УДК 681.3.06

ФУНКЦІОНАЛЬНІ ОСОБЛИВОСТІ JAVASCRIPT-ДОДАТКІВ

Щербакова М.Є., Щербаков Є.В.

FUNCTIONAL FEATURES OF JAVASCRIPT-APPLICATIONS

Shcherbakova M.E., Shcherbakov E.V.

Розглянуті базова структура та функціональні особливості додатків, які розробляються на мові програмування JavaScript з використанням мови гіпертекстової розмітки HTML та каскадних таблиць стилів CSS. В той час, як кожен додаток є унікальним, більшість з них поділяють деякі загальні елементи, такі як інфраструктура хостингу, управління ресурсами, представлення та поведінка користувача інтерфейсу. Проаналізовані особливості функціонування основних компонентів інфраструктури додатка. Досліджено вплив на роботу додатка таких елементів, як сховище даних, веб-сервіси, серверна та клієнтська частини додатка та мережа доставки контенту. Сформульовані основні критерії розробки на JavaScript і HTML чутливих, соціально значущих додатків з привабливими візуальними інтерфейсами.

Ключові слова: *JavaScript, HTML, JSON, REST, Ajax, браузер, додаток, сховище даних,віртуальна приватна мережа.*

Вступ. З появою на початку поточного десятиліття робочої версії HTML5 [1] цю мову розмітки почали використовувати не тільки як мову визначення вмісту веб-сторінок в мережі Інтернет, але і як мову розмітки візуальних інтерфейсів додатків в платформах розробки на базі мови програмування JavaScript [2]. Прикладами таких платформ програмування є система розробки для магазину додатків Windows 8.1 на базі JavaScript і HTML5, а також відкритий фреймворк PhoneGap, який дозволяє створювати додатки для мобільних пристроїв з використанням мов JavaScript і HTML5 без необхідності використання «рідних» мов програмування під всі мобільні операційні системи (Windows Phone, iOS, Android, Bada і т. д.). В даний час JavaScript є найбільш широко використовуваною мовою програмування. Майже всі, хто має комп'ютер або смартфон, має всі інструменти, необхідні для виконання та створення власних програм JavaScript. Все, що для початкового етапу розробки програм на JavaScript потрібно - браузер і текстовий редактор.

Метою роботи є аналіз функціональних особливостей сучасних додатків на мові програмування JavaScript та визначення базових критеріїв оптимального програмування основних структурних частин додатків цього типу.

Анатомія типового сучасного JavaScript-додатка. Протягом довгого часу не було ніяких засобів зовнішнього збереження даних за допомогою JavaScript. Якщо хотілося зберегти дані, потрібно було відправити форму на веб-сервер і чекати оновлення сторінки. Це перешкоджало процесу створення швидких та динамічних веб-додатків. Однак у 2000 році компанія Microsoft почала поставки технології Ajax (Asynchronous Javascript and XML) [3] у складі Internet Explorer. Незабаром після цього програмну підтримку для всіх можливостей об'єкта XMLHttpRequest, який лежить в основі цієї технології, було додано і в інші браузери.

В 2004 році Google запустила додаток електронної пошти Gmail. Цьому з самого початку аплодували, тому що додаток надавав користувачам майже необмежене сховище для своєї електронної пошти. Gmail також ознаменував важливу революцію в функціональній структурі веб-додатків: пішло в минуле оновлення сторінок. В цілому, Gmail за рахунок використання нової технології Ajax являє собою односторінковий, швидкий і чутливий веб-додаток, який назавжди змінив спосіб проектування додатків цього типу. З тих пір веб-розробники створили додатки майже всіх типів, у тому числі повномасштабні, хмарні офісні пакети, соціальні API, подібні Facebook's JavaScript SDK, і навіть графічно напружені відеоігри.

