ТЕХНІЧНІ НАУКИ

УДК 004.45
DOI: 10.15587/2313-8416.2018.127118

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ НАВЧАЛЬНИХ WEB-ДОДАТКІВ ЗА РАХУНОК КЕШУВАННЯ

© В. П. Молчанов

У статті розглянуті питання організації обчислень на клієнтської частини навчальних WEB-додатків. Обґрунтовано зв'язок між ефективністю і автономністю WEB-додатків, призначених для навчання. Визначено особливості таких додатків і можливості щодо підвищення їх ефективності. Розглянуто вплив кешування. Сформульовано проблема, яка полягає в підвищенні ступеня автономності таких додатків. Пропонується вирішення проблеми з використанням нових API HTML5. Обрані технології і розглянуті їх особливості

***Ключові слова**: навчання на робочому місці, WEB-додатки, кеш ресурсів, API, сховище, дані, розподілені додатки*

1. Вступ

Для розробки додатків, що забезпечують навчання на робочому місці, пропонується багато спеціальних засобів і технологій (від Flash до Adobe Director). Однак орієнтація на використання мережі Інтернет та розвиток її базових засобів дозволяють розглядати ці базові засоби в якості альтернативи спеціалізованим.

Якщо мова йде про можливості базових засобів WEB-технології, то в контексті розробки e-learning, можна виділити два напрямки: можливості образотворчих засобів і можливості, що впливають на архітектуру (структуру) WEB-додатків. Специфічні вимоги до образотворчих засобів створення навчальних програм існують, і використання для цих цілей HTML, CSS і JS досить докладно розглянуті. А аналіз можливостей WEB-додатків і їх відповідності вимогам (потребам) з боку розробників засобів навчання висвітлені мало, тому дане дослідження представляють певний інтерес.

2. Аналіз літературних даних та постановка проблеми

У питанні ефективності процесу навчання з використанням електронних засобів основна увага приділяється педагогічним аспектам. Ці питання досить широко висвітлені в літературі як з точки зору особливостей створення в цілому [1, 2], так і відповідного інтерфейсу [3, 4]. Розглянуто також питання впливу їх використання на ефективність навчання [5, 6]. При цьому передбачається, що розглянуті методики і педагогічні сценарії можуть бути реалізовані в повній мірі [7, 8]. Робіт, які пов'язані з технічною реалізацією, набагато менше [9]. У той же час врахування особливостей використовуваних технологій і вибір організації взаємодії використовуваних засобів може вплинути на ефективність процесу навчання [10].

Таким чином, відкритим залишилося питання ефективної організації обчислюваній у WEB-додатках, які орієнтовані на навчання.

3. Мета та задачі дослідження

Метою дослідження є аналіз впливу організації взаємодії клієнта з сервером в WEB-додатках для навчання на роботу користувача.

Для досягнення поставленої мети були сформульовані наступні задачі:

– оцінка впливу ступеня автономності додатки на доступність ресурсу сервера;

– вибір технологій для надання автономності додаткам;

– аналіз результатів використання запропонованих рішень.

4. Матеріали дослідження організації автономної роботи клієнта розподіленого додатка

Для навчальних ресурсів мережі Інтернет важливою характеристикою є доступність. Серед багатьох чинників на це впливає можливість роботи при відсутності зв'язку з сервером, тобто в автономному режимі. Користувач ресурсу повинен мати можливість в будь-який момент перервати процес, а потім продовжити його без втрати даних. При цьому доступність ресурсу тим вище, чим менше залежність від сервера. Умовно можна назвати це ступенем автономності. Мова не йде про перехід до встановленого додатка, а саме про мінімізацію залежності від стану сервера.

Для забезпечення автономної роботи браузер при початку роботи повинен повідомити не тільки адресу першої запитаної сторінки, але і сторінки (файли), до яких можливе звернення в майбутньому, для їх попереднього скачування. Це можуть бути будь-які файли-HTML, JavaScript, зображення, відео. Після такого скачування використовувати ресурс можна і без підключення до Інтернету. Браузер буде використовувати збережені раніше файли.

Дані, що забезпечують підвищення автономності навчального додатку, повинні бути поміщені в спеціальний буфер (кеш). Цей кеш повинен бути доступний розробнику для управління його роботою. Таке управління має на увазі збереження файлів для подальшого використання, їх оновлення в міру необхідності, збереження даних для подальшої відправки на сервер (при появі такої можливості), і інші дії.

Таким чином проблема існує і її рішення може бути знайдено шляхом збереження різних даних (даних користувача і статичних ресурсів додатків) в пам'яті клієнта. Серед нових інтерфейсів, що розроблені в рамках стандартів HTML5 [2], для таких цілей можуть бути використані: Application Cache, WebStorage, serviceWorker і ряд інших. Зупинимося на перших двох, оскільки вони повною мірою підтримуються сучасними браузерами.

