МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПЗВО «МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ
імені ПИЛИПА ОРЛИКА»
Економіко-технологічний факультет
Кафедра інженерних технологій

Кваліфікаційна робота
на здобуття освітнього ступеня магістра
за освітньою програмою «Комп'ютерна інженерія»
зі спеціальності 123 «Комп'ютерна інженерія»

на тему: **«ДОСЛІДЖЕННЯ ТА РОЗРОБКА ЕЛЕКТРОННОГО НАВЧАЛЬНОГО КУРСУ З JQUERY»**

Виконав:
здобувач ІІ курсу, групи КІ -20-24
Бондаренко Сергій Григорович

Керівник:
к.ф-м.н., доцент кафедри інженерних технологій
Арамян Армен Мартикович

Миколаїв – 2024

Зміст

ВСТУП

Веб-програмування на сьогоднішній день є однією із таких частин галузі інформаційних технологій (IT), що найбільш динамічно розвиваються та приносять значні прибутки відповідним компаніям. Традиційно увесь комплекс підходів, методів та засобів веб-програмування ділиться на back-end та front-end складові. Перша частина включає в себе програмування серверу (тобто того комп'ютеру, який знаходиться далеко від користувача, далеко позаду від його монітору – звідси з'являється слово «back»), друга – створення програмної компоненти, що працює на комп'ютері користувача мережі Інтернет (клієнтська частина, яка знаходиться безпосередньо перед користувачем, передня частина програмного комплексу – звідси «front»).

Комплекс засобів back-розробки відрізняється надзвичайним різноманіттям і може включати десятки мов програмування, якими допускається писати серверний програмний код, та багато різних технологій, за допомогою яких цей код можна примусити працювати (наприклад, технологія CGI і будь-яка мова програмування загального призначення, технології ASP, JSP, PHP, сервлети Java, JavaScript на основі Node.JS, і т.д., і т.п.). На відміну від першої частини, front-end розробка практично немає особливих альтернатив своєї реалізації, оскільки після відмирання мови VB.Script мова JavaScript стала єдиним засобом для програмування вікна браузера, тобто клієнтської частини усіх веб-додатків. Таким чином, усі аспекти програмування мовою JavaScript автоматично стають надзвичайно важливими і популярними в середовищі веб-програмістів. Не виключенням є і бібліотека jQuery, якій присвячено усе дане дослідження.

Як і будь-яка бібліотека (або, по-іншому, фреймворк; про питання чим конкретно вважати jQuery – бібліотекою, чи фреймворком – створено багато дискусійних публікацій, але до спільної думки програмісти так і не дійшли), jQuery є спробою об'єднання в рамках одного файлу (в даному випадку .js

файлу) усіх найбільш часто вживаних програмістами (типових) ділянок коду, написаних мовою JavaScript.

Відмітимо, що використання бібліотеки jQuery (як і інших бібліотек в цілому) не є цілком обов'язковим при вирішенні абсолютно будь-якої задачі front-end програмування – усе можна зробити засобами чистого JavaScript (pure JS), тому деякі програмісти принципово не використовують jQuery. Однак, суттєва частка розробників із задоволенням користуються можливостями, що надає ця програмна система, адже при цьому досягається економія обсягів висхідного тексту скриптів, тому використання бібліотеки безперечно є досить широким, а, отже, створення навчальних курсів із її вивчення є **актуальною** задачею викладача предметів галузі комп'ютерних наук.

В сучасному світі інформаційних технологій (далі – ІТ) практично усі процеси діяльності людини підлягають автоматизації, особливо такі, що мають складну структуру, є багатоступеневими. Навчання якраз є прикладом процесу, що добре автоматизується, і це втілюється у концепції електронного навчання. При цьому окремі дисципліни можна вивчати в рамках окремих крупних логічно замкнених одиниць, що називають електронними навчальними курсами. Отже, розробка саме навчальних курсів в електронному вигляді для вивчення запланованих дисциплін і є саме тим способом, яким може реалізовуватися сформульована вище **актуальна** задача викладача ІТ.

Виходячи з наведених міркувань, можна сформулювати наступну **мету** даного дослідження: покращити показники якості процесу вивчення популярної бібліотеки jQuery (зручність, швидкість, результативність, і т.п.) шляхом розробки та впровадження відповідного електронного навчального курсу.

Для досягнення поставленої мети слід вирішити наступні **завдання** дослідження:

- встановити особливості (зокрема, переваги та недоліки) вивчення матеріалу в електронному вигляді;

- знайти та проаналізувати особливості уже існуючих рішень-аналогів, за допомогою яких можна здійснювати вивчення бібліотеки jQuery;

- виділити недоліки, негативні риси існуючих аналогів;

- проаналізувати доступні технології та засоби розробки для створення електронних навчальних курсів;

- розробити методичні матеріали для вивчення курсу по бібліотеці jQuery;

- здійснити проектування та розробку електронного навчального курсу у вигляді програмного рішення, побудованого за обраними технологіями та за допомогою обраних засобів;

- виконати тестування та узагальнити результати роботи системи, зробити висновки по роботі.

Об’єктом дослідження виступає процес вивчення бібліотеки jQuery, розробленої для мови програмування JavaScript.

Предметом дослідження є електронний навчальний курс, за яким можна здійснювати ефективне вивчення бібліотеки jQuery.

За умови якісного виконання поставлених завдань кінцевим результатом роботи має стати електронний навчальний курс по вивченню бібліотеки jQuery, який може бути використаний в наступних випадках:

- при вивченні власне бібліотеки jQuery в рамках окремого курсу – при глибокому цілеспрямованому вивченні різноманітних аспектів front-end розробки;

- при вивченні теми jQuery в рамках курсу JavaScript – при отриманні системної освіти з IT;

- під час виконання проектів, що активно використовують jQuery, наприклад, відповідно до технічного завдання, - при здійсненні професійної діяльності програмістами-розробниками;

- при самоосвіті – в домашніх умовах;

- у будь-яких інших випадках.

Таким чином, створюваний проект має велике практичне значення і, за умови якісної реалізації, може бути впроваджений у навчальний процес різноманітних фізичних та юридичних осіб.

Розглянувши вхідні дані по роботі, можна переходити до процесу докладного аналізу та проектування.

Розділ 1. ТЕОРЕТИЧНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Теоретичні основи та прикладні аспекти сучасного веб-програмування

Важко уявити життя сучасної людини без мережі Інтернет. В першу чергу, це стосується молоді, але і серед людей старшого віку (особливо, із високим рівнем освіти та/або достатку) відсоток користувачів глобальної мережі весь час зростає. Під Інтернетом у строгому розумінні цього слова (спеціалістами) мається на увазі безпосередньо комп'ютерна мережа (що охоплює практично увесь світ та до якої підключено практично усі існуючі комп'ютери), але обиватели вважають по-іншому. Широка громадськість під Інтернетом часто розуміє, імовірно, найважливішу його частину – Всесвітнє павутиння World Wide Web, або скорочено WWW.

На відміну від інших компонентів Інтернету (хоча б тієї ж електронної пошти) WWW працює за принципами клієнт-серверної архітектури, а це визначає суттєві відмінності програмного забезпечення, призначеного для WWW, від іншого, зокрема, настільного програмного забезпечення (далі – ПЗ) – рис. 1.1.

Цей спосіб організації ПЗ забезпечується шляхом створення комплексного програмного продукту, що мінімально складається з двох частин: клієнтської та серверної (у дуже складних за структурою програмних продуктах можливо введення і більшої кількості рівнів, ніж два, при якому програмна компонента одночасно може виступати сервером для іншої програмної компоненти, ближчої до користувача, і клієнтом для ще одного програмного сервера).

Клієнтською виступає програмна складова (окрема програма), яка запитує інформацію, а серверною (або просто сервером) є та програма, що надає інформацію. Для видачі цієї інформації відповідно до запиту, сервер задіє власні ресурси, яких немає у клієнта:

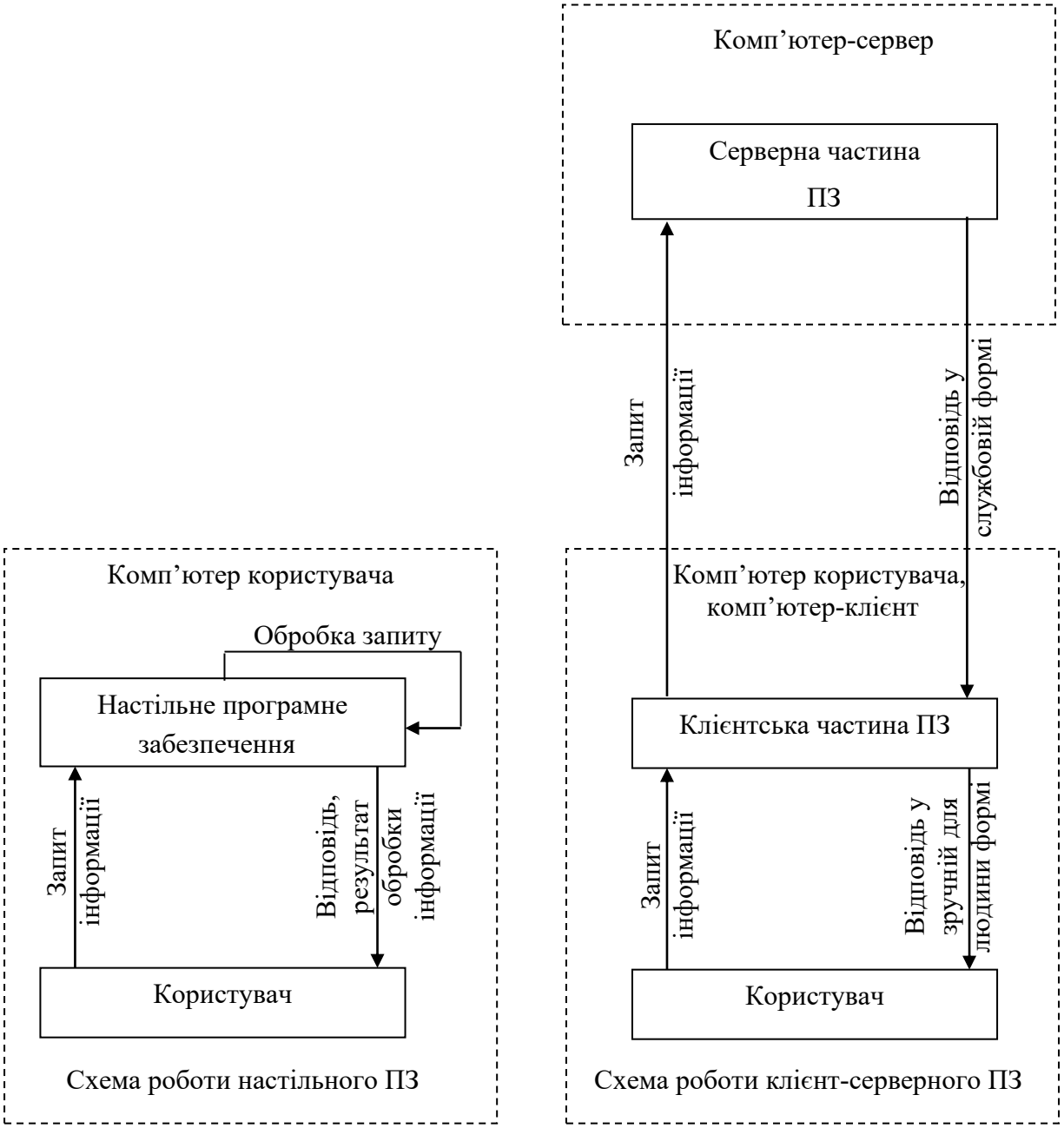


Рис. 1.1. Порівняльна схема роботи настільного програмного забезпечення та розподіленого, що має клієнт-серверну організацію.

- інформаційні (комп'ютери-сервери зазвичай мають величезні обсяги дискового простору і можуть зберігати значні обсяги інформації);

- обчислювальні (комп'ютери-сервери звичайно самі мають високу продуктивність, а, крім того, часто виступають частиною великих кластерів, ресурси яких можуть залучати при наявності відповідних дозволів);

- комунікаційні (звичайний сервер має, як мінімум декілька підключень до мережі Інтернет та безпосереднє підключення десятків, сотень, а то і тисяч клієнтів);

- алгоритмічні (з метою захисту інформації деякі секретні алгоритми не надаються клієнтам, а працюють лише на сервері, який очікує на вхідні дані від клієнта, а потім надсилає тому кінцевий результат обробки інформації);

- тощо.

Сама архітектура «клієнт-сервер» визначає особливості програмного забезпечення, побудованого по цій схемі: воно розбивається на дві окремих програми, що спілкуються між собою по спеціальним протоколам, які визначаються розробниками розподіленого ПЗ. Одна частина відповідає за інтерфейс користувача, тобто збирає від нього вхідну інформацію, що повністю визначає задачу по обробці даних, яка підлягає вирішенню. Вона ж видає у зручній для людини формі результати обробки інформації. Основну роботу по перетворенню інформації виконує серверна частина, яка і містить усі основні змістові алгоритми.

1.2. Місце jQuery в структурі комплексу сучасних засобів веб-програмування

Під терміном «веб-програмування» на сьогоднішній день можуть матися на увазі десятки різних мов програмування і навіть кардинально різних підходів до програмування однією і тією ж мовою. Відповідно, виникає потреба у класифікації засобів та мов програмування, визначенні структури всього комплексу засобів веб-програмування. Укрупнено ця структура показана на рис. 1.2 і включає самі основні на сьогодні елементи, в той час як десятки інших, менш поширених не включені до схеми.