JavaScript має дуже багаті об'єктно-орієнтовані (ОО) можливості. Стандарт JSON (JavaScript Object Notation), який використовується майже у всіх сучасних веб-додатках як для комунікацій, так і для збереження даних, є підмножиною чудової нотації об'єктних літералів JavaScript. JavaScript використовує модель прототипного спадкування. Замість класів використовуються прототипи об'єктів. Нові об'єкти автоматично успадковують методи та атрибути свого батьківського об'єкта

через ланцюжок прототипів. Прототип об'єкта можна змінювати в будь-який момент, що робить JavaScript дуже гнучкою, динамічною мовою. Прототипне спадкування набагато більш гнучке, ніж класичне спадкування, так що в JavaScript можна імітувати всі можливості моделі ОО та спадкування на основі класів C++ або Java, і в більшості випадків з меншою кількістю коду. Зворотнє не вірно. Всупереч загальноприйнятій думці, JavaScript підтримує такі можливості, як інкапсуляція, поліморфізм, множинне спадкування і композиція.

В той час як кожен додаток, написаний на JavaScript, є унікальним, більшість з них поділяють деякі загальні елементи, такі як інфраструктура хостингу, управління ресурсами, представлення та поведінка користувача інтерфейсу. Далі розглядається, де ці різні елементи знаходяться в додатку і загальні механізми, які дозволяють їм взаємодіяти [4].

Інфраструктура. Інфраструктура може мати багато особливостей і може мати багато різних механізмів кешування. Загалом, вона складається з наступних компонентів (справа наліво на рис. 1):

• Сховище даних (data store).

• Віртуальна приватна мережа (VPN) або брандмауер для захисту сховища даних від несанкціонованого доступу.

• Рівень сервісу: веб-сервіс Black Box JSON RESTful.

• Різні API сторонніх виробників.

• Серверна частина додатка або CMS для маршрутизації запитів і доставки сторінок клієнту.

• Мережа доставки статичного контенту (CDN - content delivery network) для кешованих файлів (наприклад, зображень, сценаріїв JavaScript, таблиць стилів CSS і клієнтських шаблонів).

• Клієнт (браузер).

На рис. 1 показано, як це поєднується одне з одним.

Більшість з цих компонентів не потребують пояснень, але є деякі важливі моменти, які потрібно знати про зберігання та передачу даних додатків.

Сховище даних якраз те, як це звучить: місце для зберігання даних програми. Зараз це, як правило, реляційна система управління базами даних (РСУБД) з мовою структурованих запитів SQL в якості API. Але в останні роки зростає популярність рішень, які базуються на NoSQL (нереляційних) СУБД. У майбутньому, цілком імовірно, що багато програм будуть використовувати комбінацію реляційних та нереляційних рішень для зберігання даних як окремих програм, так і пов'язаних між собою комплексів різнопланових програм і додатків.

JSON (JavaScript Object Notation): Зберігання і передача даних. Об'єктна нотація JavaScript (JSON) є відкритим стандартом, розробленим Дугласом Крокфордом. JSON являє собою підмножину синтаксису об'єктних літералів JavaScript для використання в представленні, передачі та зберіганні даних. До появи специфікації JSON більшість клієнт-серверних комунікацій здійснювалися за допомогою набагато багатослівніших фрагментів на мові розмітки XML. Розробники програм на JavaScript використовують багато веб-сервісів, які використовують нотацію JSON і часто визначають внутрішні дані за допомогою синтаксису JSON.

