Haaga-Helia
ammattikorkeakoulu Oy

Ravintolan verkkosivujen visuaalinen suunnittelu ja toteutus

Ahonen Aleksi

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2015

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä(t)
Aleksi Ahonen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Ravintolan verkkosivujen visuaalinen suunnittelu ja toteutus

Sivu- ja liitesivumäärä
26

Opinnäytetyössä käsitellään verkkosivujen luontia ravintolalle, react.js perusteita sekä produktina verkkosivujen ulkoasu Ravintola- juhlapalvelu Makuelämyksiä Oy:n ravintolalle. Toimeksianto alkoi syksyllä 2017, ja tarkoituksena on päivittää ravintolan verkkosivut nykypäivään. Produkti tehdään yhteistyössä Mikko Hallbergin kanssa, jonka tarkoituksena on omassa opinnäytetyössään tehdä back-end -koodi verkkosivuille.

Tietopohjaosuudessa käsitellään erilaisia verkkosivuja ja niiden käyttötarkoituksia. Käydään lyhyesti läpi verkkosivujen peruskirjoituskielet HTML5, JavaScript ja CSS, verkkosivujen ulkoasun muokkaantumista vuosien varrella ja sitä, mitä hyvä ulkoasu vaatii.

Teoriassa kerrotaan react.js -kielellä koodaamiseen perusteita. Teoria käsittelee aihetta samasta näkökulmasta kuin produkti on tehty aloittaen react-create-app tavasta ja lopettaen Reactin hyvä tietää -asioihin. Tiivistettynä käydään läpi sovelluksen aloittaminen, komponentit, JSX- ja ES6 -ominaisuuksia sekä virtual-DOM:in merkitys. Teorian ei ole tarkoitus opettaa kaikkea Reactista vaan, mitä React on ja miten sillä pääsee alkuun.

Viimeisessä osuudessa käsitellään produktia. Alkuun jaetaan produkti osiin selkeästi niin näkymältä kuin koodin puolelta. Tarkoituksena on antaa selkeä kuva, miten sivut ovat rakentuneet. Navigaatio jaetaan samalla tavalla osiin, jotta sen ymmärtää helposti. Lopuksi käydään läpi työssä käytettyjä moduuleja sekä sisältöä juuri Ravintola- juhlapalvelu Makuelämyksiä Oy:n ravintolalle suunnittelun kannalta.

Pohdinnassa käydään läpi työn haasteita, lopputulosta ja sitä, miten produktia voisi kehittää jatkossa.

Asiasanat
Verkkosivut, react.js, front-end, modernisointi.

Sisällys

1 Johdanto

Verkkosivujen merkitys yritykselle on suuri. Kuluttajat tekevät yhä enemmän päätöksiä verkkosivujen pohjalta. He vertailevat tuotteita, haluavatko ostaa jotain, saati päättävät lähtevätkö edes liikkeelle. Tästä johtuen verkkosivujen siisteys ja nykypäiväisyys ovat tärkeitä yrittäjälle. Tässä työssä perehdytään verkkosivujen luontiin ja tarkemmin visuaaliseen ilmeeseen, mikä on juuri se, mitä kuluttajat tarkastelevat. Työn tarkoituksena on luoda ulkoasu verkkosivuihin Ravintola- juhlapalvelu Makuelämyksiä Oy:n käyttöön, ja tästä johtuen aiheiden näkökulmat käsitellään pääsääntöisesti ravintolamailmaa silmällä pitäen.

Ravintola- juhlapalvelu Makuelämyksiä Oy on Helsingin ydinkeskustassa toimiva ravintolapalveluja tarjoava yritys. Työn aloitusvaiheessa yrityksellä oli kaksi ravintolaa, ravintola Solna Munkkiniemessä ja Kampissa sijaitseva ravintola Laulumiehet. Kuitenkin Solna myytiin työn aikana ja verkkosivujen tarve siirtyi ainoastaan Laulumiehiin. Ravintola Laulumiehet on Helsingin keskustassa toimiva tilausravintola, jonka ravintolapäällikkönä ja Ravintola- juhlapalvelu Makuelämyksiä Oy:n omistajana toimii Tarja Byman. Ravintolassa on tilaa maksimissaan 250 henkilölle ja katettuja asiakaspaikkoja on noin 120 henkilölle. Ravintola on perustettu sotien jälkeen vuonna 1949 ja toiminut siitä asti samalla paikalla perinteikkäästi.

Työn lopputuloksena on tarkoitus saada aikaiseksi uusi moderni ulkoasu verkkosivuille, mutta niistä pitää kuitenkin löytyä sitä vanhaa arvokkuutta. Nykypäiväisyys sivuilla näkyy kahdella tavalla. Konkreettisesta ulkoasusta on tarkoitus tulla moderni ja houkutteleva, mutta konepellin alla on käytössä nopeaa ja trendikästä kieltä. Nopeat ja helppokäyttöiset sivut palvelevat asiakasta hyvin ja houkuttelevat tulemaan ravintolaan. Työssä tulen perehtymään Reactiin, sen taustaan ja käyttöön. Tämä on toinen tärkeä oppimistavoite työssä.

Tässä työssä ei kuitenkaan tehdä verkkosivujen back-end -puolta, vaan sen tekee Mikko Hallberg omassa työssään. Yhteistyöllä saadaan valmiit sivut ravintolalle. Oma työ ja myös lopullinen yhteistyö testataan, jotta sivut ovat toimintavarmat, mutta käyttöönotto ei kuulu työhön.

2 Verkkosivujen luonti

Seuraavassa osiossa käsitellään verkkosivujen luontiin tarvittavia perustietoja ja -taitoja. Verkkosivujen luonti vaatii ensinnäkin ymmärryksen siitä, mitä tekee ja kenelle. Iso osa verkkosivujen tekemistä on kuitenkin itse sivujen kirjoittaminen. Kieliä löytyy nykypäivänä useita, joilla voisi sivut tehdä. Tässä osiossa kuitenkin käsittelemme kieliä, tekniikoita tai ympäristöjä, joita opinnäytetyössä tehtävässä verkkosivustossa käytetään.

2.1 Verkkosivut

Ravintola on kivijalkakauppa ja nykypäivänä verkkosivut ovat elinehto ravintolan liiketoiminnalle. Verkkosivut ovat yritykselle sekä käyntikortti että näyteikkuna (TIEKE 2015). Tästä syystä tärkeää verkkosivujen luonnissa on muistaa kohderyhmät, graafisen suunnittelun merkitys sekä sisällön suunnittelu.

Kohderyhmät ovat yksi isoimmista vaikuttajista verkkosivujen suunnittelussa. Kohderyhmän kannalta huomioitavia asioita verkkosivujen suunnittelussa ovat sisältö, visuaalisuus, tekniset ratkaisut sekä esteettömyys. Sisällön suunnittelussa otetaan huomioon, mitä halutaan viestittää lukijoille. Visuaalisuuteen vaikuttaa, mistä oletettava kohderyhmä pitää. Tekniseltä kannalta mitkä ovat hyviä ratkaisuja kohderyhmälle. Esteettömyys taas vaikuttaa edellä mainittuihin kohtiin. Kohderyhmän taustat vaikuttavat siihen, tarvitaanko isompaa tekstiä, onko nuorille sopimatonta sisältöä tai muita erityispiirteitä. (TIEKE 2015.) Ravintolan verkkosivujen kohdalla kyse ei ole niinkään sopimattomasta sisällöstä vaan teknisistä ratkaisuista ja visuaalisuudesta, siitä, millä sivuista saadaan mielenkiintoiset, mutta helppokäyttöiset kaikille. Valintapäätöksen tekee kuitenkin kuluttaja.

Sivustojen tarkoitus voidaan jakaa kolmeen osaan, läsnäoloon, markkinointiviestintään ja verkkokauppaan. Läsnäolo -sivustojen tarkoitus on vain luoda verkkoon sivut, joista selviää yrityksen nimi, tiedot ja osite. Sivustot ovat minimaaliset ja niihin ei ole käytetty paljon aikaa. Markkinointiviestintä -sivustot tai toisin sanoen mainossivut, ovat verkossa suurin vaikuttaja. Näiden sivujen tarkoitus on mainostaa ja markkinoida yritystä. Sivustot sisältävät mahdollisesti etusivun, yhteystiedot, tuoteluettelon, palveluluettelon, ajankohtaiset- ja uutiset-osio, keskustelupalstan, extranetin taikka tarjouspyyntö- ja palautelomakkeen. Verkkokauppa –sivut sekä markkinoivat että myyvät tuotteita verkossa. Tällä tavoin sivustot vaikuttavat enemmän yrityksen toimintaan. (TIEKE 2015.)