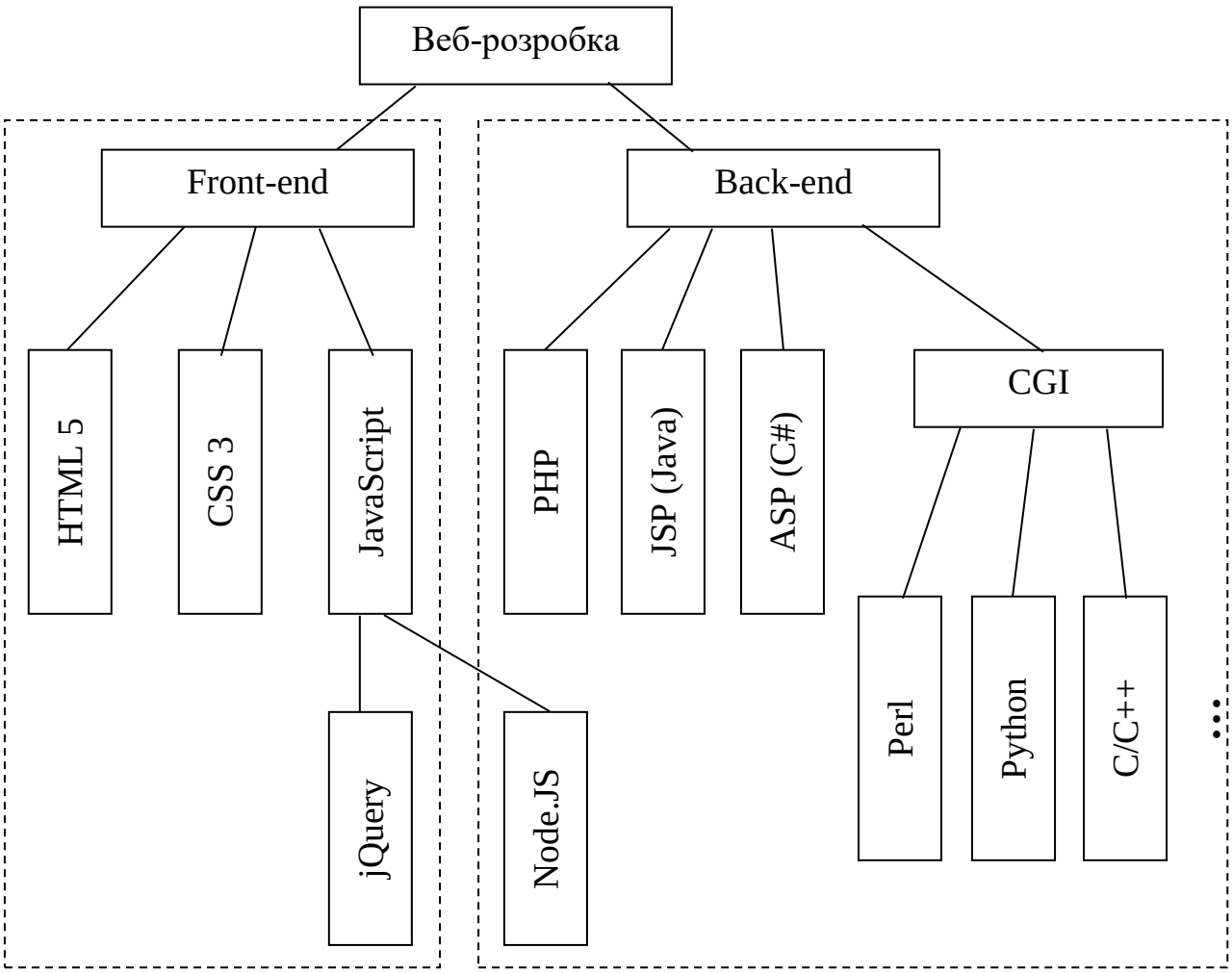


Рис. 1.2. Структура комплексу сучасних засобів веб-розробки.

Із рисунку видно, що увесь комплекс ділиться на дві частини: фронт та бек. Так називають мови, засоби та технології, призначені для програмування відповідно браузеру клієнта («фронт» – те, що бачить користувач) та комп'ютера-сервера («бек» – те, що залишається позаду монітора користувача). Нижче у підрозділі 2.2 буде докладно розглянуто особливості найголовніших складових підсистем фронт- та бек-енду, для цілей же цього підрозділу достатньо більш загальної інформації, наведеної на рис. 1.2.

Узагальнюючи цей матеріал про особливості засобів веб-програмування, можна сказати, що мова JavaScript (а, отже, і бібліотеки до неї, чим власне і є jQuery) займає у цій галузі виділене становище: практично на безальтернативній основі ця мова використовується для програмування фронт-енд складових розподілених програмних комплексів, які призначені для

виконання у програмі-браузері клієнта. Але і цього для даної мови програмування було замало і вона «замахнулася» ще й на програмування серверних складових – за допомогою платформи Node.JS. Очевидно, що будь-яка бібліотека для певної мови програмування може використовуватися у будь-якому середовищі, де може запускатися інтерпретатор цієї мови, отже jQuery, звичайно, може запускатися і на стороні серверу. Необхідно зауважити, що загалом ця бібліотека «заточена» (призначена) для програмування саме браузеру клієнта, але, зважаючи на її немалий обсяг, теоретично у ній можуть бути знайдені і універсальні компоненти, корисні для використання як у браузері, так і на сервері.

Таким чином, місце jQuery у комплексі засобів веб-розробки досить важливе і однозначно необхідним є докладне вивчення цієї бібліотеки як у відповідному загальному (оглядовому) курсі, так і спеціалізованому, присвяченому строго бібліотеці та її можливостям.

1.3. Аналіз особливостей фреймворку jQuery

В першу чергу, і це – найголовніша особливість jQuery, ця бібліотека дозволяє надзвичайно просто отримувати доступ до будь-яких елементів дерева DOM, тобто до будь-якого тега веб-сторінки [2]. Для того, щоби отримати доступ до тегів у чистому JavaScript використовуються громіздкі конструкції на зразок document.getElementById(), document.getElementsByTagName(), і т.п. Бібліотека jQuery дозволяє звертатися до елементів значно простіше (з використанням спеціальної змінної $). Відповідно, значно спрощується написання JavaScript-коду, який активно маніпулює тегами та їх властивостями. Окрім зміни властивостей елементів, jQuery дозволяє легко створювати та видаляти окремі елементи-теги, маніпулюючи DOM (об’єктною моделлю документа – Document Object Model).

По-друге, бібліотека дозволяє легше та швидше працювати з технологією AJAX (обмін інформацією з сервером без перезавантаження веб-сторінки).

Обсяги висхідного коду при цьому зменшуються в рази. Зважаючи на важливість цієї технології в цілому, переваги у використанні jQuery перед чистим JavaScript стають очевидними.

Ще однією зручною можливістю jQuery є полегшена (знову ж таки у порівнянні з чистим JavaScript) робота з подіями.

У якості частинних, але важливих особливостей цієї бібліотеки також можна згадати можливість виконання певних дій (тобто запуск окремих ділянок коду) після виконання завантаження не усієї сторінки (як дозволяє обробник стандартної події onload, що викликається після завантаження УСІЄЇ сторінки, включаючи «важкі» зображення, які на ній можуть бути присутніми), а окремих її, необхідних для програміста, елементів. Це дозволяє розпочати обробляти сторінку кодом JavaScript ще до повного її завантаження.

Також jQuery пропонує швидкий доступ до різноманітних анімаційних ефектів, як затухання, чи, навпаки, поступова поява, плавне витіснення чи розчинення. Можливо також легко реалізувати перетаскування елементів по сторінці.

Цікавою можливістю jQuery є надзвичайно просте створення на веб-сторінках оригінальних елементів управління, таких, що не входять до переліку стандартних, а саме:

- гармоніка;
- календар;
- діалогове вікно;
- індикатор виконання;
- повзунок;
- вкладки.

Ну, і мабуть, головна можливість jQuery – можливість створення кроссбраузерного коду. Професійно написаний JavaScript часто буває перевантажений умовними операторами виду

```
if (user.browser.family === 'Safari') {
```

які перевіряють сімейство/тип/версію браузера, та виконують різні дії, залежно від результатів. Якщо ж використовується бібліотека jQuery, то усі ці перевірки прописані у самій бібліотеці та непомітні для програміста. Отже, сама бібліотека дозволяє писати один і той самий код для усіх браузерів та не опікуватися численними перевірками версій – це виконується у самій бібліотеці причому прозоро для програміста.

1.4. Особливості методики викладання курсу з jQuery

При викладенні відповідного навчального матеріалу слід враховувати особливості самого фреймворку, наголошуючи на них увагу учня.

В першу чергу, слід наголосити на місці і ролі jQuery у структурі комплексу засобів веб-програмування, точніше фронт-енд розробки. Учень має зрозуміти, що немає задач, вирішення яких обов'язково потребує використання бібліотеки в цілому. Абсолютно усі завдання можна вирішити шляхом використання тільки стандартних засобів JavaScript. В той же час у галузі IT надзвичайно гостро стоять питання тайм-менеджменту, зокрема, фахівцям безперервно доводиться шукати шляхи скорочення часу виконання тих, чи інших, в основному рутинних, операцій. Відповідно, програмісти часто намагаються використовувати навіть дрібні утиліти, чи елементарні засоби, які дозволяють отримати хоча б якусь, навіть і дуже малу, економію свого робочого часу. При використанні бібліотеки jQuery можливе значне (на десятки відсотків) скорочення часу виконання типових операцій мови JavaScript, тому цей засіб і набув досить широкої популярності: майже 100% професійних фронт-енд програмістів якщо не використовують безпосередньо jQuery, то принаймні добре знайомі з її можливостями.

Також важливим аспектом викладання багатогранних можливостей бібліотеки є роз'яснення хоча би базових моментів щодо тієї, чи іншої технології, використання якої полегшується за допомогою jQuery.

Наприклад, пояснюючи застосування jQuery для спрощення використання технології AJAX, необхідно надати коротку загальну інформацію про саму технологію та, можливо, навести посилання на докладніші матеріали по ній. Аналогічна ситуація виникає і з поясненням понять «кросбраузерність», «динамічні веб-сторінки» або «динамічний HTML», і т.п. Такі додаткові відомості є особливо актуальними, якщо jQuery викладається в рамках окремого курсу, і тоді достеменно невідомо, чи знають учні усі ці додаткові поняття. Якщо ж бібліотека вивчається в рамках загальної (обширної) дисципліни «веб-програмування», то і в цьому випадку нагадування про особливості застосовуваних технологій зайвими точно не будуть (адже багатократне повторювання є найефективнішим способом запам'ятовування інформації).

Наступною особливістю методики викладання дисциплін, пов'язаних із програмуванням в цілому, є можливість невідкладного підкріплення теоретичного матеріалу численними прикладами різного рівня складності. Різнитися може і середовище, у якому реалізовуються такі приклади: найпростіші з них можна розбирати безпосередньо на дошці чи папірці вручну аналізуючи висхідний код. В той же час більш об'ємні можуть проганятися через відповідний транслятор для аналізу результату виконання цього коду. Так, у нашому випадку мова йде про бібліотеку мови програмування JavaScript, транслятор якої включений до кожної програми-браузеру Інтернет. Відповідно, розбирати приклади роботи з бібліотекою можна на будь-якому пристрої, що містить можливість виходу до Глобальної мережі (на відміну від переважної більшості інших мов програмування, транслятори яких обов'язково слід встановлювати на персональні комп'ютери, а сам процес встановлення часто займає хвилини та, навіть, десятки хвилин).

Також загальними правилами при викладенні дисциплін сучасного, модернового напрямку, зокрема пов'язаних із галуззю IT (таких, як програмування для веб, jQuery), є викладення матеріалу із використанням:

- популярної, частково навіть жаргонної, термінології, окремих слівець, американізмів, якими насичена мова спеціалістів з ІТ, і т.д.;

- доступної сучасної техніки, імовірно, цифрової, гаджетів, комунікаційних технологій та інших новомодних технічних упроваджень. Із широко доступних засобів це можуть бути проектори (в т.ч. мобільні), лазерні указки, або більш дорогі інтерактивні дошки, окуляри доповненої або віртуальної реальності, і т.п.

- згадувань, демонстрацій, пояснень із галузей, близьких до ІТ (або таких, що активно використовують продукцію ІТ-сектору) і які є модними серед молоді на даний момент, знаходяться «на слуху».

Усі вище сформовані тези можна коротко виразити фразою «найсучасніші технології слід вивчати найкрутішими способами». Ефективність цього твердження успішно підтверджується численними ІТ-школами, які не входять в структуру МОН чи офіційних структур інших держав, і тому можуть на свій розсуд використовувати навчальні засоби та методи (зокрема, стиль мовлення). Ці заклади показують надзвичайну ефективність у вигляді результативних випускників, що без додаткових зусиль (як-то навчання у ВНЗ) досить швидко стають практикуючими висококваліфікованими програмістами.

Таким чином, при викладенні розділу jQuery слід використовувати як звичайні навчальні засоби та підходи, так і деякі спеціалізовані, що використовують особливості власне бібліотеки.

1.5. Особливості електронних навчальних курсів

Електронним навчальним курсом (тут і далі – ЕНК) можна назвати програмний продукт, який включає усе необхідне для вивчення дисципліни методичне забезпечення та дещо більше (розглянуто конкретніше далі). Важливою особливістю електронних навчальних курсів є те, що робота з ними відбувається переважно в асинхронному режимі (мається на увазі окремо від

викладача). Такі програмні продукти для навчання в асинхронному режимі повинні будуватися на основі багаторічного досвіду викладання відповідної дисципліни студентам денної форми і передбачати будь-які можливі питання. Основною складовою електронного навчального курсу є електронний конспект, де, як і в гарному підручнику, повинний бути присутнім матеріал кількох рівнів складності, з виділенням необхідного мінімуму; повинні надаватися посилання (гіперпосилання на ресурси Інтернет чи паперові книги), де можна поглиблено вивчити матеріал. Також гіперпосилання можуть використовуватися для зведення внутрішньої структури матеріалу і вказувати на інші частини цього ж документу.

Окрім електронного конспекту (якщо під цим поняттям мати на увазі лише набір теоретичних відомостей), у ЕНК повинні бути присутніми практичні, та, можливо, лабораторні завдання.

Практичні завдання можна сформулювати для будь-яких дисциплін, і, звичайно, у повній мірі – для галузі програмування. Якщо такі приклади робити пасивними, тобто виключно у вигляді тексту, рисунків та формул, то принципово їх можна не вирізняти з електронного конспекту, а зробити його частиною. Таку структуру має значна кількість підручників: біля 50-60% місця відведено для теоретичного матеріалу, а потім 30-40% всього місця параграфу займає розбір типових прикладів (і часто відсотків 10 відводиться на завдання для самостійної проробки та контрольні питання).

Однак, практичні завдання можуть бути динамічними: при цьому можливим є виконання деяких змін у прикладі, що розв’язується (пояснюється), із відповідною зміною всієї іншої його частини (аж до структурних змін у розв’язку – при добрій проробці такого прикладу). У цьому випадку ці практичні завдання, звичайно не можуть бути частиною пасивного текстового документу, а повинні реалізовуватися або як окремий настільний програмний продукт (*.exe – файл), або як html-сторінка з динамічним вмістом.

Деякі навчальні предмети для повноцінного засвоєння передбачають виконання лабораторних робіт, основною метою яких є навчити учня

принципам роботи з обладнанням, яке пов’язане з предметом, що вивчається. У цьому випадку в рамках ЕНК лабораторні роботи також можуть бути реалізовані за допомогою віртуальних приладів. Наприклад, на рис. 1.3 показано головне вікно програми, що імітує роботу оператора з передньою панеллю осцилографа (можна крутити більшість регуляторів, від чого змінюється картинка, що її показує осцилограф, причому в точності так, як було би на реальному приладі).

