УДК 378:004.032.8

© **Гринько Т. Ю**., студентка 2 курсу факультету економічної інформатики, Харківський національний економічний університет імені Семена Кузнеця, Харків, Україна

ТЕХНОЛОГІЯ ФОРМУВАННЯ ІНТЕРАКТИВНИХ МУЛЬТИМЕДІЙНИХ КОМПЛЕКСІВ

The paper deals with the development of the technology of forming interactive multimedia complexes with the means of interactivity. The design is proposed, the choice of logo is justified. Developed interactive elements.

Створення різноманітних мультимедійних ресурсів вимагає чітко прописаної та ретельно обґрунтованої технології, за якою має відбуватися формування як оболонки, так і контенту.

На сьогодні в спеціалізованій літературі [1-5] є численні рекомендації відносно того, яким має бути мультимедійний комплекс, а також пропонуються різні варіанти розробки засобів інтерактивності мультимедійних ресурсів.

Проте у розглянутій літературі відсутній опис технології створення інтерактивних мультимедійних комплексів, на основі якої користувач зможе покроково сформувати необхідний контент.

Метою даної роботи є розробка технології формування інтерактивних мультимедійних комплексів із засобами інтерактивності.

Основними етапами пропонованої технології є наступні.

Етап 1. Розробка схеми інтерфейсу.

Додаток є групою файлів в підпапках. Файл index.html вважається виконуваним – для запуску програми необхідно відкрити цей файл у будь-якому сучасному веб-браузері. Після запуску файлу виконується програмний сценарій, написаний на мові javascript, що завантажує індекс статей («Зміст») і деяку вітальну інформацію. Користувачеві пропонується за допомогою миші вибрати одну із статей змісту, після чого користувач повинен побачити текст відповідної статті (рис.1)

Рис. 1. Схема інтерфейсу додатку

Кожна стаття складається з наступних структурних елементів:

1) блоку заголовка і короткого опису. У цьому блоці відображається розглянута в статті тема і її короткий опис;

2) блоків тексту. У блоках тексту міститься власне матеріал статті;

3) блоків коду. Блоки коду представляють собою приклади коду, що не можуть бути редагованими;

4) блоків, що редагуються. Редагований блок складається з візуального редактора (у лівій частині) і результуючого коду (у правій частині). Користувачеві пропонується спробувати змінювати як код, так і вміст візуального редактора. Код і візуальний редактор автоматично синхронізуються (див. рис. 2)

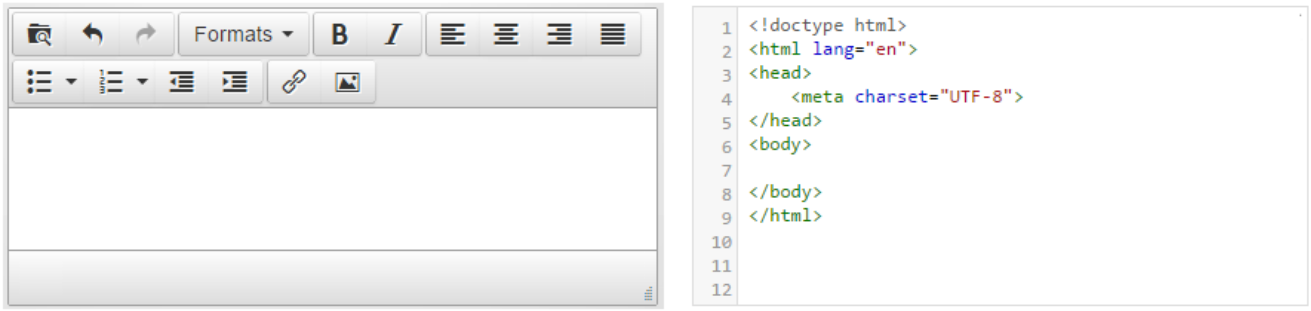

Рис. 2. Блок, що редагується

Додаток дозволяє розташовувати вищезазначені елементи в довільному порядку, завдяки чому у авторів навчального матеріалу є багато можливостей визначення зручного способу подачі цього матеріалу.