2.2 Kirjoituskielet

HTML5 on uudempi versio perinteisestä HTML -kielestä. HTML on kielenä käytetty alusta asti verkkosivujen merkkaamiskielenä, johtuen sen rakenteesta ja myöhemmästä kehityksestä (Korpela 2014, 26). HTML5 pyrkii korjaamaan kieltä varsinkin seuraavilta osa-alueilta

- erottaa sisältö ja ulkoasu
- rohkaisee tarkoituksellisiin merkintöihin
- edistää esteettömyyttä ja responsiivista suunnittelua
- vähentää päällekkäisyyksiä HTML, CSS ja JavaScriptin välillä
- tukee monipuolista kehitystä ilman Java tai Flash lisäosia.

(Wood 2015). HTML5 luo sivuille pohjan, jonka päälle rakennetaan Bootstrapillä kehys, joka muokataan CSS:n ja JavaScriptin avulla halutun näköiseksi.

JavaScript on vuonna 1995 Netscape Communications Corporationin kehittämä kirjoituskieli. Kielen ohjelmoija Brendan Eich kehitti kielen nopealla tahdilla, mikä aiheutti aikanaan kritiikkiä. Kuitenkin kehitys oli omaa aikaansa edellä, ja kesti noin 15 vuotta ennen kuin valtavirta ymmärsi, mitä JavaScript tarjoaa kehittäjälle. (Brown, 2016.) Nykyään kuitenkin JavaScript on valtavirta -koodauskieli kuuden eri version jälkeen. JavaScriptiä käytetään asiakaspuolen- ja serveri-koodaamisessa sekä työpöytä- ja mobiilisovelluksissa, että peli, tietokanta ja automaatio kehittämisessä. JavaScript on ytimekäs ja vaikuttava kieli melkein mihin tahansa koodaamiseen. (Sheiko, 2015.)

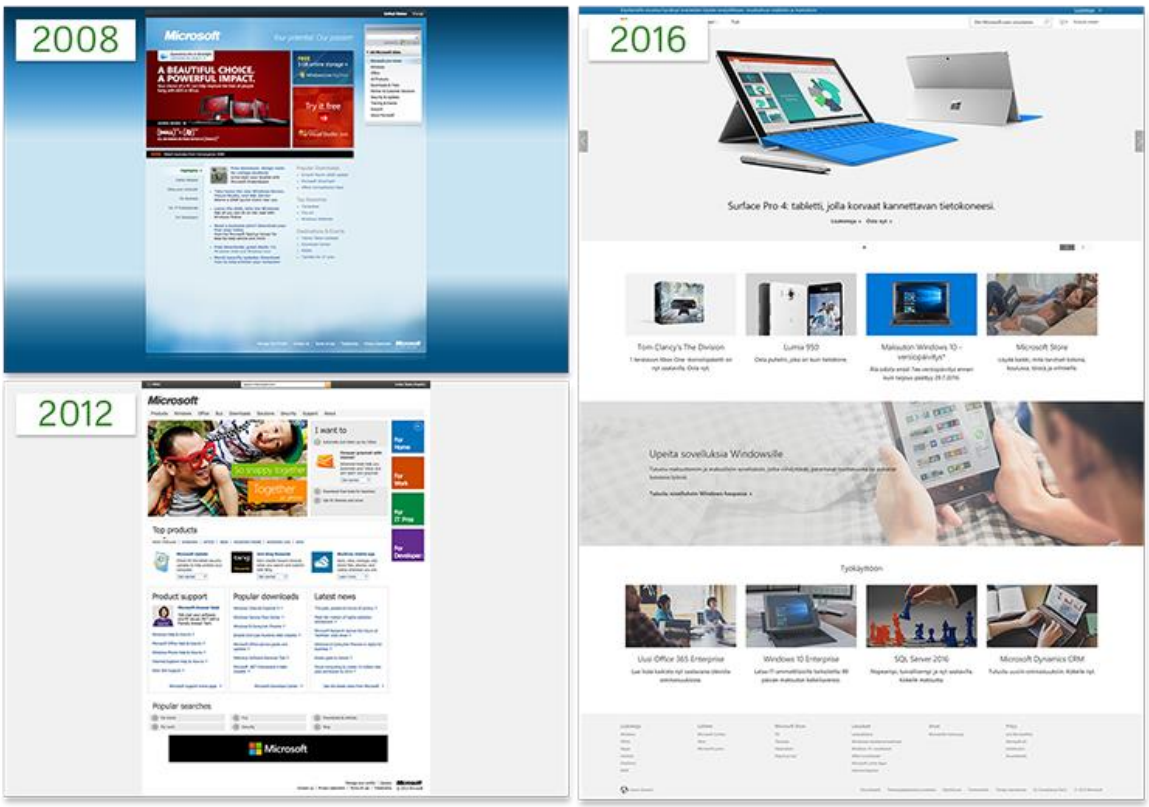
CSS:llä muokataan sivustojen erilaisia tyylejä. Sen tarkoitus on määritellä sivujen elementit ruudulle, paperille tai muulle medialle soveltuviksi. Suurin hyöty CSS:llä työskennellessä on, että kerralla pystyy hallinnoimaan useamman sivun ulkoasua kerralla, näin ollen ei joudu tekemään samaa työtä uudestaan. Muita etuja on, että se nopeuttaa ja helpottaa sivujen tekemistä. CSS tallennetaan omaan tiedostoon .css, mikä helpottaa tietojen hakemista ja työn rakenteen selkiytymistä. (w3school.com.)

2.3 Visuaalinen ilme verkossa

Verkkosivujen visuaalinen ilme on nykyään minimalistinen, mutta tarkkaan harkittu. Se sisältää värejä, kontrasteja, eri kokoja, mutta silti sen pitää olla helposti ymmärrettävä ja selkeä. (WixBlog, 2017.) Verkkosivujen kehitys on muuttunut sitten alkuaikojen ja nykyään verkkosivujen ilme ja käytettävyys huomioidaan paremmin. (koodiviidakko, 2016.)

Isoimmat muutokset web designissa johtuvat niitä käyttävien laitteiden kehityksestä. Responsiivisuudesta on tullut kulmakivi verkkosivujen suunnittelussa. Sivujen pitää skaa-

lautua isolta näytöltä pieniin puhelimiin. Tulevaisuudessa markkinoille saapuu nykyistä enemmän älykelloja ja -laseja, joiden tarve on aivan uudenlainen. Näistä muutoksista johtuen verkkosivujen ilme on muuttunut yksinkertaisemmaksi ja samalla on pyritty parantamaan sivujen nopeutta. Näyttöjen mahdollistama tila on otettu käyttöön, jotta sivut eivät olisi niin ruuhkaiset vaan lopputulos olisi rauhallinen ja huoliteltu. Värikkäiden ja liikkuvien elementtien määrää on vähennetty, koska ne vievät huomiota sivujen oleellisista asioista. Edellisiä kohtia havainnollistaa kuva 1. Sivustot pyrkivät käyttämään yhä enemmän omia kuviaan, eikä kuvapankeista hankittuja. Näillä luodaan sivuille persoonallisuutta ja niistä tulee helpommin lähestyttävät. (koodiviidakko, 2016.)



Kuva 1. Verkkosivujen visuaalisia eroja eri vuosina (koodiviidakko, 2016.)

Verkkosivua suunnitellessa kannattaa kuitenkin ottaa huomioon esteettömyys. Monet nykyajan sovellukset ja painikkeet tarvitsevat tarkkuutta painettaessa pieniä painikkeita, tai tekstissä on käytetty ilmaisuja tai sanastoa, joka ei aukene kaikille lukijoille. On huomioitava, etteivät hakukoneet tai näkörajoitteiset ymmärrä kuvia, jollei niihin ole liitetty vaihtoehtona tekstiä. Niinkin pieni asia kuin fontti voi vaikuttaa ymmärrettävyyteen. Vaalea tai ohut teksti ei näy kunnolla varsinkaan kirkkaassa ympäristössä. Tarkoitus kuitenkin verkkosivun suunnittelussa ja ulkoasussa on, että kaikki voivat ymmärtää sivuja. (Suoranta, 2017.)

