Lauri Partinen ktkt23sp

Skriptaus harjoitustyö

Osakekurssi työkalu

2025

Kaakkois-Suomen
ammattikorkeakoulu

SISÄLLYS

1 JOHDANTO

Projektin aiheena oli luoda osakekurssien seurantasovellus. Projektin tavoite oli luoda web-sovellus, joka tarjoaa tietoa eri pörssiyhtiöiden osakkeen hinnasta ja historiasta. Tarkoituksena oli luoda yksinkertainen käyttöliittymä, josta näkee tietoa osakkeen hinnasta.

2 TYÖN SUUNNITTELU

Työn suunnittelu lähti liikkeelle aiheen valinnasta. Aihe vaihtoehtoja oli useita, joissa piti ottaa huomioon harjoitustyön vaatimukset. Lopulta päädyin osakekurssi seurantasovellukseen, koska olen kiinnostunut aiheesta ja se sopi harjoitustyön aiheeksi. Tavoitteena oli luoda yksinkertainen sovellus, jossa voin soveltaa kurssin aikana opittuja asioita. Sovellus on tarkoitettu osakkeen hinnan tarkistamiseen ja edellisen päivien pörssin sulkemishintojen tarkasteluun. Sovellusta ei ole tarkoitettu treidaamiseen, koska se ei tarjoa jatkuvaa reaaliaikaista tietoa osakkeen hinnasta. Yksi sovelluksen jatkokehitysidea on luoda siitä jatkuva osakkeen hinnan seuraaja, joka soveltuu treidaamiseen.

Aiheen vallinnan jälkeen lähdin suunnittelemaan, mitä sovelluksessa tulisi olla. Käyttöliittymässä tulisi olla kenttä, johon voi syöttää haluamansa osakkeen tunnuksen. Tulosteesta tulisi nähdä nykyinen hinta ja hinnan muutos edellisestä päivästä prosentteina. Lähdin tutkimaan aihetta netistä, josta lopulta löysin Alpha vantagen. Alpha vantage tarjoaa reaaliaikaista ja historia dataa eri osakkeiden kursseista. Valitsin Alphan, koska se vaikutti fiksuimmalta ratkaisulta muihin vaihtoehtoihin nähden. Perustuen käyttäjien arvioihin ja dokumentaatioon. Alpha tarjoaa ilmaiseksi api-avaimen, jota pystyn hyödyntämään koodissa. Koodikielinä valitsin käyttöliittymään CSS ja html perustuen niiden käytettävyyteen käyttöliittymien tekemisessä ja suunnittelussa. Käytin backend puolella JavaScriptiä, joka on monipuolinen kieli ja sopiva käyttötarkoitukseen. Kurssin aikana opin kielistä perusteet, joten oli oppimisprosessin kannalta mielenkiintoista lähteä kasvattamaan tietopankkia kielistä.

Käyttöliittymän ensimmäisen version suunnittelun valmistuttua lähdin tutkimaan, miten voisin tehdä sovelluksesta mielenkiintoisemman, ja tuoda samalla vaativuutta projektiin. Löysin eri vaihtoehtoja tutkiessa chart.js kirjaston,

jossa pystyy tekemään erilaisia kaaviota ja diagrammeja. Aikani testaillessa sain alphan api-avaimen ja kirjaston keskustelemaan keskenään.

3 TEKNINEN TOTEUTUS

3.1 Arkkitehtuuri ja ulkoiset rajapinnat

Sovellus hakee reaaliaikaista dataa Alpha vantage api-avaimen kautta. Api-avain haetaan Alpha vantagen sivuilta, joka tarjoaa ilmaisen api-avaimen sovellusta varten. TIME_SERIES_DAILY tarjoaa tarvittavat tiedot osakekurssien sulkemishinnoista.

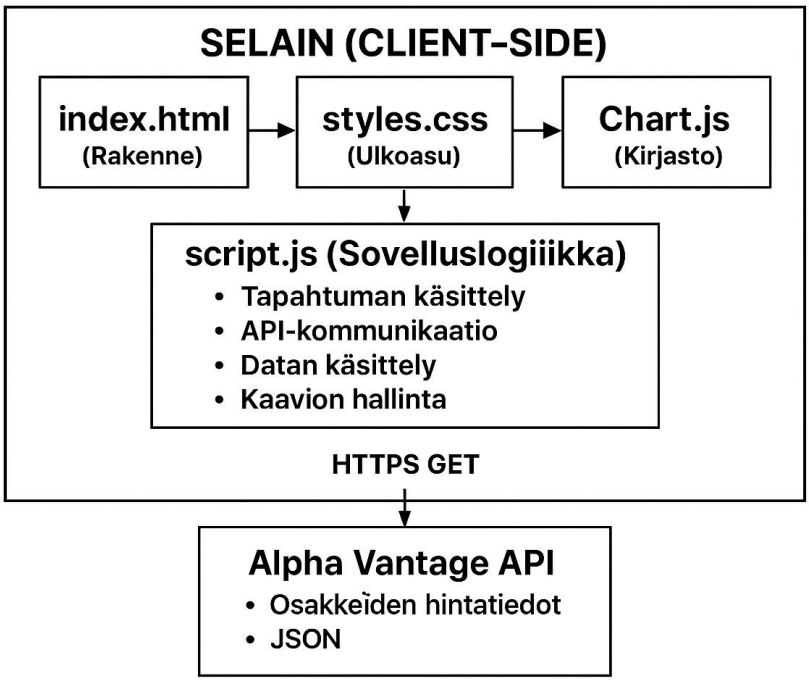
```
antage API -avain
KEY = 'KJVX2YR6OP4MU6R3';
 osaketiedot Api:sta
tion fetchStockData(symbol) {

osite, http-pyyntö ja tallennetaan json-muotoon
url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
response = await fetch(url);
data = await response.json();
```