Збереження файлів WEB-додатків може бути реалізовано за допомогою API Application Cache специфікації HTML5. Цей API дозволяє використовувати і керувати кешуванням файлів. Розглянутий API – це багато функцій, що забезпечують управління кешуванням WEB-додатків, за допомогою яких можна використовувати попередньо отримані ресурси без підключення до мережі Інтернет. Дані, які поміщені в сховище Application Cache, видаляються по команді користувача або сервера, тобто знаходяться під контролем.

Якщо порівнювати це рішення зі стандартним кешем, то головна відмінність полягає в неможливості управляти процесом збереження та використання файлів. У Application Cache допускається поміщати файли будь-яких типів, що завантажуються з сервера відповідно до спеціальної інструкції (маніфест, manifest).

Маніфест – це текстовий файл з довільним ім'ям. Інформація про цей файл поміщається в якості значення атрибута manifest, наприклад, <html lang = "ru" manifest = "/offline.manifest">. Атрибут manifest повинен бути на кожній сторінці додатка, яка повинна міститися в кеш. Така сторінка буде безумовно поміщена в буфер при її відвідуванні. Браузер не зберігає сторінку, якщо вона не має атрибута manifest або якщо адреса сторінки на міститься в списку файлів маніфесту.

Файл може містити три секції. Перша, обов'язкова – (CACHE) містить список файлів, які необхідно зберегти для автономної роботи. Сторінки, що використовують ці файли, зберігаються автоматично при відвідуванні.

Друга – (FALLBACK) містить перелік сторінок з адресами перенаправлення, при відсутності копій в кеші. Цей додатковий розділ містить посилання на резервні ресурси, якщо основні недоступні.

Третя – (NETWORK) містить шлях до файлів, які не можуть використовуватися без підключення з Інтернету. Це ресурси, які вимагають обов'язкового підключення до сервера, всі запити до них йдуть повз кеша. При завданні імен можна використовувати шаблони (типу *.js).

Слід зазначити, що використання кеша для роботи вимагає спеціального налагодження сервера. Зокрема, при передачі файла маніфесту повинен бути встановлений MIME-тип text/cache-manifest.

Якщо при програмному оновленні файл маніфеста або один з ресурсів виявиться недоступним, триватиме використовуватися раніше створений.

Для програмної роботи з кешем використовуються властивості, методи і події об'єкта window.applicationCache.

У розглянутого рішення є і недоліки. Основний полягає в недостатній гнучкості управління при роботі з буфером. Схема використання та поновлення добре працює для простих (односторінкових) ресурсів, але для складних за структурою сайтів з періодично змінюваними сторінками можливі проблеми.

Однак в цілому, використання Application Cache API може не тільки збільшити швидкість завантаження сторінок користувачем і зробити його роботу більш комфортною при відсутності зв'язку з сервером, але і зменшити розмір трафіку і знизити навантаження на сервер (число звернень).

Доповнити цю можливість дозволяє технологія сховища даних, яка забезпечує збереження даних на час сесії (об'єкт sessionStorage) і між сесіями (об'єкт localStorage).

Реалізація логіки WEB-додатка на стороні клієнта багато в чому обмежувалася неможливістю збереження даних і їх повторного використання на машині користувача.

Такими даними можуть бути як налаштування середовища роботи учня, так і результати його роботи. Дані для збереження доводиться відправляти на сервер. У багатьох випадках таке рішення знижує в цілому ефективність розподіленого додатка. Зростає трафік, ростуть затримки, з'являються обмеження на використання додатка без підключення до сервера. Наприклад, до таких додатків відносяться багато навчальних ресурсів. Для них бажано зберігати персональні налаштування, результати роботи, а також забезпечити автономну роботу.

Розробниками запропоновано кілька технологій збереження даних на клієнті без пересилання до сервера. Наприклад, специфікація HTML5 пропонує три варіанти API: WebSQL, IndexedDB і WebStorage.

Технологія WebSQL орієнтована на використання веб-браузера для роботи з базою даних на основі Transact-SQL. Технологія IndexedDB (Indexed database) надає інтерфейс для використання індексованого ієрархічного сховища типу ключ-значення, має властивості бази даних.

Серед цих рішень на даний момент найбільш прийнятним і широко підтримуваним є WebStorage (API сховища DOM або WEB-сховище).

Інтерфейс DOM-сховищ дозволяє успішно вирішити завдання збереження даних без відправки їх сервера і таким чином підвищити ефективність фун-

кціонування програми. Можливість реалізована в більшості сучасних браузерів і забезпечує збереження і використання даних об'ємом близько 5–10 Мб.

Відповідно до цього API реалізовано два рівня зберігання даних: рівень сесії (sessionStorage) і локальний (localStorage). Обидва вони прив'язані до джерела і браузера. Джерело ідентифікується протоколом, ім'ям домену та портом. Таким чином, http://koms.org/index.htm і http://koms.org/index_1.htm мають одне джерело і, відповідно, використовують загальне сховище, а https://koms.org/index.htm і http://koms.org/index.htm – різні. Звідси випливає, що дані, які зберігаються, прив'язані до домену, з яким працює браузер. Те, що зберегли при роботі з конкретним додатком в конкретному браузері недоступно іншому браузері, на іншій машині, а також з сервера.