3 Reactin teoria

Tutkimustyön teoriassa perehdytään React.js:n teoriaan ja perusteisiin. Tutkimuksen taustalla on kyky ymmärtää ja oppia kirjoittamaan Reactia yksinkertaisesti. Tarkoituksena ei ole oppia kaikkea Reactista vaan seuraavat kohdat kertovat taustat kielen oppimiseen.

3.1 Reactin tausta

Facebookin ja Instagramin tekijät päättivät kehittää Java -kirjaston, jonka tarkoituksena oli kehittää yksisivuisten sovellusten tekemistä. Reactilla työskennellessä ulkoasun lopputulos on tärkeää ja React päättelee, mitä ulkoasun toteutus vaatii. Tämä tapahtuu nopealla DOM:in (Document Object Model) muokkaamisella, visuaalisten komponenttien pilkkomisella sekä JSX:n avulla. (Chinnathambi, 2016.)

React on suosittu. Se julkaistiin vuonna 2013, ja sitä käyttävät sivustot kuten Facebook, Instagram, PayPal, BBC, The New York Times, Reddit, Yohoo ja monta muuta. (Chinnathambi, 2016.) Reactin suureen suosioon vaikutti koodaamisen tarpeen väheneminen. Reactin kyky pilkkoa monimutkaisia koodeja pienemmiksi komponenteiksi, joita voidaan uudelleen käyttää, sai suuren suosion koodareilta. Reactilla mahdollisuus luoda ulkoasua ja nähdä välittömästi muutokset on ollut yksi suosion syy. (Thinkwik, 2017)

3.2 Reactilla työskentely

Helpoin tapa aloittaa Reactilla työskentely on käyttää Create React App -tapaa. Tällä luodaan yksisivuisen applikaation front-end -osio helposti. Se ei kuitenkaan luo pohjaa backendille taikka tietokannalle. Jos haluaa käyttää create react app –tapaa, koneella täytyy olla node 6 tai uudempi. Nodella voidaan aloittaa koodi komennolla

```
npm install -g create-react-app
create-react-app my-app
```

Tämän jälkeen voidaan siirtyä kansioon, missä sovellus on ja käynnistää se `npm start`-komennolla. (React.)

Create React app -komennolla kansioon luodaan uusi hakemisto, jossa on valmis rakenne työhön. Se ei vaadi minkäänlaisia asetuksia vaan on valmis työskentelyyn.

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   └── favicon.ico
```

```
│   └── index.html
│   └── manifest.json
└── src
    └── App.css
    └── App.js
    └── App.test.js
    └── index.css
    └── index.js
    └── logo.svg
    └── registerServiceWorker.js
```

Kansiossa on kaksi tärkeää tiedostoa, public/index.html, joka on sivuston pohja, sekä src/index.js, joka on JavaScriptin toiminnan kannalta tärkeä. Muita huomioitavia asioita kansiossa on, että ainoastaan `src:n` alla toimivat js- ja css- tiedostot, muuten webpack ei niitä huomioi. Muuten tiedostoja voi poistaa ja nimetä uudelleen. (github.) Webpackin tarkoitus on kasata moduulit. Verkkosivua ladatessa jokainen <scriptt> ladataan erikseen, joka kasvattaa sivuston latausaikaa. Webpack korjaa ongelman kasaamalla kaikki moduulit yhdeksi tiedostoksi. (Vikas, 2017.)

3.3 React-komponentit

Komponentit ovat kuin legopalikoita, joita käytetään isompien sovellusten rakentamiseen (Manoj, 2017). Komponenttien tarkoitus on vähentää yhteen liittämistä ja lisätä yhteenkuuluvuutta. Ne ovat uudestaan käytettäviä, koottavia ja yksilötestattavia. Komponentit ovat JXS:n tehokkuuden takana UI:n rakentamisessa. Tarkoitus on siis tehdä monta yksinkertaista komponenttia, jotka tekevät yhden asian todella hyvin. (Chen 2016, diat 3-4.)

Alla olevassa koodissa on kuvattuna yksinkertainen komponentti, jota voidaan käyttää muualla `<MyComponent />` -ilmaisulla. (Manoj, 2017.)

```
import React from 'react';

class MyComponent extends React.Component {
  render () {
   return <div>This is a component</div>
  }
}
```

3.4 Props ja state

Props on lyhennelmä sanasta properites. Ominaisuudet siirretään komponenttiin samalla tavalla kuin argumentti funktioon. Alla olevassa koodissa luodaan ominaisuus `name`, jolla

on arvona ”Sara”. On myös mahdollista tehdä name -ominaisuudesta vaihtoehtoinen. Kun lisätään defaultProps komponenttiin, saadaan oletusominaisuus. Jos koodiin lisättäisiin dafaultProps -ominaisuus name, jonka arvo on ”World”, mutta jos const -arvoa ei ole niin, komponentti tulkitsee tekstin ”Hello World”. Props on muuttumaton. (Bain, 2016).

```
class Welcome extends React.Component {
     render () {
       return <h1>Hello {this.props.name}</h1>
     }
}
```

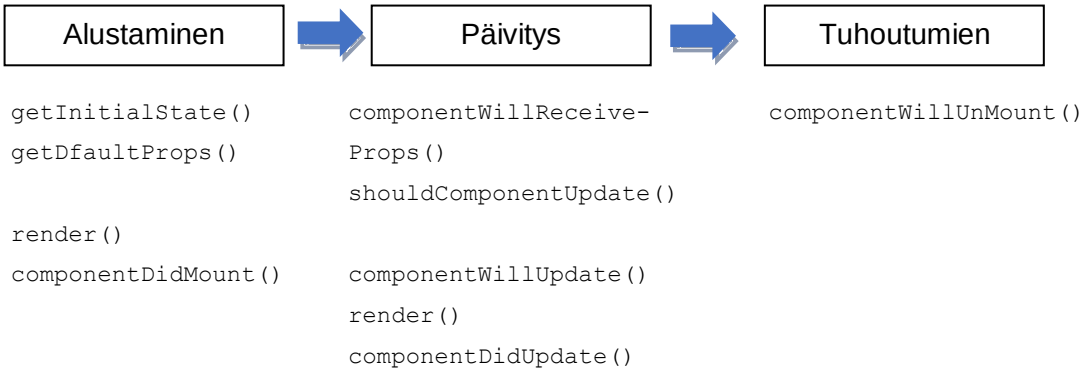
printtaa ”Hello, Sara”

Molemmat props ja state säilyttävät komponentissa tietoa. State -ominaisuutta kannattaa käyttää silloin, kun komponentin täytyy säilyttää tietoa tulkinnan aikana. State -ominaisuuden käytöstä voi käyttää yksinkertaista esimerkkiä painikkeesta, joka laskee, montako kertaa sitä on painettu. Eroina staten ja propsin käytössä on, että state luodaan komponentin sisällä. Sen sisältö voi olla käsin täytetty tai se voi tulla propsillta. Yksinkertaisesti props sisältää tietoa komponentilta, jota ei pitäisi muuttaa. State taas sisältää tietoa komponentille, jota se voi käyttää ja muuttaa itsekseen. (Bain, 2016.)