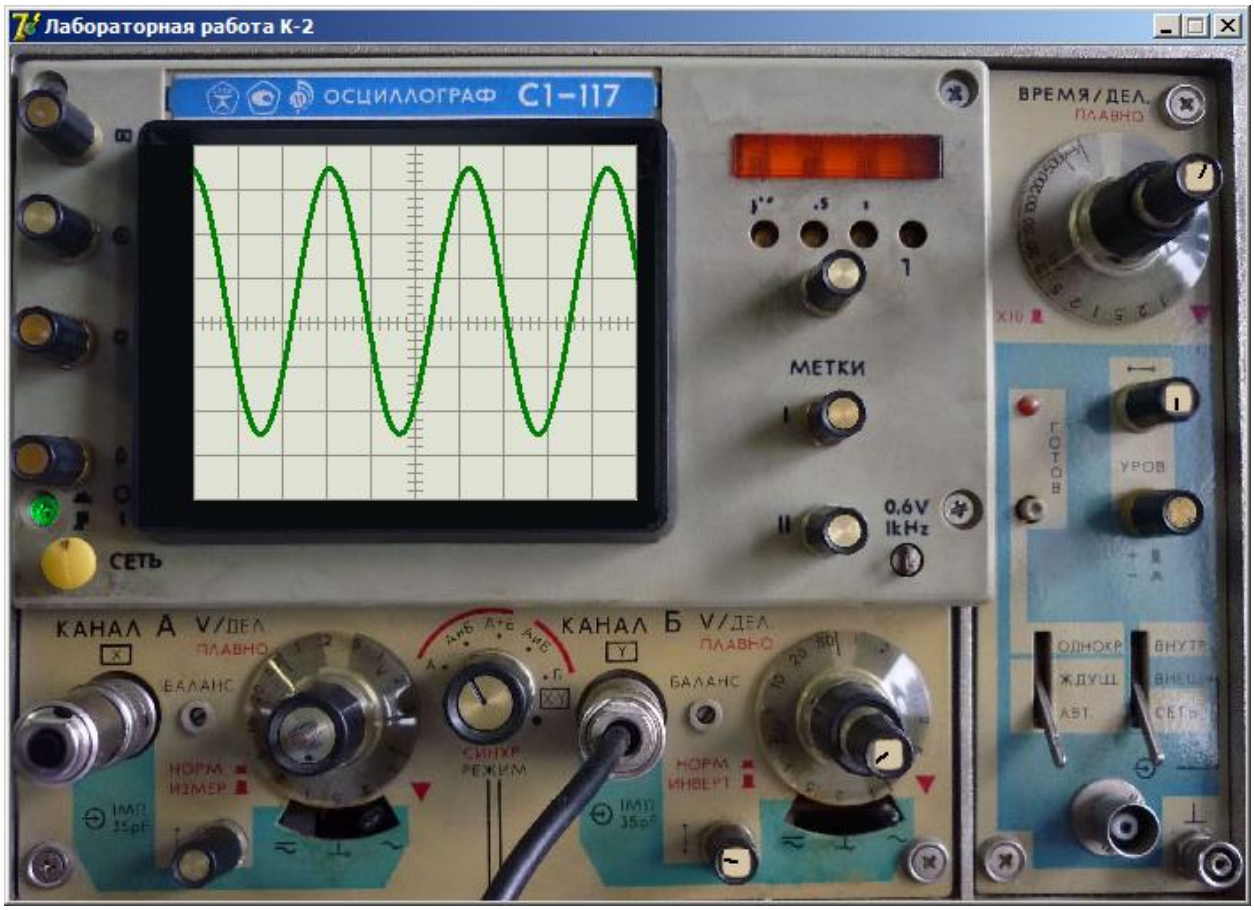


Рис. 1.3. Програма-осцилограф для виконання віртуальних лабораторних робіт, що допускає виконання основних налаштувань аналогічно до справжнього приладу.

Не зважаючи на величезну перспективу побудови програм-симуляторів, у даній роботі більше розглядати віртуальне лабораторне обладнання не будемо, оскільки вивчення бібліотеки jQuery (як і більшості предметів з програмування)

не передбачає використання якогось спеціалізованого апаратного забезпечення, для ознайомлення з яким і потрібне було б віртуальне обладнання.

Нарешті, при створенні ЕНК одним із останніх (підсумкових) елементів має бути впроваджений комплект питань, які можуть використовуватися як для самоконтролю, так і для проведення вихідного (рубіжного, підсумкового, і т.п.) контролю знань. Для підвищення якості цієї компоненти програмного ЕНК ці завдання також мають відповідати певним вимогам. Так, в тестових питаннях при наданні неправильних відповідей, учня необхідно спрямовувати на той блок матеріалу, до якого відноситься питання, отже, тестові вправи повинні бути зв’язані із електронним конспектом. Можна передбачити підказки, що їх робив би «живий» викладач, при очному опитуванні студента. Те ж саме стосується і програм - віртуальних лабораторних робіт. В них крім того, повинні даватися чіткі пояснення про хід виконання роботи, призначення приладів, обчислювані величини, і т.п.

Усі асинхронні програмні продукти повинні передбачити також будь-які дії користувача, що може бути не дуже добре обізнаний з ПК. Саме тому ці програми повинні мати інтуїтивний інтерфейс та розвинену систему підказок, аби виключити вплив досвіду роботи на комп’ютері на оцінку з предметної дисципліни.

Отже, усі асинхронні програмні продукти повинні працювати під управлінням єдиної інформаційної системи, яка контролюватиме учбовий процес, при чому якість цих програм повинна бути на рівні, що забезпечує можливість самостійної роботи учня.

1.6. Аналіз існуючих аналогічних рішень по вирішенню поставленої задачі

Усі такі рішення можна на самому верхньому рівні абстракції розподілити на два класи:

- інструменти низькорівневої розробки (за допомогою яких можна сконструювати власний, з оригінальною структурою, електронний навчальний курс), як, наприклад, засоби веб-програмування типу серверних мов та браузерних інтерпретаторів, або які-небудь інтегровані середовища розробки (Integrated Development Environment – IDE) на зразок Delphi, Microsoft Visual Studio, і т.д. – для розробки настільного програмного забезпечення;

- готові (або майже готові) програмні платформи, на основі яких можна будувати електронні навчальні курси різної направленості, в т.ч. і для галузі веб-програмування, а конкретніше, для вивчення саме бібліотеки jQuery.

Останній клас можна розглянути окремо в рамках аналізу аналогічних програмних продуктів. Як відомо, ринок програмного забезпечення на сьогоднішній день досить насичений і практично для будь-якої предметної галузі зустрічаються програми, що у більшій, чи меншій мірі реалізують необхідну для будь-якого замовника функціональність. Саме тому, аналіз існуючих програмних продуктів-аналогів є на даний час де-факто стандартною складовою процесу будь-якої розробки програмного забезпечення. Отже, засоби, що підпадають під другий клас розглянемо пізніше. Зараз же будемо говорити про ті програмні засоби, за допомогою яких необхідний електронний навчальний курс разом із засобами відображення, перевірки та контролю можна *розробити*, а не взяти готовим.

І знову усі такі засоби можна розбити на дві складові: найбільш низькорівневі та шаблонні системи.

До першого типу можна віднести усі мови програмування загального призначення (за умови реалізації електронного навчального курсу у вигляді настільного програмного продукту) та, зокрема, на основі яких можна розробляти Інтернет-сайти (мається на увазі серверний код). Сюди можна віднести мови C/C++, Perl, Python, Ruby, навіть Object Pascal та інші. Також сюди можуть бути віднесені скриптові мови, коди яких (або, як варіант, результат проміжної трансляції цих кодів типу байт-коду Java) безпосередньо вбудовуються у тіло веб-сторінки, такі як: PHP, JSP, ASP, servlets, і т.п.

До засобів шаблонного типу можна віднести уже готові універсальні розробки (тобто такі, що не є призначеними виключно для галузі освіти, але легко до неї пристосовуються), що являють собою т.зв. «движок» (причому досить гнучкий), але не наповнені інформацією. Сюди можна віднести численні різноманітні неспеціалізовані системи управління вмістом, тобто CMS (Content Management System), або уточнено WCMS (Web Content Management System):

- WordPress;
- Joomla;
- Drupal;
- 1С-Бітрікс;
- тощо.

Використання таких систем має як свої переваги, так і суттєві недоліки. Головним недоліком таких систем якраз є їхня неспецифічність. Насправді галузь освіти має чимало особливостей, що звичайно ніяк не враховуються у CMS загального призначення, тому вони потребують серйозних доробок, які можуть бути виконані лише професійними програмістами, що спеціалізуються на цих системах. Навпаки, система загального призначення пристосована для задоволення потреб якомога більшої аудиторії користувачів, має значну частину (імовірно, навіть переважну частину) зайвого функціоналу. Отже, ці системи є занадто громіздкими, повільними та вимагають більш потужного апаратного забезпечення, ніж ресурс з мінімально необхідною цільовою функціональністю.

Таким чином, замість використання та налаштування якоїсь системи загального призначення більш оптимальним видається використання спеціалізованого ресурсу, призначеного саме для освітньої галузі та, бажано, безпосередньо для вивчення дисциплін галузі ІТ, зокрема, програмування. Отже, перейдемо до розгляду таких програмних продуктів.

Дійсно, існує чимало програмних продуктів, призначених для утворення Інтернет-ресурсів в галузі освіти, і які можна адаптувати до потреб вивчення курсів програмування (зокрема, бібліотеки jQuery).

Так, найбільш активно використовуваною Інтернет-платформою для освітньої галузі є система MOODLE (Modular Object-Oriented Dynamic Learning Environment) або Moodle (читається «Мудл»).

Для вузькоспеціалізованих систем типу Moodle введено спеціальні абревіатури:

- LMS, яка розшифровується як «система управління навчанням» (Learning Management System);

- VLE (Virtual Learning Environment – «віртуальне навчальне середовище»).

Moodle можна назвати просто веб-платформою для навчання, яка надає викладачам, учням та адміністраторам дуже розвинутий набір інструментів для комп'ютеризованого навчання, в тому числі дистанційного. Moodle можна використовувати в навчанні школярів, студентів, при підвищенні кваліфікації, бізнес-навчанні, як в комп'ютерних класах навчального закладу, так і для самостійної роботи вдома.

Moodle - це найбільш досконала і поширена в Україні і в світі система такого призначення. На даний момент Moodle вже має 130 мільйонів користувачів в усьому світі й продовжує розвиватися темпами, значно швидшими, ніж її конкуренти. Тобто обрати для впровадження в навчальному закладі саме Moodle - це «те саме, що обрати, як іноземну мову для вивчення – англійську», - стверджують розробники.

Moodle - це безкоштовна, відкрита (Open Source) система. Вона не лише безкоштовна сама, а й не потребує для своєї роботи жодного платного програмного забезпечення. Тобто кожний навчальний заклад може впровадити у себе не просто безкоштовну і найбільш досконалу, а ще й абсолютно ліцензійну систему, не витративши жодної копійки на придбання програмного забезпечення. При цьому можливо вносити зміни у код у відповідності до своїх потреб.

Недоліками Moodle (та інших подібних рішень) є:

- необхідність збирати систему з нуля;

- складність, комплексність системи;

- потреба у технічних компетенціях в області веб-розробки у викладачів, які до того ж не бажають витрачати свій час на установку і налагодження системи. У цьому випадку викладач шукає готове налаштоване рішення, куди він може просто завантажити наявний у нього матеріал електронного курсу, зібрати учнів онлайн і провести заняття.

Відповідно, альтернативні розробники (особливо ті, що пропонують комерційні продукти-аналоги) наголошують, що система повністю не підходить для викладачів, які не володіють навичками веб-розробки або Інтернет-комунікацій, бо є надто складною.

Однак, в нашому випадку основний інтерес викликають не загальні дисципліни та можливості щодо проведення відповідних занять, а викладання високоспеціалізованого курсу з програмування jQuery. Нажаль, для такого конкретного завдання Moodle дійсно підходить мало. Система видається перевантаженою крутими опціями, в той час, як прості функції, пов’язані із елементарними потребами галузі програмування у цій системі відсутні, або мають виконуватися дуже незручним, непрямим чином.

Аналогічні недоліки мають і інші платформи класу LMS, такі як iSpring Online LMS, GetCourse, Blackboard Learn, Schoology, Eliademy, Teachbase, Edmodo, MySchool, EduTerra.PRO, Teachable, і т.д. До того ж програмні продукти, що мають більш простий і зрозумілий інтерфейс користувача є платними, а якщо і мають безкоштовну версію, то вона є сильно обрізаною, такою, що не дозволяє вести навчальну роботу у повній мірі.

Значним недоліком готових систем, що мають чисте онлайн розміщення (тобто користувач не має доступу до його висхідних текстів, написаних, наприклад, мовою PHP і розміщених тільки на сервері) є постійна загроза зміни політики використання, що іноді трапляється у світі вільного програмного забезпечення. Результатом є необхідність фінансових витрат для продовження використання системи, що змінила тип ліцензії. Ситуація, коли витративши чимало часу на наповнення відкритої, безкоштовної онлайн-системи реальною

інформацією, вчитель пізніше буде вимушений перестати її використовувати через необхідність оплати доступу (при зміні ліцензійної політики) є неприпустимою, тому онлайн-системи, для яких відсутній доступ до їх висхідних текстів взагалі не розглядатимемо.

Тим більше не розглядатимемо системи, що є платними вже на даний момент. Галузь освіти є чи не найбіднішою в економіці практично будь-якої країни, навіть і розвинутої (на відміну від військової, наприклад). Тому пропонувати навчальним закладам (або тим більше, окремим учителям, що взагалі неприпустимо!) заплатити гроші за програмне забезпечення (враховуючи, що у світі існує чимало повністю безкоштовного аналогічного ПЗ) є дією майже на грані криміналу.

Конкретним прикладом безкоштовної платформи для вивчення програмування є всесвітньо відомий портал Coursera. На ньому за запитом jQuery видається 81 курс – рис. 1.4, що на перший погляд може потішити, однак, при детальному аналіз виявляється, що усі ці курси є загальними, присвяченими, як мінімум, мові JavaScript в цілому, а jQuery викладається тут лише як один із розділів, тому не дуже відповідає вимогам, поставленим у даній роботі.