Datan esittämiseen käytin chart.js kirjastoa, joka näyttää käyttöliittymässä api-avaimesta saadut tiedot.

3.2 Koodin rakenne ja logiikka

Koodin rakenne koostuu index.html, styles,css ja script.js. Lisäksi koodiin liitetään valmis kirjasto chart.js. Index.html vastaa sivulla näkyvän tekstien sisällöstä ja asettelusta. Styles.css vastaa kenttien sijoittelusta, fontin koosta ja värimaailmasta sovelluksessa. Script.js vastaa tapahtumien käsittelystä, api-kommunikaatiosta Alpha vantagen kanssa, datan käsittelystä ja chart.js saatavan kaavion hallinnasta. Api-key saadaan sovellukseen https get pyynnöllä.



3.3 Script.js

Aluksi haetaan HTML tiedosta viitaukset, joita tarvitaan käyttöliittymän päivittämiseen. Lopuksi luodaan chart muuttuja, johon tallennetaan chart.js objekti.

```
const stockInput = document.getElementById('stock-input');
const searchBtn = document.getElementById('search-btn');
const stockName = document.getElementById('stock-name');
const stockPrice = document.getElementById('stock-price');
const stockChange = document.getElementById('stock-change');
const stockChart = document.getElementById('stock-chart').getContext('2d');
```

Rakennetaan api-pyyntö, joka lähettää http pyynnön url-osoitteeseen. Koodi muuntaa vastauksen json-muotoon. Luodaan if-lause, jos tieto on saatavilla, päivitetään käyttöliittymä. Tieto ei saatavilla, virhe ilmoitus tulostuu.

```
// Alpha Vantage API -avain
const API_KEY = 'KJVX2YR6OP4MU6R3';
// Haetaan osaketiedot Api:sta
async function fetchStockData(symbol) {
  try {
    //Url osite, http-pyyntö ja tallennetaan json-muotoon
    const url = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${API_KEY}`;
    const response = await fetch(url);
    const data = await response.json();


    //Tarkistetaan datan saatavuus
    if (data['Time Series (Daily)']) {
      updateUI(symbol, data['Time Series (Daily)']);
    } else {
      alert('Stock not found! Please check the symbol and try again.');
    }
  } catch (error) {
    console.error('Error fetching stock data:', error);
    alert('Failed to fetch stock data. Please try again later.');
  }
}
```

Funktiossa ensin rajataan tiedot viidentoista päivän ajalta. Haetaan päivän päätöskurssi ja muutetaan float. Se mahdollistaa desimaalilukujen näytön tulosteessa. Lasketaan viimeisin hinta miinustamalla uusi vanhasta, jakamalla edellisen päivän hinnalla ja kertomalla sadalla. Lopuksi päivitetään nykyinen hinta kahden desimaalin tarkkuudella, muutosprosentti ja osakkeen symboli isolla kirjaimilla käyttöliittymään.

```
function updateUI(symbol, timeSeries) {
  const dates = Object.keys(timeSeries).slice(0, 15); // Haetaan tiedot 15 päivältä
  const prices = dates.map(date => parseFloat(timeSeries[date]['4. close'])); // Muutetaan päiv

  // Lasketaan viimeisin hinta, edellinen hinta ja prosentuaalinen muutos
  const latestPrice = prices[0];
  const previousPrice = prices[1];
  const changePercent = (((latestPrice - previousPrice) / previousPrice) * 100).toFixed(2);

  // Päivitetään käyttöliittymään symboli, hinnat ja muutos
  stockName.textContent = symbol.toUpperCase();
  stockPrice.textContent = `Price: $${latestPrice.toFixed(2)}`;
  stockChange.textContent = `Change: ${changePercent}%`;

  updateChart(dates.reverse(), prices.reverse());
}
```

UpdateChart funktiossa luodaan ja päivitetään kaaviota. Aluksi tuhotaan aiempi kaavio uuden tieltä. Luodaan viivakaavio. Koodissa pystyy muuttamaan kaavion värimaailmaa, viivojen paksuutta ja eri efektejä. Chart.js sivustolta löytyy valmis koodi, jota muokkasin projektin vaatimuksien mukaan.

```
function updateChart(dates, prices) {
  if (chart) {
    chart.destroy();
  }
  //Luodaan chart
  chart = new Chart(stockChart, {
    type: 'line',
    data: {
      labels: dates,
      datasets: [
        {
```