Рівень sessionStorage пов'язаний з відкритою вкладкою або вікном браузера і існує до їх закриття. Відкриття тієї ж сторінки в новому вікні браузера або новій вкладці призводить до створення нової сесії сторінки і, відповідно, нового сховища.

Рівень localStorage зберігається і в перервах між сесіями, прив'язаний до браузеру і домену, доступний з усіх сторінок домену.

Загальна схема роботи зі сховищем включає таку послідовність дій: перевірка підтримки API в браузері, оцінка доступної пам'яті, додавання або зміна даних, вилучення даних, очищення сховища. Можливі й інші дії відповідно до логіки додатка.

Таким чином, відповідний інтерфейс дає можливість встановлювати, редагувати і видаляти дані. Для сховищ кожного типу і по кожному домену використовується окреме подання Storage-об'єкта, вони функціонують і управляються окремо один від одного. Наприклад, можна зберігати індивідуальні призначені для користувача налаштування різних середовищ і відновлювати їх при поверненні користувача до роботи з ресурсом. Збережені дані доступні тільки сценаріями і не пересилаються по мережі.

5. Результати дослідження та їх обговорення

Перевірка роботи зі сховищами в різних браузерах на тестових завданнях показала стійку, надійну роботу, і дозволяє зробити ряд узагальнень.

Перше, необхідно враховувати – що це синхронний API, блокуючий основний потік призначеного для користувача інтерфейсу. Локальна пам'ять працює з жорстким диском, тому операції запису і зчитування можуть внести затримки при значних обсягах даних.

Друге, сховища мають обмежений розмір, причому він різний в різних браузерах. Необхідно постійно контролювати використання пам'яті і запобігати її переповнення.

Третє, використання пар ключ/значення не дозволяє застосовувати ефективні пошукові механізми (наприклад, SQL), що може негативно впливати на продуктивність при обробці великих обсягів даних. Всі механізми пошуку зводяться до перебору.

Ще одна особливість полягає в тому, що при автономній роботі браузера (відкриття сторінок з файлової системи без звернення до сервера хоча б локального) повноцінне функціонування сховища не забезпечується. Всі роботи з налагодження необхідно вести з сервером.

6. Висновки

За результатом досліджень можна зробити наступні висновки.

1. Запропонований підхід до організації обчислень на клієнті з використанням локальних сховищ і файлу маніфесту дозволяє підвищити ступінь автономності навчального додатка.

2. Дві розглянуті технології добре доповнюють одна одну і дозволяють підвищити комфорт роботи учня за рахунок більш високої автономності додатків та збільшення швидкості завантаження сторінок.

3. Крім того зменшується розмір трафіку і знижується навантаження на сервер.

Література

1. Кларин М. В. Инновации в обучении. Метафоры и модели. Москва: Наука, 1997. 223 с.

2. Березовський В. С., Стеценко І. В., Завадський І. О. Створення електронних навчальних ресурсів та онлайнове навчання. Київ: Видавнича група BHV, 2013. 176 с.

3. Pushkar O., Lepeyko T. Design of interactive visual tools in the computer multimedia education program (by the example of management disciplines) Yeditepe university. 4tn International Symposium of Interactive Media Design. 2006. Vol. 30. P. 117–125.

4. Буланова Т. В., Стародубцев В. А., Шамина О. Б. Педагогический дизайн информационной учебной среды // Проблемы информатики. 2012. № 5. С. 208–212.

5. Назарова О. Л. Новые информационные технологии в управлении качеством образовательного процесса в колледже // Информатика и образование. 2003. № 11. С. 79–84.

6. Курдицкая О. С. Оценка эффективности обучения в высокотехнологичных информационных компаніях: мат. VIII Міжнар. наук.-пр. конф. // Теорія і практика сучасної економіки. Черкаси: ЧДТУ, 2007. С. 298–300.

7. Абрамов О. М. Про становлення, розвиток та взаємозв'язок стандартів та специфікацій електронного навчання (e-learning) // Вісник Харківської державної академії культури. 2012. № 37. С. 284–293.

8. Дистанционное обучение: теория и практика / Гриценко В. И. и др. Киев: Наукова думка, 2004. 375 с.

9. Куклев В. А. Электронное обучение с помощью мобильных устройств в любое время и в любом месте: монография. Ульяновск: УлГТУ, 2009. 356 с.

10. Молчанов В. Анализ реализации новых WEB-стандартов в массовом программном обеспечении: зб. наук. пр. // Системи обробки інформації. 2016. № 4 (141). С. 226–228.

Рекомендовано до публікації д-р техн. наук, професор Білоусов В. В.
Дата надходження рукопису 01.02.2018

Молчанов Віктор Петрович, кандидат технічних наук, доцент, кафедра комп'ютерних систем і технологій, Харківський національний економічний університет імені Семена Кузнеця, пр. Науки, 9-а, м. Харків, Україна, 61166

E-mail: Viktor.Molchanov@hneu.net