Етап 2. Розробка дизайну.

Для додатку був обраний консервативний стиль. Головною рисою стилю є «виразний мінімалізм». Скруглені рамки, м'які кольори, шрифти без зарубок. Стиль характеризують великі відступи блоків з вмістом, що робить візуальне оформлення повітряним (рис.3).

Консервативний стиль гарантує інтуїтивну простоту сприйняття інтерфейсу людьми різного віку та уподобань. Поряд з цим мінімалізм дозволяє створити у користувача «розслаблено-уважний» настрій, сприятливий для засвоєння знань, викладених у мультимедійному посібнику, а скруглені кути керуючих елементів роблять дизайн не нудним.

Рис. 3. Колірна гамма веб-додатку

Структура розташування елементів управління додатку представлена на рис. 4. Так як мультимедійний посібник має інформаційно — тематичний характер, вміст кожної сторінки — основний елемент дизайну, інші елементи на другому плані.

<table>
<tr><td>Логотип</td><td rowspan="2">Основний вміст</td></tr>
<tr><td>Меню навігації</td></tr>
<tr><td colspan="2">(c)</td></tr>
</table>

Рис. 4. Структурна схема розташування елементів на сторінці

Розробка дизайну здійснювалася в програмі Adobe Photoshop CS3 (рис. 5). Незважаючи на те, що спочатку програма була розроблена як редактор зображень для поліграфії, в даний час вона широко використовується і у веб-дизайні. У більш ранній версії була включена спеціальна програма для цих цілей — Adobe ImageReady, яка була виключена з версії CS3 за рахунок інтеграції її функцій в сам Photoshop, а також включення в лінійку програмних продуктів Adobe Fireworks, що перейшло у власність Adobe після придбання компанії Macromedia.

Photoshop забезпечує засоби нелінійного монтажу і створення таких спецефектів, як фони, текстури і т.п. Для телебачення, кінематографа і всесвітньої павутини. Photoshop також дуже популярний в колах розробників комп'ютерних ігор.

Через високу популярність Photoshop підтримка специфічного для неї формату PSD була реалізована в багатьох графічних програмах, таких як Macromedia Fireworks, Corel PHOTO-PAINT, WinImages, GIMP, Corel Paint Shop Pro та інших.

Photoshop підтримує такі колірні моделі або способи опису кольорів зображення (в нотації самої програми - режим зображення):

RGB;
LAB;
CMYK;
градації сірого;
чорно-білі;
Duotone;
з 256-колірной палітрою (Indexed);
багатоканальні (Multichannel).

Підтримується обробка зображень з глибиною кольору 8 біт (256 градацій на один канал), 16 біт (використовується 15 бітів плюс один рівень, тобто 32769 рівнів) і 32 біт (використовуються числа одинарної точності з плаваючою комою). Можливе збереження у файлі додаткових елементів, як то: напрямних (Guide), каналів (наприклад, каналу прозорості — Alpha channel), шляхів обтравки (Clipping path), шарів, що містять векторні і текстові об'єкти. Файл може включати колірні профілі (ICC), функції перетворення кольору (transfer functions). Допускаються неквадратні пікселі (Pixel Aspect Ratio). Завдяки всім цим можливостям я вважаю Adobe Photoshop незамінною програмою при розробці веб-дизайну.

В першу чергу був розроблений шаблон сайту. Верстка здійснена за допомогою DIV-ів на фреймворку Bootstrap. Кожен елемент навігації повторюється на кожному вигляді додатку. Поведінка DIV-ів, а також інших елементів, що знаходяться на сторінках, задається у файлі style.css.

В якості логотипу обрано книгу, бо книги завжди є нашою першою асоціацією з освітою (рис. 5). Виділення інформації досягається за допомогою кольорів та кеглів шрифтів.