Lopuksi luodaan käyttöiättömässä olevalle napille toiminallisuus. Aluksi koodi lukee syötekentän arvon. Lopuksi lisäsin trim ominaisuuden, jotta lopputulos olisi siistimpi. Se poistaa tyhjät välit.

```
searchBtn.addEventListener('click', () => {
  const symbol = stockInput.value.trim();
  if (symbol) {
    fetchStockData(symbol);
  } else {
    alert('Please enter a stock symbol!');
  }
});
```

3.4 styles.css

Styles.css tiedostossa pystyy muokkaaman esimerkiksi eri osien värejä, fontin kokoa tai muotoa. Päätin käyttää css lopputyössä, koska kieli käytiin tunnilla läpi ja halusin lisätä vaativuustasoa työssä.

Perustusasetuksissa (body) lisäsin oletusfontiksi sans-serif fontin ja asetin taustavärin.

```
body {
    margin: 0;
    font-family: sans-serif;
    background-color: #f0f4f8;
    color: #000000;
}
```

.app container rajoittaa sisällön maksimi leveyden 800 pikseliin ja keskittää sen margin: 0 auto.

```
.app-container {
    max-width: 800px;
    margin: 0 auto;
}
```

Koodissa keskitetään otsikko. h1 asettaa otsikon tekstin punaiseksi ja fontti-kooksi 2rem, joka tarkoittaa fontin olevan kaksi kertaa oletuskoon kokoinen.

```
.header {
    text-align: center;
    margin-bottom: 20px;
}

.header h1 {
    color: #ff0000;
    font-size: 2rem;
    margin-bottom: 10px;
}
```

.input-container käyttää flex komentoa, jolla asetetaan syötekentän ja haku-painike vierekkäin. Justify-content: center; keskittää kentän vaakasuuntaan. Syötekentässä asetetaan leveys ja fontti. Lisäsin padding komennon tuomaan lisää tilaa syötekentille.

```css
.input-container {
  display: flex;
  justify-content: center;
  gap: 10px;
}

input#stock-input {
  font-family: sans-serif;
  padding: 10px;
  width: 200px;
  font-size: 1rem;
}
/* Hakupainikke*/
button#search-btn {
  font-family: sans-serif;
  background-color: #ff0000;
  color: white;
  font-size: 1rem;
}

button#search-btn:hover {
  background-color: #ff0000;
}
```

.main-content napin painalluksen jälkeen siirretään osat keskelle ja muokataan osat pystysuorassa sarakkeeseen nähden.

```css
.main-content {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

.stock-card keskitetään ja muutetaan osiot valkoisiksi. h2 muutetaan tekstin fontti ja väri punaiseksi, jotta se erottuisi valkoisesta.

```css
.stock-card {
  background-color: white;
  padding: 20px;
  text-align: center;
  margin-bottom: 20px;
  width: 100%;
}

.stock-card h2 {
  color: #ff0000;
  font-size: 1.5rem;
  margin-bottom: 10px;
}
```

Lopuksi loin kaaviolle valkoisen tausta, jotta kaavio olisi helpommin luettavissa.

```css
.chart-container {
  width: 100%;
  max-width: 700px;
  background-color: #ffffff;
  padding: 20px;
}
```

3.5 Index.html

Tiedosto koostuu pääosin eri sovelluksessa esiintyvien tekstien sisällön määrityksestä koodaamalla. Html tiedostossa huomion arvoista on komento <link rel="stylesheet" href="styles.css">, joka linkittää styles.css tiedoston tiedostoon. <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> komennolla ladataan ulkoinen kirjasto Chart.js sivustolta, jota käytetään osakekaavion piirtämiseen. Lopuksi komennolla <script src="script.js"></script> linkitetään script.js tiedosto.

4 KEHITTYMISEN ARVIONTI

Projektin aikana kehityin monella eri osa-alueilla. Syvensin tietotaitoa kolmesta eri koodikielistä. Loin sovelluksen, joka ottaa api-avaimesta tietoa ja tulostaa ne visuaaliseen muotoon. Kokonaisuuteen olen tyytyväinen, koska en ole aiemmin tehnyt vastaavanlaista projektia. Kurssin alun tavoitteisiin pääsin kohtuullisella tasolla. Projekti oli todella hyödyllinen tapa ymmärtää koodia uudella tavalla, miten mikäkin osa vaikuttaa kokonaisuuteen. Kokonaisuutena kehityin kurssin aikana tekemään ja ymmärtämään eri skriptejä paremmin verrattuna lähtötilanteeseen. Jatkoa ajatellen haluan jatkaa tietopankin kasvattamista skriptauksessa. Kurssilta sain hyvät pohjat skriptaukseen tulevaisuutta ajatellen.