МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Савченко А.С.

«___»_____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТР"

Тема: «Функціональний плагін веб-браузера для конвертації валюти»

Виконав: Гриб Марина Олегівна

Керівник: к.т.н., доц. Моденов Юрій Борисович

Нормоконтролер з ЄСКД (ЄСПД): Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій
Спеціальність 122 "Комп'ютерні науки та інформаційні технології"
Спеціалізація «Інформаційні управляючі системи та технології (за галузями)»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Савченко А.С.

" " _____2019р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Гриб Марини Олегівни
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): «Функціональний плагін веб-браузера для конвертації валюти» затверджена наказом ректора №2175/ст. від 14.10.2019р.
2. Термін виконання проекту (роботи): з 14.10.2019р. по 09.02.2020р.
3. Вихідні данні до проекту (роботи): розроблений функціональний плагін для конвертації та порівння валюти.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): основні технології та методи розробки плагіну. Архітуктура та структурні файли. Етапи розробки функціонального плагіну. Кросбраузерність. Завантаження плагіну в браузери. Публікування в інтернет-магазині розширень. Висновки.
5. Перелік обов'язкового графічного матеріалу: модель роботи AJAX, архітектура розширення, структура файлів плагіну, представлення даних в форматі JSON, завдання background page, застосування пошуку по таблиці, відображення завантаженого в браузер розширення, html-таблиця з вихідними даними.

6. КАЛЕНДАРНИЙ ПЛАН

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Приміт ка
1.	Аналіз літератури та джерел за темою дипломного проекту.	13.10.19р. – 20.10.19р.	
2.	Розроблення та затвердження плану дипломного проекту.	21.10.19р. – 22.10.19р.	
3.	Проведення консультації з науковим керівником щодо створення першого розділу.	23.10.19р. – 27.10.19р.	
4.	Розробка розділу 1: Основні підходи та технології розробки плагіну.	30.10.19р. – 22.11.19р.	
5.	Розробка розділу 2: Архітектура та структурні файли плагіну.	23.11.19р. – 08.12.19р.	
6.	Розробка розділу 3: Розробка функціонального плагіну.	09.12.19р. – 15.12.19р.	
7.	Розробка розділу 4: Кросбраузерність плагіну. Завантаження до браузерів.	15.12.19р. – 22.12.19р.	
8.	Висновки та оформлення пояснювальної записки дипломного проекту.	25.12.19р. – 29.12.19р.	
9.	Підписання необхідних документів у встановленому порядку.	15.01.20р. – 19.01.20р.	
10.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	22.01.20р. – 31.01.20р.	

7. Дата видачі завдання: 13.10.2019р.

Керівник дипломного проекту _____ Моденов Ю.Б.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Гриб М.О.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту роботи «Функціональний плагін веб-браузера для конвертації валюти» викладена на 82 с., містить 38 рис., 11 літературних джерел.

Ключові слова: ВЕБ-ТЕХНОЛОГІЇ, ВЕБ-БРАУЗЕР, РОЗРОБКА ПЛАГІНУ, ФУНКЦІОНАЛЬНИЙ ПЛАГІН, ПЛАГІН ДЛЯ КОНВЕРТАЦІЇ ВАЛЮТИ, ЗАВАНТАЖЕННЯ ПЛАГІНУ.

***Об'єкт дослідження*:** засоби та технології розробки функціонального плагіну для веб-браузера.

Предмет дослідження: створення функціонального плагіну для конверації валюти.

Мета роботи: розробити функціональний кросбраузерний плагін, що дозволяє отримувати дані про зміну курсів валют відносно гривні, конвертувати введену суму в обрану валюту, проводити аналіз коливань курсу за допомогою графіка.

Методи дослідження: аналіз підходів та методів функціонування веб-плагіну, порівняльний аналіз існуючих технологій проектування на який буде орієнтуватись подальша розробка.

Отримані результати: функціональний кросбраузерний плагін, що дозволяє отримувати дані про зміну курсів валют відносно гривні, конвертувати введену суму в обрану валюту, проводити аналіз коливань курсу за допомогою графіка.

Результати дипломної роботи використовуються користувачами веб-браузера Google Chrome.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 9
ОСНОВНІ ПІДХОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ПЛАГІНУ 9
1.1 Web – технології для побудови плагінів 9
1.1.1 Базові HTML та CSS технології 10
1.1.2 Java Script та бібліотека JQuery 17
ВИСНОВОК ДО РОЗДІЛУ 1 32
РОЗДІЛ 2 33
АРХІТЕКТУРА ТА СТРУКТУРНІ ФАЙЛИ ПЛАГІНУ 33
2.1 Архітектура плагіну 33
2.2 Розгляд структурних файлів плагіну 34
2.2.1 Manifest.json 34
2.2.2 Background.js 36
2.2.3 Content Script 37
2.2.4 User Interface Elements 39
2.2.5 Content Security Policy 40
2.2.6 Формат передачі даних JSON 40
ВИСНОВОК ДО РОЗДІЛУ 2 42
3 РОЗДІЛ 43
РОЗРОБКА ФУНКЦІОНАЛЬНОГО ПЛАГІНУ 43
3.1 Розгляд структурних файлів плагіну 43
3.2 Навігація плагіну 49
3.3 Вкладка «Currency» 51
3.3.1 Отримання даних через API 51
3.3.2 Створення html-таблиці з вихідними даними 52

3.3.3 Пошук по таблиці 53
3.3.4 Оновлення даних 54
3.4 Вкладка «Converter» 55
3.4.1 Формування списку валют і додавання в select 55
3.4.2 Створення валідації для поля «Amount» 56
3.4.3 Реалізація конвертування за формулою 58
3.6 Вкладка «Choose Date» 59
3.7 Вкладка «Chart» 61
3.7.1 Визначення методів для побудови графіка 61
3.7.2 Побудова графіку 62
ВИСНОВОК ДО РОЗДІЛУ 3 66
РОЗДІЛ 4 67
КРОСБРАУЗЕРНІСТЬ ПЛАГІНУ. ЗАВАНТАЖЕННЯ ДО ОСНОВНИХ БРАУЗЕРІВ. ПУБЛІКУВАННЯ ПЛАГІНУ В МАГАЗИНІ РОЗШИРЕНЬ GOOGLE CHROME 67
4.1 Кросбраузерність плагіну 67
4.2 Завантаження плагіну до бібліотеки розширень браузерів 68
4.2.1 Google Chrome 68
4.2.2 Mozilla Firefox 70
4.2.3 Microsoft Edge 71
4.3 Завантаження плагіну до магазину розширень Google Chrome 73
ВИСНОВОК ДО РОЗДІЛУ 4 77
ВИСНОВКИ 78
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ 80
ДОДАТОК А 81
ДОДАТОК Б 85
ДОДАТОК В 93

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface

Sheets

Model

JS - Java Script

JSON – Java Script Object Notation

URL – Uniform Resource Located

Consortium

XPCOM - Cross Platform Component Object Model

Install

- XML-мова інтерфейсу користувача

ВСТУП

Розвиток новітніх технологій та засобів обробки інформації все частіше спонукає розробників програмного забезпечення до заощадження обсягу використання програмних ресурсів та пам’яті пристрою. Користувачі звикли отримувати шукану інформацію в зручному для них вигляді та прагнуть скоротити час на її пошук. Тому постає питання актуальності застосування плагінів в браузерах. Такий спосіб представлення робочого місця користувача Інтернету, дозволяє виокремити найбільш важливі дані і розмістити їх безпосередньо поверх відкритого вікна браузера.

Браузер, також веб-браузер, – програмний компонент для комп'ютера, та цифрових електронних пристроїв, як правило, з доступом до мережі Інтернет. Веб-переглядач надає користувачу графічний інтерфейс для інтерактивного пошуку, перегляду та обробки даних у мережі.

За статистикою найбільш розповсюдженими браузерами є Google Chrome, Mozilla Firefox, Internet Explorer. Інтерфейси всіх браузерів оснащені багатьма інструментами для розробників та звичайних користувачів, та як правило, вміщають плагіни.

Плагін – це самостійно компільований програмний модуль, що здатний підключатися до основної програми та призначений для розширення або використання її можливостей. Плагіни призначені для розширення чи використання загальних можливостей браузера, підвищують його продуктивність. Вперше плагіни з’явилися в 4 версії Chrome, а галерея плагінів офіційно відкрита 25 січня 2010 року. При відкритті вона налічувала понад 1500 плагінів.

Будь-який браузер має ряд вбудованих функцій, тому, область розширення його можливостей, як от створення функціонального плагіну для конвертації валюти є актуальним питанням серед розробників програмного забезпечення.

РОЗДІЛ 1
ОСНОВНІ ПІДХОДИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ПЛАГІНУ

1.1 Web – технології для побудови плагінів

Плагіни - (англ. Plug-in) можуть використовуються для зміни поведінки наявних функцій або для додавання нових можливостей [1]. Розширення особливо популярні у Firefox, оскільки розробники Mozilla створювали браузер, як досить мінімалістичну програму, що мало запобігти росту кількості помилок та громіздкості коду програми, зберігаючи при цьому високий степінь розширення. Таким чином даючи змогу користувачам додавати функції, яким вони віддають перевагу.

Плагін, зазвичай, використовується, щоб додати нові можливості до основних функцій програми. Приклади можливостей, які можуть бути додані за допомогою розширень: засоби веб-розробки, менеджери закладок, клієнтські програми для окремих веб-ресурсів, менеджери протоколу FTP, електронна пошта, рухи мишки, перемикання проксі-серверів, тощо.

Як правило, застосовується розширення наступних технологій: CSS, DOM, XPCOM, XP, XUL, Connect XPI.

Багато розширень можуть змінювати вміст веб-сторінки при її відтворенні на екрані. Наприклад, програми що блокують відтворення рекламного контенту на веб-сторінці, або ж навпаки, програми які доповнюють вміст основної сторінки. Загалом, це здійснюється шляхом зміни, або доповнення таблиць каскадних стилів.

Засобом перегляду веб-ресурсів є браузер, або веб-переглядач - програма, яка використовується для взаємодії з мережею Інтернет для пошуку, обробки, перегляду веб-сайтів, виведення сторінок на екран і переходу користувача між веб-сторінками в пошуку потрібної інформації.

За допомогою браузера користувач переглядає вміст веб-сайту та взаємодіє з контентом.

Сучасні браузери підтримують не лише HTTP-протокол, необхідний їм для перегляду гіпертексту. Вони, зазвичай, забезпечують роботу з електронною поштою та участь у телеконференціях, надають доступ до файлових архівів FTP, голосового та відеозв'язку. Також зберігають історію відвідин інформаційних ресурсів, щоб у разі потреби знайти потрібний, раніше відвіданий користувачем ресурс. Потрібно дещо відзначити з основного функціоналу браузера - це можливість зберігання веб-ресурсів в закладках для їх запам'ятовування і швидкого доступу.

1.1.1 Базові HTML та CSS технології

HTML — стандартна мова розмітки документів для Web, де всі веб-сторінки створюються за допомогою HTML (або XHTML). Мова HTML інтерпретується браузером у вигляді документа, зручному для відтворення та розуміння людиною.

HTML створювався в 1991-1992 роках, як мова для обміну науковою та технічною документацією, яка зручна для людей, що не є спеціалістами з верстки. Також, крім спрощення структури документу, у HTML міститься підтримка гіпертексту. Мультимедійні можливості були додані значно пізніше. Текстові документи, які містять код на мові HTML, обробляються спеціальними програмами, які відображають документ у відформатованому вигляді.

Такі програми, як веб-браузери, забезпечують зручний графічний інтерфейс для взаємодії користувача із сервером — запит до веб-сторінок, їх відображення та відправлення введених користувачем даних на сервер [2]. Від початку HTML був спроектований і створений як засіб структурування та форматування документів, без їх прив’язки до засобів відображення. Але сучасні застосування HTML далекі від його початкових цілей — додані мультимедійні можливості, з’явилися засоби для створення складних графічних представлень, додана можливість підключення плагінів та розширень. Для створення динамічних сторінок було розроблений цілий ряд технологій — JavaScript, Java Апплети,

Adobe Flash, Microsoft Silverlight. Реалізації деяких з них інтегровані в браузери (JavaScript), для роботи з іншими потрібно підключати спеціальні плагіни (доступні безкоштовно на веб-сайтах розробників або поставляються разом з операційними системами чи браузерами).

В середині 90-х років розгорнулось боротьба між розробниками найбільш популярних (на той час) браузерів — Netscape Navigator та Microsoft Internet Explorer за ринок інтернет-браузерів. Основний спосіб боротьби — розробка та впровадження нових технологій, що були не сумісні з іншими браузерами. В результаті навіть на сьогоднішній день не вдалося досягти повної сумісності між усіма браузерами, хоча їх розробники та консорціум W3C, який займається стандартизацією Web-технологій, докладають максимум зусиль для цього. З іншого боку, в результаті цієї боротьби, з’явився ряд технологій, що займають основне місце в розвитку сучасного Web, серед них — JavaScript та Ajax. Зараз важко знайти сайт, побудований згідно принципів Web 2.0, який би не використовував описаних вище технологій.

HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок. HTML представляє засоби для: створення структурованого документу, шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше; отримання інформації із Всесвітньої мережі через гіперпосилання; створення інтерактивних форм; включення зображень, звуку, відео, та інших об'єктів до тексту. У загальному з нової версії мови розмітки пропонується прибрати близько 15 тегів.

HTML5

При прийнятті рішення про введення нових тегів було розглянуто більшість популярних сайтів і виділено основні елементи, які були спільними для всіх веб-сторінок.

Структуруючи області на веб-сторінці за допомогою певних елементів, ця технологія допомагає полегшити навігацію по сторінці. Наприклад, користувач може легко скролити до потрібного розділу навігації або швидко переходити від

однієї статті до іншої, без необхідності для розробників робити відповідні посилання. Розробники також отримують вигоду в результаті заміни великої кількості блоків одним з декількох відповідних елементів, що також приводить до чистого і легкого початкового коду.

Елементами header включає інформацію, яка потрібна для коректного відображення вмісту сторінки в браузері. Це кодування символів, так звані мета-теги, заголовок сторінки, який відображається на вкладці браузера, вкладені стилі для документа, підключення шрифтів та інше.

Елемент footer визначає нижню частину розділу, до якого він відноситься. Зазвичай він містить інформацію про розділ — наприклад, посилання на соціальні мережі, пошту, ім'я автора, посилання на схожі документи, копірайт і тому подібне.

Блок nav містить список посилань для навігації. Використовується, для зручної навігації по веб-сторінці, підрозділам, що значно полегшує роботу користувача та вважається доречним способом переміщення по сторінці.

Тег section представляє окремий розділ документа, який об’єднується по певній тематиці. Наприклад: контакти, галерея, портфоліо та інше.

Тег article відзначає незалежний розділ документа, сторінки або сайту. Застосуємо для такого вмісту як новини, запису блога, повідомлення у форумі або коментарі користувачів.

В HTML5 для того, щоб підключити до сторінки відео або аудіо-доріжку, достатньо скористатись відповідними тегами.

Такі сайти як Youtube, Viddler, Revver, Myspace та десятки інших дозволяють будь-кому опублікувати свої відео або аудіо-файли. Багато сайтів застосовують технологію Flash, щоб забезпечити потрібну функціональність, оскільки в HTML 4.1 не вистачає засобів для вбудовування та широкого управління мультимедіа.

І хоча такі можливості надають різноманітні плагіни, то в наш час Flash — це єдиний широко поширений плагін, який надає крос-браузерне рішення з відповідним API для розробників.

Як доведено великою кількістю медіа-програвачів на основі Flash, автори зацікавлені в наданні інтерфейсу з їхнім власним дизайном, який, як правило, дозволяє користувачам включати, ставити на паузу, зупиняти, перемотувати і керувати гучністю відтворення медіа-файлів. Стоїть завдання надати таку функціональність за допомогою додавання можливості вбудовувати відео і аудіо засобами браузера, а також надати DOM API для доступу до скриптів.

Елементи video і audio легко дозволяють це зробити. Більшість API — загальні між цими елементами, з відмінностями лише відносно до visual і не-visual медіа.

Всі сучасні браузери (окрім Internet Explorer) вже реалізували підтримку даних елементів. Найпростіший спосіб вбудувати відео — це використовувати тег video і дозволити браузеру відобразити інтерфейс за умовчанням. Булевий атрибут controls визначає чи включати за замовчуванням цей користувацький інтерфейс.

Необов'язковий атрибут poster може використовуватися для вказання зображення, яке буде відображатися до того як відео почне програватися. Хоча є формати відео, що підтримують власний попередній перегляд, цей спосіб — рішення, що дозволяє бути незалежним від відео-формату.

Також просто підключити і аудіо — використовуючи елемент audio. Хоча по очевидних причинах у тега audio немає атрибутів height, width і poster, між video і audio більшість атрибутів спільні.

У HTML5 включений елемент source для вказівки альтернативних відео і аудіо файлів, щоб браузер міг вибрати той, який підходить до підтримуваного медіа-типу або кодеків. Атрибут media визначає вибір медіа-запиту, що базується на обмеженнях пристроїв, а атрибут type — можливості медіа-типів і кодеків. Коли використовується атрибут source, слід опускати src в елементах video (audio), інакше source буде проігнорований.

Існує набір технологій, які дуже часто помилково відносять до HTML5, хоча вони насправді стосуються інших специфікацій [3]:

- WebGL

- FileReader
- XMLHttpRequest
- querySelector(All)
- Geolocation
- ECMAScript5
- CSS3
- XBL2
- Web Workers
- Web Sockets
- Faster JavaScript

XHTML (Extensible Hypertext Markup Language) — мова розмітки, що має менш поширену ідеологію ніж HTML, але відповідає синтаксичним правилам XML. В той час, як HTML побудовано на основі правил SGML, XHTML побудовано на основі правил XML, суворішої підмножини правил SGML. Оскільки XHTML-документи мають бути коректними XML документами, їх обробку можна здійснювати стандартними інструментами обробки XML - документів на відміну від HTML, який вимагає порівняно складніших, важчих і повільніших синтаксичних аналізаторів.

XHTML 1.0 є «реформуванням трьох типів документів стандарту HTML 4 засобами XML 1.0». W3C також продовжує підтримку Рекомендації HTML 4.01 та активну роботу над специфікаціями 23 стандартів HTML5 і XHTML5.

Більшість великих виробників браузерів не бажали реалізовувати функції з нових проектів стандартів W3C XHTML оскільки вважали, що вони не відповідають сучасним потребам розвитку Інтернету, а W3C захопився формалізмом XML і не реагує на реальні вимоги виробників. Apple, Mozilla та Opera сформували робочу групу WHATWG, яка почала працювати над стандартом HTML5, який допускав, але не вимагав застосування XML.

XHTML був розроблений з метою зробити HTML більш розширюваним і підвищити сумісність з іншими форматами даних.

Стандарт XML, затверджений в 1998 році, пропонував простіший формат даних, ближче за духом до HTML 4. Існували сподівання, що за допомогою переходу на формат XML, HTML стане більш сумісним із загальними засобами XML, а проксі сервери зможуть перетворювати документи, у разі необхідності, для пристроїв з обмеженими можливостями, таких як мобільні телефони.

CSS

CSS — каскадні таблиці стилів, що використовуються для опису стилів сторінок, написаних мовами гіпертекстової розмітки [4]. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують запити CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки та їхньої візуальної презентації. CSS використовується розробниками та відвідувачами веб-сторінок, щоб задати кольори, шрифти, розміщення елементів та інші аспекти вигляду сторінки.

Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS). Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо.

CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою, у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін. Один і той самий HTML або XML документ може бути відображений по різному, залежно від використаного CSS. Стилі для відображення сторінки можуть бути:

Таблиці стилів, залежно від розміщення поділяються на:

- зовнішні таблиці стилів, найчастіше як окремий файл або файли .css;
- внутрішні таблиці стилів, включені як частина HTML-документу або блоку;
- вбудовані стилі для окремого елементу.

Стилі користувача - локальний css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера.

Стилі переглядача (браузера) - стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена більш загальними стилями.

Переваги:

- інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
- різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила

застосування і сторінка побудована з урахуванням їх;

- прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі.

1.1.2 Java Script та бібліотека JQuery

JavaScript – прототипно-орієнтована сценарна мова програмування. Є реалізацією мови ECMAScript. JavaScript зазвичай використовується, як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах, як мова сценаріїв для додання інтерактивності та динамічності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, функції як об'єкти класу. На JavaScript вплинули багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легкою для використання. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє її від ряду мов програмування, використовуваних у веб-розробці. Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation.

JavaScript є об'єктно-орієнтованою мовою, але використовуване в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з звичними класово-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, які властиві функціональним мовам, - функції як об'єкти класу, об'єкти як списки, каррінг, анонімні функції, замикання - що додає мові додаткову гнучкість.

У мові відсутні такі корисні речі , як [5]:

- модульна система: JavaScript не надає можливості управляти залежностями і ізоляцією областей видимості;
- стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управлінню потоками введення-виведення, базових типів для бінарних даних;

- стандартні інтерфейси до веб-серверів і баз даних;
- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

ECMAScript

Специфікація ECMAScript є низкою вимог до реалізації ECMAScript. Цих вимог слід дотримуватись, щоб запровадити відповідні стандартам можливості мови у реалізації ECMAScript, або, щоб створити рушій (наприклад, SpiderMonkey у Firefox або v8 у Chrome).