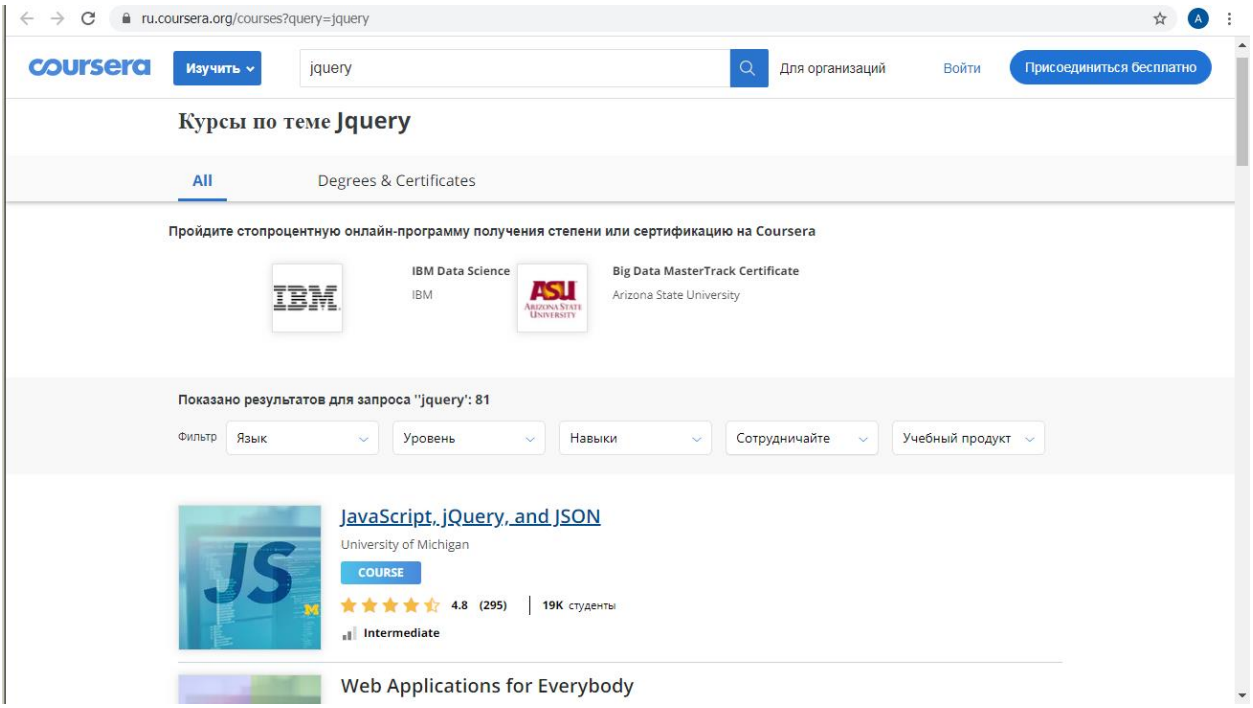


Рис. 1.4. Вікно порталу Coursera із курсами, що містять матеріали по jQuery.

Останнім класом ресурсів, на яких є можливим вивчення бібліотеки jQuery, є спеціалізовані сайти, що збудовані без використання загальновідомих LMS, а шляхом використання мов програмування для Інтернет.

Таким, наприклад, є ресурс WebforMyself – рис. 1.5. Його особливістю є використання навчального матеріалу виключно у вигляді відео лекцій, що є незручним для значної частки практикуючих програмістів, які цінують свій час, вже розуміються на програмуванні в цілому та конкретно JavaScript, і, відповідно, схиляються до читання навчальної інформації (де швидкість її отримання регулюється самим учнем, а не швидкістю відеофайлу). Важливим недоліком даного ресурсу також є необхідність оплати більшої кількості відео лекцій (окрім 2 початкових), яка складає близько 400 грн за місяць.

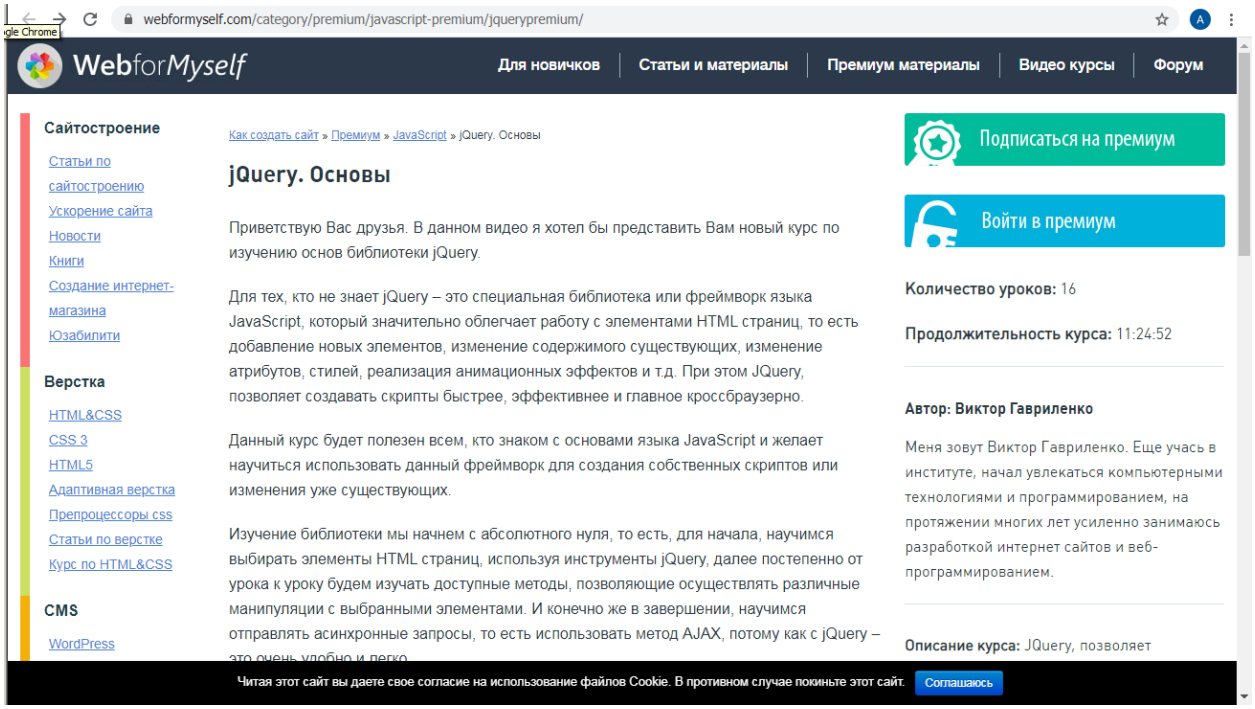


Рис. 1.5. Зовнішній вигляд ресурсу WebforMyself, присвяченого вивченню бібліотеки jQuery.

Ще одним ресурсом, що надає більш-менш повноцінний електронний навчальний курс по вивченню jQuery, є FructCode – рис. 1.6. Його особливістю є включення чималої за обсягами інформації про мову JavaScript в цілому у курс по jQuery, що не є доцільним, оскільки значна частка веб-програмістів

спочатку вивчає чистий JavaScript, а уже при переході до реальних задач, стикається з jQuery, для якого потрібен свій курс вивчення.

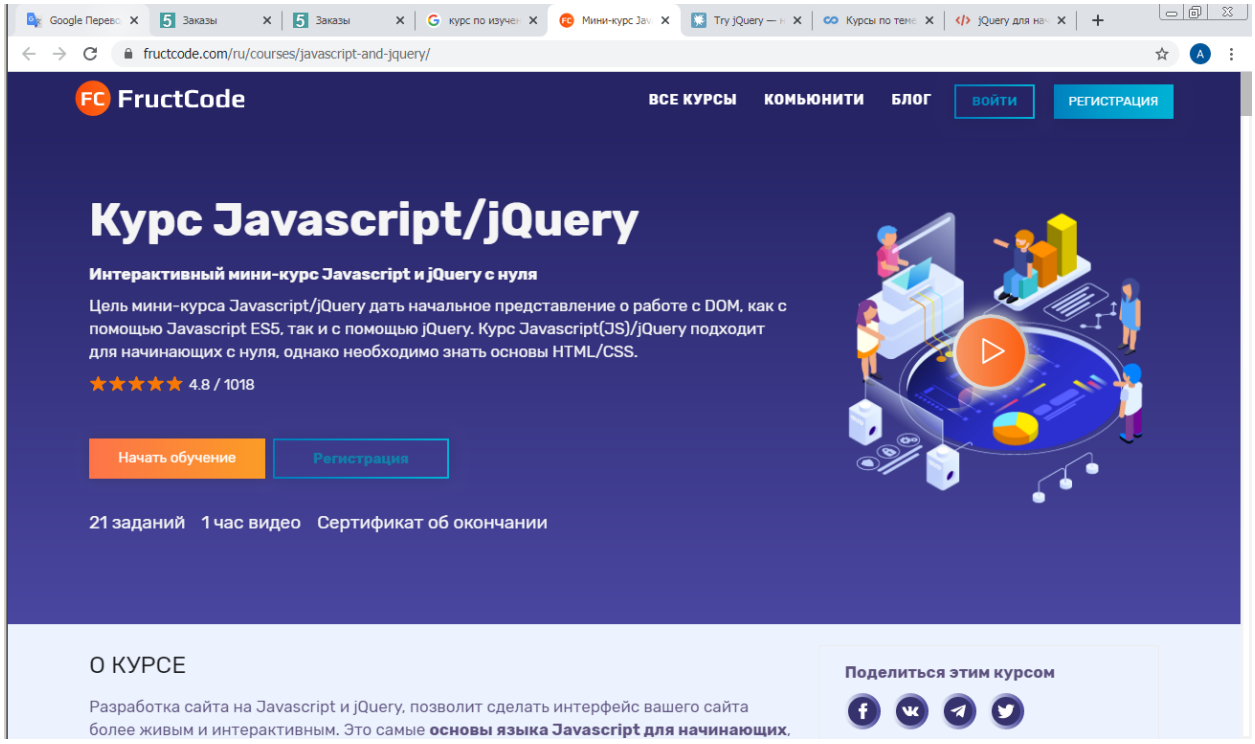


Рис. 1.6. Зовнішній вигляд ресурсу FructCode, присвяченого вивченню бібліотеки jQuery та мови JavaScript в цілому.

Відмітимо, що доступ до усіх уроків на FructCode також є платним і складає близько 450 грн на місяць (19 у.о.).

Нарешті останнім із характерних ресурсів, за допомогою яких можна вивчати jQuery, є itProger – рис. 1.7. Тут присутні досить якісні відео презентації та відповідний ним текстовий матеріал, отже існує можливість вибору способу здійснення навчання. Також відкрито доступ до малих практичних завдань, але «великі» (за термінологією розробників сайту) завдання – закриті, так само, як і тестові вправи трьох рівнів складності. Ці елементи також є платними (біля 400 грн на місяць).

Рис. 1.7. Зовнішній вигляд ресурсу itProger, присвяченого вивченню бібліотеки jQuery.

Таким чином, основними недоліками аналогічного програмного забезпечення, що могло би використовуватися для створення електронного навчального курсу є:

- відсутність вузькоспеціалізованого програмного продукту, призначеного чітко для вивчення програмування (тим більше, саме бібліотеки jQuery), і відповідно такого, що враховує усі аспекти даної предметної галузі;

- тотальне переважання універсальних освітніх систем LMS (серед яких найвизначнішою є Moodle), які мають надзвичайно складну, надлишкову для даного випадку будову і, відповідно, завищують вимоги до кваліфікації користувача, що міг би користуватися такою системою; всеосяжність таких систем також спричинює завищені вимоги до апаратної частини комп'ютера, що також суттєво, зважаючи на широке різноманіття наявної у вітчизняних користувачів комп'ютерної техніки (яка включає чимало застарілого та обмеженого по характеристиках обладнання);

- значна частина присутніх у вільному доступі LMS та спеціально розроблених ресурсів являють собою лише обмежені версії більш зручних та

цікавих комерційних програмних продуктів, що також не підходить для практичного використання, зважаючи на невисокі зарплати працівників галузі освіти.

Беручи до уваги вищенаведене, можна зробити висновок, що доцільним є створення власного програмного продукту, простого за інтерфейсом та структурою, спеціалізованого, що враховує лише важливі для викладання курсу програмування бібліотеки jQuery аспекти, безкоштовного у своїй повнофункціональній версії. На виконання цього завдання і направлена дана магістерська робота.

2. ОСОБЛИВОСТІ ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ РОЗРОБКИ

2.1. Постановка задачі дослідження

Беручи до уваги вищенаведену інформацію (особливо підрозділ 1.6) по аналізу існуючих аналогічних рішень, слід констатувати, що існує нагальна потреба у розробці оригінального вузькоспеціалізованого електронного навчального курсу по вивченню бібліотеки jQuery. Відповідно до мети та задач дослідження, а також аналізу доступних засобів розробки (підрозділи 1.2 та 1.6) створення курсу слід вести на базі більш універсальних методів та підходів веб-програмування, а саме, з використанням наступного стеку мов та технологій:

- HTML – для формування безпосередньо сторінки (сторінок), присвячених окремим темам, відображення змістовного тексту, картинок, і т.п.;
- CSS – для забезпечення уніфікованого зовнішнього вигляду усіх елементів курсу;
- JavaScript – для забезпечення динамічної поведінки сторінок, зокрема динамічного оновлення вмісту – технологія AJAX;
- PHP – для виконання серверного коду, який забезпечує реалізацію усієї бізнес-логіки проекту;
- MySQL – система управління базами даних для збереження усієї змістовної інформації курсу.

При такому підході уся міць веб-програмування поєднується з кросплатформеністю кінцевого рішення.

Маючи конкретні завдання та озброївшись відповідними технологіями та засобами розробки, можна переходити безпосередньо до процесу проектування кінцевого рішення яким є електронний курс по jQuery у вигляді динамічного Інтернет-сайту.

2.2. Обгрунтування вибору мов та засобів розробки

Першим питанням, яке постає перед розробниками практично будь-якого програмного забезпечення, є вибір моделі або технології його розробки. Такими, що широко використовуються на сьогоднішні день у виробничій практиці, є технології структурного (процедурного) та об'єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне, або як його ще називають практикуючі програмісти, процедурне програмування засноване на використанні окремих структурних блоків - в першу чергу, підпрограм (процедур і функцій).

Історично перші комп'ютерні програми були відносно простими і мали пакетний режим роботи: отримуючи на вхід якусь інформацію (можливо, навіть на перфокарті) вони виконували певний обсяг операцій з обробки цих даних і видавали результат. При цьому існувала сувора функціональна залежність виходу від входу – рис. 2.1.

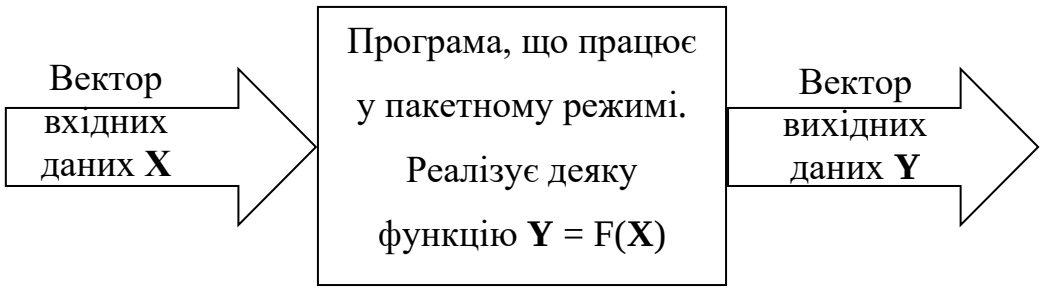


Рис. 2.1. Схема роботи пакетної програми.