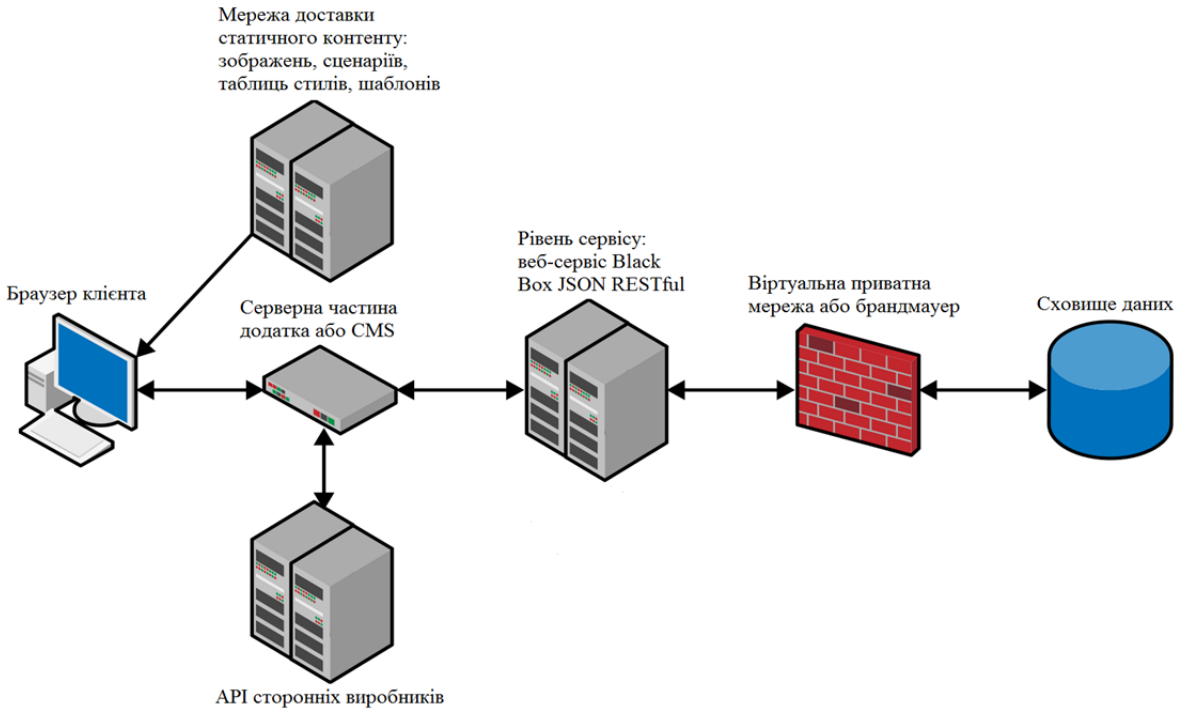

Рис. 1. Інфраструктура JavaScript-додатків

Розглянемо приклад повідомлення, яке описує колекцію книг:

```
[
    {
        "title" : "JavaScript: The Good Parts",
        "author" : "Douglas Crockford",
        "ISBN" : "0596517742"
    },

    {
        "title" : "JavaScript. Подробное
    руководство",
        "author" : "Дэвид Флэнаган",
        "ISBN" : "5-93286-103-7"
    },

    {
    "title" : "Мовні засоби системного
програмування",
    "author" : "Є. В. Щербаков, М. Є.
Щербакова",
    "ISBN" : "966-590-570-8"
    }
]
```

Як можна побачити, цей формат майже ідентичний синтаксису об'єктних літералів мови JavaScript з парою важливих відмінностей:

• Всі імена атрибутів та строкові значення повинні бути взяті в подвійні лапки. Інші значення можуть з'являтися в їх літеральній формі.

• JSON-записи не можуть містити циклічні посилання.

• JSON не може містити функції.

Сховища даних NoSQL. До появи розширеної мови розмітки XML (Extensible Markup Language) і сховищ даних JSON майже всі веб-сервіси підтримували реляційні системи управління базами даних (РСУБД). РСУБД містять дискретні точки даних в таблицях і групують дані, які виводяться, переглядаючи таблиці у відповідь на запити, написані на мові структурованих запитів SQL.

Навпаки, сховища даних NoSQL зберігають всі записи у формі документів чи фрагментів документів, не використовуючи заснованої на таблицях структурованої пам'яті. Сховища даних, орієнтовані на документи, зазвичай зберігають дані у форматі XML, в той час як об'єктно-орієнтовані сховища даних, як правило, використовують формат JSON. Останні особливо добре підходять для розробки веб-додатків, оскільки JSON-формат внутрішньо використовується для обмінів даними в мові JavaScript.