Документ ECMAScript не призначений для того, щоб надавати допомогу у програмуванні скриптів. Для отримання інформації про написання скриптів, зазвичай, використовують документацію JavaScript.

У докуметації ECMAScript використовується термінологія і синтаксис, з якими JavaScript-програмісти можуть бути і не знайомі. Не зважаючи на те, що опис мови у ECMAScript може бути дещо іншим, сама мова залишається такою ж. JavaScript підтримує весь функціонал, закладений у специфікації ECMAScript.

Документація JavaScript описує аспекти мови, які використовує розробник ПО на JavaScript.

JavaScript доволі компактна та гнучка мова. Розробники забезпечили велике розмаїття інструментів, що доповнюють основу мови JavaScript, які відкривають величезну кількість додаткового функціоналу з мінімальними зусиллями. Серед них:

- Програмні інтерфейси (APIs) для браузерів — API, які вбудовані у браузери, що надають функціонал на зразок динамічного створення HTML та застосування CSS-стилів, збір та обробка відео-потоків з вебкамери користувача, генерація 3D-графіки та аудіо-семплів.
- API третіх осіб, що дозволяють розробникам інтегрувати у власні сайти функціонал інших провайдерів, таких як Twitter або Facebook.

- Фреймворки та бібліотеки третіх осіб, які ви можете застосувати до вашого HTML, щоб прискорити створення сайтів та застосунків.

JavaScript найчастіше використовується як частина браузера , що надає можливість виконання коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером , змінювати структуру та зовнішній вигляд веб-сторінки.

Мова JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор , стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF -документів тощо.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів функції як об'єкти першого класу обробка винятків
- автоматичне приведення типів
- автоматичне прибирання сміття
- анонімні функції

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів.

Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки , поширилося використання JavaScript поза браузером.

JQuery

JQuery — популярна JavaScript-бібліотека з відкритим сирцевим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом. Згідно з дослідженнями організації W3Techs, jQuery використовується понад половиною від мільйона найвідвідуваніших сайтів.

JQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день. Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків.

JQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

Основне завдання jQuery — це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

Існують два варіанти під’єднання jQuery UI:

- Локальне під’єднання. Даний спосіб вимагає завантаження файлу з офіційного сайту.
- Віддалене під’єднання. Даний спосіб не вимагає завантаження файлу, а замість цього використовує його віддалено.

JQuery намагається відокремити поведінку від загальної структури HTML-документа. Наприклад, якщо мова йде про роботу обробника подій натискання кнопки, то замість безпосереднього його зазначення, управління переходить до

JQuery. Бібліотека спочатку визначає кнопку, після чого трансформує сигнал в обробник подій кліку.

Серед модулів бібліотеки jQuery є багато вигідних компонентів, які можна застосувати в широкому спектрі завдань розробника. Разом з тим, команда розробників бібліотеки не намагалася створити набір інструментів, які дозволяли б поєднувати велику кількість jQuery-функцій, оскільки це може призвести до створення великих фрагментів коду. Саме тому вибрали архітектуру універсального ядра, яка містить бібліотеку і плагіни. Це дає можливість об'єднати тільки ті JavaScript-функції, які по-справжньому потрібн.

1.2 Технології та підходи для побудови інтерфейсу

1.2.1 Технологія Ajax

AJAX (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-додатків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX – один з компонентів 21 концепції DHTML.

Про AJAX заговорили після появи в лютому 2005 р. статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до веб-додатків». Гарретт придумав термін, коли йому довелося якось назвати новий набір технологій, пропонований ним клієнту. Проте в тій чи іншій формі багато технологій були доступні і використовувалися набагато раніше, наприклад в підході «Remote Scripting», запропонованому компанією Microsoft в 1998 р., або з використанням HTML-елемента IFRAME, що з’явився в Internet Explorer 3 в 1996 р.

AJAX – це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів [6]:

- використання DHTML для динамічної зміни змісту сторінки;

- використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю;
- альтернативний метод – динамічне підвантаження коду JavaScript.

Розглянемо класичну модель веб-додатку:

1. Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
2. Браузер надсилає запит серверу.
3. У відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т. д.
4. З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

Модель AJAX (рис. 1.1):

1. Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
2. Браузер відправляє відповідний запит на сервер.
3. Сервер віддає тільки ту частину документа, яка змінилася.

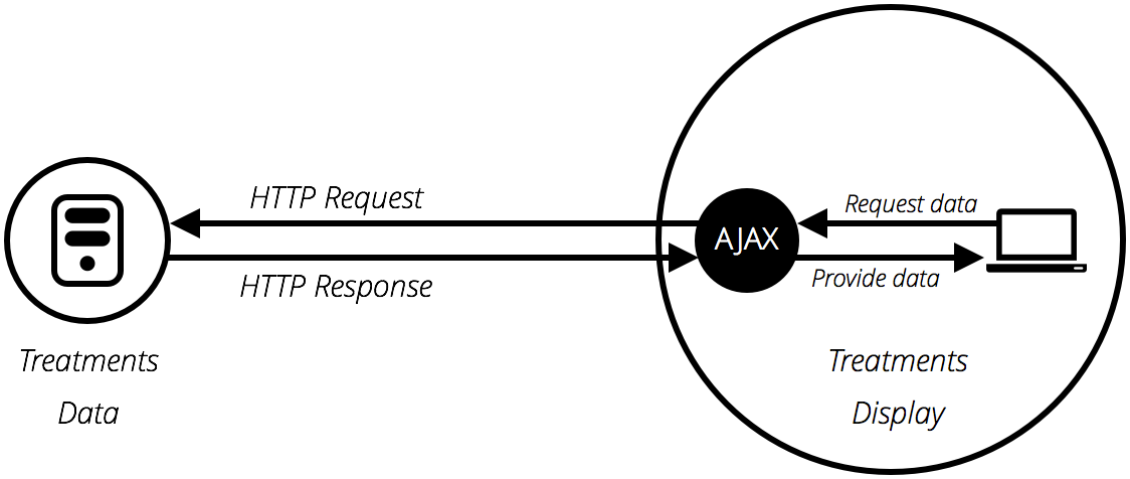

Рис. 1.1 Модель роботи AJAX

В деяких додатках використовуються певні варіації з форматом відповіді сервера, такі варіації набули напівофіційних назв. AHAH (Asynchronous HTML and HTTP) – це споріднений AJAX підхід для динамічного оновлення веб-сторінок, з використанням JavaScript.

Основною його відмінністю від AJAX є те, що відповіді сервера повинні бути звичайним HTML. Перевага підходу полягає у більшій сумісності і функціональності (підтримка навігаційних кнопок браузера, аплоад файлів тощо).

Реалізується у вигляді звичайних фреймів, що автоматично змінюють свій розмір під розмір вмісту, або у вигляді прихованих фреймів, що виконують тільки функції завантаження даних. Asynchronous XHTML and HTTP, або абревіатура AXAH – це майже те ж саме що і AHAH. Різниця тільки в тому, що в AHAH сервер клієнтові повертає HTML, а в AXAH вже XHTML.

Порівняємо стандартний підхід і AJAX.

У класичній моделі веб-додатку:

- Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
- Браузер формує і відправляє запит серверу.
- У відповідь сервер генерує абсолютно нову веб-сторінку і відправляє її браузеру і т. д. Після чого браузер повністю перевантажує всю сторінку.

При використанні AJAX: користувач заходить на веб-сторінку і натискає на який-небудь її елемент. Скрипт (на мові JavaScript) визначає, яка інформація необхідна для оновлення сторінки. Браузер відправляє відповідний запит на сервер. Сервер повертає лише ту частину документа, на яку прийшов запит. Скрипт вносить зміни з урахуванням отриманої інформації (без повного перезавантаження сторінки).

AJAX – не самостійна технологія, а концепція використання декількох суміжних технологій. AJAX базується на двох основних принципах:

- використання технології динамічного звернення до сервера «на льоту», без перезавантаження всієї сторінки повністю, наприклад з використанням XMLHttpRequest (основний об'єкт);
- через динамічне створення дочірніх фреймів;
- через динамічне створіння тега, як це реалізовано в Google Analytics;

CSS, DOM і JavaScript – складають DHTML (англ. Dynamic HTML). Як формат передачі даних можуть використовуватися фрагменти простого тексту,

HTML-коду, JSON або XML. Дії з інтерфейсом перетворюються в операції з елементами DOM (англ. Document Object Model), за допомогою яких обробляються дані, доступні користувачеві, в результаті чого подання їх змінюється. Тут же проводиться обробка переміщень і клацань мишею, а також натискань клавіш.

Каскадні таблиці стилів, або CSS (англ. Cascading Style Sheets), забезпечують узгоджений зовнішній вигляд елементів програми та спрощують звернення до DOM- об'єктів. Об'єкт XMLHttpRequest (або подібні механізми) використовується для асинхронного взаємодії з сервером, обробки запитів користувача і завантаження в процесі роботи необхідних даних. Розглянемо переваги та недоліки даної технології.

Переваги:

1. Економія трафіку. Використання AJAX дозволяє значно скоротити трафік при роботі з веб- додатком завдяки тому, що замість завантаження всієї сторінки достатньо завантажити тільки частину, що змінилася, або взагалі тільки отримати / передати набір даних в форматі JSON або XML, а потім змінити вміст сторінки за допомогою JavaScript.

2. Зменшення навантаження на сервер. При правильній реалізації, AJAX дозволяє знизити навантаження на сервер в кілька разів. Зокрема, всі сторінки сайту найчастіше генеруються за одним шаблоном, включаючи незмінні елементи («шапка», «навігаційна панель», «підвал» і т.д.) Для їх генерації потрібні звернення до різних файлів, час на обробку скриптів (а іноді і запити до БД) – все це можна опустити, якщо замінити повне завантаження сторінки генерацією та передачею лише змістовної частини.

Дизайн сторінки також зазвичай містить безліч файлів, пов’язаних з оформленням (картинки, стилі), на повторну обробку яких не треба витрачати час, використовуючи AJAX (економія на кількості HTTP-з’єднань значно вигідніше, ніж на скороченні трафіку кожного з них).

1. Прискорення реакції інтерфейсу. Оскільки завантаження частини, що

змінилася, значно швидше, то користувач бачить результат своїх дій швидше і без мерехтіння сторінки (виникає при повному перезавантаженні). Майже безмежні можливості для інтерактивної обробки. Наприклад, на багатьох сайтах при реєстрації користувач вводить ім’я, і відразу ж бачить, доступне це ім’я чи ні.

AJAX зручний для програмування чатів, адміністративних панелей та інших інструментів, які виводять мінливі з часом дані.

Недоліки:

1. Відсутність інтеграції зі стандартними інструментами браузера. Динамічно створювані сторінки не реєструються браузером в історії відвідування сторінок, тому не працює кнопка «Назад», що надає користувачам можливість повернутися до переглянутих раніше сторінок, але існують скрипти, які можуть вирішити цю проблему.

2. Інший недолік – зміна вмісту сторінки при постійному URL полягає в неможливості збереження закладки на бажаний матеріал. Проблему можна успішно вирішити за допомогою History.pushState.Динамічно завантажувати вміст не доступно пошуковикам (якщо не перевіряти запит, звичайний він або XMLHttpRequest). Пошукові машини не можуть виконувати JavaScript, тому розробники мають подбати про альтернативні способи доступу до вмісту сайту.

3. Старі методи обліку статистики сайтів стають неактуальними. Багато сервісів статистики ведуть облік переглядів нових сторінок сайту. Для сайтів, сторінки яких широко використовують AJAX, така статистика втрачає актуальність.

4. Ускладнення проекту. Перерозподіляється логіка обробки даних – відбувається виділення і часткове перенесення на сторону клієнта процесів первинного форматування даних. Це ускладнює контроль цілісності форматів і типів. Кінцевий ефект технології може бути знівельовано необґрунтованим зростанням витрат на кодування і управління проектом, а також ризиком зниження доступності сервісу для користувачів.

5. Потрібен включений JavaScript в браузері. JavaScript може бути

вимкнений з міркувань безпеки. І, звичайно ж, 26 AJAX-сторінки важкодоступні неповно функціональним браузерам, роботам і веб-архівам.

6. Низька швидкість при грубому програмуванні. Здавалося б, AJAX призначений саме для підвищення швидкості. Але, коли AJAX-запитів на одній сторінці багато і, наприклад, по кожному кліку довантажується список, AJAX-сторінка стає навіть повільніше традиційної.

Було б не правильно не згадати про таку технологію, як Flash, яка в свою чергу використовується на всіх популярних Інтернет ресурсах в вигляді різноманітних додатків, але поступово на заміну цій технології приходить HTML5.

Стек технологій Flash (раніше – Macromedia Flash) у вигляді ActionScript 3, Adobe Flex і Flash Remoting становить технологічну основу RIA (Rich Internet Applications), активно просуваються Macromedia (поглиненої компанією Adobe).

Технологія Flash підходить для самих різних додатків – від 27 комп'ютерних ігор до складних інтерфейсів бізнес-додатків. В межах даної технології реалізовані потужні засоби підтримки графіки, відсутні в базових засобах Ajax (хоча дедалі зростаючу кількість можливостей реалізується в рамках стандарту HTML5 та CSS3).

1.2.2 Взаємодія з Web за допомогою Ajax

В цьому механізмі доступу з'єднуючою ланкою між сервером та сторінкою є JavaScript-об'єкт XMLHttpRequest. В різних движках та їх версіях він реалізований по різному тому потрібно використовувати спеціальну функцію, яка враховує всі можливі варіанти.

При певних діях користувача (наприклад при активізації кнопки в складі користувацького інтерфейсу) браузер генерує запит і за допомогою JavaScript-об'єкта XMLHttpRequest відправляє його на сервер. При цьому метод доступу може бути GET або POST.

Користувацький інтерфейс під час відправлення запиту і отримання відповіді не блокується і користувач може продовжувати виконувати певні дії,

результатом яких можуть бути нові запити до сервера — Ajax підтримує декілька одночасних взаємодій сторінки з сервером.

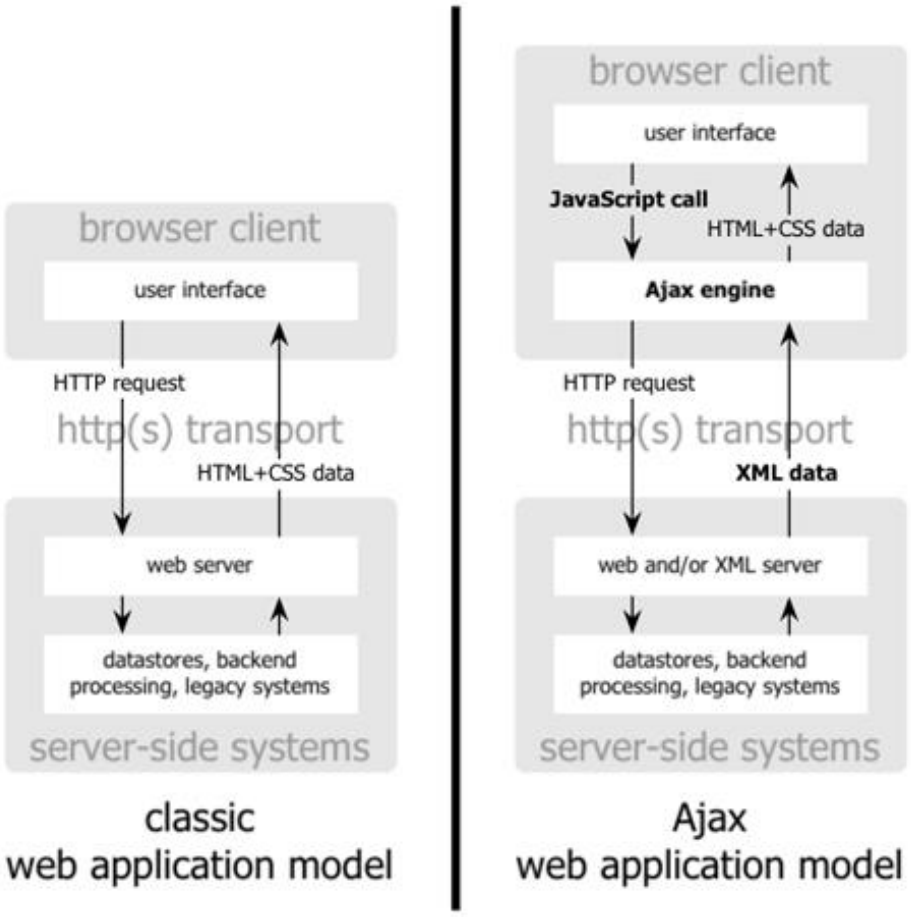

Рис. 1.2 Модель роботи Ajax

Користувацький інтерфейс виглядає і реагує на дії користувача як звичайна програма, що полегшує роботу з ним. Сервер оброблює запит і відправляє браузеру відповідь у форматі XML, JSON або подібних. При цьому не відбувається генерації усієї сторінки (як у класичному механізмі доступу), тому час обробки запиту скорочується. Це дозволяє зменшити навантаження на сервер або збільшити кількість клієнтів, що можуть працювати одночасно. Браузер, за допомогою JavaScript, обробляє отриману відповідь і модифікує сторінку без перезавантаження за допомогою DHTML (рис. 1.2).

Переваги цього механізму доступу — сторінка модифікується без повного перезавантаження, збільшується швидкість роботи з Web-програмою,

зменшується трафік між сервером та клієнтом, метод роботи користувача з web-програмою є зручним.

Недоліки методу:

- важкість у розробці та налагодженні через використання мови сценаріїв JavaScript, що має специфічне застосування тому вона мало пристосована до розробки багатих web-програм.
- зміст сторінок, згенерованих за допомогою Ajax, не індексується пошуковими системами і сторінку не можна зберегти за допомогою браузера, збережеться лише початкова сторінка та сценарії JavaScript.
- на сторінку, згенеровану за допомогою Ajax, не можна поставити пряме посилання — при модифікації сторінка не змінює адреси.

Для подолання вказаних недоліків потрібно:

1. Обмежети використання мови сценаріїв JavaScript і використати технологію Java Апплетів.

2. Повністю відмовитись від використання JavaScript не можна, але якщо перенести більшу частину функціональних можливостей з сценарію JavaScript до Java Апплету і використовувати JavaScript лише для зв’язку HTML сторінки з Апплетом то складність розробки та налагодження такої системи буде на порядок нижча.

3. Створювати окремі статичні сторінки, що матимуть той самий вміст, що і динамічні сторінки, але їх зможуть прочитати та обробити пошукові системи а також переглянути ті користувачі, що використовують застарілі браузери або браузери із відключеними або заблокованими додатковими можливостями (JavaScript, Java, Flash і т.п.). Також ці статичні сторінки користувач може зберегти на свій комп’ютер для перегляду оффлайн або редагування за допомогою HTML-редакторів.

4. Створити спеціальний елемент користувацького інтерфейсу — «посилання на цю сторінку», що міститиме спеціально сформоване посилання, перейшовши за яким відкривається сторінка з таким самим вмістом, як і

згенерована динамічно. Це потребує модифікації серверної частини (в більшості випадків ця модифікація є незначною), але подолання цього недоліку є дуже важливим для комфортної роботи з Web-сторіками, що побудовані динамічно.

Другий та третій пункти вже доволі широко використовується на сайтах, побудованих за допомогою концепції Web 2.0. Спосіб, вказаний в першому пункті ще мало вивчений, тому майже не зустрічається на сайтах. Реалізація цього способу призведе до створення базового набору засобів, за допомогою якого розробники web-програм зможуть більш ефективно та з меншими витратами часу створювати web-програми.

1.2.3 Фреймворк Bootstrap

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript [7]. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap (початкова назва — Twitter Blueprint) був розроблений Марком Отто та Джейкобом Торнтоном як фреймворк для забезпечення однаковості внутрішніх інструментів Twitter. До появи Bootstrap у розробці інтерфейсу застосовувалися різні бібліотеки, що призводило до появи суперечностей та ускладнювало супровід.

Через кілька місяців до розробки рішення долучилося багато розробників компанії Twitter. Проект було перейменовано з Twitter Blueprint на Bootstrap. Реліз із відкритим сирцевим кодом вийшов 19 серпня 2011 року. Нині проект підтримується невеликою групою розробників на чолі з Марком Отто та Джейкобом Торнтоном, а також широкою спільнотою прихильників.

Bootstrap сумісний з останніми версіями браузерів Google Chrome, Firefox, Internet Explorer, Opera і Safari (деякі з цих браузерів підтримуються не на всіх платформах).

Структура і функції

Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Основні інструменти Bootstrap:

- Сітки (grid) — наперед задані, готові до використання колонки
- Шаблони (template) — фіксовані чи адаптивні шаблони сторінок
- Типографіка (typography) — опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо
- Мультимедіа (media) — засоби управління зображеннями та відео
- Таблиці (table) — засоби оформлення таблиць, які зокрема забезпечують сортування
- Форми (form) — класи для оформлення як форм, так і деяких подій
- Навігація (nav, navbar) — класи для оформлення вкладок, сторінок, меню і панелей навігації
- Сповіщення (alert) — класи для оформлення діалогових вікон, підказок і спливаючих вікон
- Іконочний шрифт (icon font) — набір іконок у вигляді шрифту, складається майже з 500 компонентів.