Незважаючи на відсутність явного зв'язку функціональності і структури, пакетні програми в основному мали просту лінійну послідовність виконання. Тобто в них практично відсутні будь-які підпрограми, принаймні, використання підпрограм не було наріжним каменем самої методики програмування.

В міру ускладнення функціональності програмного забезпечення, змінювалася і його внутрішня структура: поступово розвинулася інтерактивна модель взаємодії користувача і програми – рис. 2.2. Програми стали запитувати інформацію і активно реагувати на дії людини. Ускладнення функціональності

привело до відповідного ускладнення програмного коду, якого, в першу чергу, стало просто багато. Багато для того, щоб людина-програміст без всяких спеціальних хитрувань швидко і легко розібралася з незнайомим кодом. Розміщувати код у простій лінійній послідовності без виділення великих блоків стало незручно, в першу чергу, для розуміння цього коду.

Рис. 2.2. Схема роботи програми, що активно взаємодіє з користувачем.

Тут слід зазначити психологічні особливості сприйняття людиною складних «великих» завдань. Неструктуроване «велике» завдання (наприклад, написання дипломної роботи) зазвичай викликає певний психологічний ступор і, як результат, повну неможливість поступово розібратися з ним. Людині зручно розбити проблему на не надто велику (зазвичай до десятка, а краще 3-4) кількість завдань (наприклад, розділів у дипломній роботі), не замислюючись про реалізацію кожного з них. Коли є ясність і розуміння проблеми на найвищому рівні абстракції, слід приступати до деталізації підзадач, кожну з яких слід розбити на окремі «підпідзадачі» тобто підзадачі нижчого рівня, більш дрібні. Уже після такого розбиття слід аналізувати всі перераховані підзадачі. На певному етапі зупиняються і виконують не розбиття чергової підзадачі на більш дрібні, а безпосередню її реалізацію в програмних кодах.

Отже, можна сказати, що розвиток методики програмування відбувався в ногу з розвитком призначеного для користувача інтерфейсу: і грубо кажучи,

розвиненому інтерфейсу командного рядка відповідає парадигма структурного програмування.

Говорячи більш строго, структурне програмування має на увазі побудова програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

Слідування має на увазі, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if, і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (що йдуть підряд багаторазових повторів одного і того ж ділянки коду), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування у деякій мірі є застарілою методикою програмування, на зміну якій разом з віконним інтерфейсом прийшло об’єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об’єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування більш виправдано і код краще сприймається, ніж його об’єктно-орієнтований варіант.

Суть же методології об’єктно-орієнтованого програмування полягає в тому, що система розглядається, як сукупність окремих сутностей - об’єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у

яких типом виступає клас об’єкта. Клас - це просто опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт - це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності)? Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу показана на рис. 2.3. З графіка рис. 2.3 слід, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою, порядку кількох тисяч рядків), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

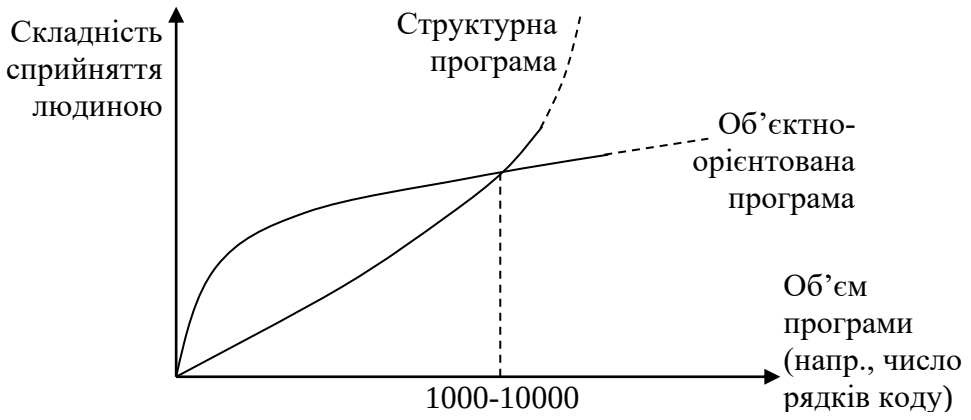


Рис. 2.3. Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об'єктно-орієнтованому та структурному підходам.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний підхід до програмування.

Крім розбиття (декомпозиції) всієї предметної області на об'єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об'єднання даних (значення властивостей класу у деякого конкретного об'єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об'єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас,

який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент - як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і той саме.

Поліморфізм є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і в загальному багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, у програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(),

а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Грунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо структурний підхід як більш простий, що відповідає невеликим (не промисловим) масштабам проектованого ПЗ, а також вимогам до її складності. Також структурний підхід прекрасно реалізується засобами та мовами (практично усіма) для веб-розробки, яку доцільніше всього застосувати для створення електронного навчального курсу.

Наступним кроком після вибору технології програмування має бути докладний глибокий аналіз засобів, що можуть бути використані для створення як серверної частини (умовно їх називають back-end засоби), так і клієнтської (засоби front-end розробки) – рис. 1.2.

2.2.1. Засоби Front-end розробки

Спрощено кажучи, під «передньою» частиною – Front-end – у програмуванні мають на увазі те, що бачить користувач. Для настільних систем під цим мають на увазі інтерфейс, а під back-end’ом мають на увазі логіку роботи програми, яка теоретично (зокрема, так часто роблять у Linux-системах) взагалі може бути реалізована окремою програмою (типу утиліти) із текстовим консольним інтерфейсом. Задачею фронт-енду при цьому є зручний для широкого кола користувачів збір усіх вхідних даних, необхідних для виконання змістовних операцій консольною утилітою back-end’ом. У багатьох же випадках фронт та бек поєднані разом в рамках одного програмного продукту. Такою є ситуація для настільних систем.

Якщо конкретніше говорити про веб-програмування, то тут ситуація є максимально поляризованою (ще більше, ніж у описаному вище випадку написання фронт-енду одним програмістом для бек-енд-утиліти, створеної

іншим розробником), оскільки користувач працює і «бачить» систему через браузер на одній віддаленій машині (на комп’ютері, що називається клієнтом), а логіка додатку та практично уся супутня інформація розміщується на зовсім іншому комп’ютері-сервері.

Таким чином, усі програмні коди та висхідні тексти, що виконуються у браузері, тобто на стороні клієнта, відносяться до front-end частини. З іншого боку, все, що виконується на сервері, відноситься до back-end складової.

Якщо глибоко не вдаватися у детальний аналіз, то коротко можна сказати, що набір інструментів для front-end розробки значно менший, ніж для беку. Серед них можна виділити:

- HTML – п’ятої версії (HTML5), найсучасніший стандарт мови гіпертекстової розмітки (HyperText Markup Language), де традиційно під гіпертекстом мають на увазі текст, оснащений гіперпосиланнями на інші частини документів та, можливо, супроводжуваний картинками, звуками, відео, і т.п.;

- CSS – третьої версії (CSS3) – правила запису стилів різноманітних елементів веб-сторінки, або, якщо говорити точно – каскадні таблиці стилів;

- JavaScript – найпотужніший інструмент front-end розробки, що дозволяє вивести html-сторінки на новий рівень практично повноцінних додатків, на зразок настільних, що активно реагують на дії користувача, передають і отримують інформацію, від нього.

Ці технології є безальтернативними, тому розглянемо їх докладніше.

1) Hyper Text Markup Language (HTML) - мова розмітки гіпертексту - призначена для написання усіх документів в мережі World Wide Web - WWW.

HTML документ - це текстовий файл, який має спеціальні мітки, що називаються тегами, які при затребуванні клієнтом передаються із комп’ютера-серверу на браузер клієнта і використовуються ним для відображення вмісту файлу па екрані комп'ютера.

За допомогою цих міток можна виділяти заголовки документа, змінювати колір, розмір і написання літер, вставляти графічні зображення і таблиці. Але

основною перевагою гіпертексту перед звичайним текстом є можливість додавання до вмісту документа гіперпосилань - спеціальних конструкцій мови HTML, які дозволяють клацанням миші перейти до перегляду іншого документа. Таким чином «гіпертекст» означає «надтекст» - розвинення ідеї текстового документу, але із більш широкими можливостями.

Сама по собі мова HTML є різновидом більш загальної мови XML (eXtensible Markup Language), причому з одного боку звужує її властивості, а з іншого – навпаки деталізує. Так, у мові XML допустимим є використання будь-яких тегів, тобто з довільними іменами, в той час, як назви тегів мови HTML є цілком фіксованими, а інші слова окрім стандартних, використовувати не можна. З іншого боку, той вміст веб-сторінки, що розміщений біля конкретних тегів (точніше – між парою однакових тегів), набуває визначених стандартом властивостей (на відміну від тексту, що розміщений поруч із тегами XML, які самі по собі практично нічого не означають, а використовуються лише при наявності певних домовленостей про відображення й обробку того, чи іншого тегу) відображується браузером відповідно до суті цих тегів: якщо тегом є <B>, то текст зображуватиметься жирним, <i> - курсивом, і т.д.

HTML-документ має дві складові: власне текстова інформація, тобто дані, що складають вміст документа, і теги - спеціальні конструкції мови HTML, які використовуються для розмітки документа і керують його відображенням. Теги мови HTML визначають, в якому вигляді буде представлений текст, які його компоненти будуть виконувати роль гіпертекстових посилань, які графічні або мультимедійні об’єкти повинні бути включені в документ.

Графічна та звукова інформація, що включається в HTML-документ, зберігається в окремих файлах. Програми перегляду HTML-документів називають браузерами, і вони інтерпретують теги розмітки і оперують текстом і графікою, розміщуючи їх на екрані відповідним чином. Для файлів, що містять HTML-документи використовується розширення .htm або .html.

У переважній більшості випадків теги використовуються парами. Пара складається з тега, що відкриває <ім’я_тега> і тега, що закриває </ім’я_тега>. Дія будь-якого парного тега починається з того місця, де зустрівся відкриваючий тег, і закінчується при зустрічі відповідного закриваючого тега. Часто пару, що складається з відкриваючого і закриваючого тегів, називають контейнером, а частину тексту між відкриваючим і закриваючим тегом - елементом.

Послідовність символів, яка утворює текст, може складатися з пробілів, табуляцій, символів переходу на новий рядок, символів повернення каретки, букв кириличного та латинського написань, знаків пунктуації, цифр, і спеціальних символів (наприклад #, +, $, @), за винятком наступних чотирьох символів, що мають в HTML спеціальний сенс: < (менше), > (більше), & (амперсанд) і «(лапки)». Якщо необхідно включити в текст будь-який із цих спецсимволів, то слід закодувати його особливою послідовністю символів:

```
<     -     &lt
>     -     &gt
&     -     &amp
“     -     &quot
```

Найпершим із тегів HTML розміщується однойменний тег <html>. Він завжди відкриває документ, так само, як тег </html> повинен неодмінно стояти в останньому його рядку. Ці теги позначають, що рядки, які між них знаходяться, представляють собою єдиний гіпертекстовий документ. Без цих тегів браузер або інша програма перегляду не в змозі ідентифікувати формат документа і правильно його інтерпретувати.

Далі, якщо говорити укрупнено, HTML-документ має дві частини: заголовок (head) і тіло (body), розташованих в наступному порядку:

```
<Html>
<Head> Заголовок документа </ head>
<Body> Тіло документа </ body>
</ Html>
```

Найчастіше в заголовок документа включають парний тег <title> ... </title>, що визначає назву документа. Багато програм перегляду

використовують його як заголовок вікна, в якому виводиться документ. Програми, що індексують документи в мережі Інтернет, використовують назву для ідентифікації сторінки. Гарна назва повинна бути досить довгою для того, щоб можна було коректно вказати відповідну сторінку, і в той же час воно має міститися в заголовку вікна. Назва документа вписується між відкриваючим і закриваючим тегами title.

Тіло документа є обов'язковим елементом, так як в ньому розташовується весь матеріал веб-сторінки. Тіло документа розміщується між тегами <body> і </body>. Все, що розміщено між цими тегами, інтерпретується браузером відповідно до правил мови HTML дозволяють коректно відображати сторінку на екрані монітора.

Текст в HTML розділяється на абзаци за допомогою тега . Він розміщується на початку кожного абзацу, і програма перегляду, зустрічаючи його, відокремлює абзаци один від одного порожнім рядком. Використання закриваючого тега є необов'язковим.

Якщо потрібно «розірвати» текст, перенісши його залишок на новий рядок, при цьому, не виділяючи нового абзацу, використовується тег розриву рядка
. Він змушує програму перегляду виводити символи, що стоять після нього з нового рядка. На відміну від тега абзацу, тег
 не додає порожній рядок. У цього тега немає парного закриваючого тега.

Мова HTML підтримує логічне н фізичне форматування вмісту документа. Логічне форматування вказує на призначення даного фрагмента тексту, а фізичне форматування задає його зовнішній вигляд.

При використанні логічного форматування тексту браузером виділяються різні частини тексту відповідно до структури документа. Щоб відобразити назву, використовується один з тегів заголовка. Заголовки в типовому документі поділяються за рівнями. Мова HTML дозволяє задати шість рівнів заголовків: h1 (заголовок першого рівня), h2, h3, h4, h5 і h6. Тема першого рівня має зазвичай більший розмір і насиченість в порівнянні з заголовком другого рівня. Приклад використання тегів заголовків:

```
<Hl> l. Назва розділу </hl>
<H2> l.l. Назва підрозділу </h2>
```

Теги фізичного форматування безпосередньо задають вид тексту на екрані браузера, наприклад пара   виділяє текст напівжирним шрифтом, <u> </u> задає підкреслення тексту,   керує шрифтом тексту. Саме ці елементи вважаються застарілими, і замість них у специфікації HTML5 рекомендується користуватися стилями, що розглянемо нижче.