```
Class SearchBar extends Component {
 constructor(props) {
  super(props);

this.state = { term: '' };
 }

render() {
  return (
   <div className="search-bar">
   <input
   value={this.state.term}
   onChange={event => this.onInputChange(event.target.value)} />
   </div>
   );
 }

onInputChange(term) {
  this.setState({term});
 }
}
```

3.5 Komponentin elinkaari

Yksinkertaisesti komponentin elinkaaressa on kolme vaihetta. Initialization/Mounting, State/Property Updates ja Destruction/Unmounting. Näiden vaiheiden ymmärtäminen helpottaa valitsemaan, mitä logiikkaa kannattaa käyttää missäkin tilanteessa.



Kaavio 1. Komponentin elinkaari

Ylläolevassa kaaviossa 1 kaikki metodit, joissa lukee `will`, kutsutaan ennen toimenpidettä ja kaikki, jossa lukee `did`, kutsutaan toimenpiteen jälkeen. (Rachiele, 2018.)

Alustaminen on komponentin ensimmäinen vaihe. Tällöin komponentti luodaan ensimmäisen kerran ja siirretään DOM:iin. `ComponentDidMount()` tapahtuu heti `render()` metodin jälkeen. Tästä johtuen esimerkiksi dataa, jota pitää pyytää ohjelmointirajapinnasta, kannattaa suorittaa `ComponentDidMount()` komennolla. Jos taas käyttää `setState()` –metodia, aiheuttaa uudelleen renderöinnin, koska alkuperäinen renderöinti on jo tapahtunut.

`ShouldComponentUpdate()` -metodilla kerrotaan, pitääkö komponentin päivittää itsensä, kun muutetaan props- tai state- tilaa. `ComponentWillUpdate()` toteutetaan ennen renderöimistä kun muutetaan props tai statea. Tällä keinolla voidaan valmistella tilaa ennen päivitystä. `ComponentDidUpdate()` tapahtuu heti renderöitymisen jälkeen eikä tapahdu ensimmäisellä kerralla, kun komponenttia luodaan.

`ComponentWillUnmount()` on ainoa tallainen metodi ja sitä kutsutaan, kun komponentti poistetaan DOM:ista. Sitä on hyvä käyttää, kun esimerkiksi ajastin nollataan tai tapahtumalista tyhjennetään. (Rachiele, 2018.)

3.6 JSX

JavaScript extension tai lyhennettynä JSX on suositeltu ulkoasun kehittämiseen. Se näyttää HTML-koodilta, mutta on täysin JavaScriptiä. Sillä halutaan vähentää logiikan ja merkinnän eroa. (React). Se on Reactin kehittäjien suosittelema lisä Reactiin, mutta sen käyttö ei ole pakollista. Sen käyttö edistää kehittäjän kokemusta, koska sitä on helpompi lukea XML-mallisen sisäkkäisten rakenteiden ansiosta. Se edistää tiimissä työskentelyä. JSX muistuttaa HTML koodia, joka on helpommin ymmärrettävää ja muokattavaa vähemmän kokeneille kehittäjille. Lisäksi koodia on vähemmän kirjoitettavana, mikä vähentää virheiden määrää. (Mardan, 2017.)

Ennen kehittäjien piti kirjoittaa staattista HTML:ää sekä CSS:ää ja pikkuisen JavaScriptiä, jotta saatiin teksti vilkkumaan. Nykyään JSX vähentää työn määrää yhdistämällä UI- ja JS- logiikkaa ja helpottaakseen kirjoittamista se muistuttaa HTML:llää, joten sitä on helppo lukea ja kirjoittaa. (Mardan, 2017.)

JSX on tarkoitettu ihmisen luettavaksi, mistä johtuen tietokone ei ymmärrä JSX:ää sellaisenaan vaan se muutetaan JavaScriptiksi konetta varten. Seuraavassa koodissa on esimerkki JSX:n käytöstä. JSX:ää on käytetty div-elementin sisällä. (Chinnathambi, 2016.)

```
class Home extends Component {
  render() {
    return (
      <div className="Home">
        <PageHeader className="Header">
          Tervetuloa <small>ravintola Solnaan</small>
        </PageHeader>
        <div className="Intro">
        <p>
          Lorem Ipsum is simply dummy text…
        </p>
        <br/>
        </div>
        <div className="Photo-gallery">
          <Images />
        </div>
      </div>
    );
  }
```

```
}
```

Kyseistä koodia tietokone ei sellaisenaan ymmärrä. Kun koodi saavuttaa selaimen, se muutetaan perinteiseksi JavaScriptiksi. Kaikki tämä tapahtuu automaattisesti taustalla. Kaikki sisäkkäiset HTML-tyyliset elementit, niiden attribuutit muutetaan CreateElement -kutsuiksi ja alustetuiksi arvoiksi. (Chinnathambi, 2016.)

3.7 ES6 -periaate

ES& on ECMAScriptin uusin standardi, mitä JavaScript käyttää. Isoimpia uudistuksia mitä ES6 tuo on luokat, super ja let avainsanan, oletus parametrit, nuoli (=>) funktio syntaksin. Esimerkiksi nuoli-funktio helpottaa yksinkertaisten funktioiden kirjoittamista. Se vähentää täytetekstin kirjoittamista ja kykenee käsittelemään jopa monimutkaisempia ja useampirivisiä koodeja. Kuitenkaan moni selaimista ei kykene vielä käsittelemään ES6:tta vaan tiedostot useasti muutetaan ES5 -muotoon kääntäjän avulla. Kääntäjänä käytetään useasti Googlen Traceur- tai Babel-kääntäjää, joka on hyvin suosittu React -yhteisössä. (Brennan, 2015.)

Luokat ovat yksi keskeisimmistä uudistuksista ES6:teen. Perinteisesti JavaScriptissa ei ole käytetty luokkia, koska se on oliopohjainen kieli, jossa kaikki oliot periytyvät toisesta oliosta prototype -ominaisuuden kautta. Seuraavassa koodissa Person funktio tehty ES6:lla. Kuten koodista huomaa, prototype -avainsanaa ei käytetä ja ulkoasu on yksinkertaisempi ja tuttavallisemman näköinen muihin luokkaan pohjautuviin kieliin verrattuna. (Brennan, 2015.)

```
class Person {
    constructor(name = 'John', age = 0, gender = '?') {
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    greet() {
        alert('Hello ' + this.name + '!');
    }
}
var John = new Person ('John', 21, 'M');
console.log(John.greet()); // Hello John!;
```

Yllä olevasta koodista huomaa, että ES6:den mukana tulee uusi tapa kirjoittaa julistus-syntaksi, jossa funktio-avainsana jätetään pois. ES6:ssa JavaScript saa oletusparametrit ja `let-` sekä `const-` avainsanat. `Let` antaa määritellä muuttujia, jotka on määritelty lohkoon, lausuntoon tai ilmaisuun. Kuitenkin on hyvä huomioida, että muuttujat on usein julistettu globaalisti funktiossa, mikä saata johtaa eri lopputulokseen. Toinen, mikä saattaa aiheuttaa haittaa on, että samaa muuttujaa ei voi määrittää kahdesti `let` -avainsanalla. Toinen avainsana `const` sallii taas ainoastaan vakiomuuttujat. (Brennan, 2015.)

Erilaiset taulukkofunktiot helpottavat logiikan kirjoittamista, kuten suodattamista. `ForEach` tulostaa taulukon elementit argumentteina. `Map` taas luo uuden taulukon, jossa on sama määrä elementtejä, mutta se muuttaa ne joksikin muuksi. Esimerkiksi taulukon elementit voidaan vaihtaa kaikki kirjoitetuiksi isoilla kirjaimilla. `Find` -funktiolla kykenee etsimään ensimmäisen elementin, joka läpäisee asetetut vaatimukset. `Every` taas vastaavasti katsoo, läpäisevätkö kaikki elementit vaatimuksen. (Lukasz, 2017.)

Luokkien lisäys JavaScriptiin ei paljon muuta JavaScriptin olemusta, mutta helpottaa kirjoittamista. Pitkien merkkijonojen kirjoittamista ja lukemista helpottaa `template stings`. Se tuo helposti käytettävät paikkamerkinnät muuttujille. (Lukasz, 2017.)

```
function hello(firstName, lastName) {
return `Good morning ${firstName} ${lastName}!
How are you?`
}

console.log(hello('Jan', 'Kowalski'))
```

printtaa

```
Good morning Jan Kowalski!
How are you?
```

Vakiofunktiot helpottavat työn määrää. Alla kuvattu yksinkertaisesti niiden toimintaa. (Lukasz. 2017).