Bootstrap.js

Окрім стилів, фреймворк містить також функціональні компоненти, які побудовані на js з використанням jQuery і містять такі плагіни:

- Transitions — плавні зміни, плагін використовується для налаштування останніх компонентів фреймворку.
- Modal — модальні вікна, як спливні, так і вбудовані в сторінку.
- Dropdown — випадні списки, побудовані без тегу select.

- Scrollspy — плагін, що автоматично змінює активний пункт у меню залежно від позиції прокручування сторінки.
- Tab — вкладки, використовується зазвичай для стилізованої навігації.
- Tooltip — спливні підказки, текстові елементи, які з'являються поряд із вказаним об'єктом після наведення курсору.
- Popover — аналог спливних підказок, але з більшими можливостями. У підказку можна додавати заголовок, до того ж блок з'являється після кліку на об'єкті.
- Alert — інформаційні повідомлення, які створюються класом .alert, але з можливістю закриття.
- Button — плагін для керуваннями станами кнопок. Завдяки методам плагіну можна змінювати стан і тип кнопки, а також створювати елементи, які поводяться як checkbox або radio button, але при цьому є звичайними блочними елементами.
- Collapse — згортання блочних елементів.
- Carousel — мультимедійна галерея зображень.
- Affix — плагін, що «приліплює» меню до одного з країв екрану під час прокручування сторінки.

ВИСНОВОК ДО РОЗДІЛУ 1

В першому розділі розглянуті технології та підходи до розробки плагінів, проаналізовані підходи до побудови користувацьких інтерфейсів веб-додатків.

Визначено, що мова гіпертекстової розмітки разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

Переваги використання технології AJAX при розробці розширення — сторінка модифікується без повного перезавантаження, збільшується швидкість роботи з веб-програмою та зменшується трафік між сервером та клієнтом.

Для встановлення динамічності плагіну доцільно використовувати фреймворк Bootstrap, що надає шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу.

РОЗДІЛ 2
АРХІТЕКТУРА ТА СТРУКТУРНІ ФАЙЛИ ПЛАГІНУ

2.1 Архітектура плагіну

Плагін, як правило, складається зі стислих пакетів HTML, CSS, JavaScript коду, зображень та інших файлів, які використовуються в веб-платформі та налаштовують роботу з браузером. Розширення створюються з використанням веб-технологій і можуть використовувати ті ж API-інтерфейси, які браузер надає для відкритої мережі.

Плагіни мають широкий спектр функціональних можливостей. Вони можуть змінювати веб-контент, який користувачі бачать, з яким взаємодіють або розширюють, а також змінюють поведінку самого браузера.

Розробка плагіну дозволяє розширити функціональні можливості веб-браузера без детального розгляду його власного коду. Для розробки плагіну використовуються технології HTML, CSS, Java Script, Ajax та бібліотека JQuery [8]. За допомогою даних технологій розробляються структурні файли, що містять код для реалізації плагіну.

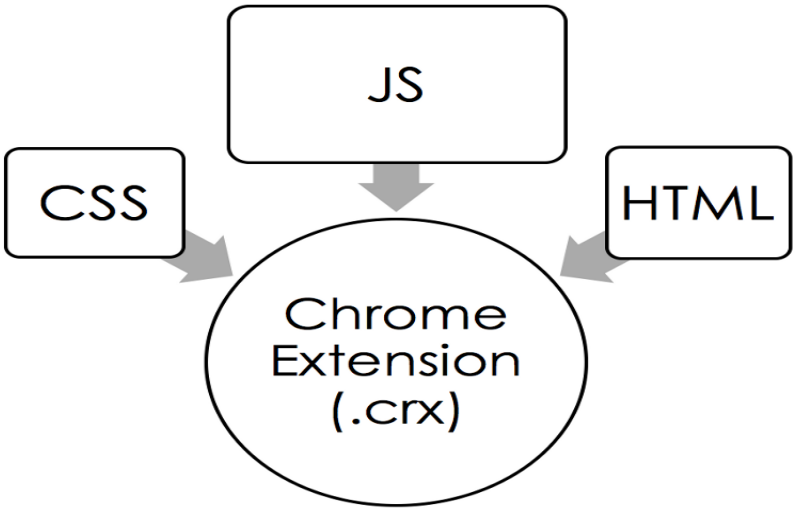

Рис. 2.1 Архітектура розширення

Залежно від визначених потреб та особливостей реалізації, загальна структура плагіну не змінюється.

Архітектура розширення буде залежати від його функціональності, але практично всі розширення включають основні компоненти:

- Manifest
- Background Script
- User Interface Elements
- Content Script
- Options Page

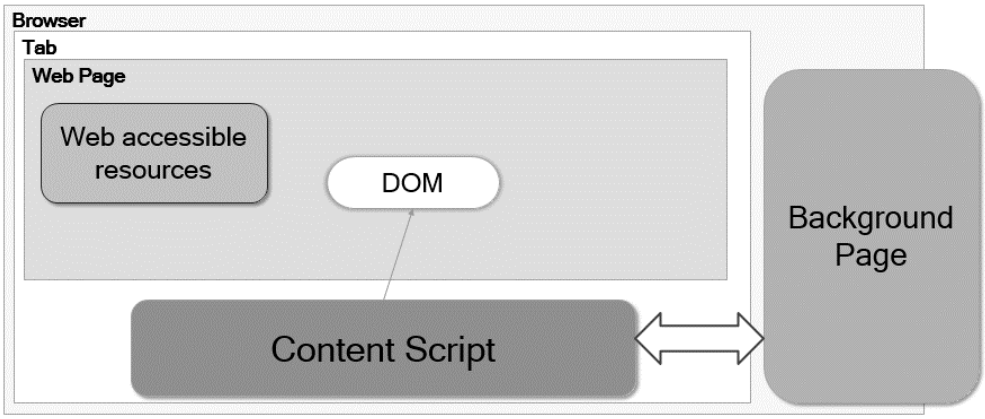

Рис. 2.2 Архітектура стандартного розширення

2.2 Розгляд структурних файлів плагіну

2.2.1 Manifest.json

Manifest.json – головний файл, що містить інформацію про доступи, які потрібні для коректної роботи плагіну, інформацію про підключені файли, метадані: версію, автора, опис та налаштування безпеки плагіну.

Основні скрипти, що включає файл (рис.2.2):

- Manifest_version – версія маніфест файлу.
- Description – опис плагіну.
- Permission – масив з назвами доступів, які необхідні для коректної роботи плагіну.

- Content_scripts – масив файлів, що підключені як контент скрипти.
- Background – опис файлів, які будуть виконуватись фоновому режимі.
- Browser_action – опис файлів, які виконуватимуться при натиску на іконку в панелі інструментів.
- Icons – іконка за стандартним розміром 28, 48 або 128.

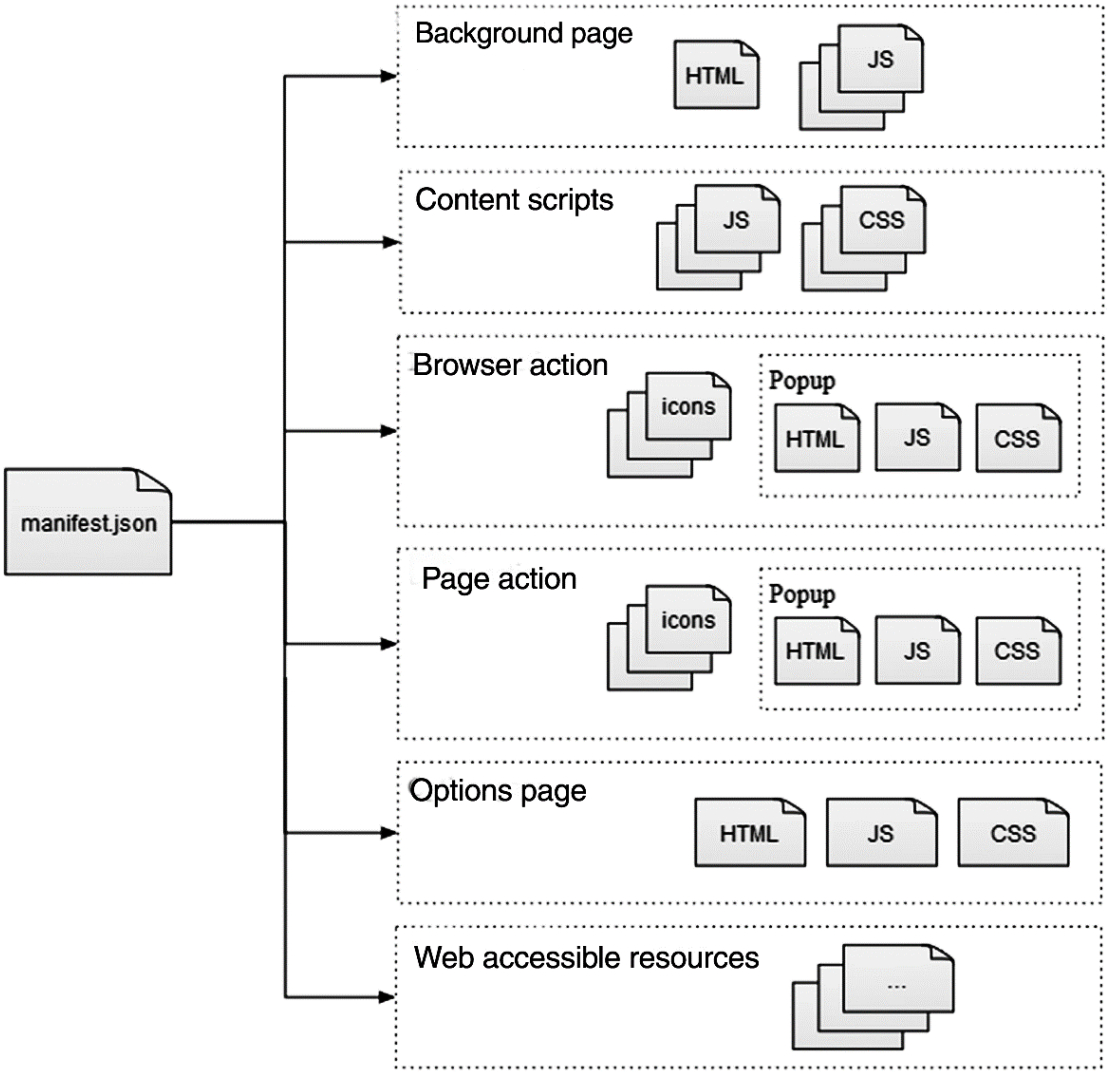

Рис.2.2 Зв'язок файлу manifest.json з іншими файлами

Крім уже перерахованих в маніфесті посилань, розширення може також включати додаткові сторінки і допоміжні файли.

Маніфест надає інформацію про програму (таку як ім'я, авторство, іконку і опис) в форматі JSON-файлу. Мета маніфеста - встановити веб-додаток на екран пристрою, надаючи користувачеві більш швидкий доступ і більше можливостей.

Маніфест веб-додатки є частиною колекції веб-технологій, поряд з іншими потужними можливостями.

2.2.2 Background.js

Background.js – файл, що описує основну логіку плагінуF. Головна особливість цієї сторінки в тому, що вона активується під час запуску браузера і тримається в його оперативній пам’яті як фоновий процес, впродовж всієї сесії (рис.2.3).

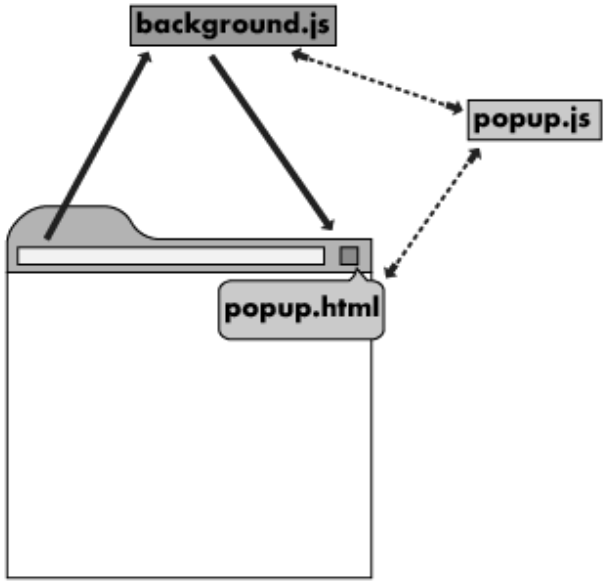

Рис.2.3 Завдання background page

Фоновий скрипт можна підключити, безпосередньо з файлу manifest.json, прописавши наступну частину коду:

```
"background": {
  "scripts": ["background.js"] }
```

Щоб оптимізувати використання ресурсів, в 2012 році була розроблена концепція івент сторінок (Event Pages).

Головна відмінність такого підходу - замість безперервної роботи у фоновому режимі, івент сторінка запускається тільки тоді, коли потрібно – наприклад, щоб обробити визначену подію [9]. Після чого вивантажується, звільняючи пам'ять, до того моменту, поки визначена подія не спрацює повторно.

Зі сторони написання коду, різниці між цими двома підходами немає, а єдине що потрібно вказати в manifest.json файлі, це коректне значення властивості persistent. За замовчуванням це значення буде мати значення «true» для стандартних background сторінок і «false» для івент сторінок.

2.2.3 Content Script

Контент скрипти - це javascript файли або код, які виконуються не в окремому фоновому процесі (як бекграунд скрипти), а в контексті веб- сторінки.

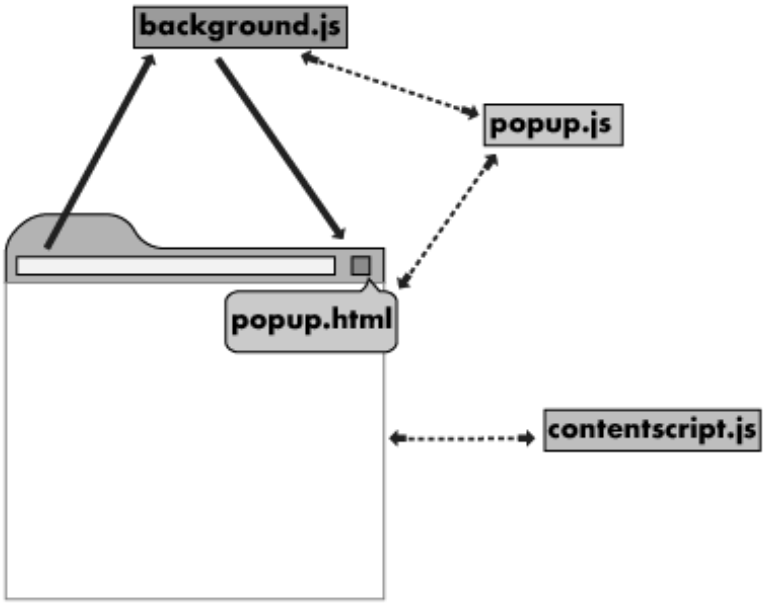

Рис. 2.4 Структура роботи Content Script

Контент скрипти використовують в обмеженому вигляді API. Але при цьому вони ізольовані і не можуть використовувати, як функції і змінні які оголошені, наприклад на бекграунд сторінці, так і змінні, функції зі скриптів, які знаходяться на веб-сторінці (рис. 2.4). Контент скрипти зчитують і змінюють DOM веб-сторінок, які відвідує користувач в браузері.

Контент скрипти можуть взаємодіяти зі своїм батьківським розширенням, обмінюючись повідомленнями і зберігаючи значення за допомогою API сховища (рис. 2.5).

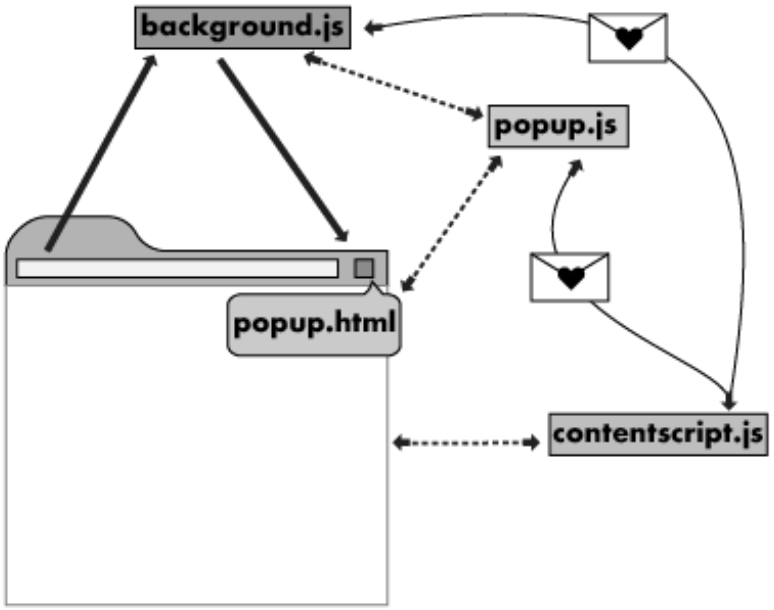

Рис. 2.5 Робота Content Script з обміном API

Тобто, повноцінний доступ до скриптів є тільки через DOM-дерево сторінки. З контент скрипта можна ініціювати івенти та змінювати DOM. Також є змога додавати script-тег в сторінку і підключати потрібні файли (рис. 2.6).

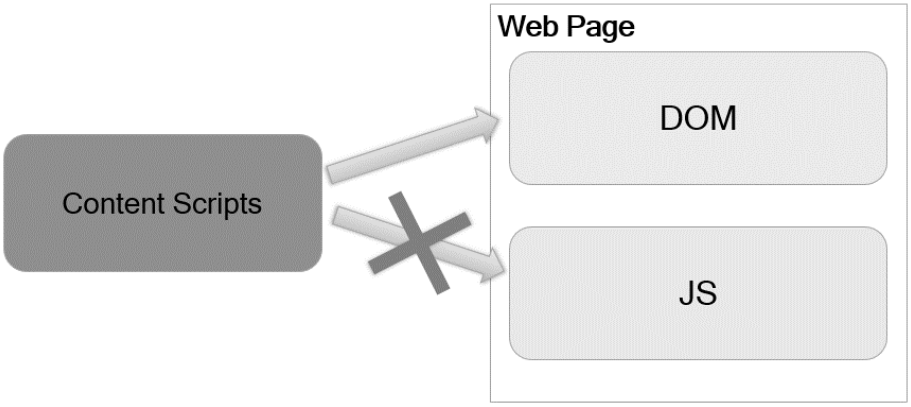

Рис. 2.6 Доступ до Content Script

2.2.4 User Interface Elements

Index.html – файл, в якому описана загальна структура розмітки плагіну, яка складається з декларації типу документу, шапки та тіла документа. Елементи та атрибути, які реалізують зовнішній вигляд, зображені на рис.2.6.

Index.html активується під час натискання на іконку плагіну та відображається в відкритому вікні.

Main.css – файл включає таблиці стилів для оформлення елементів, що входять до структури документу index.html (рис 2.7). Для зв'язання документу з цим файлом застосовується тег <link>. Файл main.css не зберігає ніяких інших даних, крім синтаксису css. Таким способом реалізується повне розділення коду і його оформлення.

Браузер встановлює за замовчуванням власні значення стилів для HTML-елементів. За допомогою файлу reset.css можна зрівняти цю різницю, тобто обнулити початкові значення стилів для забезпечення кросбраузерності стилів.

Основні файли, які використані для розробки плагіну :

- Manifest.json
- Background.js
- Index.html
- Main.css
- Popup.js

Додаткові файли, які використані для розробки плагіну:

- Reset.css
- Jquery.js
- bootstrap-datepicker.min.js
- bootstrap-datepicker.min.css

2.2.5 Content Security Policy

Для Chrome розширень, діє так зване Content Security Policy - це набір строгих правил, які необхідні для того, щоб зробити розширення безпечнішим і контролювати контент, може бути завантажений і виконаний в розширенні [10].

За замовчуванням, якщо використовувати маніфест 2 версії, плагін матиме такі обмеження:

- заборонено використовувати eval і схожі функції,
- inline java script виконуватися не буде,
- можливість завантажувати тільки локальні скрипти і ресурси.

Браузери, які не підтримують CSP, все ще можуть працювати з серверами, які підтримують CSP, і навпаки: браузери, в яких підтримка CSP відсутня, будуть її ігнорувати, продовжуючи роботу відповідно до стандартних правил обмеження домену для завантаження контенту. У разі, якщо сайт не надає CSP-заголовки, браузери, в свою чергу, будуть використовувати стандартні правила обмеження домену.

Налаштування CSP включає в себе додавання на сторінку HTTP-заголовка Content-Security-Policy і його налаштування відповідно до списку довірених джерел, з яких користувач може отримувати контент.

Наприклад, сторінка, на якій відбувається завантаження і відображення зображень може дозволити їх отримання з будь-яких джерел, але обмежити відправку даних форми конкретною адресою.

При вірному налаштуванні, Content Security Policy допоможе захистити сторінку від атак між сайтового скриптингу.

2.2.6 Формат передачі даних JSON

JavaScript Object Notation (JSON) - це стандартний текстовий формат для подання структурованих даних на основі синтаксису об'єктів JavaScript. Він, зазвичай, використовується для передачі даних в веб-додатках (наприклад, для

відправки даних з сервера клієнту, щоб їх можна було відобразити на веб-сторінці або навпаки).