Тег <img> вставляє зображення в документ, причому так, якби воно було просто одним великим символом. Закриваючий тег для зображення не потрібний. Приклад застосування тега:

```
<Img src = "picture.gif">
```

Для створення гіпертекстового посилання використовується пара тегів <a> ... . Фрагмент тексту, зображення або будь-який інший об'єкт, розташований між цими тегами, відображається у вікні браузера як гіпертекстове посилання. Активація такого об’єкта за допомогою щигля мишею призводить до завантаження у вікно браузера нового документа або до відображення іншої частини поточної Web-сторінки. Гіпертекстове посилання формується за допомогою формули:

```
<A href = "document.html"> посилання на документ </a>
```

Href тут є обов'язковим атрибутом, значення якого і є URL-адресою запитуваного ресурсу. Лапки в завданні значення атрибута href не обов'язкові (як і при завданні значень і для інших атрибутів будь-яких тегів, але використання лапок – подвійних, чи одинарних, є вкрай бажаним). Якщо задається посилання на документ на іншому сервері, то вид гіперпосилання такий:

```
<A href = "http://www.school.dn.ua/11.jpg">Фотографія 11-А </
a>
```

Підсумовуючи можливості мови HTML, можна сказати, що за допомогою різних тегів можна малювати таблиці, форматувати текст, вставляти в документ зображення, відео-, звукові файли та інше. В процесі свого розвитку все більша

увага приділялася тегу <style> та пов’язаними з ним каскадними таблицями стилів.

Каскадні таблиці стилів (Cascading Style Sheets, CSS) - це мова, яка містить набір властивостей для визначення зовнішнього вигляду документа. Специфікація CSS (CSS3 – на даний момент) визначає властивості і описову мову для встановлення зв'язку властивостей з елементами в документі. Розуміння таблиць стилів необхідно для додавання динамічного стилю сторінки. Під динамічним стилем (dynamic style) тут маємо на увазі модифікацією таблиці стилів, пов'язану з документом за допомогою сценарію.

На вузлі консорціуму W3C (www.w3.org) можна знайти останню інформацію про нововведення і елементах, підтримуваних таблицями стилів.

Таблиці стилів представляють собою абстракцію, в якій стиль документа визначається окремо від змісту або структури. Існує три методи додавання таблиць стилів в документ, доступних для Web-майстра - в цілому, з підвищенням рівня складності розширюються надані можливості з одночасним збільшенням ступеня абстракції. Перший метод полягає в використанні таблиці внутрішніх стилів (inline style sheet). Внутрішні стилі (inline styles) визначаються безпосередньо в елементі. Другий метод полягає в використанні таблиці глобальних стилів (global style sheet) для визначення стилю на початку документа. Третій, найбільш абстрактний і потужний метод полягає в використанні таблиці пов'язаних стилів (linked style sheet) для визначення стилю окремо в іншому документі.

Внутрішні стилі мало відрізняються від традиційного HTML. При використанні внутрішніх стилів зовнішній вигляд документа важко змінити. Перевага даного методу полягає в скороченому обсязі розмітки і в тому, що HTML може бути більш повноцінно використаний для подання вмісту презентації. Використання таблиці глобальних стилів дозволяє більш ефективно відокремити представлення від вмісту, а також дає можливість швидкої і незалежної зміни стилю і обробки документа. Використання таблиці

пов'язаних стилів дає більше переваг, представляючи вміст у вигляді набору сторінок або визначаючи цілий Web-вузол за допомогою одного файлу.

Термін cascading (каскадні) в назві CSS вказує на можливість злиття різних таблиць стилів для створення єдиного визначення стилю для елемента або для цілого документа. Це дозволяє проводити передбачуване злиття таблиці стилів Web-вузла з таблицею стилів документа і навіть з внутрішнім стилем.

Отже, внутрішній стиль (inline style) є по суті таблицею стилів для одиночного екземпляру елемента і його визначено безпосередньо у тегу елемента. Таблиця внутрішніх стилів визначається з використанням атрибута STYLE, а дані для атрибута визначаються за допомогою мови таблиці стилів. Нижче наведено код HTML, який збільшує шрифт відображення вмісту параграфа і вирівнює параграф по центру на жовтому фоні:

```
    <P STYLE = "font-size: 120%; text-align: center; background: yellow">
    Створює жовтий вирівняний по центру параграф з великим розміром шрифту.
    </ P>
```

Внутрішні стилі допомагають при вивченні мови внутрішніх стилів або за необхідності швидко змінити одиночний екземпляр елемента. Однак, внутрішні стилі не відповідають ідеології структурованого документа і погано працюють при необхідності зміни зовнішнього вигляду ряду елементів в документі, коли презентація та вміст розділені в повному обсязі. Для відділення стилю документа від його структури таблиця стилів повинна бути визначена в заголовку документа або як окремий файл, який пов'язаний з документом.

Наступною, більш ідеологічно вірною можливістю, є використання окремого тегу <STYLE> для завдання таблиць глобальних стилів, який зазвичай розміщується в заголовку документа. Розміщення усіх стилів документа в одному місці спрощує зміну режиму відтворення документа. Наведений нижче приклад таблиці стилів визначає зовнішній вигляд усіх параграфів в документі. Для зміни режиму відтворення параграфів потрібно

змінити тільки елемент STYLE. Якби використовувалися внутрішні стилі, то довелося б міняти кожен параграф в документі окремо.

```
<HTML>
  <HEAD>
     <STYLE TYPE = "text / css">
        P {font-size: 120%; text-align: center; background:
yellow}
     </ STYLE>
  </ HEAD>
  <BODY>
     <P> Все параграфи тепер більше і вирівняні по центру на
жовтому
        тлі. </ P>
  </ BODY>
</ HTML>
```

Для зв'язку стилю з певним елементом використовується селектор (selector). У наведеному вище прикладі був створений простий селектор, який пов'язаний зі стилем у всіх параграфах. Можуть бути також визначені більш функціональні контекстуальні селектори, що описуються нижче у цьому розділі.

Ще одним із трьох способів завдання стилів є введення таблиці пов'язаних стилів (linked style sheet), яка знаходиться у зовнішньому файлі. Перевага використання таблиці пов'язаних стилів полягає в тому, що всі правила і стилі можуть бути визначені і вміщені в одному файлі, який може бути спільно використаний багатьма сторінками або навіть цілим Web-вузлом. При використанні таблиці пов'язаних стилів обробка всіх параграфів цілого Web-вузла може бути змінена в одному документі. Таблиця пов'язаних стилів може також підвищити продуктивність, оскільки вона кеширується локально на диску клієнта, окремо від документа, так що кожен документ має менший розмір, а інформацію про стилі буде потрібно завантажити тільки один раз.

Для визначення таблиці пов'язаних стилів у заголовку документа розміщується тег <LINK>:

```
<HTML>
  <HEAD>
     <LINK REL = "stylesheet" TYPE = "text / css" HREF =
"fancy.css">
  </ HEAD>
```

```
        <BODY>
           <P>  This  document  uses  the  styles  specified  in
fancy.css. </ P>
        </ BODY>
     </ HTML>
```

Атрибут REL визначає, що пов'язаний файл є таблицею стилів, а атрибут TYPE визначає тип MIME таблиці стилів. Атрибут HREF є покажчиком URL, що вказує на зовнішню таблицю стилів. Таблиця пов'язаних стилів повинна містити тільки контекстуальні правила і визначення стилю і не може включати код HTML.

Для створення таблиці стилів всередині документа використовується той же синтаксис, який використовувався при створенні таблиці пов'язаних стилів. Мова CSS складається з селекторів і правил подання (presentation rules). Селектори (selectors) визначають елементи, які пов'язані з певним правилом, а правила подання встановлюють методи обробки даних елементів.

CSS містить два типи селекторів: прості і контекстуальні. Простий селектор пов'язує елемент на основі його атрибутів або типу незалежно від контекстуального положення усередині інших елементів. Контекстуальні елементи є більш потужними - вони можуть зв'язати правило з контейнерами певного елемента, наприклад, усі теги <EM> всередині тегів <P>.

У базовій формі простий селектор може бути створений для зв'язку певного елемента, класу елементів або ідентифікатора (ID) з певним стилем. Наведений нижче код демонструє ряд простих селекторів і їх правил подання:

```
      <STYLE TYPE = "text / css">
        / * Change all H1s to red. * /
        H1 {color: red}
        / * Встановлює напівжирний шрифт всіх елементів з тегом
CLASS = "Special" boldface. * /
        .special {font-weight: bold}
        / * Розміщує елемент з ідентифікатором ID = special на
жовтому тлі.*/
        #special {background: yellow}
        / * Встановлює висновок елементів H1 з тегом CLASS = "cool"
з великим інтервалом.* /
        H1.cool {letter-spacing: 2px}
     </ STYLE>
```

Селектори можуть бути перераховані через кому, що дозволяє одночасно описати кілька селекторів:

```
/ * Встановлює для всіх заголовків однакові правила. * /
H1, H2, H3, H4, H5, H6 {color: red; background: yellow}
```

Контекстуальні селектори визначають ієрархію контейнерів, з якою повинен бути пов'язаний стиль. Ієрархія контейнерів визначається порядком елементів в списку через кому. Наприклад, наведений нижче оператор визначає правило для всіх елементів EM, що знаходятцься в елементі P:

```
P EM {color: blue}
```

Кожен селектор може посилатися на теги CLASS, ID або тип елемента. Нижче наведена більш складна версія контекстуального селектора:

```
 / * Будь-який елемент з CLASS = "cool", який знаходиться всередині елемента LI з CLASS = "special" і далі перебуває всередині елемента UL, буде використовувати даний стиль. * /
UL LI.special .cool {font-weight: bolder; font-size: 120%}
```

Всі елементи контекстуального селектора є нечутливими до регістру - наприклад, .cool це те ж саме, що і .cOoL.

Псевдоклас (pseudo-class) складається з елементів одного типу, які задовольняють певному контекстуальному критерію. Наприклад, переглянуті елементи Anchor (посилання) представляють собою псевдоклас visited. Активні і не переглянуті посилання являють собою псевдокласи active і link, відповідно. Псевдоклас в таблиці стилів відділяється двокрапкою:

```
A: link {color: green}
: Link {color: green}
```

У другому прикладі опущено ім'я елемента (A), так як тільки посилання мають псевдоклас link. Псевдоклас може бути використаний так само, як клас або покажчик ID і є нечутливим до регістру.

На одні й ті ж самі елементи можуть посилатися кілька селекторів. CSS визначає послідовність каскадування (cascading order), яка використовується для вирішення проблем перекривання областей дії селекторів і правил. Послідовність каскадування об'єднує всі правила, які застосовуються до елементу, шляхом сортування на основі їх визначень. Наприклад, елемент

Strong, що знаходиться в елементі H1, може мати правила подання, визначені селектором H1, селектором STRONG і контекстуальним селектором для елементів Strong всередині елементів H1. Параметр каскадування в CSS визначає порядок об'єднання даних трьох правил. Загалом, правило для більш конкретного контекстуального селектора відкидає правило для менш конкретного селектора, а правила, подані нижче у початковій таблиці стилів або документі, мають більш високий пріоритет.

Докладно розглянувши каскадні таблиці стилів слід також подивитися основні особливості останнього дуже потужного засобу front-end-розробки, а саме – мови програмування JavaScript. Це спеціалізована мова програмування, яка традиційно використовується для управління об’єктною моделлю документа у браузері під час перегляду сторінок мережі Інтернет (існують продукти, що перетворюють JavaScript на мову загального призначення, команди якої виконуються на сервері, наприклад, Node.JS; але такий підхід сильно протирічить початковій концепції використання цієї мови програмування, і, за умови наявності значної кількості традиційних засобів back-end-розробки перетворення на серверну мову ще й JavaScript значна кількість програмістів вважає абсолютно недоцільною, тому про цей варіант використання говорити більше не будемо).

Особливістю JavaScript є те, що його висхідні програмні коди інтерпретуються, що є досить гнучким, але повільним рішенням.

Оператори у цій, в цілому C-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює C.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до

необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.

Мова JavaScript об’єктно-орієнтована, проте заснована на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані об’єкти, об’єкти браузера, об’єкти документа і об’єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням передбачається, що доступ до локальної файлової системи заборонений. Однак браузери можуть надавати спеціальні об’єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку виконання файлових операцій.

Сценарії JavaScript активно взаємодіють з об’єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, слід впровадити код сценарію в текст HTML-документа. Існує кілька способів зв'язати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розміщують всередині контейнерного тега <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде перебувати безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть прямо в HTML-документі або в спеціальних текстових файлах, які можна викликати з головного HTML-документа. Для початку розглянемо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь наступний текст намагається обробити як скриптовий код. І так до тих пір, поки не зустріне закриваючий тег </script>. Після цього всі наступні символи будуть вважатися HTML-текстом. Будь-який HTML-документ може містити довільне число «скриптових включень», але кожне має відкриватися і завершуватися

відповідним тегом. Від їх розташування у тілі HTML-документа іноді може залежати функціонування всієї Web-сторінки, але про це буде сказано пізніше.

Контейнерний тег <script> може містити атрибут SRC, який вказує ім'я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай використовують js:

```
<SCRIPT SRC = «myscripts.js»> </ SCRIPT>.
```

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

У браузерах, що потенційно підтримують сценарії, цю функцію користувач може відключити (зокрема, із міркувань підвищеної безпеки). Крім того, існують браузери, які принципово не підтримують сценарії. У даній ситуації бажано хоча б вивести повідомлення про те, що на сторінці був сценарій, але в даному конкретному випадку він не виконується. З цією метою в HTML-документі використовують контейнерний тег <noscript>, в якому розміщують текст, який з'явиться користувачеві за умови, що сценарій з тієї, чи іншої причини не виконуватиметься. Всі браузери, які підтримують сценарії, проігнорують вміст тегів <noscript> крім тих випадків, коли підтримка сценаріїв відключена. Браузери, що принципово не підтримують сценарії, навпаки, вміст тега <script> опустять (особливо, якщо він вкладений у теги <!-- -->, які є HTML-коментарем), а тега <noscript> - відобразять. Приклад наведено нижче:

```
<HTML> <HEAD>
<TITLE> Приклад застосування тега NOSCRIPT </ TITLE>
</ HEAD>
<SCRIPT TYPE = "text / JavaScript»>
<! -
alert ( «Підтримка JavaScript включена»); // ->
</ SCRIPT>
<NOSCRIPT>
<H2> Ваш браузер не підтримує JavaScript або його підтримка відключена </ H2>
```

```
</ NOSCRIPT>
</ HTML>
```

Тут був використана функція alert (повідомлення) для виведення повідомлень в маленькому діалоговому вікні.

Як уже зазначалося, в окремих файлах зазвичай розміщують бібліотеки функцій (визначення функцій), а також сценарії, які використовуються в декількох HTML-документах одного або декількох сайтів. Сценарій можна також писати у вигляді рядка операторів, між якими ставиться крапка з комою. Такий рядок, взятий в лапки, служить у якості значення атрибута-події, наприклад:

```
<IMG SRC = "mypicture.gif” ONCLICK = “alert ('Привіт!')”.
```

Виклики функцій і їх визначення можуть слідувати в довільному порядку, але тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо вони розміщуються у різних контейнерах <script>, то необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його необхідно завантажити в HTML-документ раніше викликів цих функцій.

