

МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №8**

з дисципліни «Веб-технології та веб-дизайн-1.»

Варіант №12

**Виконав:**

студент гр. БС-83

Милейко К.Ю.

**Перевірив:**

ас. Давидько О. Б.,

ас. Матвійчук О. В.

Зараховано від

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Завдання:**

Створити 1 сторінковий сайт який має містити наступну інформацію:

Час з моменту останнього оновлення даних. (Бажано в header, та в секундах з точністю до 0.1)

Дані по позиціях на біржі для 8 найменувань (на ваш вибір)

**Пункт 1:**

Доступ до наних біржі відбувається за допомогою API описаного тут: <https://iexcloud.io/docs/api/>

Ключ доступу: pk\_ba9bb61a36a24ec3a5ae70362941f41c

Частота оновлень: 20 сек

p.s: даний ключ має задавитись конфігураційно та не повинен фігурувати на gitHub

**Пункт 2:**

Таблиця з результатами повинна містити наступні поля:

Назву компанії (Приклад: "Apple Inc.")

Назву позиції (Приклад: "AAPL")

Останню ціну позиції (Приклад: "158.73")

Різницю з моменту останнього оновлення (Приклад: "-1.67")

**При виконанні роботи обов'язково використовувати бібліотеку RXjs**

PS1: Потрібно чітко відділяти потоки роботи з даними

PS2: Статичні поля  необхідно запитувати лише 1 раз

Різницю в часі з моменту останньої угоди (Приклад: "2.9с") // постійно оновлювати

**index.html**<!DOCTYPE html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width,initial-scale=1">

<title>Exchange Tracker</title>

<script src="https://unpkg.com/rxjs/bundles/rxjs.umd.min.js"></script>

<link rel="stylesheet" href="./css/style.css">

</head>

<body>

<h1>Exchange Tracker</h1>

<div id="time-wrapper">

<h3>Last update was <div id="time">0</div> seconds ago.</h3>

</div>

<div id="container"></div>

<script src="./js/main.js"></script>

</body>

</html>

**style.css**

h1 {

text-align: center;

}

#container {

display: flex;

justify-content: center;

align-items: center;

}

#time-wrapper {

text-align: center;

}

#time-wrapper h3 {

display: inline-block;

}

#time {

font-size: 25px;

}

table {

border-collapse: collapse;

width: 70%;

}

th,

td {

text-align: left;

padding: 8px;

}

tr {

background-color: green;

}

th {

background-color: yellow;

color: black;

}

**main.js**

class ExchangeTracker {

constructor(baseURL, symbols, staticFilters, dynamicFilters, token) {

this.baseURL = baseURL;

this.symbols = symbols;

this.static = staticFilters;

this.dynamic = dynamicFilters;

this.token = token;

this.time = Date.now();

}

getUrl(isStatic = false) {

const smb = this.getQuery(this.symbols, "symbols");

const flt = this.getFilters(isStatic);

return `${this.baseURL}?${smb}&${flt}&types=quote&token=${this.token}`

}

getQuery(args, name) {

let query = name + "=";

for (const arg of args) {

query += arg + ',';

}

return query.substring(0, query.length - 1);

}

getFilters(isStatic) {

if (isStatic) {

return this.getQuery(this.static.concat(this.dynamic), "filter")

}

return this.getQuery(this.dynamic, "filter")

}

async getData(fetchUrl) {

let data = await this.fetchData(fetchUrl);

return this.parseData(data);

}

parseData(data) {

let dataArray = [];

for (var key in data) {

let quote = data[key]["quote"];

quote.previousClose = (quote.previousClose - quote.latestPrice).toFixed(2)

dataArray.push(quote);

}

return dataArray;

}

async fetchData(url) {

let response = await fetch(url);

if (!response.ok) {

console.log("Error HTTP code: " + response.status);

return "";

}

return response.json();

}

updateTable(data) {

let tableData = [];

const keys = Object.keys(data[0])

for (const key of keys) {

tableData.push(document.getElementsByClassName(key));

}

for (let i = 0; i < data.length; i++) {

let values = Object.values(data[i])

for (let j = 0; j < values.length; j++) {

tableData[j][i].innerHTML = values[j];

}

}

this.time = Date.now();

}

updateTimeWorker() {

let timeDiff = (Date.now() - this.time) / 1000

document.getElementById('time').textContent = timeDiff.toFixed(1);

}

async updateTableWorker(partialUrl) {

let data = await this.getData(partialUrl);

this.updateTable(data);

}

async createTable(fullUrl, columnNames) {

let data = await this.getData(fullUrl);

this.buildTable(data, columnNames);

}

buildTable(rows, columnNames) {

let html = '<table id="data-table">';

html += '<tr>';

for (let j of columnNames) {

html += '<th>' + j + '</th>';

}

html += '</tr>';

for (let i = 0; i < rows.length; i++) {

html += '<tr>';

for (let j in rows[i]) {

html += `<td class="${j}">` + rows[i][j] + '</td>';

}

html += '</tr>';

}

html += '</table>';

document.getElementById('container').innerHTML = html;

}

run() {

const fullURL = this.getUrl(true);

const partialURL = this.getUrl();

let columnNames = ["Company name", "Position name", "Latest priсe", "Latest priсe diff"];

this.createTable(fullURL, columnNames)

rxjs.interval(100)

.subscribe(() => {

this.updateTimeWorker();

});

rxjs.interval(20000)

.subscribe(() => {

this.updateTableWorker(partialURL);

});

}

}

companies = [

"AAPL",

"FB",

"TSLA",

"AMZN",

"MSFT",

"KO",

"SPOT",

"NFLX",

"NOK",

"SNE"

]

staticFilters = ["symbol", "companyName"]

dynamicFilters = ["latestPrice", "previousClose"]

new ExchangeTracker("https://cloud.iexapis.com/v1/stock/market/batch",

companies,

staticFilters,

dynamicFilters,

"token-placeholder").run();