JSON використовує розширення .json. Якщо файл задається в інших файлових форматах, наприклад як .html, він з'являється в лапках як JSON-рядок, або може бути об'єктом, призначеним на змінну. Такий формат легко передавати між сервером і клієнтською частиною, ну або браузером (рис. 2.8).

JSON є альтернативою формату XML і вимагає менше форматування контенту. Пари ключ-значення розділені двокрапкою, як наприклад "key": "value". Кожна пара значень розділена двокрапкою, таким чином середина JSON виглядає так: "key": "value", "key": "value", "key": "value" [11].


```
[{"r030":986,"txt":"Бразильський реал","rate":7.529564,"cc":"BRL","exchangeda
{"r030":643,"txt":"Російський рубль","rate":0.42405,"cc":"RUB","exchangedate"
{"r030":933,"txt":"Білоруський рубль","rate":13.063640000000001,"cc":"BYN","e
динар","rate":0.228415,"cc":"DZD","exchangedate":"29.05.2018"},{"r030":944,"t
{"r030":36,"txt":"Австралійський долар","rate":19.793523,"cc":"AUD","exchange
{"r030":51,"txt":"Вірменський драм","rate":0.05419939,"cc":"AMD","exchangedat
{"r030":124,"txt":"Канадський долар","rate":20.183478,"cc":"CAD","exchangedat
{"r030":191,"txt":"Куна","rate":4.130915,"cc":"HRK","exchangedate":"29.05.201
{"r030":208,"txt":"Данська крона","rate":4.098749000000001,"cc":"DKK","exchan
долар","rate":3.333318,"cc":"HKD","exchangedate":"29.05.2018"},{"r030":348,"t
рупія","rate":0.3857149,"cc":"INR","exchangedate":"29.05.2018"},{"r030":360,"
ріал","rate":0.0006213,"cc":"IRR","exchangedate":"29.05.2018"},{"r030":368,"t
ізраїльський шекель","rate":7.347306999999999,"cc":"ILS","exchangedate":"29.0
{"r030":392,"txt":"Єна","rate":0.2390348,"cc":"JPY","exchangedate":"29.05.201
{"r030":410,"txt":"Вона","rate":0.024238399999999997,"cc":"KRW","exchangedate
{"r030":422,"txt":"Ліванський фунт","rate":0.0174057,"cc":"LBP","exchangedate
рингґіт","rate":6.694363,"cc":"MYR","exchangedate":"29.05.2018"},{"r030":484,
```


Рис. 2.6 Дані в форматі JSON

Переваги використання формату JSON для передачі даних:

- Швидкість передачі даних, що знижує на 28% загальний трафік,
- Підтримка всіх основних типів даних, які можуть легко серіалізовані (перетворені) з будь-якої мови програмування
- Мінімальна семантичне уявлення, що дозволяє в рази знизити розмір відповіді сервера, а отже, і навантаження на нього.

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі розглянуті архітектура та структурні файли плагіну.

Плагін, як правило, складається зі стислих пакетів HTML, CSS, JavaScript коду, зображень та інших файлів, які використовуються в веб-платформі та налаштовують роботу з браузером.

Розширення створюються з використанням веб-технологій і можуть використовувати ті ж API-інтерфейси, які браузер надає для відкритої мережі.

Архітектура плагіну буде залежати від його функціональності, але практично всі плагіни включають основні компоненти:

- Manifest
- Background Script
- User Interface Elements
- Content Script
- Options Page

Для передачі даних використовують формат даних JSON , який має високу швидкість передачі даних, що знижує на 28% загальний трафік користувача.

3 РОЗДІЛ
РОЗРОБКА ФУНКЦІОНАЛЬНОГО ПЛАГІНУ

3.1 Розгляд структурних файлів плагіну

Структура плагіну складається з HTML розмітки, до якої застосовуються CSS таблиці стилів та описується динаміка в поведінці за допомогою логіки JavaScript. Також для коректної роботи плагіну, підключаються бібліотеки, що надають додаткові можливості розробнику.

Для структуризації коду, всі компоненти плагіну виокремлені в папки з відповідними назвами, що включають основні файли (рис.3.1).

Manifest.json

Manifest.json – головний файл, що містить інформацію про доступи, які потрібні для коректної роботи плагіну, інформацію про підключені файли, версію, автора, опис та налаштування безпеки плагіну (рис.3.2).

Основні скрипти, що включає файл (дод. А):

- Manifest_version – версія маніфест файлу.
- Name – назва плагіну.
- Author – вказує на розробника плагіну.
- Version – поточна версія продукту.
- Web accessible resources – надає доступ до медіа-файлів, які описані в цьому скрипті.
- Description – опис плагіну.
- Permission – масив з назвами доступів, які необхідні для коректної роботи плагіну.
- Content_scripts – масив файлів, що підключені як контент скрипти.

- Background – опис файлів, які будуть виконуватись фоновому режимі.
- Browser_action – опис файлів, які виконуватимуться при натиску на іконку в панелі інструментів.
- Icons – іконка за стандартним розміром 28, 48 або 128.

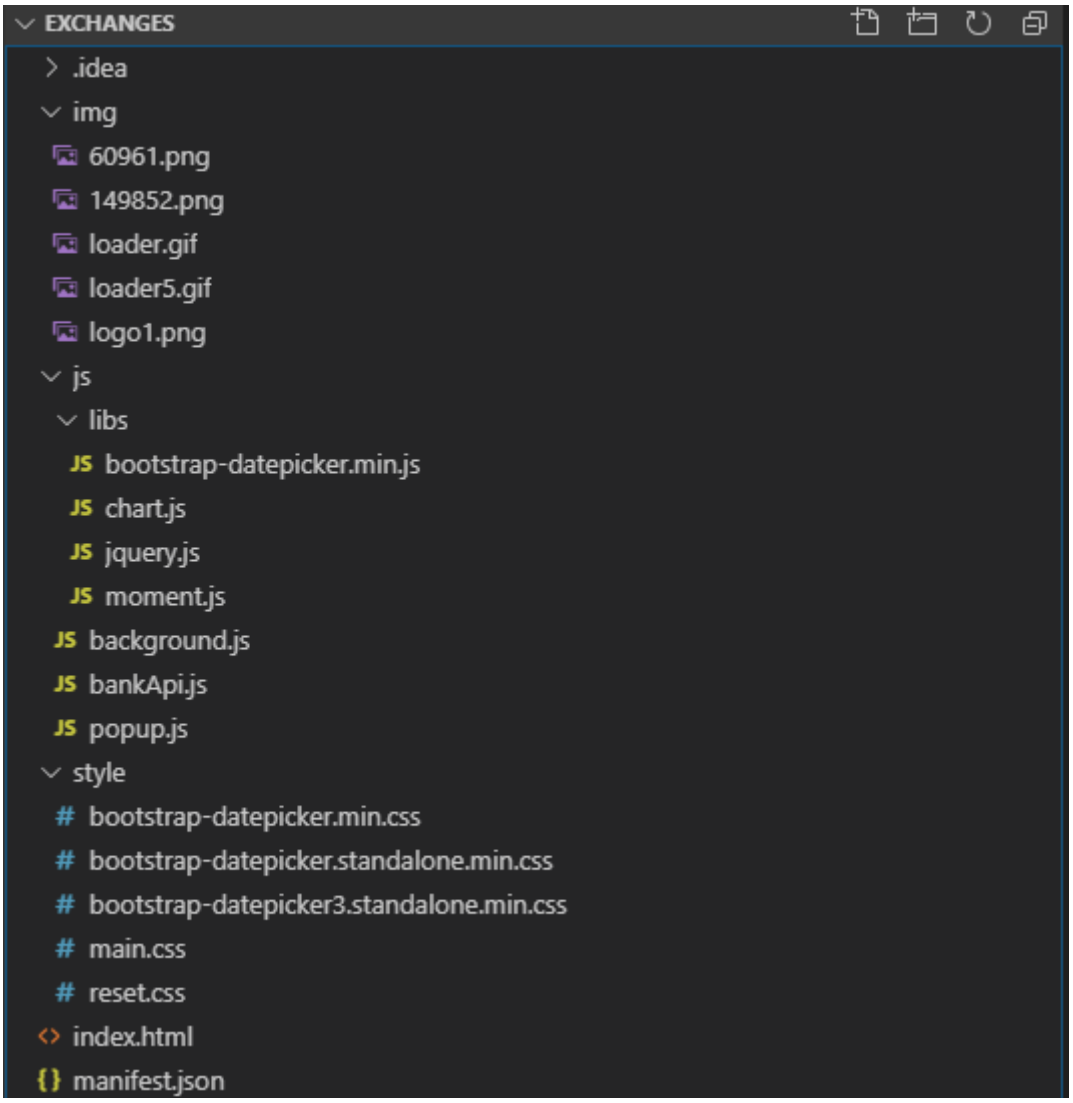

Рис. 3.1 Структура файлів плагіну

Маніфест надає інформацію про програму (таку як ім'я, авторство, іконку і опис) в форматі JSON-файлу.

Мета маніфеста - встановити веб-додаток на екран пристрою, надаючи користувачеві більш швидкий доступ і додаткові функціональні можливості.

Маніфест веб-додатки є частиною колекції веб-технологій, поряд з іншими потужними можливостями.

manifest.json

```
{
    "manifest_version": 2,
    "name": "Exchange Rates",
    "author": "Marina Grib",
    "description": "Exchange Rates and Converter",
    "version": "2.0",
    "icons": {
        "128": "img/logo1.png"
    },
    "background": {
        "scripts": ["js/background.js"],
        "persistent" : false
    },
    "browser_action": {
        "default_popup": "index.html"
    },
    "web_accessible_resources": ["img/60961.png", "img/149852.png", "img/logo.png", "loader.gif"]
}
```

Рис. 3.2 Файл manifest.json

Background.js

Background.js – файл, що описує основну логіку плагіну (дод. Б). Головна особливість цієї сторінки в тому, що вона активується під час запуску браузера і тримається в його оперативній пам'яті як фоновий процес, впродовж всієї сесії (рис.3.3).

Рис.3.3 Завдання background page

Використовуючи комбінацію клавіш Shift+Esc, можна переглянути список завдань, які виконуються в браузері. Вони займають частину пам'яті та системних ресурсів, але не виконують жодних функцій, так як плагін ще не запущений (дод. А).

Саме в цьому списку відображається background-сторінка плагіну (рис.3.4).

Рис 3.4 Файл Background.js

Index.html

Index.html – файл, в якому описана загальна структура розмітки плагіну, яка складається з декларації типу документу, шапки та тіла документа. Елементи та атрибути, які реалізують зовнішній вигляд, зображені на рис.3.5.

Index.html активується під час натискання на іконку плагіну та відображається в відкритому вікні (дод. А).

За допомогою розмітки виділено чотири основні вкладки та їхні компоненти:

- Currency,
- Converter,
- Choose date,
- Chart.

Рис.3.5 Файл Index.html

Файли .css

Файли включають таблиці стилів для оформлення елементів, що входять до структури документу index.html. Для зв’язування документу з цим файломи застосовується тег <link>, що відповідає за прив’язку файлу стилів до основної html-структури. Файл.css не зберігає жодних інших даних, окрім синтаксису css (дод. В), що дозволяє реалізувати повне розділення коду і його оформлення.

Всі файли даного розширення об’єднанні в папку «style», що видно на рис.3.1. Папка вміщує:

- bootstrap-datepicker.min.css
- bootstrap-datepicker.standalone.min.css
- bootstrap-datepicker3.standalone.min.css
- main.css
- reset.css

Перші три файли відповідають за стилізування календаря, що вбудований у вкладку «Choose date», файл «main.css» відповідає за основне оформлення DOM-структури (рис.3.7), файл «reset.css» відміняє стандартні стилі браузера, які встановленні за замовчуванням.

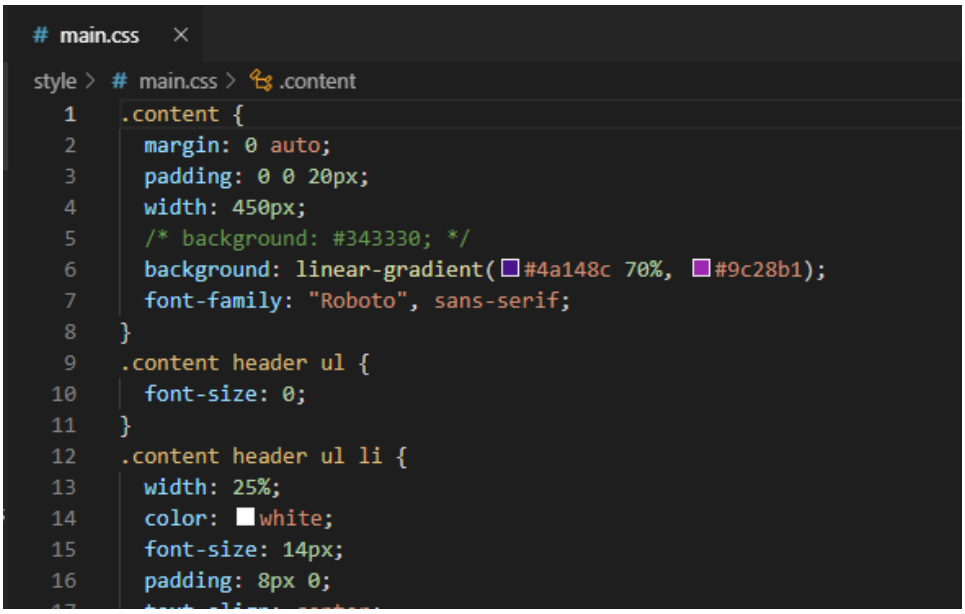

Рис.3.7 Файл «main.css»

Reset.css

Браузер встановлює за замовчуванням власні значення стилів для HTML-елементів. За допомогою файлу reset.css можна зрівняти цю різницю, тобто обнулити початкові значення стилів для забезпечення кроссбраузерності стилів.

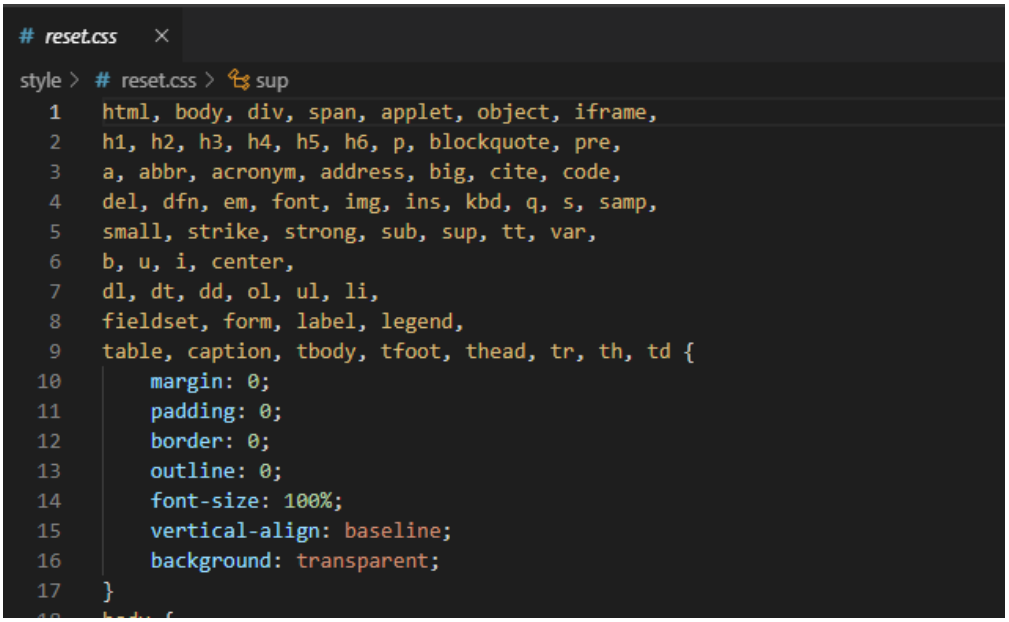

Рис. 3.8 Файл «reset.css»

Основні файли, які використані для розробки плагіну :

- Manifest.json
- Background.js
- Index.html
- Main.css
- Popup.js

Додаткові файли, які використані для розробки плагіну:

- BankApi.js
- Chart.js
- Reset.css
- Jquery.js
- Moment.js
- bootstrap-datepicker.min.css
- bootstrap-datepicker.standalone.min.css
- bootstrap-datepicker3.standalone.min.css

3.2 Навігація плагіну

Базуючись на проведеному аналізі існуючих плагінів по курсу валют, були враховані їхні переваги. Тому, при розробці навігаційної панелі реалізовано горизонтальне меню, що має чотири вкладки: Currency, Converter, Choose date та Chart перехід між якими здійснюється по кліку.

Це реалізовано наступними кроками:

1. В файлі index.html, створюється список, що має чотири пункти, які відповідають назвам вкладок. Список вкладений в контейнер <header> та має заданий клас <class=’active’>.

2. За допомогою файлу reset.css, стилі браузера за замовчуванням, відносно елементів документу, будуть обнулені.

Задаються стилі для оформлення навігації. При наведені курсору на вкладку та для виділення активної вкладки, застосовуються наступні стилі:

```
.content header ul li {
  width: 25%;
  color: white;
  font-size: 14px;
  padding: 8px 0;
  text-align: center;
  cursor: pointer;
  display: inline-block;
  border-bottom: 3px solid white;
  transition: all 0.2s ease;
}
.content header ul li:hover,
.content header ul li.active {
  border-bottom: 3px solid #9c28b1;
  background: #000;
}
```

3. За замовчуванням, при відкритті плагіну, активною буде вкладка «Currency». При переході між вкладками спрацьовує обробник подій navigator.click, який позначає виділену вкладку активною і активує контент, який прикріплений до цієї вкладки, шляхом зрівняння атрибутів data-active і data-content.

```
navigator.click(function () {
const currActive = $(this).attr("data-active");
if ($(this).hasClass("active")) return false;

navigator.removeClass("active");
$(this).addClass("active");

allTabsCont.removeClass("active");
```

```
$(".item-tab[data-content=" + currActive + "]").addClass("active");
});
```

3.3 Вкладка «Currency»

3.3.1 Отримання даних через API

Для реалізації списку валют необхідно отримати дані про курси валют. Тому, надсилається крос доменний запит до API Національного банку України, в якому запитуються всі данні по курсам відносно гривні. Сервіс надає безкоштовну інформацію через відкритий API.

Для запиту використовується URL - посилання на адресу ресурсу, в якому вказується, що дані повертаються в форматі json (рис.3.9).

Після повернення відповіді, коли не виникає помилок в надсиланні даних зі сторони сервера, спрацьовує подія success і всі дані записуються в рядок. Реалізація виконується за допомогою коду:

```
(() => {
   const mainUrl = "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange
?";
   const bankApi = {
      currency: (callback, errorHandler = () => { }) => {
         $.ajax({
            url: `${mainUrl}json`,
            type: "GET",
            dataType: "json",
            success: callback,
            error: errorHandler
         });
      },
```

Курс гривні відносно інших валют встановлюється Національним банком України до 18.00 години поточного дня на наступний день.

[{"r030":986,"txt":"Бразильський реал","rate":7.529564,"cc":"BRL","exchangeda
{"r030":643,"txt":"Російський рубль","rate":0.42405,"cc":"RUB","exchangedate"
{"r030":933,"txt":"Білоруський рубль","rate":13.063640000000001,"cc":"BYN","e
динар","rate":0.228415,"cc":"DZD","exchangedate":"29.05.2018"},{"r030":944,"t
{"r030":36,"txt":"Австралійський долар","rate":19.793523,"cc":"AUD","exchange
{"r030":51,"txt":"Вірменський драм","rate":0.05419939,"cc":"AMD","exchangedat
{"r030":124,"txt":"Канадський долар","rate":20.183478,"cc":"CAD","exchangedat
{"r030":191,"txt":"Куна","rate":4.130915,"cc":"HRK","exchangedate":"29.05.201
{"r030":208,"txt":"Данська крона","rate":4.098749000000001,"cc":"DKK","exchan
долар","rate":3.333318,"cc":"HKD","exchangedate":"29.05.2018"},{"r030":348,"t
рупія","rate":0.3857149,"cc":"INR","exchangedate":"29.05.2018"},{"r030":360,"
ріал","rate":0.0006213,"cc":"IRR","exchangedate":"29.05.2018"},{"r030":368,"t
ізраїльський шекель","rate":7.347306999999999,"cc":"ILS","exchangedate":"29.0
{"r030":392,"txt":"Єна","rate":0.2390348,"cc":"JPY","exchangedate":"29.05.201
{"r030":410,"txt":"Вона","rate":0.024238399999999997,"cc":"KRW","exchangedate
{"r030":422,"txt":"Ліванський фунт","rate":0.0174057,"cc":"LBP","exchangedate
рингґіт","rate":6.694363,"cc":"MYR","exchangedate":"29.05.2018"},{"r030":484,

Рис. 3.9 Дані в форматі json

3.3.2 Створення html-таблиці з вихідними даними

Після виконання запиту, результат зберігається в змінній dataCurrency. Для виводу всіх даних в вигляді таблиці застосовується функція $.each(result, function (k, v)), яка перебирає кожен елемент масиву відповідно до їх індексу і значення (k,v).

Формується html-таблиця (рис.3.10), кожен рядок якої має три значення : назва валюти(v.txt), абревіатура(v.txt) та курс(v.rate), які задані для кожної наступної валюти.