При спробі завантажити і виконати неправильний варіант HTML-коду з'явиться діалогове вікно з повідомленням «Помилка: Передбачається наявність об’єкта». Браузер інтерпретує теги HTML послідовно. Так, зустрівши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції), розташованого в іншому контейнері <script>. Якщо ж визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується. Коли інтерпретатор зустрічає виклик функції, то він шукає її визначення в пам'яті і в разі успіху виконує код цієї функції.

Якщо необхідно, щоб сценарій виявився в браузері перш, ніж буде завантажено елементи HTML-документа, то його слід розташувати у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа.

2.2.2. Засоби Back-end розробки

Як зазначалося вище, під Back-end'ом мають на увазі програмні компоненти, що працюють на сервері, і вибір засобів для реалізації цих компонентів є вкрай широким – рис. 1.2, справа.

Умовно усі засоби для серверного веб-програмування можна поділити на два класи: окремі програми, що працюють відповідно до технології CGI та мови, висхідні коди яких вбудовуються прямо у веб-сторінку та проганяються препроцесором веб-серверу перед видачею у браузер клієнта. Другий підхід є більш зручним, про що свідчить все більше занепадання CGI-засобів. У якості альтернативи для них виступають такі серйозні продукти, як:

- серверні сторінки Java - JSP, що активно просуваються любителями цієї відкритої, сучасної мови програмування (загального призначення);

- активні серверні сторінки ASP від корпорації Microsoft (відповідно ступінь підтримки цього рішення надзвичайно високий);

- використання мови PHP, яку і обирає більшість розробників серверних додатків через цілий комплекс позитивних рис цього продукту, які розглянемо докладніше.

На сьогоднішній день PHP є найбільш поширеною мовою веб-програмування. Переважна більшість сайтів і веб-сервісів в Інтернеті написано за допомогою PHP. За деякими оцінками PHP застосовується більш ніж на 80 % сайтів, серед яких такі сервіси, як facebook.com, vk.com, baidu.com і інші. І така популярність не видається дивною. Простота мови дозволяє швидко і легко створювати сайти і портали різної складності.

PHP був створений в 1994 році датським програмістом Расмусом Лердорфом і спочатку являв собою набір скриптів на іншій мові програмування Perl. Пізніше цей набір скриптів був переписаний в інтерпретатор на мові C. Із самого виникнення PHP представляв зручний набір інструментів для спрощеного створення веб-сайтів і веб-додатків.

Розглянемо, які ж переваги надає PHP:

- для всіх найбільш поширених операційних систем (Windows, MacOS, Linux) є свої версії пакетів розробки на PHP, а це означає, що можна створювати веб-сайти на будь-якій з цих операційних систем;

- PHP може працювати в зв'язці з різними веб-серверами: Apache, Nginx, IIS, тощо;

- простота і легкість освоєння є особливістю цієї мови. Як правило, маючи навіть невеликий досвід в програмуванні на PHP, можна створювати простенькі веб-сайти;

- PHP схожий на мову C, тому, знаючи C, або одну з мов з C-подібним синтаксисом, буде простіше опановувати PHP;

- PHP підтримує роботу з великою кількістю систем баз даних (MySQL, MS SQL Server, Oracle, Postgres, MongoDB, та інші);

- поширеність хостингових послуг і їх дешевизна. Так як, як правило, хостингові компанії розміщують веб-сайти на PHP на веб-серверах Apache або Nginx, які працюють на одній з операційних систем сімейства Linux. І веб-сервери, і операційні системи на базі Linux безкоштовні, що знижує загальну вартість використання хостингу;

- постійний розвиток, адже PHP продовжує розвиватися, виходять все новіші версії, які несуть нові функції, адаптуючи мову програмування до нових викликів. І, як правило, перехід на нову версію не викликає ніяких труднощів;

- існує надзвичайно багато якісних Інтернет-ресурсів з підтримки процесу програмування на PHP, де можна задати власне питання і найскоріше отримати кваліфіковану відповідь.

2.3. Проектування бази даних розроблюваного ресурсу

Жодний справжній (не учбовий) Інтернет-ресурс не може обійтися без використання бази даних (далі – БД). Інформацію слід в якомусь місці та вигляді зберігати, і база даних якраз є стандартизованим способом збереження структурованих відомостей самого широкого плану. У попередньому реченні

особливу увагу слід звернути на слово «структуровані», тобто перед будь-яким використанням БД (внесенням, пошуком, зміною чи видаленням даних), інформацію необхідно структурувати, тобто фактично розібратися якими елементарними поняттями (полями БД) описується предметна область, та як їх сгрупувати (формуючи таблиці БД), а також встановити зв'язки між групами (таблицями).

Але, перш ніж виконувати структурування, або інші перетворення інформації, слід вибрати інструментальний засіб для збереження бази даних ресурсу, що розробляється. Оскільки у якості серверної мови програмування було обрано PHP, то традиційно з ним використовують продукт MySQL, що є системою управління базами даних із численною кількістю переваг, у порівнянні з конкурентами:

- порівняно легкий продукт, що займає малу кількість оперативної пам'яті сервера (а також, що менш критично, і малі обсяги постійної пам'яті жорсткого диску сервера);

- порівняно функціональний продукт, що включає практично усі основні функції по обробці даних та створенню ефективних БД;

- це програмне забезпечення є вільним для використання (на відміну від деяких корпоративних продуктів, ліцензії на які можуть коштувати тисячі умовних одиниць, тобто сотні тисяч грн.);

- існує значна кількість уже розробленого програмного забезпечення (фрагментів, модулів, і т.п.) для роботи з цією СУБД;

- існує чимало Інтернет-ресурсів, на яких здійснюється технічна підтримка програмістів, що здійснюють розробку для MySQL.

Зважаючи на усі ці особливості, СУБД MySQL береться за основу для збереження даних, що використовуватиме Інтернет-ресурс, який розробляється.

Тож, повертаючись до теми розробки конкретної бази даних, вкажімо, що проектувати БД можна відповідно до загальнонаукових принципів синтезу («знизу-догори», починаючи розглядати найпростіші поняття та елементи і об'єднуючи їх у все ширші, доходячи до рівня однієї цілісної бази) або аналізу

(«зверху-вниз», починаючи із найширшого поняття всієї бази і деталізуючи, з яких саме елементів все нижчого рівня вона має складатися). Обираємо метод аналізу і будемо вибудовувати структуру БД відповідно до наступних правил:

- увесь матеріал електронного навчального курсу розділяємо на окремі теми (Theme);

- теми будемо ділити на розділи (Section);

- в рамках одного розділу можуть співіснувати наступні сутності:

а) окремі текстові блоки теоретичного матеріалу або практичних прикладів (TextBlock);

б) зображення (Image), що доповнюють певний текстовий блок (як крайній випадок, текстовим блоком може бути і одне слово, як, наприклад, проста назва рисунку);

в) гіперпосилання (Anchor), що відносяться до якогось текстового блоку, або зображення;

г) завдання (Question), що можуть використовуватися для різних цілей: самостійної проробки і самоконтролю, рубіжного контролю, підсумкового та вихідного контролю.

Отже, виділяємо у структурі БД шість таблиць, детальна інформація про які наведена нижче:

а) таблиця themes, що містить перелік усіх доступних для вивчення тем (найвищий рівень розбиття навчального матеріалу), кількість тем є невеликою (прийнято 6 тем):

- вибір елементів сторінки із DOM;

- маніпулювання елементами сторінки;

- робота з подіями;

- візуальні ефекти;

- робота з технологією AJAX;

- створення оригінальних елементів управління.

Таблиця themes міститиме наступні поля:

- ThemeID – ідентифікатор теми; це число буде виступати первинним ключем цієї таблиці;

- ThemeName – безпосередньо сама тема (її назва);

- ThemeOrder – номер цієї теми по порядку при вивченні всього курсу, що розглядається;

- ThemeInfo – короткий текстовий опис кожної теми загального характеру.

Типи кожного поля можна подивитися на рис. 2.4, де показана структура цієї таблиці.

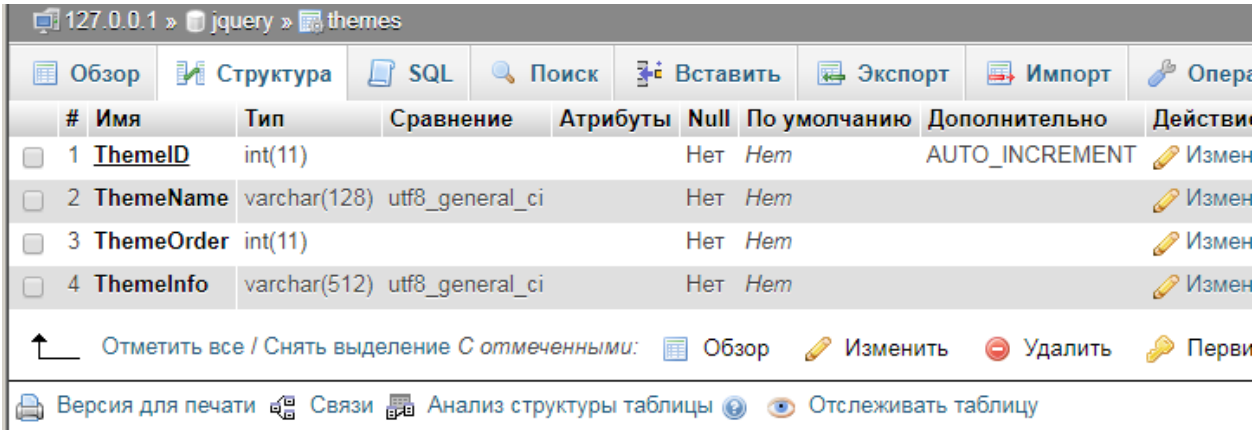


Рис. 2.4. Структура таблиці themes.

б) таблиця sections містить перелік усіх розділів (секцій), що відносяться до усіх без виключення. Кількість розділів, на які розбивається кожна тема невелика і складає число менше 10. Ця таблиця містить наступні поля:

- SectionID – ідентифікатор розділу, який є первинним ключем цієї таблиці;

- SectionName – назва розділу;

- SectionTheme – ідентифікатор тієї теми, до якої відноситься даний розділ; це – зовнішній ключ таблиці.

Інформацію про поля таблиці можна продивитися на рис. 2.5, де показана її структура.

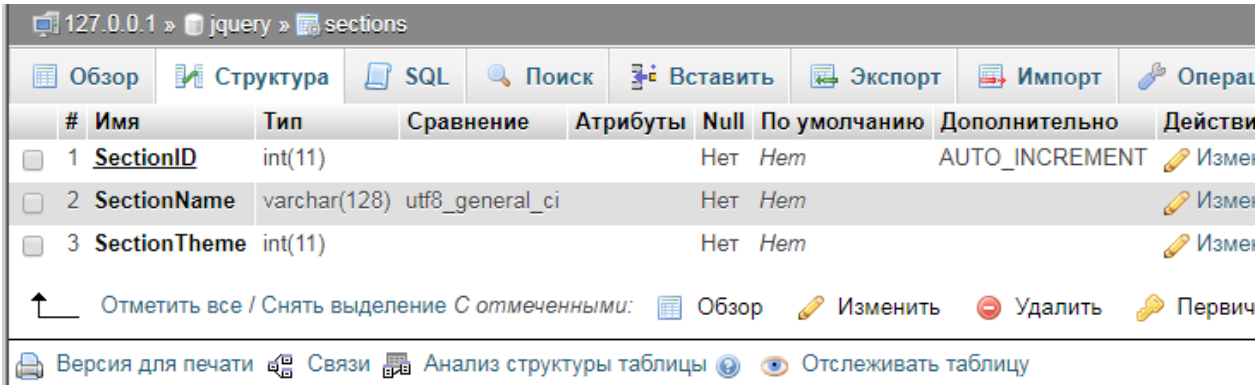


Рис. 2.5. Структура таблиці sections.

в) таблиця textblocks включає інформацію про текстові блоки, які є складовими розділів. Число текстових блоків, що утворюють розділ, не лімітується, тому блоки можуть бути як завгодно малими (до одного слова мінімум). Для забезпечення можливості гарного форматування тексту для кожного текстового блоку задається тип шрифту та його розмір. Таблиця textblocks складається з наступних полів:

- TextBlockID – ідентифікатор текстового блоку, що є первинним ключем таблиці;

- TextBlockText – безпосередньо текст, що утворює блок;

- TextBlockFontName – тип шрифту, яким слід відображувати даний текстовий блок;

- TextBlockFontSize – розмір шрифту, яким слід відображувати даний текстовий блок;

- TextBlockSection – ідентифікатор розділу, частиною якого є даний текстовий блок; це – зовнішній ключ таблиці;

- TextBlockNumberInSection – порядковий номер текстового блоку у розділі, до якого він належить.

Більш конкретну інформацію про назву та тип полів цієї таблиці можна знайти на рис. 2.6.

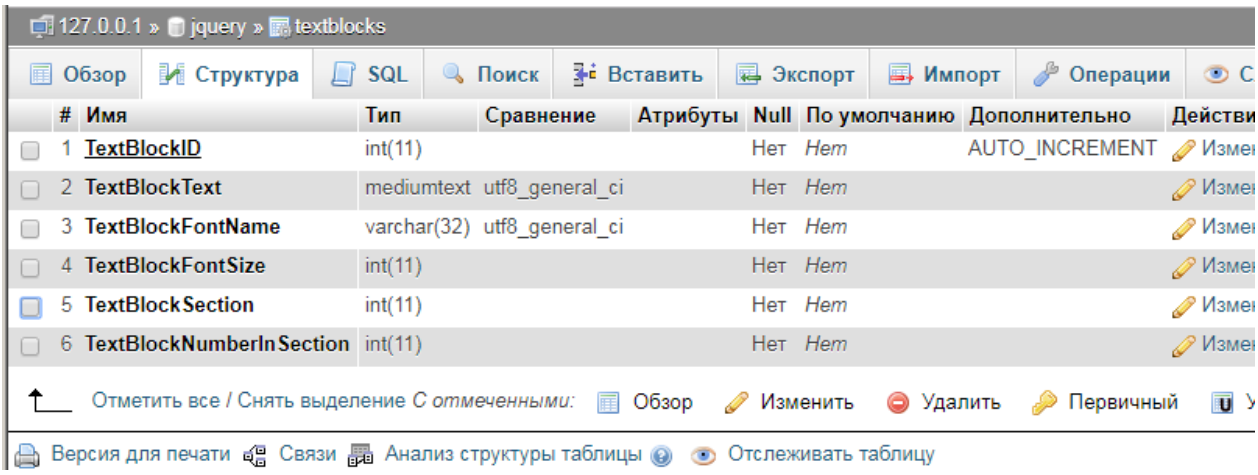


Рис. 2.6. Структура таблиці textblocks.

