä pitää kasvattaa. Palautelähettämisen prosessissa hyödynnetään ikkunatapahtumia.



KUVA 19. Palautesivusto kokonaisuudessaan

4 JOHTOPÄÄTÖKSET

Komponenttien käyttäminen verkkosivun käyttöliittymän tekemisessä on usealta osalta vaativaa. Se helpottaa käyttäjän näkökulmasta sivun eri ominaisuuksia, mutta kehittäjällä on vaikeuksia järkevästi päätellä sivuston komponenttien koostumusta. React-kirjasto on moniosainen koostumus, joka vaikeuttaa komponenttien rakentamista, ja joitakin komponentteja ei edes voida rakentaa Reactin rajoitusten vuoksi, ja niiden toteuttamisessa pitää käyttää toisenlaista keinoa. Mitä luultavammin komponenttien rakentamisen helpottaminen voi onnistua vain yhdistämällä useita eri kirjastoja, jotka ovat pääosin tarkoitettu käyttöliittymän parantamiseen.

Reactin suurin ongelma on se, että se on lähdekoodiltaan rakenteellisesti sekava. React on periaatteessa kykenevä lähes kaikkeen, mutta sen kykyjen tutkimista estää sen monimutkaisuus. Perusjavascript on huomattavasti heikompi Reactiin verrattuna, mutta sen ohjelmointi on helpompaa. Kuten edellä mainittiin, Reactin vaikeudet voidaan korjata yhdistämällä yhteenkuuluvia kirjastoja.

Jos Puustisen Koneen verkkosivua voitaisiin kasvattaa, sen kokonaisuuteen voidaan yrittää lisätä muita verkkokauppoihin kuuluvia komponenttia. Jokaiselle tuotekategorialle tulisi todennäköisesi omat yksityiskohtaiset valikoimansa, jotka olisivat myös helppokäyttöisiä. Myös rahanmuunnos olisi enemmän reaaliaikainen ja tehokkaampi, ja se hyödyntäisi reaaliaikaisen rahanmuuntamisen kirjastoa, sekä komponenttia, joka tarjoaa helposti muunnettavia arvoja.

Sivulle voisi myös yrittää asentaa tilille kirjautumiskyvyn, jolla käyttäjät voisivat vahvistaa ja ylläpitää tilauksiaan. Javascriptin rooli pienenee sivun uusien komponenttien lisäämisen myötä, sillä tilintekeminen vaatii PHP;tä. Tilien lisäksi on kannattavaa myös luoda simulaatio itse verkkopankin omistajan näkökulmasta, luomalla tuoterekisterin MySQL;n kautta, josta voi lisätä tai poistaa tuotteita. Myös sivunhaltijalla on tilausrekisteri, jonne sivu merkitsee kaikki viimeisimmät tilaukset ja ilmoittaa sivunhaltijalle tiedot uusimmista tilauksista sähköpostiin.

Seuraavassa verkkosivunsuunnitteluprojektissa kannattaa kehittää isompi kokonaisuus, jossa yhdistetään monta eri kirjastoa ja monta eri ohjelmointikieltä. Näitä kokonaisuuspohjia voidaan käyttää yritysten verkkosivujen suunnittelua varten.

LÄHTEET

Agarwal, H., 2018. Variables And Datatypes In Javascript - Geeksforgeeks. GeeksforGeeks. Saatavissa: https://www.geeksforgeeks.org/variables-datatypes-javascript Viitattu: 4. 4. 2020.

Bui, T., 2019. Web Components: Concept And Implementation. theseus.fi. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2019052311527 Viitattu 19. 10. 2019.

Goalkicker.com, 2018. ReactJS – Notes for Professionals, versio 16.3.2 Saatavissa: https://books.goalkicker.com/ReactJSBook/

Heinonen, O., 2014. Front End -Kehitystyökalujen Hyödyntäminen Verkkosivuston Kehityksessä. Theseus.fi. Saatavissa: http://urn.fi/URN:NBN:fi:amk-2014060511985 Viitattu: 15. 1. 2020.

MDN Web Docs. 2020. Introduction To The DOM. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction Viitattu 6. 4. 2020

MDN Web Docs. 2019. Using shadow DOM. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM Viitattu 20.11.2019

Peltomäki, J. 2004. Javascript. 3. painos. Jyväskylä: WSOY

Reactjs.org. 2020. React – A Javascript Library For Building User Interfaces. Saatavissa: https://reactjs.org Saatavissa: 4. 11. 2019.

Selecttutorial. The Document Tree – It's a Family Thing. Css.maxdesign.com.au. Saatavissa: http://css.maxdesign.com.au/selectutorial/document_tree.htm Viitattu 8.4.2020.

Smith, D. & Negrino, T. 2006. Javascript – Rakenna Dynaamisia Verkkosivuja, 2007. Jyväskylä: Gummerus Kirjapaino