```
function sort(arr = [], direction = 'ascending') {
console.log('I\'m going to sort the array', arr, direction)
}

sort([1, 2, 3])
sort([1, 2, 3], 'descending')
```

Muita huomioitavia muutoksia ovat operaattorit `spread` ja `rest`. `Spread` mahdollistaa toisen taulukon ”kopioimisen” ja tekee siitä yhden elementin. Sillä pystyy myös yhdistä-

mään eri taulukoita. `Rest` mahdollistaa taulukosta ensimmäisten elementtien sitomisen eri muuttujiin ja loput yhteen muuttujaan. (Lukasz, 2017.)

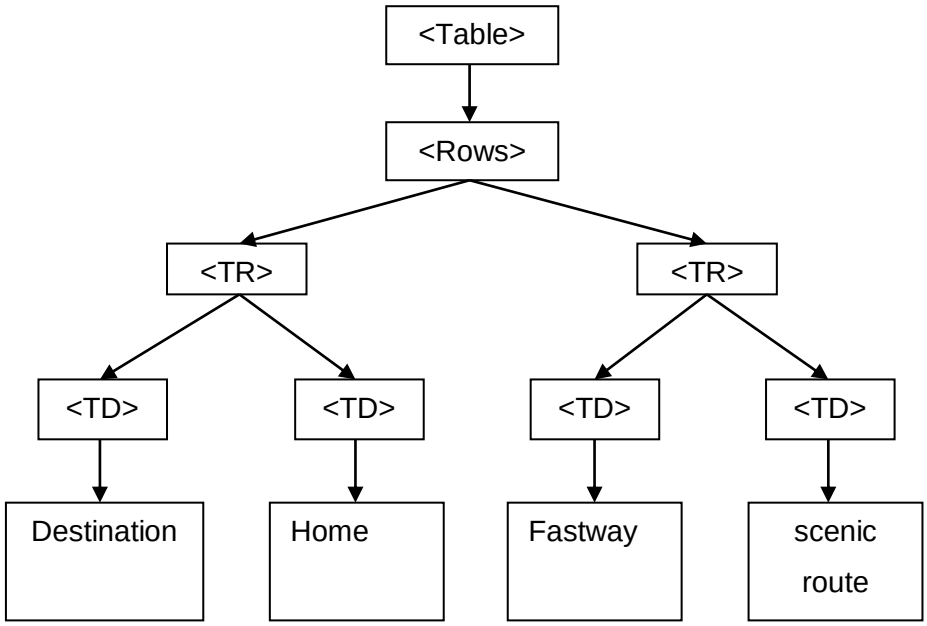
```
function printColors(first, second, third, ...others) {
console.log('Top three colors are ' + first + ', ' + second + '
and ' + third + '. Others are: ' + others)
}
printColors('yellow', 'blue', 'orange', 'white', 'black')
```

printtaa `Top three colors are yellow, blue and orange. Others are: white,black.` (Lukasz, 2017.)

3.8 Virtuaalinen Dom

Document Object Model on ohjelmointirajapinta HTML- ja XML- tiedostoihin, ja se voidaan esittää loogisesti puuna: Table -niminen HTML-tunniste ensin koodina ja sitten kuvassa 2 DOM puuna. DOM:ia voidaan käyttää dokumenttien sisällön muokkaamiseen, poistamiseen taikka lisäämiseen. Kaikki, mikä löytyy HTML- tai XML-dokumentista voidaan DOM:ia käyttäen muokata, muutamaa poikkeusta lukuun ottamatta. (Robie.)

```
<TABLE>
<ROWS>
<TR>
<TD>Destination</TD>
<TD>Home</TD>
</TR>
<TR>
<TD>Fastway</TD>
<TD>scenic route</TD>
</TR>
</ROWS>
<TABLE>
```



Kuva 2. DOM puu.

DOM:in muokkaaminen on osa modernia verkkokehittämistä, valitettavasti suurin osa JavaScript- kehyksistä päivittää DOM:ia liian usein. Pienillä sivuilla päivittäminen ei ole ongelma, mutta modernit sivut sisältävät paljon tietoa. Jos listasta pitää poistaa yksi kohta, koko loppulista pitää päivittää turhaan. Tästä johtuen React käyttää Virtual DOM:ia. Jokaista DOM-oliota kohden on olemassa virtual DOM-olio, ikään kuin kevyt versio. Se on vastaava, mutta kevyempi käsitellä, kun mitään ei piirretä ruudulle. Se ikään kuin suunnittelee talon pohjapiirustuksen uusiksi eikä suoraan siirrä huoneita. Kun JXS-elementti tulkitaan, Virtual DOM päivittyy kokonaan. Tämän jälkeen Virtual DOM:ia verrataan aikaisempaan DOM:iin ja React päättelee, mitä on muutettu. Tämän jälkeen se päivittää ainoastaan muutetut kohdat. Tämä nopeuttaa sivujen päivittämistä huomattavasti, mistä johtuu osittain Reactin suosio. (Codecademy.)

4 Ravintola Solnan verkkosivut

Seuraavassa osiossa käydään läpi, millaisiin ulkoasuratkaisuihin sivustolla on päädytty, mitä sen toteutuksessa on käytetty hyödyksi ja esimerkkejä koodista, jolla ratkaisut on toteutettu.

4.1 Sivuston pohja

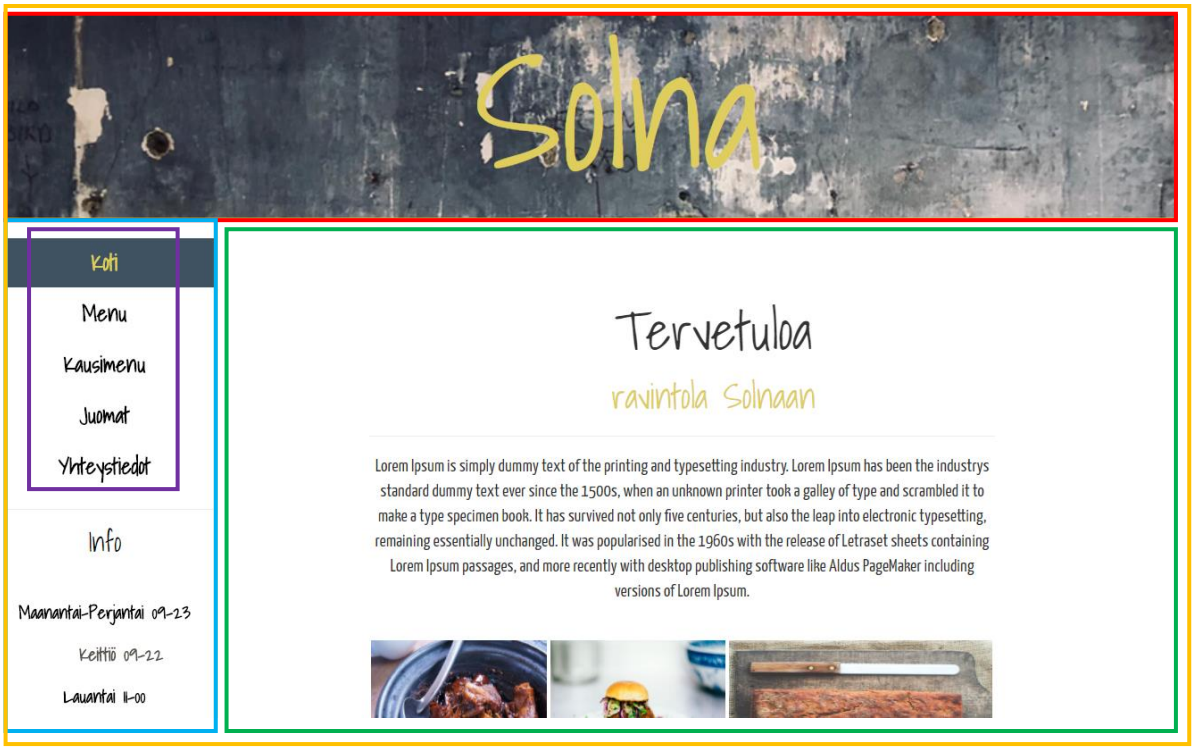
Sivusto on tehty react.js -kieltä käyttäen. React on tällä hetkellä hyvin suosittu front-end-kehityskieli. Sivuston yksi tärkein tarkoitus oli luoda uudet nykyaikaiset sivut. React mahdollistaa helposti päivitettävän sivun nyt ja tulevaisuudessa. Sillä on tarkoitus tehdä yksisivuinen sivusto, joka on nopea päivittää selaimessa. Tästä johtuen valittiin kieleksi react.js. Hyvänä puolena siinä oli se, etten kieltä osannut aikaisemmin, joten opin uuden kielen opinnäytetyötä tehdessä. Kielenä react.js on haluttu eri työpaikoissa, näin ollen kielen osaaminen helpottaa työn hankintaa.

Bootstrap on maailman suosituin viitekehys verkkosivujen ja sovellusten ulkoasun kehittämiseen. Se on avoin työkalulaatikko HTML-, CSS- ja JavaScript -työskentelyyn, ja se sisältää responsiivisia ruudukkokehyksiä ja sisäänrakennettuja komponentteja. (Bootstrap.) Tästä johtuen sivuston pohja on toteutettu Bootstrapiä käyttäen. Bootstrapillä ei kuitenkaan ole vielä suoraan valmista react.js-tukea vaan sivuilla käytetään Reactstrap- ja react-bootstrap- moduuleja, jotka mahdollistavat Bootstrapin käytön.

Sivuston on yksisivuinen ja se on luotu `npm- create-react-app` -komennolla. Tästä johtuen sivu rakentuu index.html -tiedostoon app.js:n kautta. Alla oleva koodi on app.js-tiedostosta, mihin koko sivu pohja perustuu. Koodista on jätetty sisältöä pois selkeyden takia, ja niitä käydään läpi myöhemmissä kohdissa. Koodissa ja kuvassa 3 on värillä korostettu Bootstrap osioita.