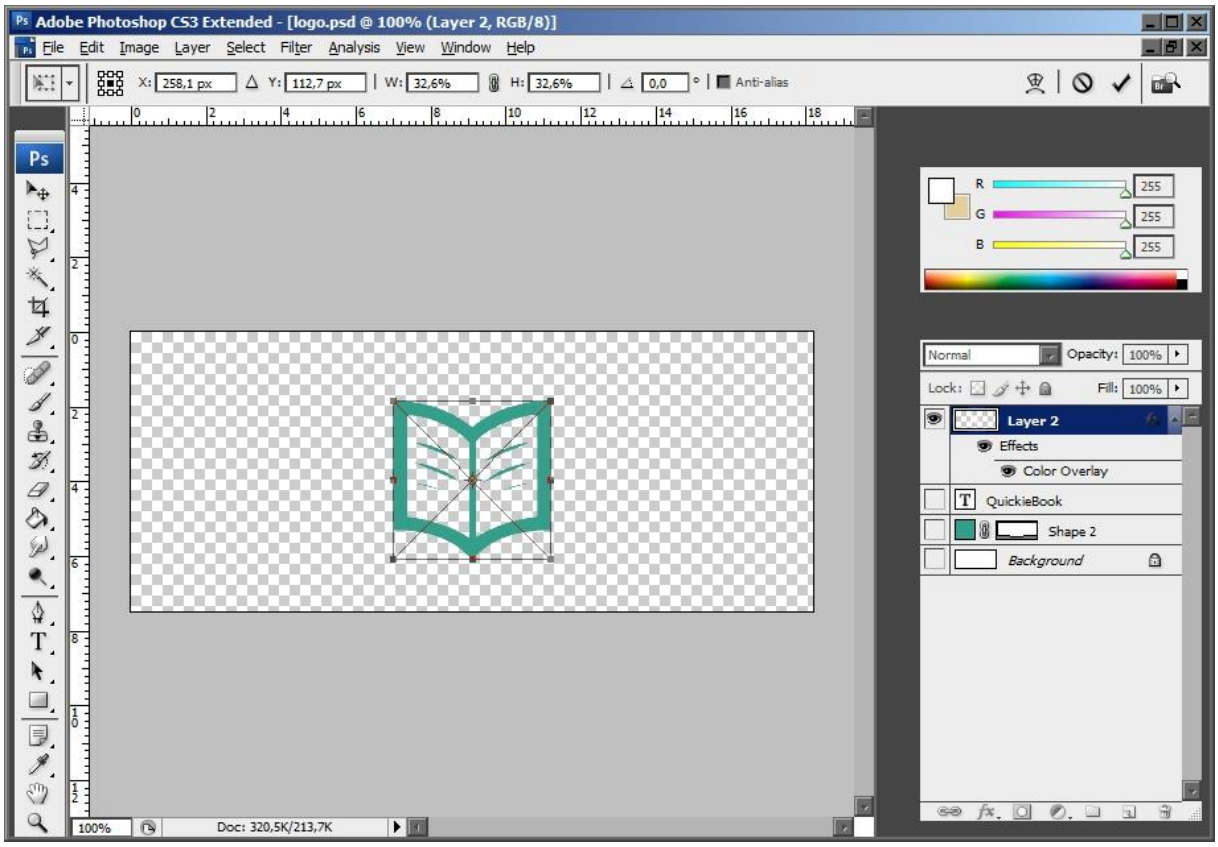

Рис. 5. Розробка логотипу

Основний знак логотипу (книга) було розроблено із застосуванням інструменту «криві» програмного комплексу Adob Illustrator CS3.

Етап 3. Розробка інтерактивних елементів

Завантаження файлів.