Приклади популярних сховищ даних NoSQL на основі JSON включають MongoDB та CouchDB. Незважаючи на теперішню популярність NoSQL, як і раніше часто можна знайти сучасні програми на мові JavaScript, які базуються на базах даних MySQL та аналогічних РСУБД.

Веб-сервіси RESTful JSON. Архітектура передачі станів REST (Representational State Transfer) є архітектурою комунікацій клієнт-сервер, яка забезпечує поділ сутностей між ресурсами даних та інтерфейсами користувача (або іншими споживачами інформаційних ресурсів, такими як інструменти аналізу даних та агрегатори). Сервіси, які реалізують архітектуру REST в повному обсязі, називаються RESTful. Сервер управляє ресурсами даних (такими, як користувацькі записи), але не реалізує і не включає в себе інтерфейс користувача. Клієнти можуть вільно реалізовувати користувальницький інтерфейс (або не реалізовувати) в будь-якій формі і на будь-якій мові. Архітектура REST не має справи з тим, як користувацькі інтерфейси реалізовані. Вона має справу тільки з підтриманням стану додатка між клієнтом та сервером.

Веб-сервіси RESTful використовують дієслова методів HTTP, щоб повідомити сервер, які дії має на увазі клієнт. Забезпечуються наступні дії:

• Створити новий елемент в колекції ресурсів: HTTP POST.

• Отримати представлення ресурсу: HTTP GET.

• Оновити (замінити) ресурс: HTTP PUT.

• Видалити ресурс: HTTP DELETE.

Це відповідає чотирьом базовим функціям інтерфейсу GRUD (create - створити, retrieve - отримати, update - оновити, delete - видалити), призначеного для роботи з персистентними сховищами даних. Просто потрібно пам'ятати, що у відображенні REST дія «оновити» насправді означає дію «замінити».

На рис. 2 представлений типовий потік передачі станів.

1. Клієнт запрошує дані з сервера за допомогою запиту HTTP GET, задаючи уніфікований індикатор ресурсу URI (uniform resource indicator). Кожен ресурс на сервері має унікальний URI.

2. Сервер отримує дані (зазвичай з бази даних або кеш-пам'яті) та упаковує їх в представлення, зручне для використання клієнтом.

3. Дані повертаються у вигляді документа. Ці документи, як правило, є текстовими рядками, які містять JSON-кодовані об'єкти; при цьому REST не цікавить те, як упаковані дані. Часто можна бачити RESTful-сервіси на основі XML. Найновіші сервіси за замовчуванням використовують JSON-відформатовані дані, а багато з них підтримують як формат представлення XML, так і формат представлення JSON.

4. Клієнт обробляє отримане представлення даних.

5. Клієнт виконує запит HTTP PUT до того ж URI, посилаючи назад оброблені дані.