```
class App extends Component {
  render() {
    return (
<Router>
      <div className="App">
        <header className="App-header">
          <h1 className="App-title">Solna</h1>
        </header>
        <Container fluid >
```

```
                <Row className="show-grid">
                      <div>
                      <Example className="Navigation"/>
                        <Col xs={12} md={3} className="Leftside">
                         <div className="Sidebar">
                         <div className="Links">
                         <Nav stacked>
                          <NavItem><Link to="/">Koti</Link></NavItem>
                         …
                         </Nav>
                            </div>
                            <hr/>
                            <Info />
                          </div>
                        </Col>
                        <Col xs="auto"></Col>
                      </div>
                  </Row>
                </Container>
          </div>
</Router>
    );
  }
}
```



Kuva 3. Kotisivun väreillä jaoteltu rakenne

4.2 Sivuston navigaatio

App.js -tiedostossa sijaitsee myös verkkosivujen navigaatio. Alkuun työssä kokeilin `react-router` -navigaatiota, mutta kyseinen moduuli oli vanhentunut. Tästä päädyin `react-router-dom` -moduuliin, joka toimi hyvin enkä kokenut muiden kokeilua tarpeelliseksi. Navigaatio koostuu kolmesta osasta, ja ne ovat nähtävissä koodissa alapuolella värikoodattuna. Ensimmäisenä koko sivusto rakentuu `<Router>` -elementin sisälle, tämä on koodissa merkattu punaisella. Toiseksi luodaan linkit, jotka on app.js -tiedostossa merkattu violetilla ja viimeisinä ovat route-elementit, jotka on merkattu vihreällä. Linkkien paikat ja se, mihin route-elemetti luo sivun, näkyvät samoilla väreillä yllä olevassa kuvassa 3.

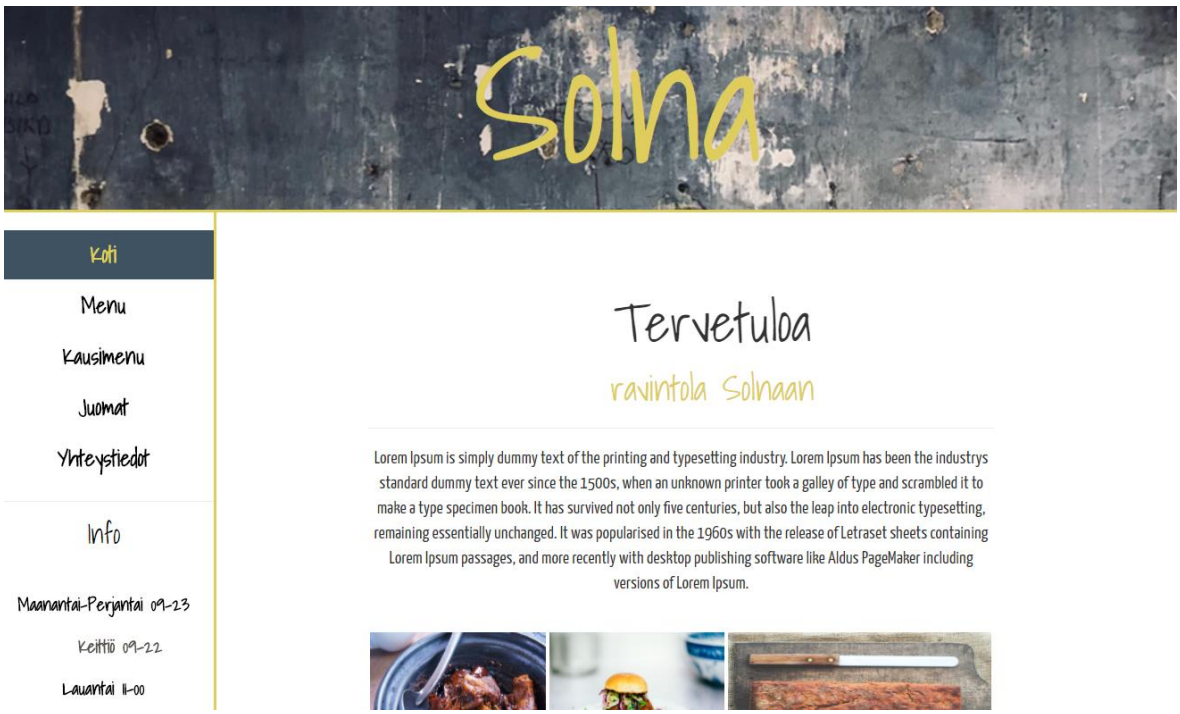
```
class App extends Component {
  render() {
    return (
<Router>
      <div className="App">
        <header className="App-header">…</header>

        <Container fluid >
            <Row className="show-grid">
               <div>
```