Через те, що передбачається, що додаток буде запускатися з електронного носія (а не з веб-сервера), було необхідно реалізувати можливість підвантаження файлів прямо з папки приложени. З усіх можливих варіантів реалізації цього підходу для завантаження даних був обраний формат JSONp. Принцип однакового джерела (англ. Same origin policy) — це важлива концепція безпеки для деяких мов програмування на стороні клієнта, таких як JavaScript. Політика дозволяє сценаріями, що знаходяться на сторінках одного сайту, доступ до методів і властивостей один одного без обмежень, але запобігає доступ до більшості методів і властивостей для сторінок на різних сайтах. Однакові джерела — це джерела, у яких співпадають три ознаки:

домен;

порт;

протокол.

Для простоти розгортання, додаток передбачається запускати поза веб-сервера. Тому у сценаріях додатку немає і не може бути ні домену, ні порту, ні протоколу (в браузері вони відображаються з псевдо-протоколом file://). У зв'язку з цим звичайні AJAX-запити до даних неможливі — всі сучасні браузери блокують такі запити як потенційно небезпечні.

JSONP (англ. JSON with padding, «JSON з підкладкою») є розширенням JSON, коли ім'я функції зворотного виклику вказується як вхідний аргумент. В основу технології покладено той факт, що політика безпеки браузера дозволяє використовувати тег

<script type="text/javascript" src="..."></script>

для звернення до сторонніх доменів і, у нашому випадку, до файлів на электронному носії.

Спочатку ідея була запропонована в блозі MacPython в 2005 році, і зараз використовується багатьма Web 2.0 застосунками, такими, як Dojo Toolkit Applications, Google Toolkit Applications, Kendo UI і zanox Web Services. Без використання технології JSONP (тобто використовуючи просто JSON кодування даних) сервер може повернути тільки дані. Наприклад так:

{"type": "employee", "count": 15}

Однак це лише дані та вони не можуть впливати на браузер. Використовуючи JSONP, сценарію передається в рядку виклику (GET) ім'я callback функції:

<script type="text/javascript" src="getjson?jsonp=parseResponse"></script>

Тут параметр jsonp містить ім'я callback функції parseResponse. Тепер сторонній сервер example.com може повернути такий код:

parseResponse({"type": "employee", "count": 15})

Тепер код викликає javascript-функцію першого домену.

Оскільки JSONP використовує скрипт-теги, виклики відкриті світу. З цієї причини, JSONP може бути недоречними для зберігання конфіденційних даних. Включення скриптових тегів від віддалених сайтів дозволяє їм передати будь-який контент на сайті. Якщо віддалений сайт має вразливості, які дозволяють виконати Javascript ін'єкції, то початковий сайт також може зачеплений ними.

У нашому випадку в додатку не використовується ніяких скриптових мов крім JavaScript. Перевага нашого застосування в даному випадку полягає в тому, що немає необхідності в зберіганні даних користувача, а все інформаційне наповнення сайту може бути підготовлено заздалегідь.

У нашому випадку всі дані програми зберігаються в численних файлах .js, які динамічно підключаються до додатка і на підставі даних з яких будується вміст веб-додатки.

Вся логіка додатки представлена файлом app.js і файлами розширень (editor.js і code.js), також завантажуються динамічно.

Шаблон Model-View-ViewModel.

Архітектура програми побудована за шаблоном MMVM. Шаблон архітектури MMVM спочатку був представлений спільноті Джоном Госсманом в 2005 році як модифікація шаблону Presentation Model. MVVM орієнтований на сучасні платформи розробки, такі як Windows Presentation Foundation, Silverlight від компанії Microsoft, ZK framework.

MVVM використовується для розділення моделі і її уявлення, що необхідно для зміни їх окремо один від одного. Наприклад, розробник задає логіку роботи з даними, а дизайнер відповідно працює з інтерфейсом. MVVM зручно використовувати замість класичного MVC і йому подібних в тих випадках, коли в платформі, на якій ведеться розробка, присутній «зв'язування даних».