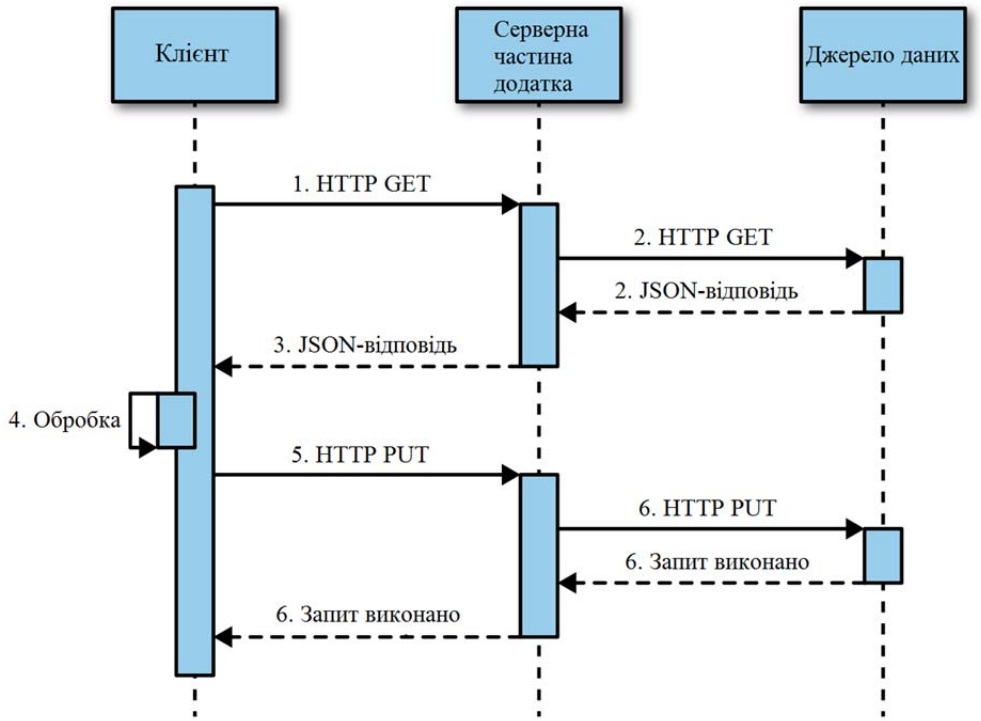

Рис. 2. Послідовність REST

6. Дані ресурсу на сервері замінюються даними, переданими в запиті HTTP PUT.

Зазвичай плутають, використовувати PUT чи POST для зміни ресурсу. REST полегшує вирішення цієї колізії. PUT використовується, якщо клієнт здатний генерувати свої власні безпечні ідентифікатори (IDs). В усіх інших випадках, щоб створити новий ресурс, виконується запит POST. В таких випадках сервер генерує ідентифікатор (ID) ресурсу і повертає його клієнту.

Наприклад, можна створити нового користувача, включивши /users/ в запит POST, внаслідок чого сервер згенерує унікальний ID, який потім можна використовувати для доступу до нового ресурсу в /users/userid. Сервер поверне нове представлення користувача зі своїм власним унікальним URI. Не можна модифікувати існуючий ресурс з запитом POST; можна тільки додати до нього нащадка.

Запит PUT потрібно використовувати, якщо треба змінити відображуване ім'я користувача, задавши /users/userid для запису користувача, який оновлюється. Треба зауважити, що цей запит повністю замінює запис користувача, тому потрібно переконатися, що представлення PUT містить все те, що необхідно.

Висновки. Синтаксис JavaScript зрозумілий всім, хто має досвід роботи з такими мовами, як C++, Java, C# або PHP. Частково популярність JavaScript можна пояснити її близькістю до цих мов, хоча важливо розуміти, що внутрішньо JavaScript реалізується зовсім інакше, ніж компілятори цих мов. JavaScript має дуже багаті об'єктно-орієнтовані можливості. Стандарт JSON, використовуваний майже у всіх сучасних веб-додатках як для комунікацій, так і для збереження даних, є підмножиною чудової нотації об'єктних літералів JavaScript. У сучасних браузерах використовується оперативна компіляція (just-in-time compiling), і більшість коду JavaScript - це відкомпільований, добре оптимізований і виконуваний як машинний код, і тому швидкість виконання близька до програм, написаних на C або C++. В даний час на JavaScript розробляються додатки, візуальні інтерфейси (UI's) яких перевершують візуальні інтерфейси (UI's) настільних додатків. JavaScript також просувається в світ технічних засобів. Такі проекти, як Arduino, Tessel, Espruino і NodeBots в найближчому майбутньому віщують час, в якому JavaScript може бути спільною мовою для вбудованих систем і робототехніки. Сучасні програми JavaScript є найбільш чутливими, найбільш соціально привабливими з коли-небудь написаних. Таким чином, розробники JavaScript знаходяться в центрі того, що може бути найбільшою революцією в історії обчислювальної техніки: на початку мережі Інтернет реального часу.