```
                <Example className="Navigation"/>
                <Col xs={12} md={3} className="Leftside">
                 <div className="Sidebar">
                  <div className="Links">
          <Nav stacked>
        <NavItem><Link to="/">Koti</Link></NavItem>
        <NavItem><Link to="menu">Menu</Link></NavItem>
        <NavItem><Link to='seasonalMenu'>Kausimenu</Link></NavItem>
        <NavItem><Link to='drinks'>Juomat</Link></NavItem>
        <NavItem><Link to="contact">Yhteystiedot</Link></NavItem>
        </Nav>
                              </div>
                              <hr/>
                              <Info />
                            </div>
                          </Col>
            <Col xs="auto">
              <Route exact path="/" component={Home} />
              <Route path="/menu" component={Menu} />
              <Route path="/contact" component={Contact} />
              <Route path='/drinks' component={Drinks} />
              <Route path='/seasonalMenu' component={SeasonalMenu} />
            </Col>
                      </div>
             </Row>
           </Container>
      </div>
</Router>
    );
  }
}
```

React-router-domissa linkkien ja route-elementtien pitää olla routerin sisällä. App.js -tiedoston lisäksi linkkejä löytyy `<Example className="Navigation"/>`komponentista. Kyseinen komponentti hoitaa navigoinnin pienemmillä näytöillä. Komponentin paikka on routerin sisällä, joten linkit toimivat, vaikka ovat eri .js -tiedostossa. Linkkien tarkoitus on siis olla painikkeita, joilla vaihdetaan sivua. Route-osio taas merkkaa paikan ja reitin mistä sivu tarkemmin ottaen komponentti haetaan ja mihin sivu tulostetaan.

4.3 Sivuston sisältö

Sivuston värit on päivitetty vanhoista sivuista. Solnan värit ovat olleet aina kultaan taipuva keltainen ja ruskea. Rusehtavat värit on jätetty pois, ja tilalle on vaihtunut valkoisen ja harmaan vaaleat värit. Kontrastina menu-osioissa on tausta vaihdettu mustaksi, jotta sivut eivät jää liian kalpeaksi. Tehosteväriä on käytetty edelleen kultaista. Kuvissa 4 ja 5 on kuvattuna aloitussivun ilme ja menu-sivun musta tausta.



Kuva 4. Kotisivu

Kuva 5. Menu-sivu

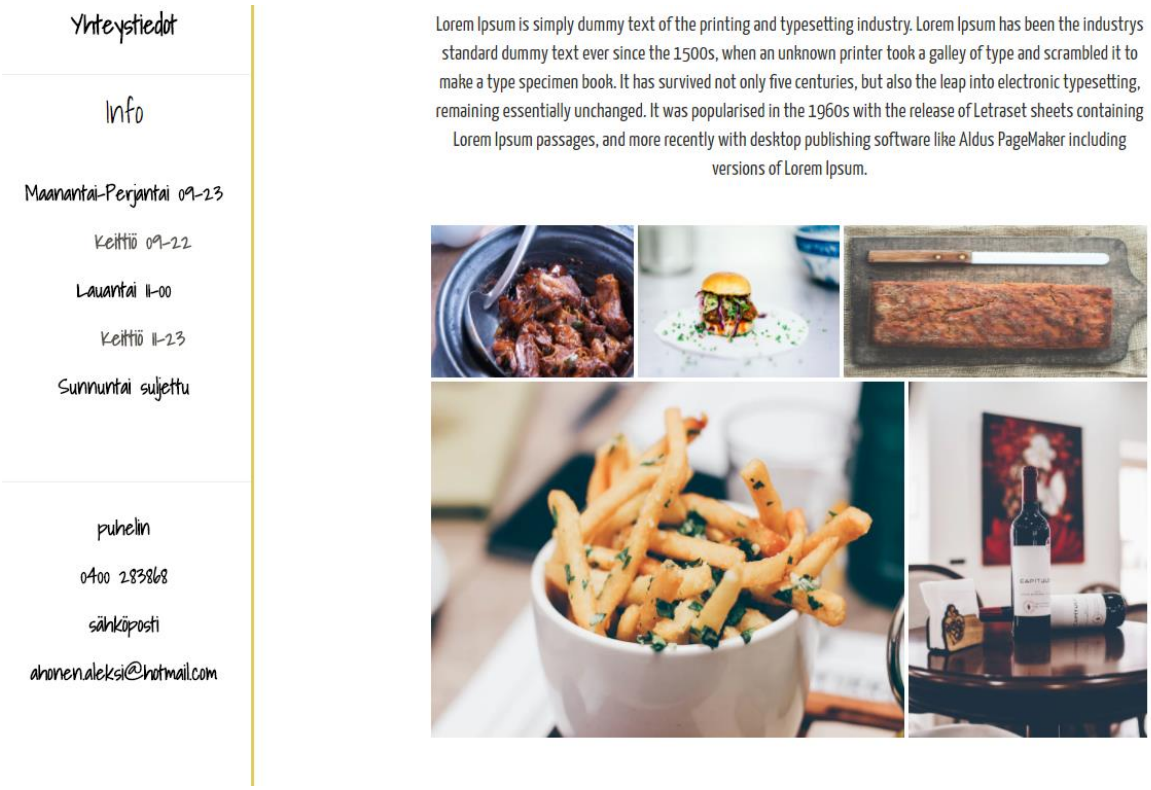
Kuten kuvista 4 ja 5 huomaa, vain sivuston pääväri vaihtuu. Sivuston yläosa ja vasemmassa laidassa sijaitseva infopalkki pysyvät samana selkeyden vuoksi. Värien vaihtuminen on helppo tapa lisätä kontrastia eri osioiden välille. Nykyaikaiset sivut ovat usein vaaleita ja minimalistisia. Tätä noudattaen aloitussivu on vaalea ja hyvin yksinkertainen. Kuitenkin eri menusivuille vaihtaessa tausta muuttuu mustaksi ja tekstit vaaleiksi. Mielestäni efekti on hyvä ja menut ovat siistit ja asialliset. Jos sivusto olisi kokonaan valkoinen tai musta, ne olisivat liian yksitoikkoiset. Varsinkin kun menu sivuilla ei ole muuta kuin tekstiä, niin valkoinen kävisi tylsäksi.

Navigoinnin toteutuksessa oli kaksi vaihtoehtoa, sivupalkki tai ylhäällä oleva navigaatio. Päädyin sivupalkkiin, koska halusin ravintolan yhteystiedot ja ajat näkyviin suoraan ja helposti. Monen ravintolan sivuilla aukioloaikoja joutuu etsimään, mikä ei ole mielestäni tarkoituksenmukaista. Yläpalkissa en nähnyt hyvää tapaa saada tietoja esille selkeästi. Pienemmässä ruutukoossa sivupalkki kuitenkin poistuu ja muuttuu tiputusvalikoksi (Kuva 5) tilan säästämisen takia. Tästä johtuen aukioloajat on sisällytetty myös yhteystietsivulle.

Kuva 5. Tiputusvalikko

Sivustolla käytetään kahta fonttia `Shadows Into Light cursive` ja `Yanone Kaffeesatz sans-seriff`. Molemmat fontit on ladattu Google Fonts -palvelusta. Shadows fontilla on tehty pääasiassa sivupalkki, otsikot ja ravintolan nimi. Kevyempi ja käsinkirjoitetun näköinen fontti on valittu modernisoinnin takia. On haluttu luopua vanhoillisesta jäykästä fontista ja saada tilalle jotain maanläheisempää. Tekstiosuuksissa on kuitenkin käytetty perinteisempää fonttia, jotta pienemmän tekstin lukeminen olisi helpompaa. Tähän tarkoitukseen valittu Yanone -fontti sopii, sillä se on yksinkertainen ja siisti.

Sivuille on toteutettu kuvagalleria `react-photo-gallery` –moduulilla. Tämä mahdollistaa suoraan siistin responsiivisen gallerian. Alla olevassa kuvassa 6 on galleria etusivulla. Moduuli mahdollistaa kuvien lisäämisen ja niiden koon muokkaamisen, minkä pohjalta moduuli itse asettelee kuvat lomittain.



Kuva 6. Etusivun kuvagalleria

Koko sivustolla esteettömyyden lisäämiseksi on tehty pieniä mutta tärkeitä asioita. Sivustolla on pyritty väreissä selkeisiin eroihin, jotta alueet tai tekstit eivät sulautuisi yhteen. Tämä helpottaa näköhaitoista kärsivien asiakkaiden verkkosivujen käyttöä. Kuviin on lisätty vaihtoehtoiset tekstit, jotta automaattiset lukukoneet ymmärtävät kuvat. On kuitenkin tärkeää, että mahdollisimman moni kykenisi sivuja selaamaan mahdollisimman pienellä vaivalla.

5 Pohdinta

Tässä työssä on ollut omat haasteensa ja helppoutensa. Isona tekijänä oli, että alkuperäisesti sivut suunniteltiin Solnalle, mutta kyseisen ravintolan toimitilat myytiin opinnäytetyön aikana. Kuitenkin toimeksianto pysyi ennallaan. Ravintoloitsija ei siis myynyt ravintolan nimeä ja saattaa aloittaa uuden toiminnan jonain päivänä. Verkkosivujen on tarkoitus palvella ravintolaa silloin. Tästä johtuen sain vapaat kädet työskennellä ulkoasun kanssa, eikä tarvinnut noudattaa vanhan ravintolan perinteitä täydellisesti. Tämä teki oman haasteen siinä, ettei ollut enää kuvia ravintolan tiloista tai infoa ravintolasta. Siitä johtuen varsinaista täytettä ei ole suoraan sivuille saatavissa.

Isona haasteena työssä oli react.js -kirjoituskielen opettelu. Alkuperäisestä käsityksestäni poiketen Reactilla tehtiin asiat aivan eri tavalla. Tarkoituksena oli tehdä sivun pohja HTML:llä ja Bootrapillä ja tämän jälkeen lisätä sivustolle elävyyttä Reactia käyttäen. Kuitenkin teoriaa tehdessä selvisi, että Reactilla tehdään käytännössä koko työ ja HTML-koodia sivuilla ei juurikaan ole. Koodaamisen piti olla alkuperäisessä suunnittelussa huomattavasti nopeampi osuus. Työ kuitenkin venyi itselle tuntemattomien ratkaisujen etsimisessä ja niiden soveltamisessa. Lopputuloksena sivut eivät ole myöskään samanlaiset kuin alun perin olin ajatellut, jos olisin käyttänyt html:ää, css:ää ja JavaScriptiä.