У шаблонах проектування MVC / MVP зміни в інтерфейсі не впливають безпосередньо на Модель, а попередньо йдуть через Контролер або Presenter. У таких технологіях як WPF і Silverlight є концепція «зв'язування даних», що дозволяє пов'язувати дані з візуальними елементами в обидві сторони. Отже, при використанні цього прийому застосування моделі MVC стає вкрай незручним через те, що прив'язка даних до подання безпосередньо не вкладається в концепцію MVC / MVP.

Шаблон MVVM ділиться на три частини:

Модель (англ. Model), так само, як у класичній MVC, Модель являє собою фундаментальні дані, необхідні для роботи програми.

Подання (англ. View) — це графічний інтерфейс, тобто вікно, кнопки і т.п. Вистава є передплатником на подію зміни значень властивостей або команд, що надаються Моделлю уявлення. У випадку, якщо в Моделі представлення змінилося якесь властивість, то вона сповіщає всіх передплатників про це, і Уявлення, в свою чергу, запрошувати оновлене

значення властивості з Моделі представлення. У випадку, якщо користувач впливає на який-небудь елемент інтерфейсу, Подання викликає відповідну команду, надану Моделлю Подання.

Модель Подання (англ. ViewModel) є, з одного боку, абстракцією Подання, а з іншого, надає обгортку даних з Моделі, які підлягають скріпленню. Тобто, вона містить Модель, яка перетворена до Поданню, а також містить у собі команди, якими може користуватися Подання, щоб впливати на Модель.

В додатку, що розробляється шаблон MMVM взятий за основу при проектуванні архітектури. В якості вистави використовується основний файл програми, в якості моделі даних - функція подгрузки JSONp-файлів, в якості моделі представлення - функція, що реагує на зміну якірної посилання

Якорне посилання

Для реалізації можливості запам'ятовування стану програми були використані якірні посилання. Кожній одиниці даних (статті) у додатку відповідає якірна посилання. При зміні якірної посилання на сторінці додатка виконується функція, яка намагається завантажити відповідний імені якірної посилання файл даних і побудувати на його основі сторінку (рис.6)

```
$(window).on('hashchange',function(a){
    var location = window.location.hash.substr(1);
    page.load(location);
});
```

Рис. 6. Фрагмент коду з якорним посиланням

Ініціалізація додатка відбувається в кілька етапів

Завантажується файл config.js, шлях до якого записаний в основному файлі програми. Цей файл зберігає шляху до файлів і імена файлів програми. У майбутньому, завдяки чіткому завданням місця розташування компонентів програми на диску, можна буде з легкістю змінювати структуру файлів і директорій, а також створювати кілька варіантів однієї програми, перемикатися між якими можна буде всього лише змінюючи одну єдину строчку налаштувань.

Завантажується файл мови. Текст всіх системних повідомлень зберігається в окремому файлі, lang.js. На підставі ключів об'єкта translate в цьому файлі проводиться пошук елементів зі значенням атрибута data-translate, рівному ключу, і, якщо такі елементи знайдені, їх текст замінюється значенням за даним ключу.

Завантажуються розширення з папки plugins. Імена завантажуваних розширень задані у файлі config.js. Розширення - це обробники блоків даних. Завдяки реалізації розширень в майбутньому можна буде з легкістю додати нові блоки даних та їх обробники, не зачіпаючи при цьому основного функціоналу програми.

З файлу toc.js завантажується і будується зміст (бічне меню). Генеруються пункти меню, кожен з яких є anchor-посиланням.

Завантажується перша сторінка з перерахованих у файлі змісту.