Фрагмент коду з файлу background.js, що відповідає за формування таблиці:

```
dataCurrency = result;
    result.forEach((v, k) => {
      stringCurrency += `<div class="currency-row">
        <span class="name-curr">${v.txt}(${v.cc})</span>
        <span>${v.rate.toFixed(2)} UAH</span>
      </div>`; });
```

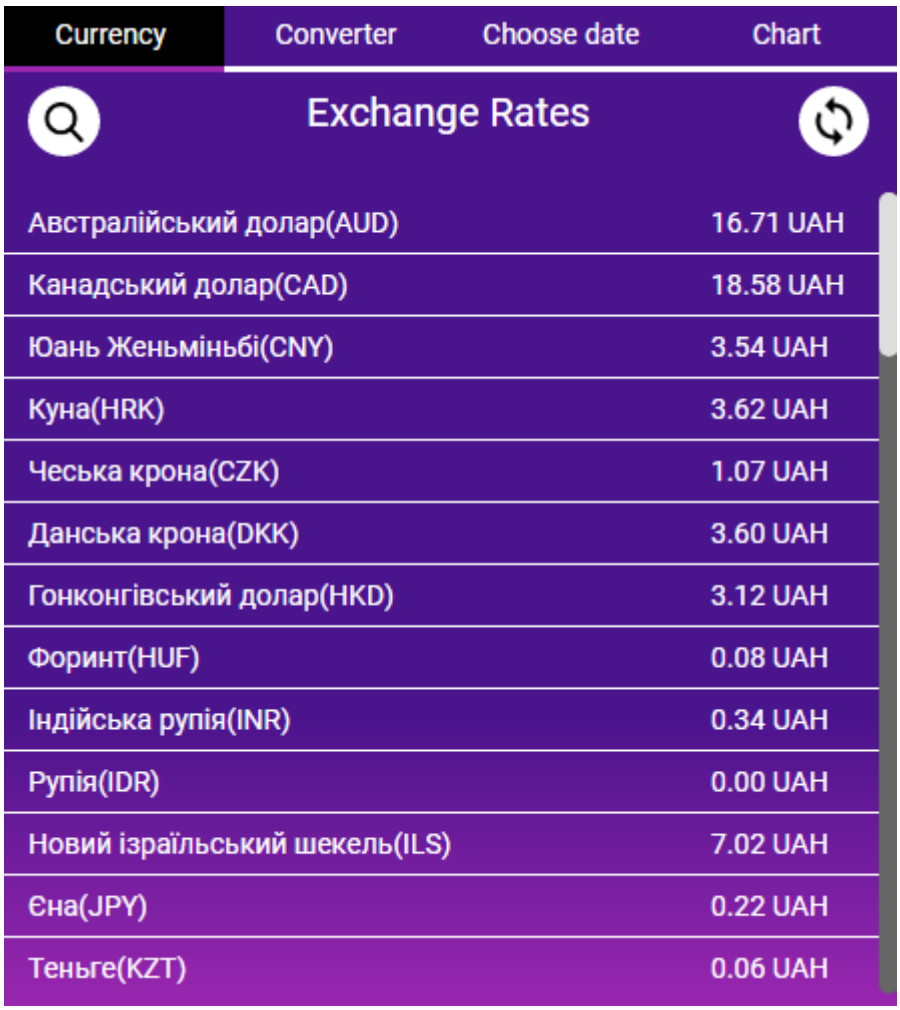

Рис.3.10 Html-таблиця з вихідними даними

3.3.3 Пошук по таблиці

В вкладці Currency організований пошук по елементам html-таблиці, який здійснюється при введені в текстове поле назви шуканої валюти. Пошук реалізований порівнянням введеного в поле тексту з назвою валюти з таблиці.

Введений в поле текст та дані з таблиці зводяться до нижнього регістру та порівнюються лише після введення двох символів в поле пошуку, що вважається раціональним. Метод indexOf() повертає значення індексу елемента, який задовольняє умову, після чого елемент списку виводиться на екран (рис.3.11).

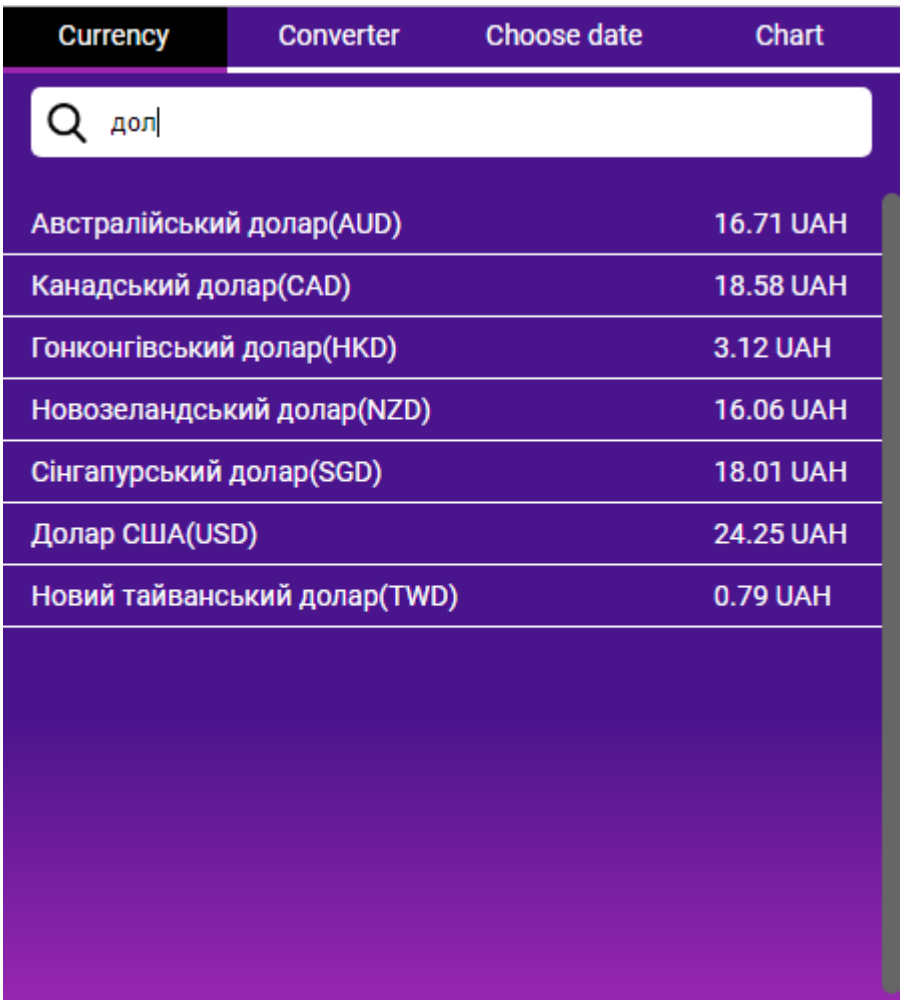

Рис.3.11 Застосування пошуку по таблиці

3.3.4 Оновлення даних

Оновлення даних застосовується лише в вкладці Currency та відбувається після натискання на кнопку з іконкою оновити. Під час натискання спрацьовує обробник подій update.click(), який виконує дві команди,- обнуляє поле пошуку та викликає функцію stateFunction{}.

StateFunction виконує надсилання запиту для отримання даних таблиці, який описаний в пункті 3.3.1. Оновлення виконується за допомогою технології Ajax, що дозволяє без перезавантаження сторінки, надсилати запити на сервер і довантажувати необхідні дані.

Також при оновленні вкладки відтворюється анімація (рис 3.12), розроблена за допомогою класу loader та методу show(), яка після формування html-таблиці - зникає.

Рис 3.11 Зображення для gif-анімації

3.4 Вкладка «Converter»

3.4.1 Формування списку валют і додавання в select

При переході на вкладку «Converter», активується прикріплений до неї контент(рис. 3.12). Розроблені два списки, що відповідають за вибір валют для конвертування, - відповідно список валюти, з якої буде здійснюватись конвертація та список валюти, в яку конвертується введена сума.

Оскільки, по надісланому запиту, надходять валюти представлені відносно гривні, тому потрібно додати в список дані про гривню, відношення якої до інших валют рівняється 1. Це потрібно для того, щоб мати можливість здійснити конвертування з гривні в іншу валюту, чи навпаки.

Для цього використовується метод додавання елементу в початок масиву – unshift:

```
dataCurrency.unshift({
    cc: "UAH",
    exchangedate: "",
    r030: 0,
   rate: 1,
   txt: "Гривня"
```

```
});
```

Реалізовані два випадаючих списки: selectFrom та selectTo. Методом append() здійснюється вставка, раніше сформованих елементів в списки. За це відповідає наступна частина коду:

```
setSelectCurrency: currencyData => {
    const selectFrom = $(".selectFrom");
    const selectTo = $(".selectTo");
    const chooseCurrency = $(".choose-currency");
    let allCurrencies = "";
    let allCurrenciesChart = "";
    currencyData.forEach((v, k) => {
      allCurrencies += `<option value = "${v.rate}"> ${v.txt} (${v.cc})</option>`;
      k && (allCurrenciesChart += `<option value = "${v.cc}"> ${v.txt} (${v.cc})
</option>`);
    });
    selectFrom.append(allCurrencies);
    selectTo.append(allCurrencies);
    chooseCurrency.append(allCurrenciesChart);
  },
```

3.4.2 Створення валідації для поля «Amount»

Вкладка «Converter» містить текстове поле «Amount» (рис.3.12), що слугує для введення кількості одиниць для конвертування. Введений вираз може бути цілим, або дробовим числом, але не може містити букви та спец символи, для цього здійснюється валідація поля «Amount». Валідація розроблена за допомогою коду:

```
amountInput.on('input', function(){
      var replace = /[^0-9\.]/gi;
      var exeption = /^([0-9]+\.)$/;
      var valid = /^([0-9]+(\.?([0-9]+))?)$/;
```

```
    var cut = $(this).val().replace(replace, '');
    if(!valid.test(amountInput.val())){
    amountInput.val(cut);
        if(!valid.test(cut) && !exeption.test(cut) && cut.length){
        amountInput.val(amountState);
         }
 }
amountState = amountInput.val();
});
```

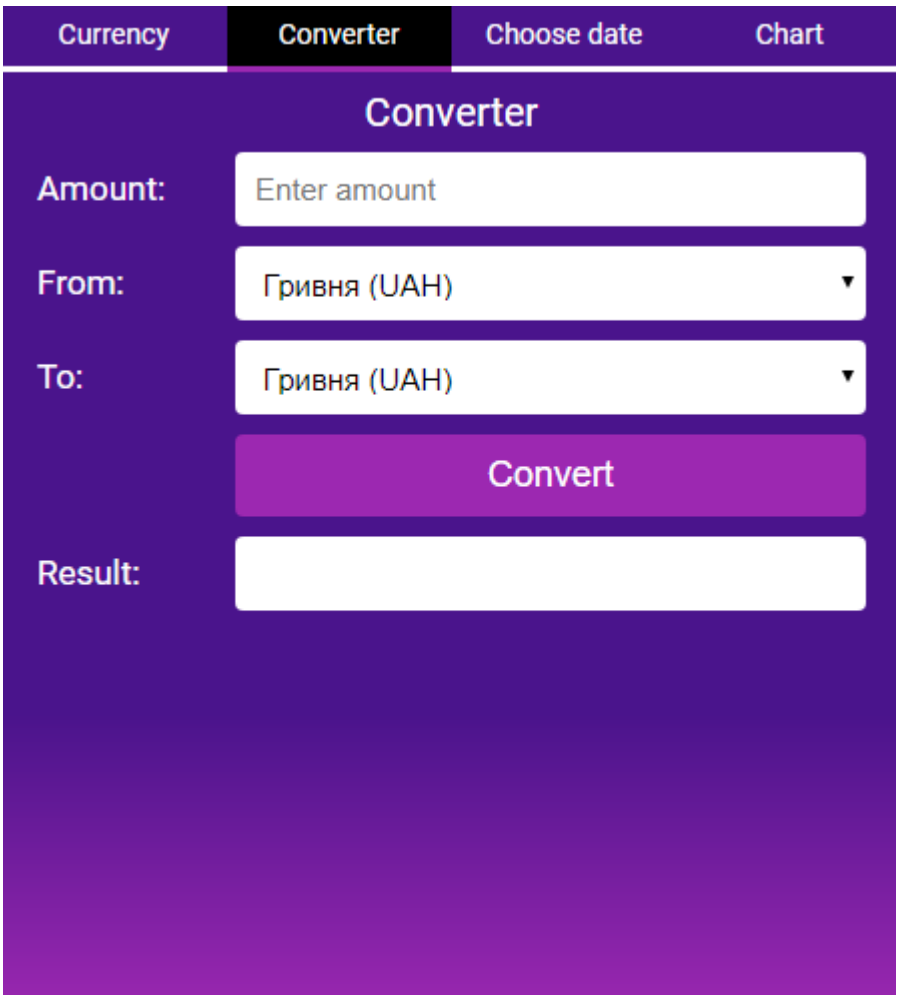

Рис.3.12 Вкладка «Converter»

3.4.3 Реалізація конвертування за формулою

Для того, щоб конвертувати валюту, потрібно задати кількість одиниць в полі «Amount» та обрати зі списків «From» і «To» відповідно, з якої та в яку валюту здійснюватиметься перевід.

Конвертування відбувається за допомогою кнопки «Convert» та обробника подій convertBtn.click(), який викликає функцію:

```
convertBtn.click(function () {
    const resultConvert = $(".resultConvert");
    const amount = amountInput.val();
    const selectFrom = $(".selectFrom").val();
    const selectTo = $(".selectTo").val();
    resultConvert.text((+amount * +selectFrom) / +selectTo);
  });
```

Функція обраховує результат за формулою ((amount * selectFrom) / selectTo) та записує його в поле «Result» (рис 3.13).

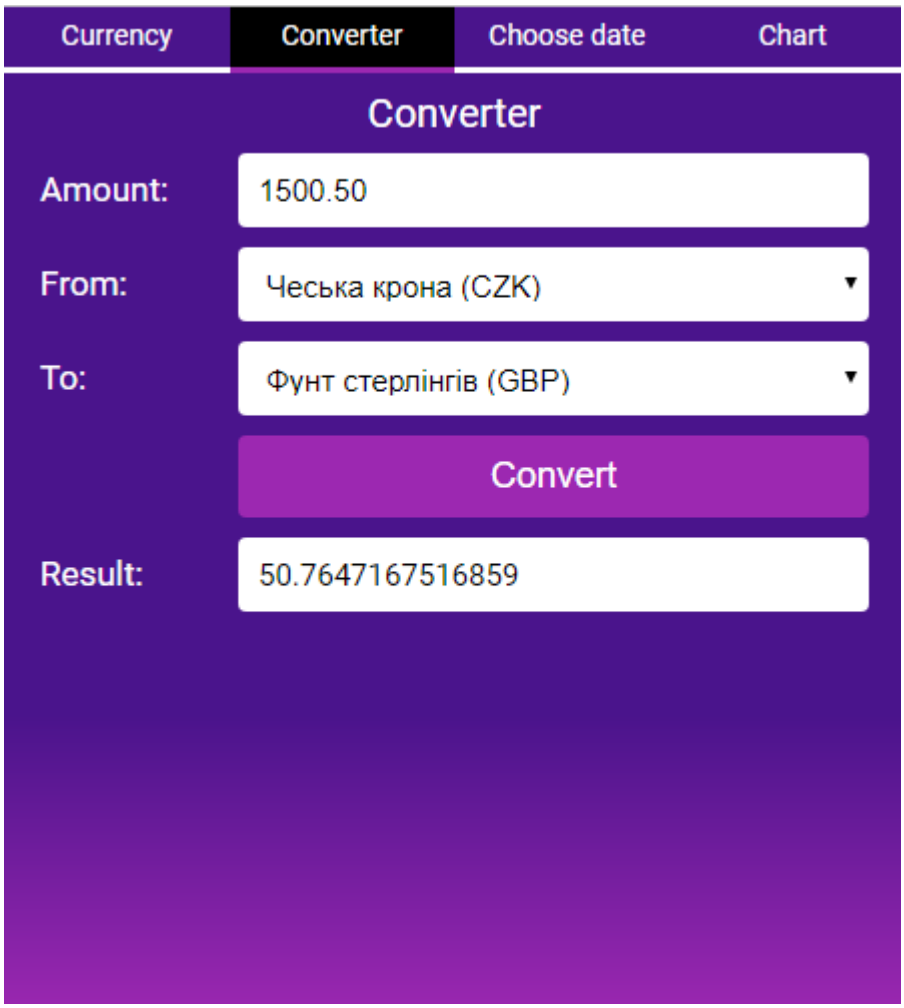

Рис.3.13 Конвертування валют

3.6 Вкладка «Choose Date»

DatePicker

Вкладка «Choose Data» розроблена для пошуку курсу валют по введеній даті. Для того, щоб користувач правильно увів формат дати для пошуку, застосовується зручний віджет у вигляді календаря – «DatePicker» (рис.3.14).

Обрати дату можна по кліку на полі «Choose Data», після чого спрацьовує обробник подій, що розгортає календар. Після вибору дати, календар згортається завдяки методу hide(). Дата представлена у форматі: 'yyyy.mm.dd', що дозволяє здійснити прямий запит до ресурсу на оновлення даних.

```
dataPicker.datepicker({
        format: 'yyyy.mm.dd',
```

```
}).on('changeDate', function(){
    dataPicker.datepicker('hide');
stateFunction.getCurrencyDate(dataPicker.val().replace(/[^0-9]/gi, ''))
});
```

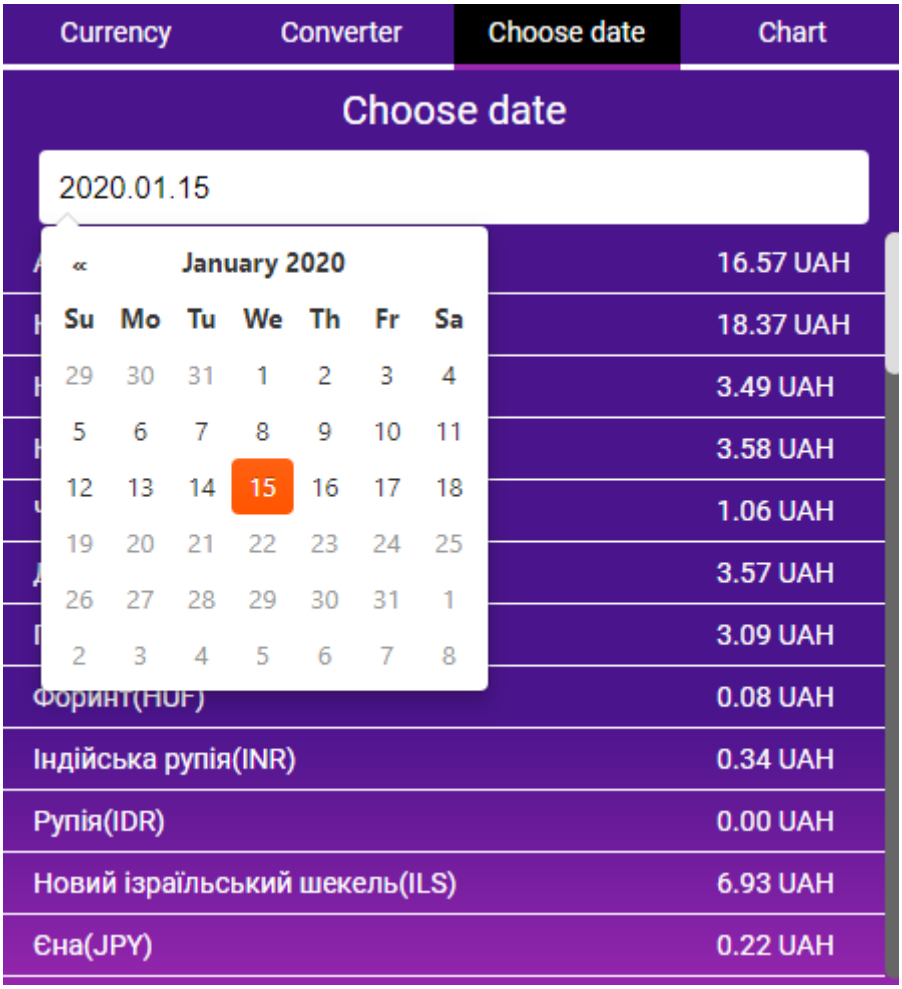

Рис.3.14 Календар «DatePicker»

Запит даних по даті

Дані по курсу валют відносно обраної дати, отримуються за допомогою методу getCurrencyDate, що надсилає запит на оновлення даних таблиці по даті:

```
getCurrencyDate: date => {
    const tableCurrency = $(".choose-data .table-currency");
    loader.show()
    window.currencyByDate(date, result => {
```

```
        let stringCurrency = "";
        result.forEach((v, k) => {
          stringCurrency +=
            `<div class="currency-row">
              <span class="name-curr">${v.txt}(${v.cc})</span>
              <span>${v.rate.toFixed(2)} UAH</span>
            </div>`;
        });
        tableCurrency.empty().append($(stringCurrency));
        loader.hide();
      });
    },
```

Після отримання оновлених даних відносно певної дати формується html-таблиця, що вказано в пункті 2.4.1.

3.7 Вкладка «Chart»

3.7.1 Визначення методів для побудови графіка

На вкладці «Chart» користувачеві надається можливість вибрати валюту з випадаючого списку та проаналізувати графік, який будується в проміжку 5 днів відповідно до вибору (рис.3.15).