Opinnäytetyö opetti ehkä eniten järjestelmällisyyttä. Kirjoittaminen ei ole minulle luonnollista toimintaa, ja tekstin tuottaminen oli aika ajoin hyvinkin haastavaa. Kuitenkin kun tehtävän jakoi pienempiin osiin, taakka tuntui pienemmältä. Tietenkin uutena asiana oli react.js. Näin jälkikäteen mielestäni kielessä on omat haasteensa. Joskus kuulin lausahduksen, että react.js on helppo aloittaa mutta vaikea hallita. Siinä vaiheessa, kun ymmärtää komponenttien käytön ja valmiiden moduulien helppouden, kieli on helppo aloittaa. Kieli on front end –puolella suosittu mutta koska se on uusi, ulkoasun tekeminen ei aina ole helppoa. Tapa, jolla asiat on oppinut tekemään HTML:n ja CSS:n avulla, ei aina päde, ja aikaisemmin yksinkertainenkin asia saattaa paisua isommaksi ongelmaksi. Tästä johtuu mielestäni kielen vaikeus: se verhotaan HTML -tyyliseksi koodiksi mutta toimii aivan eri tavalla.

Verkkosivujen tulos on kuitenkin omasta mielestäni hyvä. Vaikka sivuilla ei ole niin paljon eläviä elementtejä kuin olin alun perin suunnitellut, niin sivut ovat nykyaikaiset. React.js:llä tehdyt sivut ovat koodin puolesta uutta, ja niitä on tarvittaessa helppo päivittää. Jatkoa ajatellen sivut tarvitsisivat mielestäni enemmän liikkuvuutta, elementtejä, jotka reagoivat käyttäjän tekemisiin. On se sitten sivustolla liikkumisen elävöittämistä tai sivujen

antamaa infoa ruokalistasta. Kysymys on enemmänkin siitä, mitä ravintola haluaa ja miten toiveet voidaan toteuttaa mielenkiintoisesti ja houkuttelevasti.

Yhtenä jatkokehitysmahdollisuutena on luoda sivustolle uusia komponentteja, joilla saadaan sivustoa muokattua nopeasti ja helposti uudenlaiseksi toisia ravintoloita ajatellen. Sivustolle on myös myöhemmin tarkoitus luoda pöytävarausjärjestelmä, jota voi markkinoida muille pienille ravintoloille. Nykyiset pöytävarausjärjestelmät ovat kalliita ja sisältävät usein kuukausimaksuja. Pöytävarausjärjestelmän idea olisi luoda helposti muokattava systeemi, jonka voisi myydä kerralla ja päivittää vain tarvittaessa erillismaksulla.

Opinnäytetyön ajankäyttö venyi pahasti. Tuntimäärässä en ole tarkoitetusta 400 tunnista niin paljon jäljessä kuin alkuperäisestä työn valmistumisajasta. Työn piti valmistua tammikuun alkuun, mutta kuitenkin venyi toukokuulle. Olen tehnyt työtä noin 450 tuntia. Pitkät aikatauluvenymiset mutta pieni työmäärän ylitys johtuvat kahdesta täysin erillisestä asiasta. Alkuperäisessä työsuunnitelmassa en ollut ottanut ollenkaan huomioon henkilökohtaisia asioita. Tuntien osalta en osannut arvioida, että koodin tuottaminen vei enemmän aikaa kuin oletin. Tämä johtui uuden kielen opiskelusta ja sen haasteista.

Lähteet

Bain, L. 2016. ReactJS: Props vs. State. Luettavissa: http://lucybain.com/blog/2016/react-state-vs-pros/. Luettu 01.02.2018.

Bootstrap. Bootstrap. Luettavissa: http://getbootstrap.com/. Luettu 02.11.2017.

Brennan, M. 2015. ES6 Basics. Luettavissa: https://www.martin-brennan.com/es6-basics/. Luettu: 22.1.2018.

Brown, E. 2016. Learning JavaScript, 3rd Edition. O'Reilly Media Inc.

Chen, M. 2016. React for Dummies. Luettavissa: https://www.codecademy.com/articles/react-virtual-dom. Luettu: 07.11.2017.

Chinnathambi, K. 2016. Learning React. Addison-Wesley Professional.

Codecademy. React: The Virtual DOM. Luettavissa: https://www.slideshare.net/mitchbox/react-for-dummies. Luettu: 19.01.2018.

Github. Facebook/create-react-app. Luettavissa: https://github.com/facebook/create-react-app/blob/master/packages/react-scripts/template/README.md#folder-structure Luettu: 17.04.2018.

Horton, A; Vice, R. 2016. Mastering React. Packt Publishing.

Koodiviidakko. 2016. Blogi – Web designin ja uutiskirjeiden trendit. Luettavissa: https://www.viidakko.fi/ajankohtaista/koodiviidakko-blogi/kirjoitus/web-designin-ja-uutiskirjeiden-trendit.html. Luettu: 05.11.2017.

Korpela, J. 2014. HTML5-käsikirja. Docendo Oy. Jyväskylä.

Lukasz, Kyc. 2017. Top 10 ES6 features by example. Luettavissa: https://blog.pragmatists.com/top-10-es6-features-by-example-80ac878794bb. Luettu: 21.4.2018.

Manoj, S. 2017. React Components Explained. Luettavissa: https://codeburst.io/react-components-explained-96718311f20b. Luettu: 09.04.2018.

Mardan, A. 2017. React Quickly: Painless web apps with React, JSX, Redux, and GraphQL. Manning Publications.

Rachiele, G. 2018. React's Component Lifecycle. Luettavissa: https://itnext.io/reacts-component-lifecycle-6c13e09d10ad Luettu: 17.4.2018.

React. add React to a New Application. Luettavissa: https://reactjs.org/docs/add-react-to-a-new-app.html. Luettu: 17.4.2018.

Robie, J. What is the Document Object Model? Luettavissa: https://www.w3.org/TR/WD-DOM/introduction.html. Luettu: 7.1.2018.

Sheiko, D. 2015. JavaScript Unlocked. Packt Publishing.

Suoranta, T. 2017. Verkkopalvelun esteettömyys ei koske vain neliraaja-halvaantuneita. Luettavissa: https://www.aucor.fi/blogi/verkkopalvelun-esteettomyys-ei-koske-vain-neliraajahalvaantuneita/. Luettu: 22.04.2018.

Thinwik. 2017. Why ReactJS is gaining so much popularity these days. Luettavissa: https://medium.com/@thinkwik/why-reactjs-is-gaining-so-much-popularity-these-days-c3aa686ec0b3. Luettu: 11.5.2018.

TIEKE Tietoyhteiskunnan kehittämiskeskus ry. Graafisen ulkoasun suunnittelu. Luettavissa: https://www.tieke.fi/display/julkaisut/Graafisen+ulkoasun+suunnittelu. Luettu: 04.11.2017.

TIEKE Tietoyhteiskunnan kehittämiskeskus ry. Kohderyhmän määrittely. Luettavissa: https://www.tieke.fi/pages/viewpage.action?pageId=3441045. Luettu: 04.11.2017.

Vikas, K. 2017. Learn React Essentials in 5 Minutes. Luettavissa: https://www.codementor.io/kavithavikas/learn-react-essentials-in-5-minutes-9ftje553w. Luettu: 11.5.2018.

W3School.com. CSS Introduction. Luettavissa: https://www.w3schools.com/Css/css_intro.asp. Luettu: 04.11.2017.

WixBlog. 2017. 5 Crucial Web Design Tips for a Professional Site. Luettavissa: https://www.wix.com/blog/2017/10/5-design-tips-for-a-professional-site/ . Luettu: 23.04.2018.

Wood, A 2015. HTML5 – New & Deprecated Features. Luettavissa: https://html.com/html5/#What_is_HTML5. Luettu: 02.11.2017.