Література

1. Brown T. Jump Start HTML5 / Tiffany B. Brown, Kerry Butters, Sandeep Panda. - Collingwood : SitePoint Pty Ltd, 2014. – 313 p.
2. Flanagan D. JavaScript: The Definitive Guide, Sixth Edition / David Flanagan. - Sebastopol : O'Reilly Media, Inc., 2011. – 1098 p.
3. Revill L. jQuery 2.0 Development Cookbook / Leon

Revill. - Birmingham : Packt Publishing Ltd, 2014. – 410 p.
4. Elliott E. Programming JavaScript Applications / Eric Elliott. - Sebastopol : O'Reilly Media, Inc., 2014. – 253 p.

References

1. Brown T. Jump Start HTML5 / Tiffany B. Brown, Kerry Butters, Sandeep Panda. - Collingwood : SitePoint Pty Ltd, 2014. – 313 p.
2. Flanagan D. JavaScript: The Definitive Guide, Sixth Edition / David Flanagan. - Sebastopol : O'Reilly Media, Inc., 2011. – 1098 p.
3. Revill L. jQuery 2.0 Development Cookbook / Leon Revill. - Birmingham : Packt Publishing Ltd, 2014. – 410 p.
4. Elliott E. Programming JavaScript Applications / Eric Elliott. - Sebastopol : O'Reilly Media, Inc., 2014. – 253 p.

Щербакова М. Е., Щербаков Е. В. Функциональные особенности JavaScript-приложений.

Рассмотрены базовая структура и функциональные особенности приложений, разрабатываемых на языке программирования JavaScript с использованием языка гипертекстовой разметки HTML и каскадных таблиц стилей CSS. В то время, как каждое приложение является уникальным, большинство из них разделяют некоторые общие элементы, такие как инфраструктура хостинга, управление ресурсами, представление и поведение пользовательского интерфейса. Проанализированы особенности функционирования основных компонентов инфраструктуры приложения. Исследовано влияние на работу приложения таких элементов, как хранилище данных, веб-сервисы, серверная и клиентская части приложения, а также сеть доставки контента. Сформулированы основные критерии разработки на JavaScript и HTML чувствительных, социально значимых приложений с привлекательными визуальными интерфейсами.

Ключевые слова: *JavaScript, HTML, JSON, REST, Ajax, браузер, приложение, хранилище данных, виртуальная частная сеть.*

Shcherbakova M. E., Shcherbakov E. V. Functional features of JavaScript-application.

Considered the basic structure and functional features of applications developed in programming language JavaScript with hypertext markup language HTML and cascading style sheet CSS. At that time, as each application is unique, most of them share some common elements, such as infrastructure of hosting, resource management, performance and behavior of the user interface. W*ere analyzed features of the functioning of the main components of the application infrastructure. Was investigated influence of the application performance elements such as data storage, web services, server and client-side of the application, and content delivery network. Formulated basic criteria for the development in JavaScript and HTML sensitive, socially relevant applications with attractive visual interface.*

Keywords: *JavaScript, HTML, JSON, REST, Ajax, browser, application, data storage, virtual private network.*

Щербакова Марина Євгенівна – к.т.н., доцент, доцент кафедри комп'ютерної інженерії, Технологічний інститут Східноукраїнського національного університету імені Володимира Даля (м. Сєвєродонецьк), ms432@mail.ru
Щербаков Євген Васильович – к.т.н., доцент, доцент кафедри комп'ютерної інженерії, Технологічний інститут Східноукраїнського національного університету імені Володимира Даля (м. Сєвєродонецьк), gkvarc@gmail.com

Рецензент: **Суворін О. В. –** д.т.н., доцент

Стаття надана 19.11.2014