При переході на вкладку з графіком спрацьовує наступна частина коду:

```
chartTab.click(async () => {
    loader.show();
    const initialCurrency = 'AUD';
    const periodByDate = await window.rateByPeriod(initialCurrency);
    const convertedData = mapData.convertToChartData(periodByDate);
```

```
    stateFunction.setChart(convertedData, initialCurrency);
    loader.hide();
  });
```

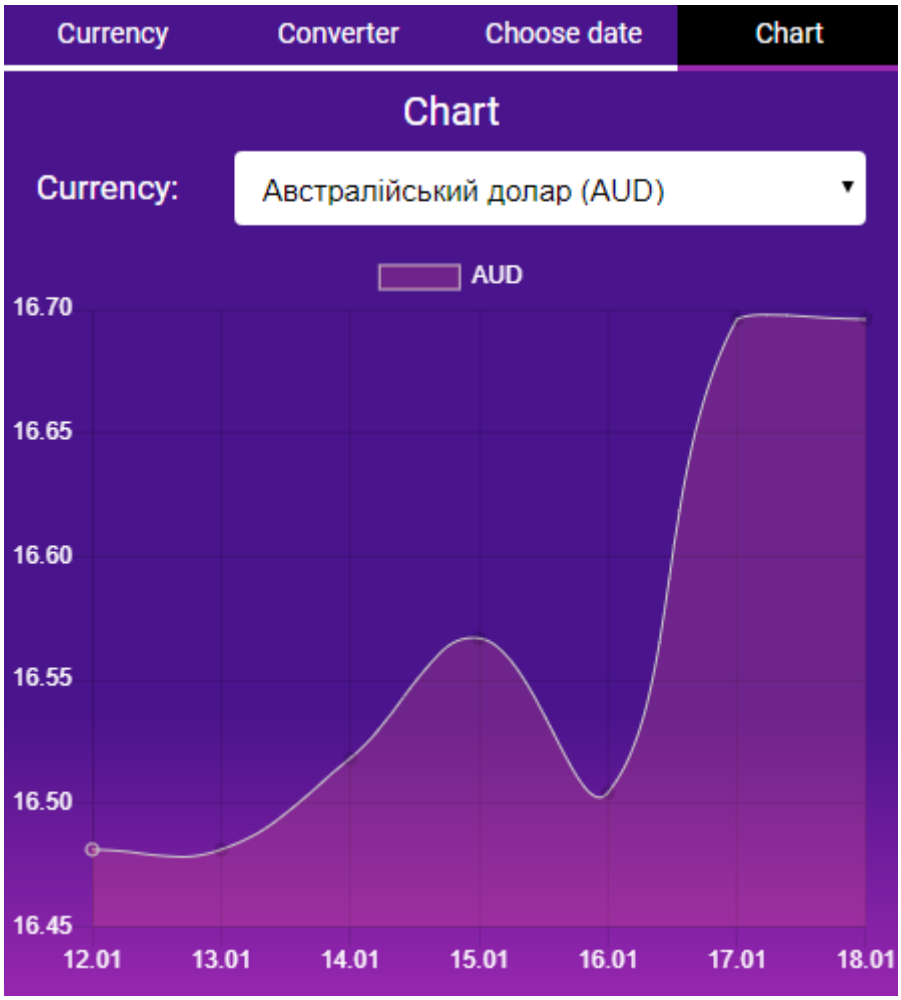

Рис.3.15 Вкладка «Chart»

3.7.2 Побудова графіку

Для побудови графіку необхідні результати опрацювання двох методів convertedData та initialCurrency , які визначають період для побудови графіку та обрану валюту відповідно.

При виборі валюти з списку надсилається запит, результати якого опрацьовує так званий «listener», який визначає проміжок часу та передає його

значення для побудови по осі X і визначає величину курсу обраної валюти та передає її значення для побудови по осі Y.

За це відповідає наступний код:

```
chooseCurrency.change(async ({ target: { value } }) => {
    loader.show();
    const periodByDate = await window.rateByPeriod(value);
    const convertedData = mapData.convertToChartData(periodByDate);
    myChart.data.labels = convertedData.x;
    myChart.data.datasets[0].label = value;
    myChart.data.datasets[0].data = convertedData.y;
    myChart.update();
    loader.hide();
  });
```

Період за який формується графік визначений величиною в 5 днів дозволяє аналізувати зміну курсу вибраної валюти. Для того, щоб показати період на графіку з врахуванням поточного дня та чотирьох попередніх і визначити велиину курсу за ці дні, розроблено реверсну функцію:

```
convertToChartData: (currencyData) => {
let chartData = {
 x: [],
 y: [],  };
currencyData.reverse().forEach(([currencyByDay]) => {
chartData.x.push(currencyByDay.exchangedate.slice(0, 5));
chartData.y.push(currencyByDay.rate);
});
 return chartData;
```

};

Щоб привести формат дати, що надходить в файлі json до потрібного вигляду, використовується бібліотека «Moment».

Функція передає отримані дані до методу побудови графіку setChart відповідно до осі X та Y.

```
setChart: (data, currency) => {
    myChart = new Chart(currencyChart, {
      type: 'line',
      data: {
        labels: data.x,
        datasets: [{
          label: currency,
          data: data.y,
          borderWidth: 1,
          backgroundColor: ['rgba(255, 99, 132, 0.2)'],
          borderColor: ['#ccc']
 }],
 }
```

Після отримання необхідних даних будується графік за допомогою додаткової бібліотеки «Chart» (рис.3.16).

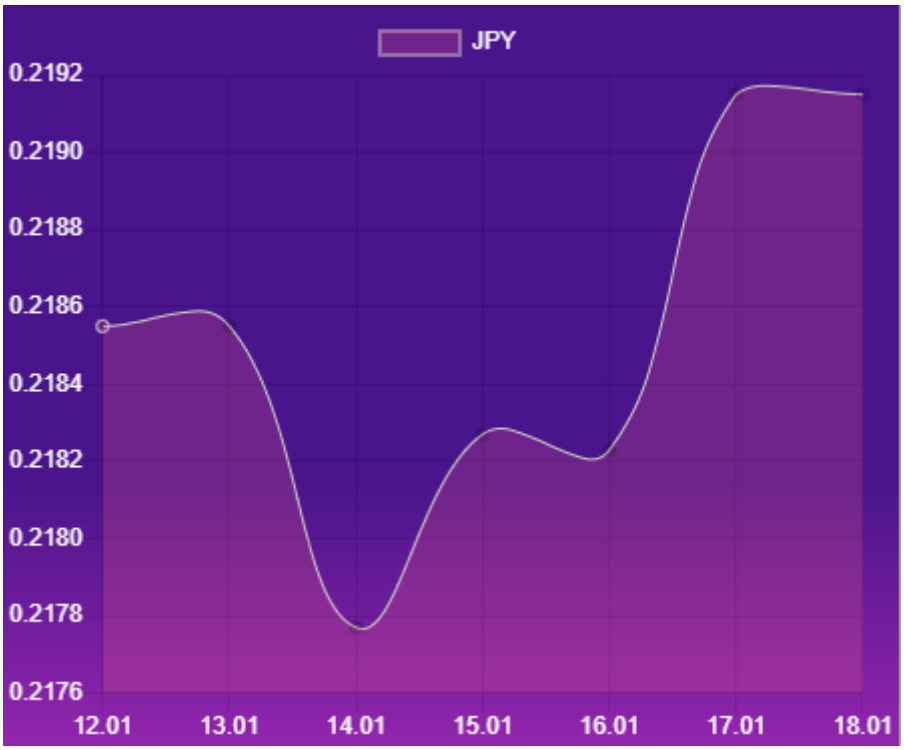

Рис.3.16 Візуалізація графіка

ВИСНОВОК ДО РОЗДІЛУ 3

Визначено, що для структуризації коду, всі компоненти плагіну виокремлені в папки з відповідними назвами, що включають основні файли: manifest.json, background.js, index.html та css - файли.

Додаткові файли, які використані для розробки плагіну:

- BankApi.js
- Chart.js
- Reset.css
- Jquery.js
- Moment.js
- bootstrap-datepicker.min.css
- bootstrap-datepicker.standalone.min.css
- bootstrap-datepicker3.standalone.min.css

При розробці навігаційної панелі реалізовано горизонтальне меню, що має чотири вкладки: Currency, Converter, Choose date та Chart перехід між якими здійснюється по кліку. Перехід між вкладками дозволяє користувачу виконати певну послідовність дій: переглянути список валют з відповідними курсами, конвертувати необхідну суму, переглянути курс валюти за певну дату та проаналізувати графік зміни курсу.

Отже, плагін має структуровані файли та зручний інтерфейс, що дозволяє оцінити високу якість розробки.

РОЗДІЛ 4
КРОСБРАУЗЕРНІСТЬ ПЛАГІНУ. ЗАВАНТАЖЕННЯ ДО ОСНОВНИХ БРАУЗЕРІВ. ПУБЛІКУВАННЯ ПЛАГІНУ В МАГАЗИНІ РОЗШИРЕНЬ GOOGLE CHROME

4.1 Кросбраузерність плагіну

Під терміном «кросбраузерність» розуміється коректне відображення вмісту плагіну в різних браузерах. В даному випадку за основу взято три найбільш використовуваних браузера, це Google Chrome, Mozilla Firefox та Microsoft Edge.

Кросбраузерність обумовлена можливістю плагіну підлаштовуватись під поведінку того чи іншого браузера. Головним чином, вона проявляється в точному відтворенні інтрейфейсу плагіну та наданні однакових функціональних можливостей, незалежно від середовища влаштування.

Кросбраузерність забезпечується шляхом розробки плагіну у відповідності до технології WebExtentions, яка вміщає в себе всю інформацію про розширення, як от інформацію про автора, версію, необхідність встановлення додаткового забезпечення, взаємодію з браузером, контент сценарії та бекграунд скрипти, що потрібні для роботи плагіну.

Запити в даній технології реалізовані асинхронними API. Найчастіше вони сумісні з extension API, та підтримуються більшістю браузерів.

Також, кросбраузерність обумовлена синтаксисом та семантикою використовуваної мови програмування, що дозволяє врахувати підтримувані браузером атрибути, властивості та поведінку.

4.2 Завантаження плагіну до бібліотеки розширень браузерів

4.2.1 Google Chrome

Після завершеної розробки плагіну здійснюється завантаження архіву з розробкою «Exchange.zip» до браузера. Для цього потрібно:

1. Відкрити браузер Google Chrome на комп’ютері.
2. Угорі праворуч натиснути значок із трьома крапками ⋮ > Інші інструменти > Розширення.
3. Включити функцію «Режим розробника».
4. На верхній панелі обрати «Завантажити розпаковане/запаковане розширення».
5. З’явиться вікно з вибором каталогу для загрузки (рис 4.1), в якому потрібно обрати папку зі створеним розширенням, або запакований zip-архів, який в процесі публікації перетвориться в формат .crx.

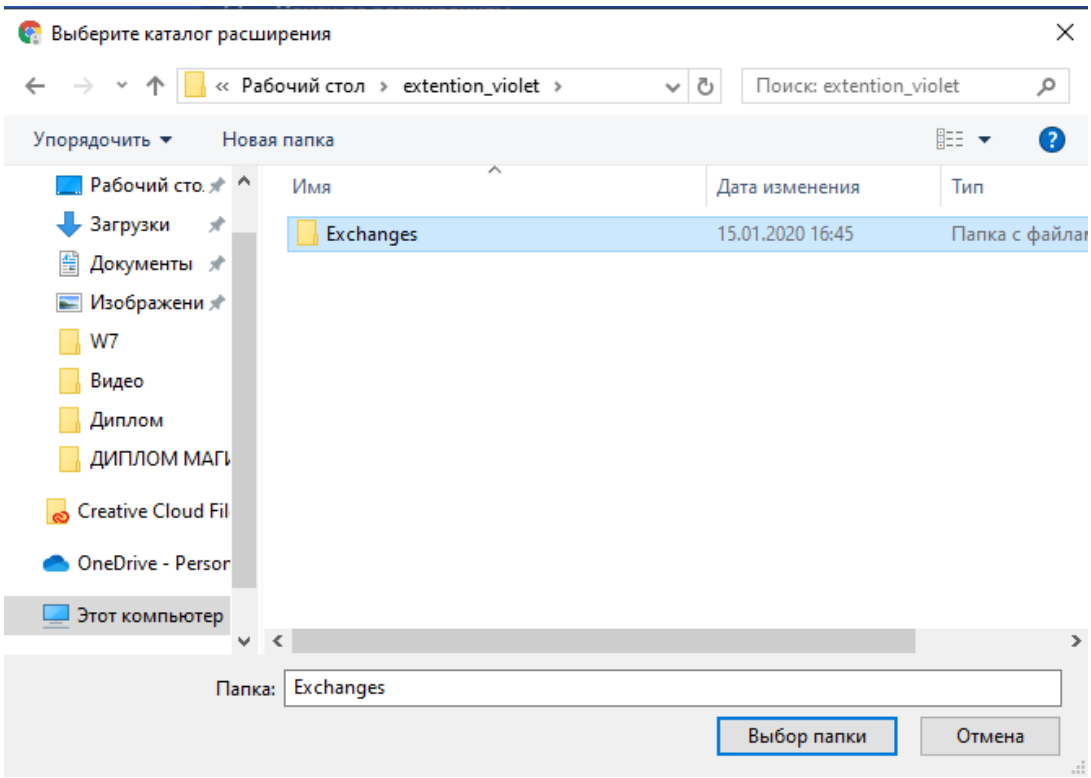

Рис.4.1 Вибір каталогу з плагіном

6. Натиснути кнопку «ОК».

7. Після чого, в вкладці «Розширення» з’явиться вікно з назвою розширення «Exchange Rates» (рис.3.2), а на панелі інструментів – його іконка.

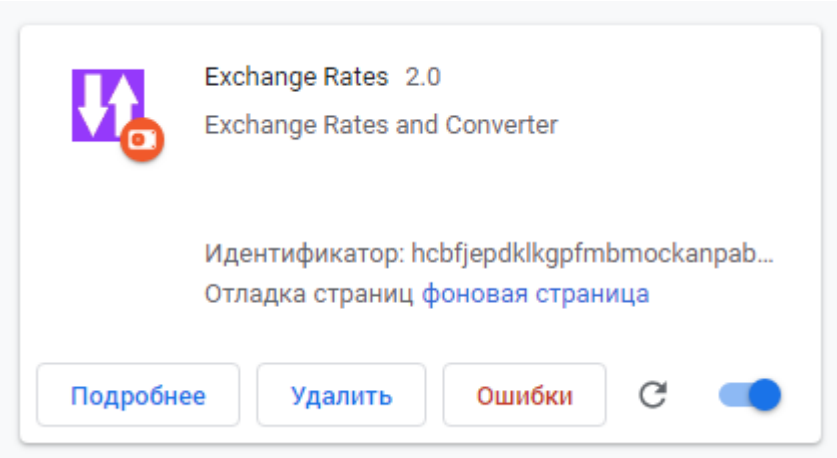

Рис.4.2 Вікно розширення «Exchange Rates»

8. На панелі інструментів з’явилась іконка розширення, натиснувши на яку, можна користуватись завантаженим плагіном (рис. 4.3).

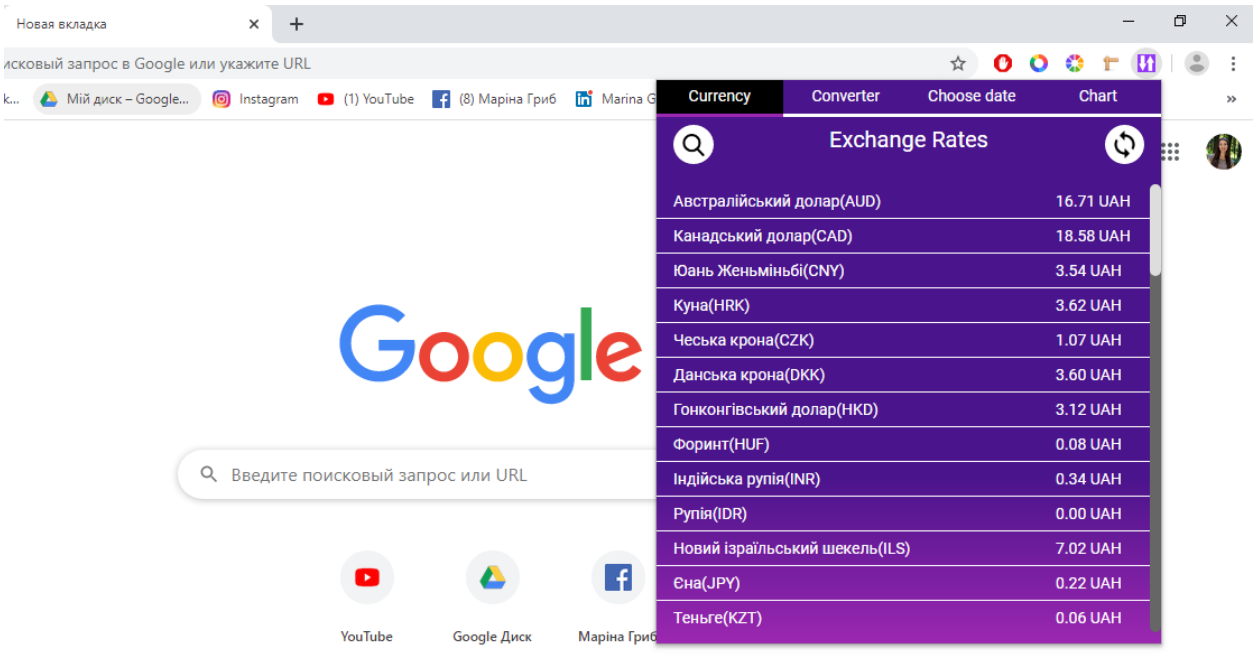

Рис. 4.3 Відображення завантаженого плагіну в Google Chrome

4.2.2 Mozilla Firefox

Щоб завантажити плагін в браузер Mozilla Firefox, потрібно виконати наступні дії:

1. Відкрити браузер на комп’ютері.

2. В пошукову рядку ввести «about:debugging» та перейти у вкладку «Цей Firefox».

3. Натиснути кнопку «Завантажити тимчасовий додаток». З’явиться вікно з вибором каталогу для загрузки.

4. Обрати в вікні файл manifest.json або увесь архів «Exchange».

5. Натиснути кнопку «ОК».

6. Після чого, у вкладці «Розширення» з’явиться вікно з назвою розширення «Exchange Rates» (рис.4.4), а на панелі інструментів – його іконка.

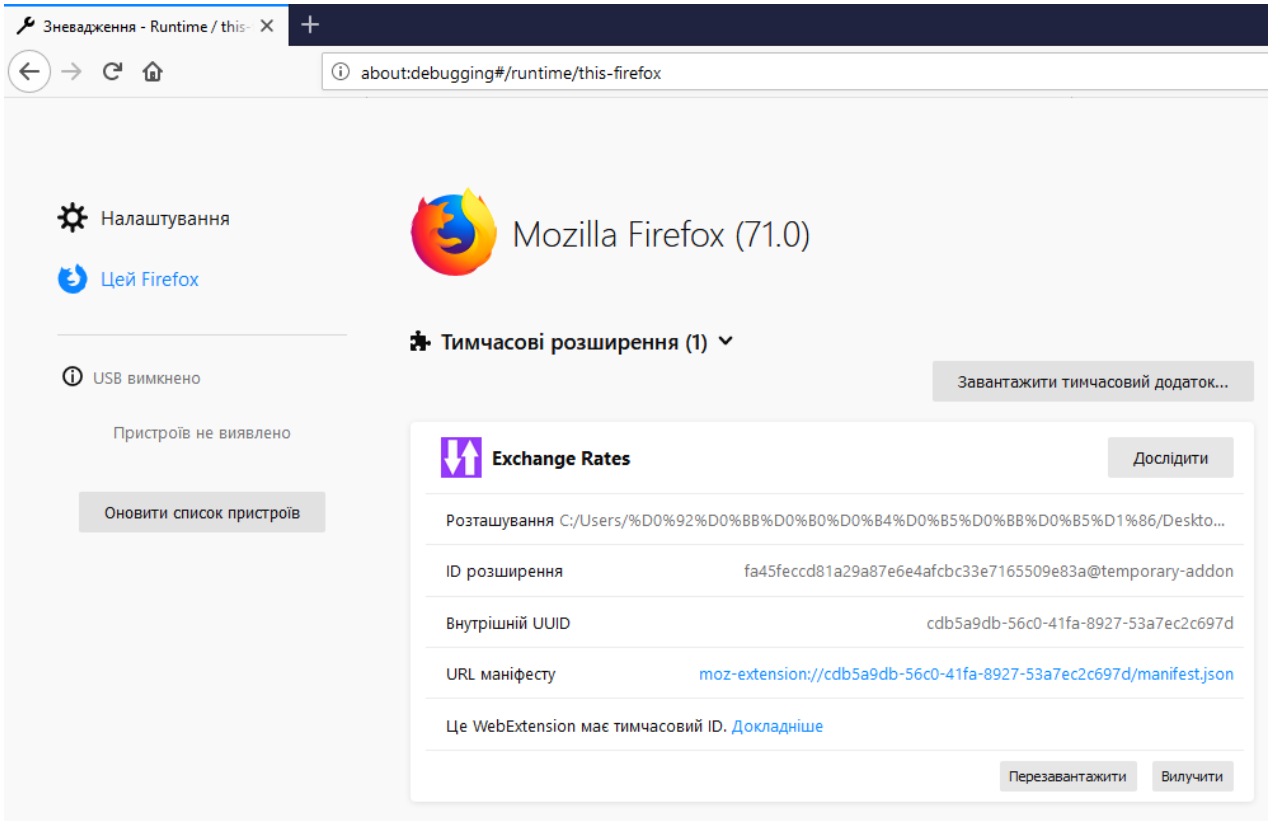

Рис. 4.4 Вікно з відображенням завантаженого розширення

7. Після натискання на іконку розширення, що знаходиться на панелі інструментів, з’явиться повноцінний інтерфейс плагіну (рис. 4.5). Плагін буде доступний впродовж поточної сесії браузера та зникне після його перезавантаження.