Далі, при змінені anchor-посилання відбувається динамічне завантаження даних з файлів сторінок і побудова на їх підставі блоків інформації. В даний момент реалізовано три типи блоків інформації:

Блоки абзаців текста (p). Мають наступний формат (рис.7).

```
{
    type: 'p',
    data: [
        'Це - абзац тексту',
        'Це - інший абзац'

    ]
},
```

Рис. 7. Фрагмент коду з блоками абзаців текста

Блоки редактора (editor). Ці блоки дозволяють користувачеві експериментувати з розміткою. Вони поєднують у собі два редактори, TinyMCE і CodeMirror, що синхронізуються між собою (рис.8).

```
{
    type: 'editor',
    data: {
        value:"Текст у редакторі",
        tinymce:{
            // Тут задається властивості редактора TinyMCE.
            // наприклад:
            valid_elements: "b,b/strong,p,i/em,em,u,br",
            toolbar: "bold italic underline"
        }
    }
},
```

Рис. 8. Фрагмент коду з блоками редактора

Блоки коду (code). Дозволяють виводити користувачеві блоки з прикладами коду з підсвіткою синтаксису. Мають наступний формат (рис.9)

```
{
    type:'code',
    data:{
        value: "Будь-який код",
        lang:'html'
    }

},
```

Рис. 9. Фрагмент коду з блоками коду

Слід зауважити, що розробка велася з оглядом на можливість подальшої роботи доданка в якості веб-сервіса. Завдяки використаним бібліотекам та фреймворкам доданок пристосований для роботи у браузерах Microsoft Internet Explorer (починаючи з версії 8, яку було встановлено у ОС Windows 7 за замовченням і яка й понині (на початок 2016 року) займає 8% ринку за даними netmarketshare.com), Google Chrome, Mozilla Firefox та інших.

Також додаток працює в середовищах Android та iOS (у розрахунку на використання ним з планшетних ПК) (додаток А).

Таким чином, розроблений додаток, завдяки технології JSONp, може працювати в веб-браузері, при цьому не залежачи від веб-сервера (за псевдопротоколом file: //, що дозволяє браузеру обмежений доступ до файлової системи).

Дизайн додатка відповідає всім естетичним і ергономічним нормам, будучи приємним зовні і зручним у використанні. Ненав'язлива колірна гамма і друкарська система фреймворку bootstrap забезпечують зручність і легкість читання і сприйняття дидактичного матеріалу.

Підготовка дидактичного матеріалу до публікації за допомогою додатка досить складна, але це не є проблемою з огляду на орієнтованості додатки на кінцевого користувача і того, що підготовка дидактичного матеріалу повинна забезпечуватися кваліфікованим персоналом. Варто зауважити, що в майбутньому можлива автоматизація процесу інтеграції дидактичного матеріалу в інфраструктуру розробленого в рамках даної роботи мультимедійного навчального комплексу – при виникненні необхідності інеграціі в додаток великої кількості навчального матеріалу а також подальшого доопрацювання додатка з метою публікціі матеріалу в мережі Інтернет.

Перелік посилань:

1. N. I. Aralova and O. Y. Kyiashko,“The Method of Technology Evaluation Based on Improved Cost Approach” // Science and innovation. 2017. Vol. 13, Issue 3. pp. 65–76. DOI: 10.15407/scine13.03.065

2. R. Hryshchuk and K. Molodetska,“Synergetic control of social networking services actors’ interactions”. Recent Advances in Systems, Control and Information Technology. Vol. 543,pp. 34–42, 2017. DOI: 10.1007/978-3-319-48923-0_5

3. P. V. MartinsandM. A Zacarias,“Web-based Tool for Business Process Improvement”. International Journal of Web Portals. 2017. Vol. 9, Issue 2. pp. 68–84. DOI: 10.4018/ijwp.2017070104

4. C. Hu, Y. Zhao,andM. Guo,“AHP and CA Based Evaluation of Website Information Service Quality: An Empirical Study on HighTech Industry Information Center Web Portals”. Journal of Service Science and Management. 2009. Vol. 02, Issue 03. pp. 168–180. DOI: 10.4236/jssm.2009.23020

5. I. Safonov, Adaptive Image Processing Algorithms for Printing Springer, 2018. 304 p. DOI: 10.1007/978-981-10-6931-4