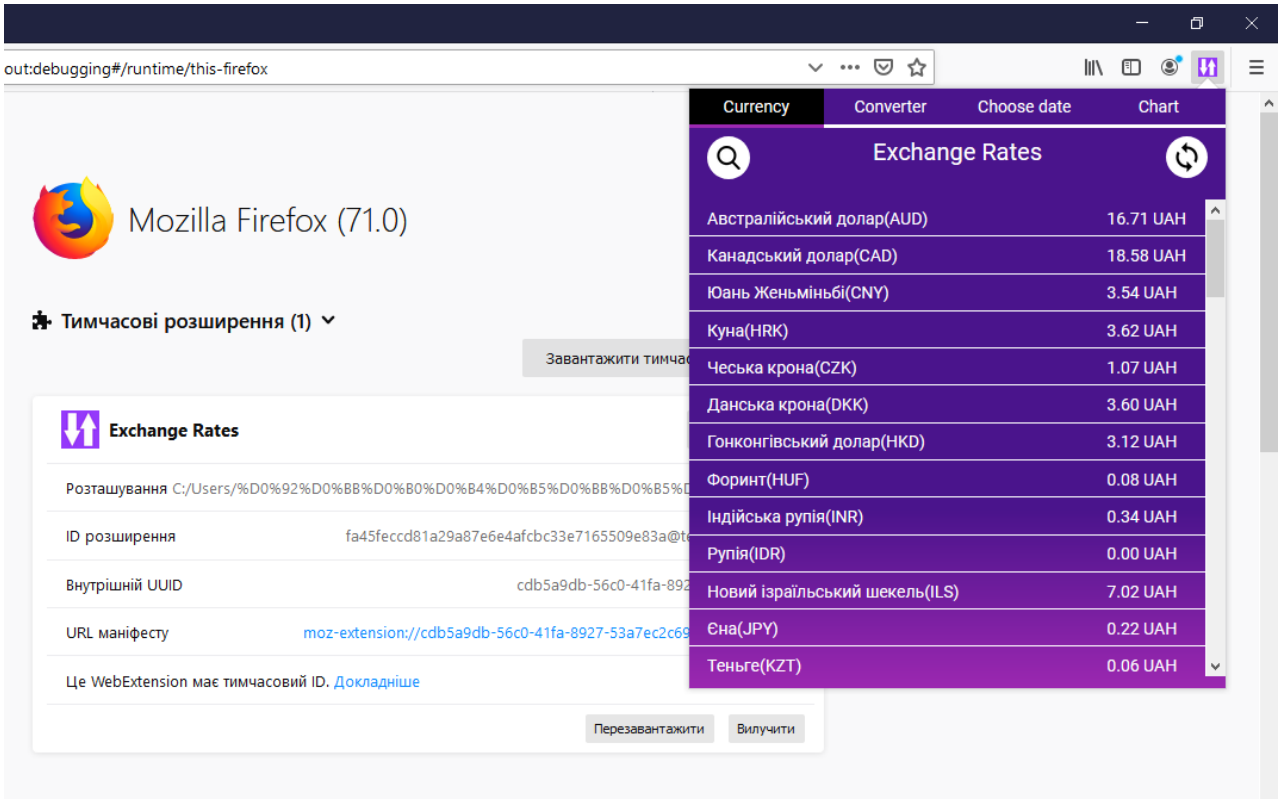

Рис. 4.5 Відображення завантаженого плагіну в Mozilla Firefox

4.2.3 Microsoft Edge

Для завантаження архіву з розробкою «Exchange.zip» до браузера Microsoft Edge потрібно:

1. Відкрити браузер Google Chrome на комп’ютері.

2. Угорі, на панелі інструментів праворуч натиснути значок із трьома крапками ⋮ > Розширення, або ввести в пошуковому рядку «edge://extensions/».

3. Натиснути перемикач «Режим розробника».

4. На екрані з’являться налаштування. Обрати «Загрузити розпаковане розширення».

5. У вкладці відобразиться завантажене розширення (рис. 4.6).

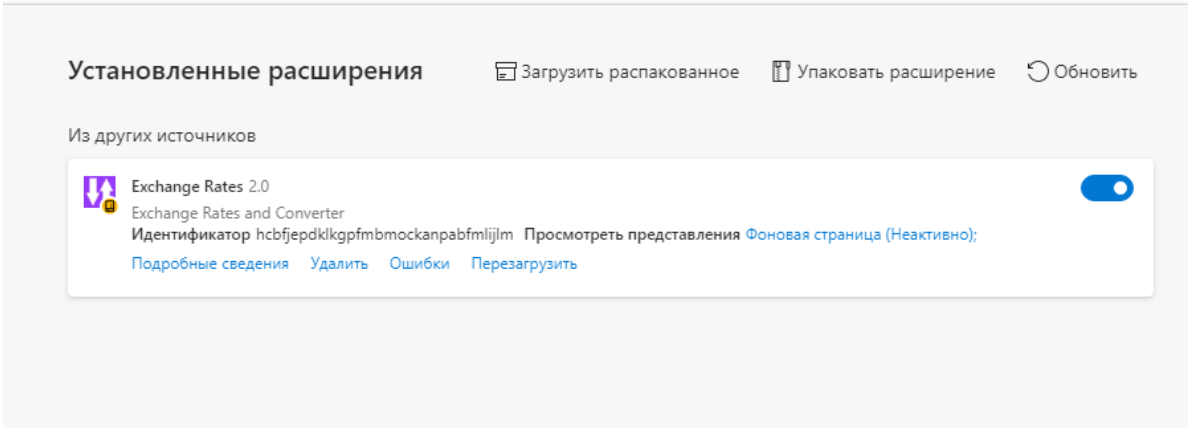

Рис. 4.6 Вікно з відображенням завантаженого розширення

6. Після натискання на іконку розширення, що знаходиться на панелі інструментів, з’явиться повноцінний інтерфейс плагіну (рис. 4.7).

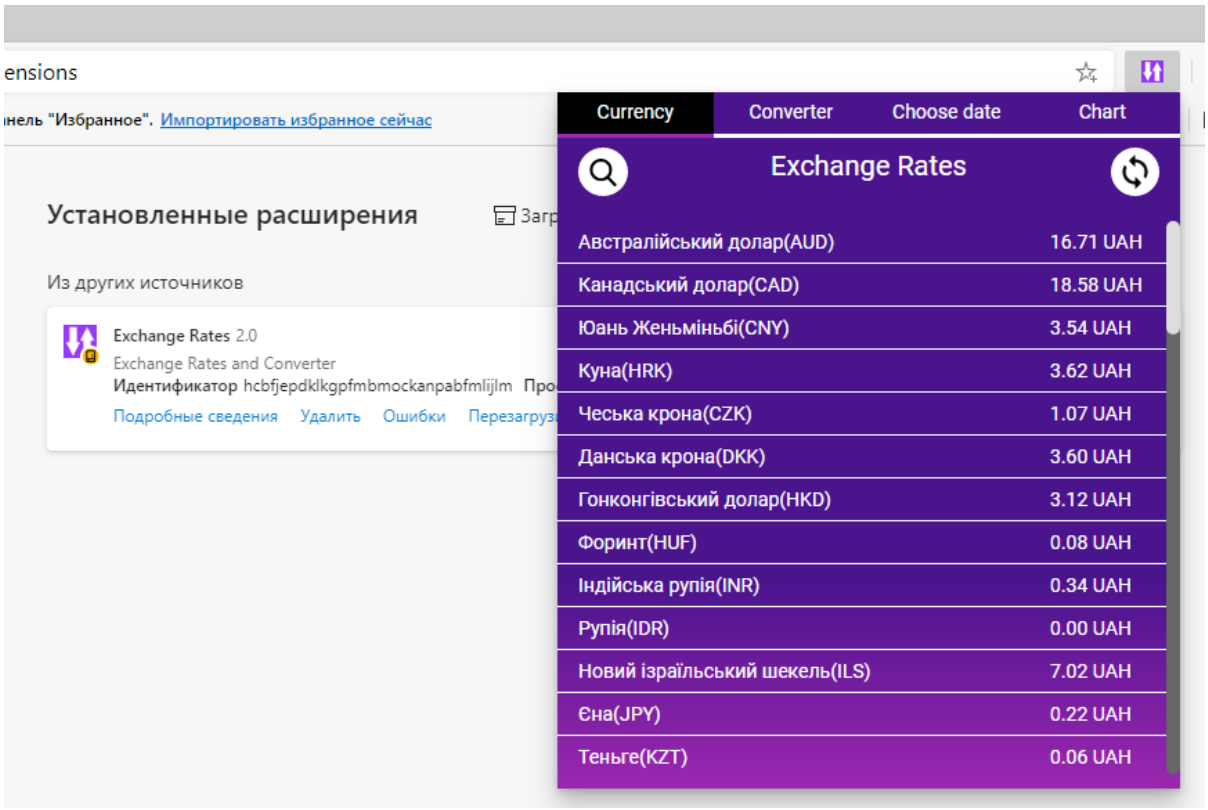

Рис. 4.7 Відображення завантаженого плагіну в Microsoft Edge

4.3 Завантаження плагіну до магазину розширень Google Chrome

Користувач певного браузера має змогу завантажити власний плагін локально в браузер, що описано в пункті 4.2. Тобто, плагін буде доступним лише для користувача, який його завантажив.

Для того, щоб мати можливість розповсюдити власну розробку для всіх користувачів певного браузера, потрібно завантажити плагін до магазину розширень.

З огляду на те, що найбільш часто використовуваним веб-переглядачем в Україні є Google Chrome, завантаження до магазину розширень виконане саме в ньому.

Поетапне завантаження:

1. Відкрити браузер Google Chrome

2. Угорі, праворуч на панелі інструментів, натиснути значок із трьома крапками ⋮ > Інші інструменти > Розширення.

3. Виконаний перехід на сторінку «Розширення». В лівому куті натиснути на меню та обрати пункт «Відкрити інтернет-магазин Chrome».

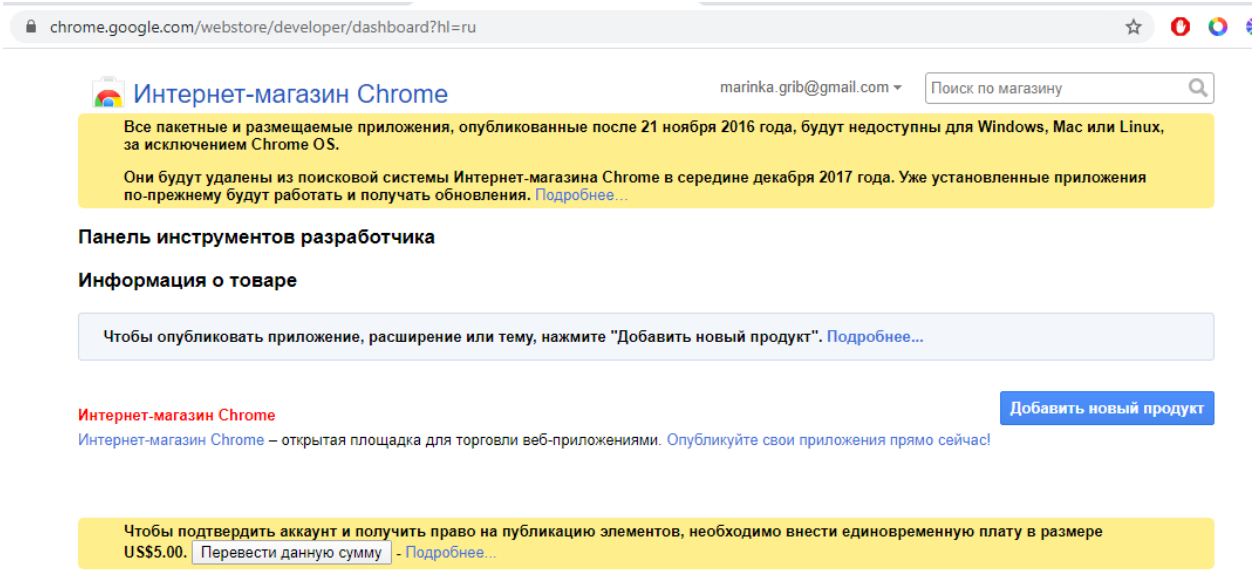

Рис.4.8 Інтерфейс інтернет-магазину Chrome

4. Зліва на панелі обрати «Умови використання» і в розділі «Для розробників» натиснути «Опублікувати додаток Chrome».

5. В цілях безпеки, браузер проводить ідентифікацію користувача та повторний вхід в обліковий запис.

6. Після підтвердження входу в обліковий запис, здійснюється перехід до магазину завантажень (рис.4.8).

7. Натиснути на кнопку «Додати новий продукт» та прийняти умови політики конфіденційності.

8. Завантажити архів з розширенням .zip, який містить основні структурні файли плагіну (рис.4.9).

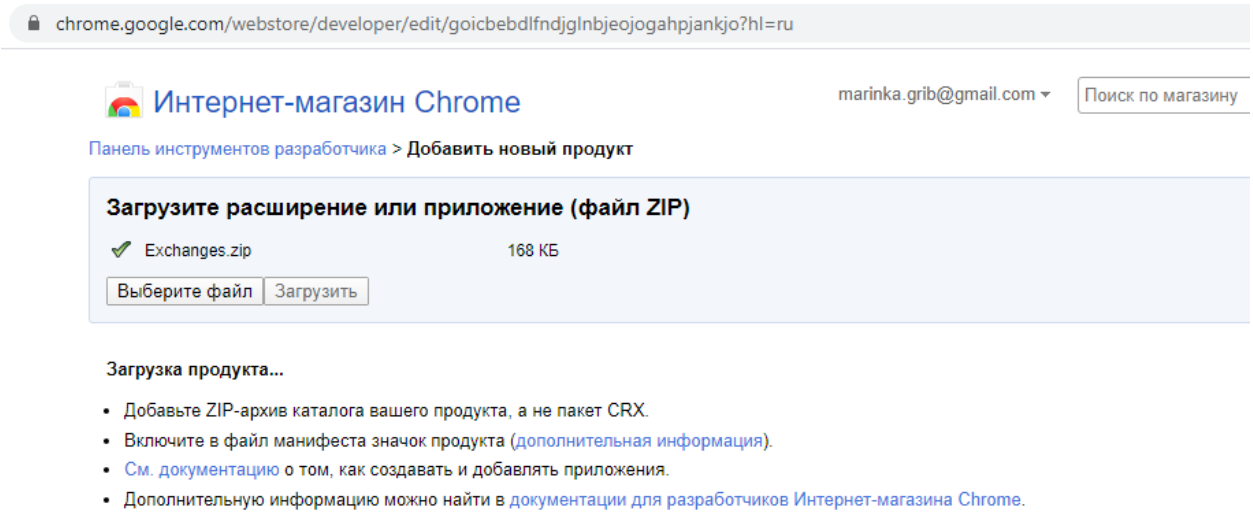

Рис.4.9 Завантаження архіву

9. Натиснути «Завантажити».

10. Заповнити необхідні поля форми, які характеризують розширення: логотип, детальний опис роботи, скріншоти, категорія до якої належить плагін, регіони розповсюдження, надання безкоштовного доступу та інші.

11. Для опублікування розширення в інтернет-магазині потрібно внести плату в розмірі п’яти доларів (рис.4.10). За внесені кошти, розробнику надається право опублікування двадцяти продуктів.

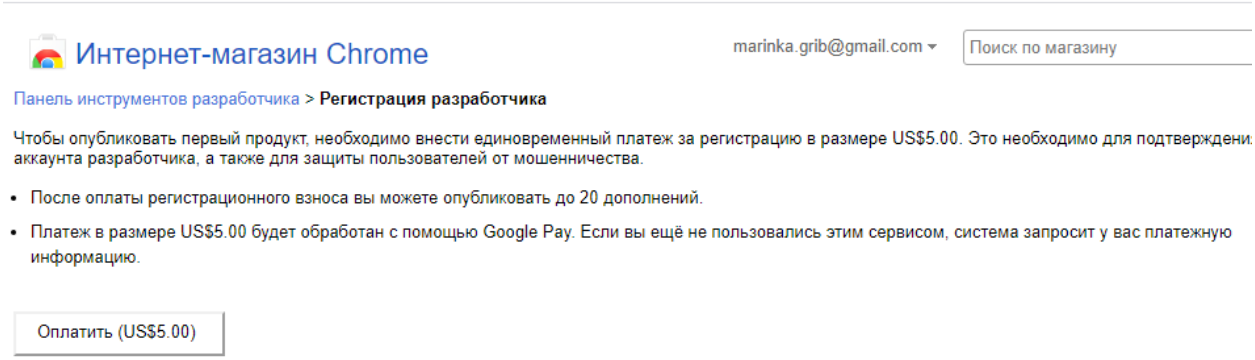

Рис. 4.10 Запит на оплату публікування розширення

12. Після успішної оплати перейти до «Акаунта розробника» та в правій частині натиснути «Опублікувати», підтвердити публікацію натиснувши на кнопку (рис.4.11).

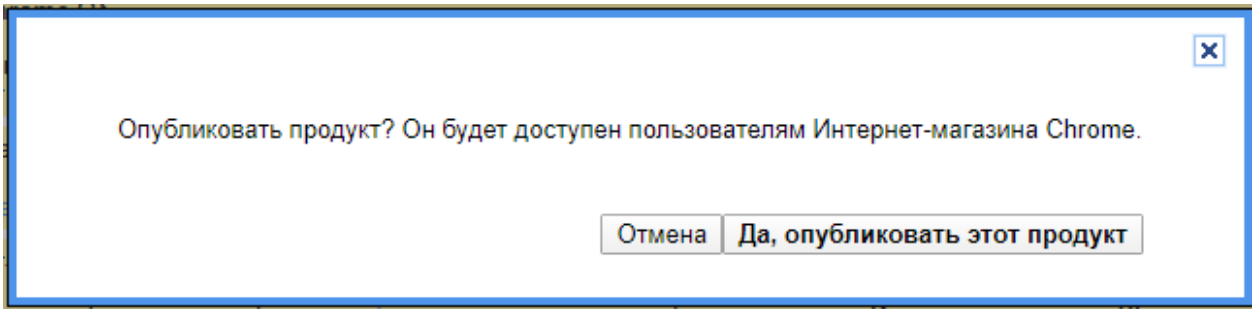

Рис.4.11 Запит на публікування плагіну

13. На панелі інструментів розробника з’явиться відповідний статус «Очікує розгляду» (рис.4.12). який вказує, що завантажене розширення буде розглянуте на вміст вірусного зараження, порушення правил розробника,

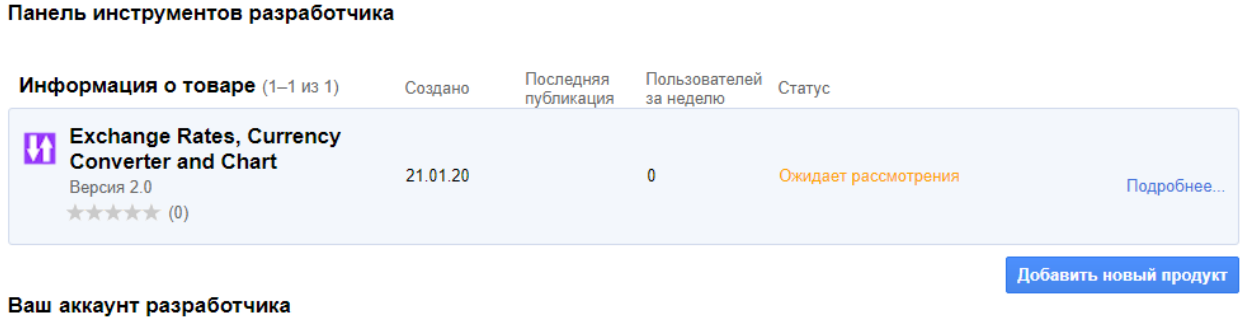

Рис. 4.12 Статус розширення

Під час розгляду, плагін перевіряється на:

- вміст вірусного програмного забезпечення та його передачі,
- порушення правил розробника,
- на повторне завантаження видаленого за порушення закону контенту,
- запит на додаткові дозволи, які вимагають перевірки.

Отримавши позитивний результат розгляду, розширення перебуває в загальному доступі та може бути встановлене користувачами браузера Google Chrome.

ВИСНОВОК ДО РОЗДІЛУ 4

Під терміном «кросбраузерність» розуміється коректне відображення вмісту плагіну в різних браузерах. В даному випадку, за основу взято три найбільш використовуваних браузера, це Google Chrome, Mozilla Firefox та Microsoft Edge.

Головним чином, вона проявляється в точному відтворенні інтрейфейсу плагіну та наданні однакових функціональних можливостей, незалежно від середовища влаштування.

Тому, кросбраузерність забезпечується шляхом розробки плагіну у відповідності до технології WebExtentions, яка вміщує всю інформацію про розширення, як от, інформацію про автора, версію, необхідність встановлення додаткового програмного забезпечення, взаємодію з браузером, контент сценарії та бекграунд скрипти, що потрібні для коректної роботи плагіну.

Користувач веб-переглядача має змогу завантажити власний плагін локально в браузер, або опублікувати в магазині розширень.

Під час публікації плагін перевіряється на:

- вміст вірусного програмного забезпечення та його передачі,
- порушення правил розробника,
- на повторне завантаження видаленого за порушення закону контенту,
- запит на додаткові дозволи, які вимагають перевірки.

ВИСНОВКИ

В першому розділі розглянуті технології та підходи до розробки плагінів, проаналізовані підходи до побудови користувацьких інтерфейсів веб-додатків.

Визначено, що мова гіпертекстової розмітки разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

Переваги використання технології AJAX при розробці розширення — сторінка модифікується без повного перезавантаження, збільшується швидкість роботи з веб-програмою та зменшується трафік між сервером та клієнтом.

У другому розділі розглянуті архітектура та структурні файли плагіну.

Плагін, як правило, складається зі стислих пакетів HTML, CSS, Java Script коду, зображень та інших файлів, які використовуються в веб-платформі та налаштовують роботу з браузером.

Архітектура плагіну залежить від його функціональності, але практично всі плагіни включають основні компоненти:

- Manifest
- Background Script
- User Interface Elements
- Content Script
- Options Page

Визначено, що для структуризації коду, всі компоненти плагіну виокремлені в папки з відповідними назвами, що включають основні файли: manifest.json, background.js, index.html та css - файли.

При розробці плагіну реалізовано: навігаційна панель з горизонтальним меню, що має чотири вкладки: Currency, Converter, Choose date та Chart перехід між якими здійснюється по кліку. Перехід між вкладками дозволяє

користувачу виконати певну послідовність дій: переглянути список валют з відповідними курсами, конвертувати необхідну суму, переглянути курс валюти за певну дату та проаналізувати графік зміни курсу.

Під терміном «кросбраузерність» розуміється коректне відображення вмісту плагіну в різних браузерах. В даному випадку, за основу взято три найбільш використовуваних браузера, це Google Chrome, Mozilla Firefox та Microsoft Edge.

Тому, кросбраузерність забезпечується шляхом розробки плагіну у відповідності до технології WebExtentions, яка вміщує всю інформацію про розширення, як от, інформацію про автора, версію, необхідність встановлення додаткового програмного забезпечення, взаємодію з браузером, контент сценарії та бекграунд скрипти, що потрібні для коректної роботи плагіну.

Користувач веб-переглядача має змогу завантажити власний плагін локально в браузер, або опублікувати в магазині розширень.

Отже, за допомогою основних веб-технологій розроблено кросбраузерний функціональний плагін, що дозволяє:

- отримати курси п’ятдесяти двох валют та чотирьох сплавів,
- конвертувати необхідну суму в обрану валюту,
- отримати курс обраної валюти за певну дату,
- переглянути графік зміни обраної валюти за період часу.

Плагін має структуровані файли та зручний інтерфейс, що дозволяє оцінити високу якість розробки.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Jack M. KHTML vs. Gecko vs. Trident vs.Presto: Behind the Browser / M. Jack//ECT NewsNetwork : news sites TechNewsWorld.com[Електронний ресурс]. – Режим доступу:http://www.technewsworld.com/rsstory/59309.html.

2. Портал Wikipedia.org [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Comparison_of_web_browsers.

3. Блог проекту Mozilla [Електронний ресурс]. – Режим доступу:http://blog. mozilla.org/blog/2008/07/02/were-official.

4. Статистичний портал StatCounter [Електронний ресурс]. – Режим доступу:http://gs.statcounter.com/#browser-ww-monthly-201002-201305.

5. Звіт «Порівняльний аналіз безпеки браузерів 2019» компанії NSS Labs [Електронний ресурс]. – Режим доступу:https://www.nsslabs.com/reports/categories/test-reports/browser-security.

6. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010. — 656 с.

7. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, Edition. — М.: «Диалектика», 2010. — 656 с.

8. Дубаков М.А. Веб-мастеринг средствами CSS. - СПб.: БХВ-Петербург, 2002. – 544 с.

9. Дейв Крейн, Бер Бибо, Джордон Сонневельд. Ajax in Practice. / Дейв Крейн, Бер Бибо, Джордон Сонневельд.— М.: Вильямс, 2007.- 348с.

10. Фролов А.В., Создание Web-приложений: Практическое руководство. - М.: Издательско-торговый дом "Русская редакция", 2001. - 1040 с.

11. Блог проекту Envatotuts+ [Електронний ресурс]. – Режим доступу: https://code.tutsplus.com/uk/tutorials/developing-google-chrome-extensions--net-33076.

ДОДАТОК А

manifest.json

```
{
   "manifest_version": 2,
   "name": "Exchange Rates, Currency Converter and Chart",
   "short_name": "Exchange Rates",
   "author": "Marina Grib",
   "description": "Курс валют, конвертація валюти та графіки зміни курсу за п
еріод",
   "version": "2.0",
   "icons": {
      "128": "img/logo1.png"
   },
   "background": {
      "scripts": ["js/background.js"],
      "persistent" : false
   },
   "browser_action": {
      "default_popup": "index.html"
   },
   "web_accessible_resources": ["img/60961.png", "img/149852.png", "img/logo.p
ng", "loader.gif"]
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport"
```

```
    content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>Exchange Rates</title>
  <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet" />
  <link rel="stylesheet" href="style/reset.css" />
  <link rel="stylesheet" href="style/main.css" />
  <!--<link rel="stylesheet" href="style/bootstrap-datepicker.min.css">-->
  <link rel="stylesheet" href="style/bootstrap-datepicker.standalone.min.css" />
</head>

<body>
  <div class="content">
    <header>
      <ul class="navigator">
        <li class="active" data-active="currency">Currency</li>
        <li data-active="converter">Converter</li>
        <li data-active="chooseData">Choose date</li>
        <li data-active="chart">Chart</li>
      </ul>
    </header>

    <div class="main-content">
      <div class="tab-content">
        <div class="loader"></div>
        <div class="item-tab active" data-content="currency">
          <div class="wrapper">
            <input type="text" class="search-
currency" placeholder="Search" value="" />
            <h1>Exchange Rates</h1>
            <div class="update"></div>
          </div>
```

```
  <div class="table-currency"></div>
</div>
<div class="item-tab converter" data-content="converter">
  <h1>Converter</h1>
  <div class="wrap-convert">
    <h5>Amount:</h5>
    <input type="text" class="amount" placeholder="Enter amount" value=""/>
  </div>
  <div class="wrap-convert">
    <h5>From:</h5>
    <select class="selectFrom"></select>
  </div>
  <div class="wrap-convert">
    <h5>To:</h5>
    <select class="selectTo"></select>
  </div>
  <button class="convertBtn" type="button">Convert</button>
  <div class="wrap-convert">
    <h5>Result:</h5>
    <div class="resultConvert"></div>
  </div>
</div>
<div class="item-tab choose-data" data-content="chooseData">
  <h1>Choose date</h1>
  <input class="dataPicker" data-date-end-date="0d" value="" />
  <div class="table-currency"></div>
</div>
<div class="item-tab chart-tab" data-content="chart">
  <h1>Chart</h1>
  <div class="wrap-convert">
    <h5>Currency:</h5>
    <select class="choose-currency"></select>
```

```
          </div>
          <div class="chart-wrapper">
            <canvas class="currencyChart" width="400" height="330"></canvas>
          </div>
        </div>
      </div>
    </div>
  </div>
  <script src="js/libs/jquery.js"></script>
  <script src="js/libs/bootstrap-datepicker.min.js"></script>
  <script src="js/libs/chart.js"></script>
  <script src="js/libs/moment.js"></script>
  <script src="js/bankApi.js"></script>
  <script src="js/background.js"></script>
</body>
</html>
```

background.js

```
$(function () {
  const searchCurrency = $(".search-currency");
  const loader = $(".loader");
  const navigator = $(".navigator li");
  const allTabsCont = $(".item-tab");
  const convertBtn = $(".convertBtn");
  const amountInput = $(".amount");
  const dataPicker = $(".dataPicker");
  const update = $(".update");
  const chartTab = $(".navigator li[data-active='chart']");
  const chooseCurrency = $(".choose-currency");
  const currencyChart=document.querySelector(".currencyChart").getContext('2d');
  let amountState = "",
    dataCurrency;
  let myChart;
  const mapData = {
    currency: setOptions => result => {
      const tableCurrency = $(".table-currency");
      let stringCurrency = "";
      /* save data */
      dataCurrency = result;
      result.forEach((v, k) => {
        stringCurrency += `<div class="currency-row">
          <span class="name-curr">${v.txt}(${v.cc})</span>
          <span>${v.rate.toFixed(2)} UAH</span>
        </div>`;
      });
      dataCurrency.unshift({
        cc: "UAH",
        exchangedate: "",
```

```
      r030: 0,
      rate: 1,
      txt: "Гривня"
    });
    tableCurrency.empty().append($(stringCurrency));
    setOptions && setOptions(dataCurrency);
    loader.hide();
  },
  convertToChartData: (currencyData) => {
    let chartData = {
      x: [],
      y: [],
    };
    currencyData.reverse().forEach(([currencyByDay]) => {
      chartData.x.push(currencyByDay.exchangedate.slice(0, 5));
      chartData.y.push(currencyByDay.rate);
    });
    return chartData;
  }
};
const errorHandlers = {
  getCurrencyError: () => window.currency(mapData.currency, errorHandlers.getCurrencyError)
};
const stateFunction = {  getCurrency: setOptions => {
    loader.show();
    window.currency(mapData.currency(setOptions));
  },
  getCurrencyDate: date => {
    const tableCurrency = $(".choose-data .table-currency");
    loader.show();
```

```
    window.currencyByDate(date, result => {
      let stringCurrency = "";
      result.forEach((v, k) => {
        stringCurrency +=
          `<div class="currency-row">
            <span class="name-curr">${v.txt}(${v.cc})</span>
            <span>${v.rate.toFixed(2)} UAH</span>
          </div>`;
      });
      tableCurrency.empty().append($(stringCurrency));
      loader.hide();
    });
  },
  setSelectCurrency: currencyData => {
    const selectFrom = $(".selectFrom");
    const selectTo = $(".selectTo");
    const chooseCurrency = $(".choose-currency");
    let allCurrencies = "";
    let allCurrenciesChart = "";
    currencyData.forEach((v, k) => {
      allCurrencies += `<option value = "${v.rate}"> ${v.txt} (${v.cc})</option>`;
      k && (allCurrenciesChart += `<option value = "${v.cc}"> ${v.txt} (${v.cc})</option>`);
    });
setChart: (data, currency) => {
    myChart = new Chart(currencyChart, {
      type: 'line',
      data: {
        labels: data.x,
        datasets: [{
          label: currency,
          data: data.y,
```

```
          borderWidth: 1,
          backgroundColor: ['rgba(255, 99, 132, 0.2)'],
          borderColor: ['#ccc'],
        }],
      },
      options: {legend: {labels: {fontColor: 'white', }},
        scales: {yAxes: [{ticks: {fontColor: "white",}}],
          xAxes: [{ticks: {fontColor: "white",}}]
        }
      }
    });
  },
};
    selectFrom.append(allCurrencies);
    selectTo.append(allCurrencies);
    chooseCurrency.append(allCurrenciesChart);
  },
(() => {
  /*set current currency data*/
  stateFunction.getCurrency(stateFunction.setSelectCurrency);
  /*initialize datapicker*/
  dataPicker
    .datepicker({
      format: "yyyy.mm.dd"
    })
    .on("changeDate", () => {
      dataPicker.datepicker("hide");
      /*update table with chose date*/
      stateFunction.getCurrencyDate(dataPicker.val().replace(/[^0-9]/gi, ""));
    });
  /* search currency */
  searchCurrency.on("input", function () {
```

```
  const searchString = $(this)
    .val()
    .toLowerCase();
  const allCurrencyRow = $(".currency-row");
  if (searchString.length <= 2) {
    return allCurrencyRow.show(), false;
  }
  allCurrencyRow.each((i, el) => {
    const curName = $(el)
      .find(".name-curr")
      .text()
      .toLowerCase();
    if (curName.indexOf(searchString) >= 0) $(el).show();
    else $(el).hide();
  });
});
/*active tab content*/
navigator.click(function () {
  const currActive = $(this).attr("data-active");
  if ($(this).hasClass("active")) return false;
  navigator.removeClass("active");
  $(this).addClass("active");
  allTabsCont.removeClass("active");
  $(".item-tab[data-content=" + currActive + "]").addClass("active");
});
/*input amount currency*/
amountInput.on("input", function () {
  const replace = /[^0-9\.]/gi;
  const exeption = /^([0-9]+\.)$/;
  const valid = /^([0-9]+(\.?([0-9]+))?)$/;
  const cut = $(this)
    .val()
```

```
    .replace(replace, "");
  /*validate only number*/
  if (!valid.test(amountInput.val())) {
    amountInput.val(cut);
    if (!valid.test(cut) && !exeption.test(cut) && cut.length) {
      amountInput.val(amountState);
    }
  }
  amountState = amountInput.val();
});
/*event convert currency*/
convertBtn.click(function () {
  const resultConvert = $(".resultConvert");
  const amount = amountInput.val();
  const selectFrom = $(".selectFrom").val();
  const selectTo = $(".selectTo").val();
  resultConvert.text((+amount * +selectFrom) / +selectTo);
});
/*update btn*/
update.click(async () => {
  searchCurrency.val("");
  stateFunction.getCurrency();
});
/*chart*/
chartTab.click(async () => {
  loader.show();
  const initialCurrency = 'AUD';
  const periodByDate = await window.rateByPeriod(initialCurrency);
  const convertedData = mapData.convertToChartData(periodByDate);
  stateFunction.setChart(convertedData, initialCurrency);
  loader.hide();
});
```

```
  // choose currency listener
  chooseCurrency.change(async ({ target: { value } }) => {
    loader.show();
    const periodByDate = await window.rateByPeriod(value);
    const convertedData = mapData.convertToChartData(periodByDate);
    myChart.data.labels = convertedData.x;
    myChart.data.datasets[0].label = value;
    myChart.data.datasets[0].data = convertedData.y;
    myChart.update();
    loader.hide();
  });
 })();
});
```

bankAPI.js

```
(() => {
  const
mainUrl="https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?";
  const bankApi = {
    currency: (callback, errorHandler = () => { }) => {
      $.ajax({
        url: `${mainUrl}json`,
        type: "GET",
        dataType: "json",
        success: callback,
        error: errorHandler
      });
    },
    currencyByDate: (date, callback, errorHandler = () => { }) => {
      $.ajax({
        url: `${mainUrl}date=${date}&json`,
```

```
            type: "GET",
            dataType: "json",
            success: callback,
            error: errorHandler
        });
    },
    rateByPeriod: currency => {
        const period = 7;
        let rateCalls = [];
        for (let i = 0; i < period; i++) {
            let date = moment();
            date = date.subtract(i, "days");
            date = date.format("YYYYMMDD");
            const call = new Promise((resolve, reject) => {
                $.ajax({
                    url: `${mainUrl}valcode=${currency}&date=${date}&json`,
                    type: "GET",
                    dataType: "json",
                    success: resolve,
                    error: reject
                });
            });
            rateCalls.push(call);
        };
        return Promise.all(rateCalls);
    },
};
window.currency = bankApi.currency;
window.currencyByDate = bankApi.currencyByDate;
window.rateByPeriod = bankApi.rateByPeriod;
})();
```

main.css

```
.content {
  margin: 0 auto;
  padding: 0 0 20px;
  width: 450px;
  /* background: #343330; */
  background: linear-gradient(#4a148c 70%, #9c28b1);
  font-family: "Roboto", sans-serif;
}
.content header ul {
  font-size: 0;
}
.content header ul li {
  width: 25%;
  color: white;
  font-size: 14px;
  padding: 8px 0;
  text-align: center;
  cursor: pointer;
  display: inline-block;
  border-bottom: 3px solid white;
  transition: all 0.2s ease;
}
.content header ul li:hover,
.content header ul li.active {
  border-bottom: 3px solid #9c28b1;
  background: #000;
}
.content .update {
  width: 35px;
  height: 35px;
```

```
  position: absolute;
  background: url("../img/60961.png");
  background-repeat: no-repeat;
  background-color: white;
  background-position: center;
  background-size: 25px;
  top: 7px;
  right: 15px;
  cursor: pointer;
  border-radius: 50%;
}
.wrapper {
  position: relative;
  margin: 0 0 20px;
}
.content .search-currency {
  position: absolute;
  top: 7px;
  left: 15px;
  z-index: 1;
  width: 0;
  padding: 10px 10px 10px 26px;
  background-image: url("../img/149852.png");
  background-repeat: no-repeat;
  background-color: white;
  background-position: center;
  background-position-x: 8px;
  background-size: 20px;
  border-radius: 50%;
  border: 0;
  box-sizing: border-box;
  transition: all 0.3s ease;
```

```
}
.content .search-currency:focus {
  width: calc(100% - 30px);
  padding: 10px 10px 10px 40px;
  border-radius: 5px;
}
.content h1 {
  color: white;
  text-align: center;
  margin: 0;
  padding: 10px 0;
  font-size: 20px;
  font-weight: 300;
}
.content .tab-content {
  position: relative;
  width: 100%;
  height: 450px;
}
.content .tab-content .loader {
  position: absolute;
  width: 100%;
  height: 100%;
  background: rgba(255, 255, 255, 0.2) url("../img/loader.gif");
  background-repeat: no-repeat;
  background-position: center;
  background-size: 60px;
}
.content .item-tab,
.content .item-tab,
.content .item-tab {
  display: none;
```

```
}
.content .item-tab.active,
.content .item-tab.active,
.content .item-tab.active {
  display: block;
}
.content .table-currency {
  height: 400px;
  overflow-y: scroll;
}
.content .table-currency::-webkit-scrollbar {
  width: 10px;
}

.content .table-currency::-webkit-scrollbar-track {
  background: #666;
  border-radius: 20px;
}

.content .table-currency::-webkit-scrollbar-thumb {
  background: #ddd;
  border-radius: 20px;
}
.content .table-currency .currency-row {
  width: 100%;
  padding: 7px 0;
  border-bottom: 1px solid white;
}
.content .table-currency .currency-row span {
  display: inline-block;
  color: white;
  font-size: 14px;
```

```
  width: 19%;
  vertical-align: middle;
  box-sizing: border-box;
}
.content .table-currency .currency-row span:first-child {
  padding: 0 0 0 15px;
  width: 80%;
}
/*tab converter*/
.converter .amount {
  display: inline-block;
  width: 70%;
  padding: 10px;
  height: 37px;
  border: none;
  box-sizing: border-box;
  margin: 0;
  border-radius: 4px;
  font-size: 15px;
  vertical-align: top;
}

.converter .convertBtn {
  display: inline-block;
  margin: 0 0 10px 26%;
  width: 70%;
  background: #9c28b1;
  border: none;
  color: white;
  font-size: 18px;
  font-weight: 500;
  padding: 10px 10px;
```

```
  cursor: pointer;
  border-radius: 4px;
}
.converter .resultConvert {
  display: inline-block;
  width: 70%;
  box-sizing: border-box;
  background: white;
  height: 37px;
  overflow: hidden;
  vertical-align: top;
  white-space: nowrap;
  text-overflow: ellipsis;
  border-radius: 4px;
  font-size: 15px;
  padding: 12px 10px 10px 10px;
}

.content .choose-data .table-currency {
  height: 373px;
}
.content .dataPicker {
  display: inline-block;
  width: 92%;
  padding: 10px;
  height: 37px;
  border: none;
  box-sizing: border-box;
  margin: 0 0 4px 4%;
  border-radius: 4px;
  font-size: 15px;
  vertical-align: top;
```

```
}

.wrap-convert h5 {
  display: inline-block;
  font-size: 17px;
  width: 22%;
  text-align: left;
  font-weight: normal;
  vertical-align: top;
  color: white;
  margin: 10px 0 0 4%;
}
.wrap-convert {
  width: 100%;
  font-size: 0;
  margin: 0 0 10px;
  box-sizing: border-box;
}
.wrap-convert .selectFrom,
.wrap-convert .choose-currency,
.wrap-convert .selectTo {
  display: inline-block;
  margin: 0;
  vertical-align: top;
  width: 70%;
  height: 37px;
  background: white;
  font-size: 15px;
  border: none;
  padding: 10px;
  border-radius: 4px;
}
```

reset.css

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    font-size: 100%;
    vertical-align: baseline;
    background: transparent;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}
```

```
/* remember to define focus styles! */
:focus {
   outline: 0;
}
/* remember to highlight inserts somehow! */
ins {
   text-decoration: none;
}
del {
   text-decoration: line-through;
}
/* tables still need 'cellspacing="0"' in the markup */
table {
   border-collapse: collapse;
   border-spacing: 0;
}
```