г) таблиця images описує зображення, які прикріпляються до текстових блоків: до одного текстового блоку можна прикріпити одне зображення. Зважаючи на те, що текстові блоки можуть бути як завгодно малими, а їх загальна кількість не є обмеженою, умова «одне зображення на один текстовий блок» ніяк не утискає розробників електронних курсів для даної платформи. Перелік полів є наступним:

- ImageID – ідентифікатор зображення, який є первинним ключем таблиці;

- ImageURL – адреса зображення у вигляді універсального локатора ресурсу (URL);

- ImageTextBlock – ідентифікатор текстового блоку, до якого відноситься (прикріплюється) дане зображення; це поле є зовнішнім ключем таблиці.

Структура таблиці показана на рис. 2.7.

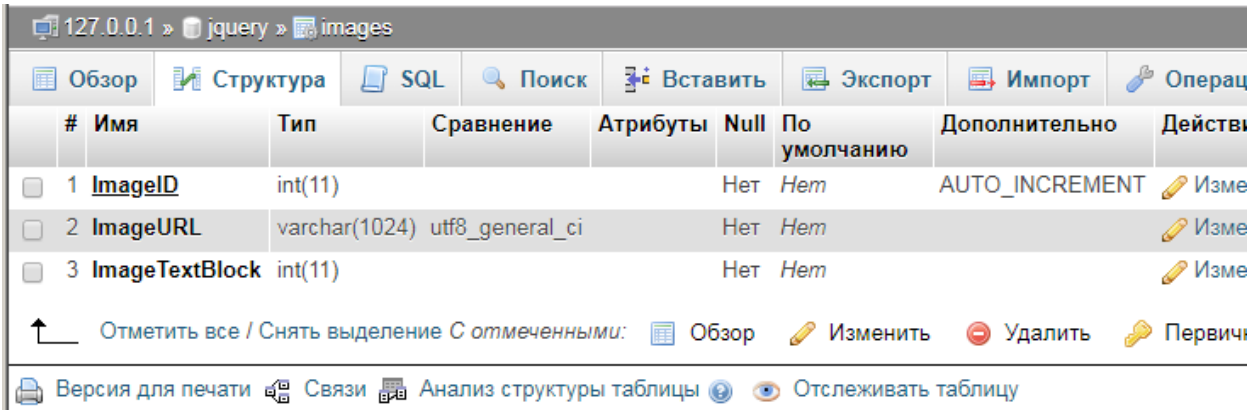


Рис. 2.7. Структура таблиці images

д) таблиця anchors описує зображення, які прикріпляються до текстових блоків: до одного текстового блоку можна прикріпити одне зображення. Зважаючи на те, що текстові блоки можуть бути як завгодно малими, а їх загальна кількість не є обмеженою, умова «одне зображення на один текстовий блок» ніяк не утискає розробників електронних курсів для даної платформи. Перелік полів є наступним:

- AnchorID – ідентифікатор посилання, який є первинним ключем таблиці;

- AnchorLink – адреса, за якою треба перейти браузеру, при щиглі мишею на гіперпосиланні;

- AnchorTextBlock – ідентифікатор текстового блоку, до якого відноситься (прикріплюється) дане посилання; це поле є зовнішнім ключем таблиці;

- AnchorImage – ідентифікатор зображення, до якого відноситься (прикріплюється) це ж саме посилання; це – зовнішній ключ таблиці.

Посилання може відноситися до текстового блоку, або до зображення, або одне і те саме посилання може відноситися одночасно до якогось текстового блоку та зображення.

Більш докладна інформація про структуру цієї таблиці наводиться на рис. 2.8.

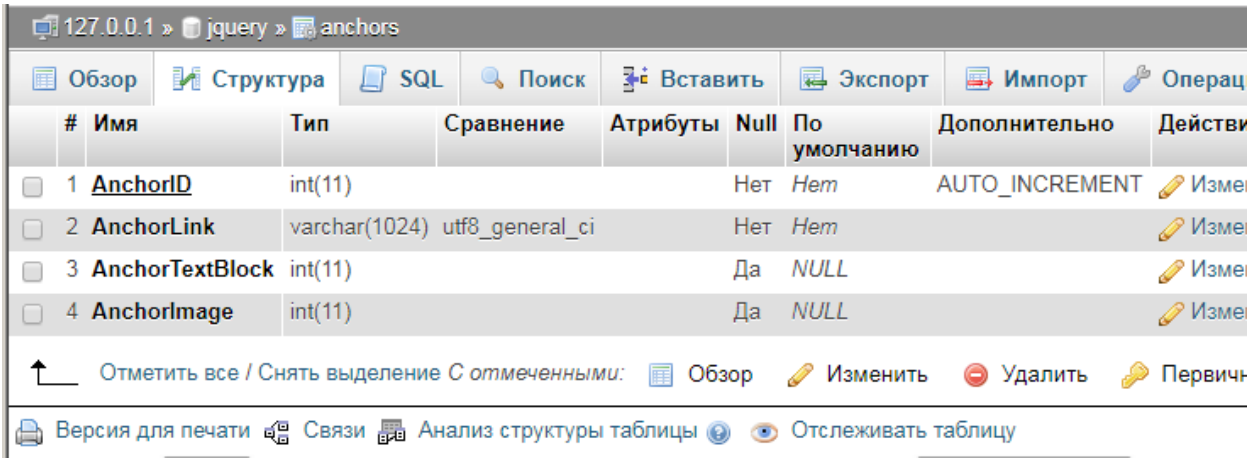


Рис. 2.8. Структура таблиці anchors.

е) у таблиці questions містяться тестові питання, які можуть використовуватися як контрольні для самоконтролю, або для інших цілей. Питання прикріплюються до окремих розділів. Перелік полів, якими описується конкретне питання, є наступним:

- QuestionID – ідентифікатор чергового питання, що є первинним ключем таблиці;

- QuestionSection – ідентифікатор розділу, до якого відноситься дане питання, зовнішній ключ таблиці;

- QuestionText – текст тестового питання;

- QuestionVar1 – текст варіанту відповіді №1;

- QuestionVar2 – текст варіанту відповіді №2;

- QuestionVar3 – текст варіанту відповіді №3;

- QuestionVar4 – текст варіанту відповіді №4;

- QuestionAnswer – номер варіанту, який є правильною відповіддю на дане питання;

- QuestionImageURL – адреса зображення, яке може додаватися до тексту питання;

- QuestionExplanationTextBlock – ідентифікатор текстового блоку, який пов’язаний із даним питанням, пояснює його за умови, що користувач надав неправильну відповідь; це поле є зовнішнім ключем таблиці.

Найбільш повна інформація про структуру таблиці з питаннями наведена на рис. 2.9.

127.0.0.1 » jquery » questions

Обзор | Структура | SQL | Поиск | Вставить | Экспорт | Импорт | Операции | Слежен

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действ
1	QuestionID	int(11)			Нет	Нем	AUTO_INCREMENT	Изме
2	QuestionSection	int(11)			Нет	Нем		Изме
3	QuestionText	varchar(1024)	utf8_general_ci		Нет	Нем		Изме
4	QuestionVar1	varchar(256)	utf8_general_ci		Нет	Нем		Изме
5	QuestionVar2	varchar(256)	utf8_general_ci		Нет	Нем		Изме
6	QuestionVar3	varchar(256)	utf8_general_ci		Нет	Нем		Изме
7	QuestionVar4	varchar(256)	utf8_general_ci		Нет	Нем		Изме
8	QuestionAnswer	int(11)			Нет	Нем		Изме
9	QuestionImageURL	varchar(1024)	utf8_general_ci		Нет	Нем		Изме
10	QuestionExplanationTextBlock	int(11)			Нет	Нем		Изме

Отметить все / Снять выделение С отмеченными: Обзор | Изменить | Удалить | Первичный | Уникал

Версия для печати | Связи | Анализ структуры таблицы | Отслеживать таблицу

Рис. 2.9. Структура таблиці questions.

За наведеними відомостями будуємо таблиці, встановлюємо зв’язки між ними, та реалізовуємо БД jQuery, схема даних якої наведена на рис. 2.10.

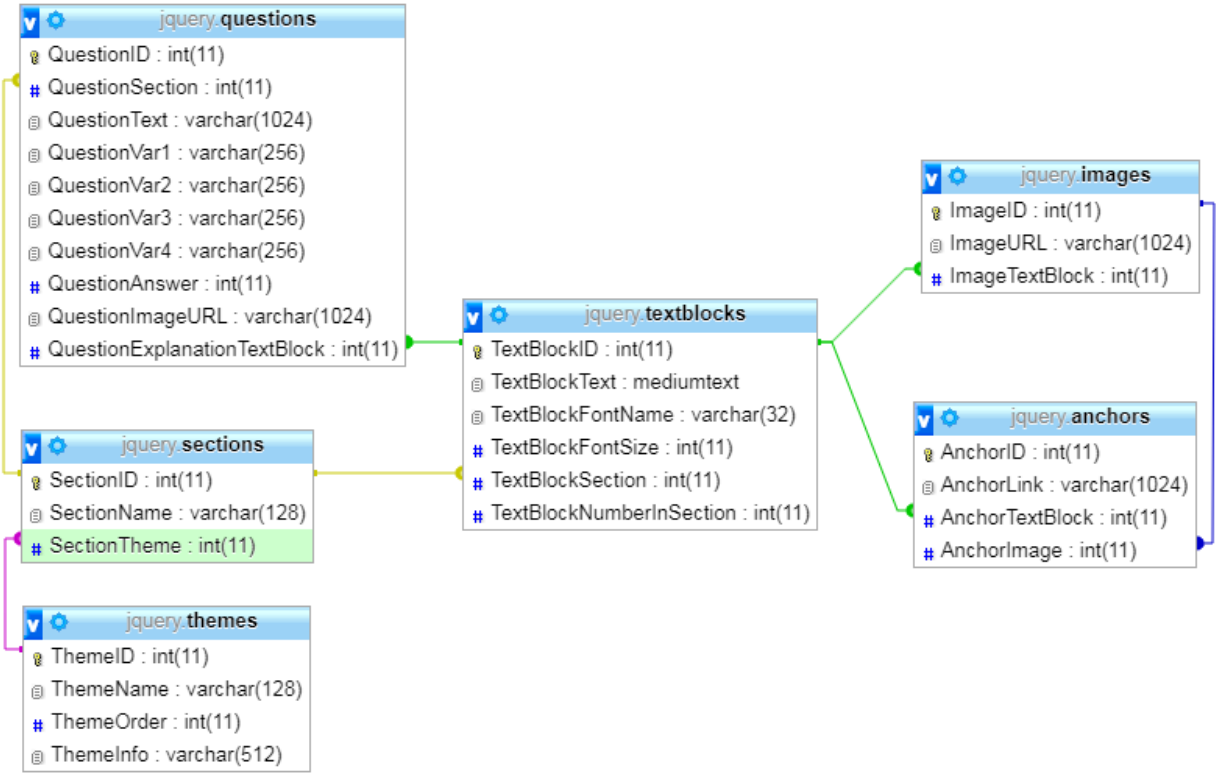


Рис. 2.10. Схема даних спроектованої БД для вивчення бібліотеки jQuery.

Таким чином, у підрозділі спроектовано базу даних, необхідну для створення та підтримки електронного навчального курсу з вивчення бібліотеки jQuery.

2.4. Зовнішній вигляд сторінок системи (інтерфейс користувача)

При проектуванні інтерфейсу користувача слід визначитися із наступними моментами, що на нього впливають:

- встановити технології, що змінюють (можуть змінювати) поведінку користувача в системі, а, отже, і особливості її інтерфейса;

- визначити найважливіші елементи, що відображатимуться на «головному екрані» програмного забезпечення, яке проектується;

- встановити перелік опціональних елементів управління, які можуть викликатися за командою користувача.

Оскільки проектуванню підлягає веб-система, то відповідними є і технології, що визначають її користувацький інтерфейс. В першу чергу, це використання динамічного HTML, і, як споріднений факт – активне використання технології AJAX.

На ранніх етапах функціонування World Wide Web під час роботи в Інтернет відбувалися часті перезавантаження сторінок, навіть після виконання кожної елементарної дії користувача, що не є ознакою гарного інтерфейсу. Тому тут будемо орієнтуватися на технологію AJAX, яка дозволяє здійснювати активний обмін інформацією (як завгодно інтенсивний, включаючи звернення до баз даних, що і потрібно для цілей даної роботи), без перезавантаження всієї сторінки.

AJAX (англ. Asynchronous Javascript and XML) - спосіб побудови призначених для користувача web-додатків за допомогою фонового обміну інформацією браузера з сервером. Термін AJAX ввів Джессі Джеймс Гаррет у 2005 році. Першими додатками, що використовували дану технологію, стали сервіс карт Google Maps і поштовий клієнт Gmail.

Використання AJAX для розкрутки сайту дозволяє поліпшити його юзабіліті (додатки стають більш зручними і швидкими для відвідувачів), покращуються функціональність і зовнішній вигляд сторінок.

Коротко розглянемо принцип роботи цієї технології.

AJAX базується на технології звернення до сервера без перезавантаження сторінки (XTMLHttpRequest, створення дочірніх фреймів або тега script) або використанні DHTML, що дозволяє динамічно змінювати вміст. Формат передачі даних - XML або JSON. AJAX можна реалізувати в різних мовах програмування: PHP, Ruby on Rails, ASP.NET і інших. У коді web-сторінок широко використовується JavaScript для прозорого обміну даними клієнта з сервером. Користувачі взаємодіють зі стандартними HTML елементами, динамічна поведінка яких описується на JavaScript.

Для просування сайту застосування такої технології, як AJAX, має ряд суттєвих переваг:

- економія трафіку користувача (замість оновлення всієї сторінки, завантажується її невелика частина, що змінилася);

- зниження навантаження на сервер. Наприклад, на сторінці особистих повідомлень форуму при виділенні користувачем прочитаних листів сервер вносить зміни в БД і відправляє скрипту клієнта відповідь про виконання операції без повторного створення сторінки і її передачі;

- прискорення реагування інтерфейсу на команди користувача.

В той же час AJAX має і свої недоліки, серед яких можна назвати наступні:

- не завжди можлива інтеграція зі стандартним набором інструментів браузера, а саме, так як Інтернет-переглядачі не реєструють в історії AJAX-переходи по сторінках, не можна скористатися кнопкою «Назад». У деяких випадках немає можливості додати в закладки потрібний матеріал;

- контент, що завантажується динамічно, не доступний пошуковим системам, тому необхідно забезпечити альтернативний доступ до вмісту ресурсу;

- здійснюється неправильний облік статистики переміщення користувача по сайту;

- відбувається ускладнення програмного контролю цілісності типів і форматів, так як процеси форматування даних частково переносяться на сторону клієнта;

- в браузері користувача повинен бути включений JavaScript (не дуже суттєве на сьогоднішній день уточнення, оскільки з кожним роком частка користувачів із увімкненим JavaScript наближається до 100 %).

Альтернативою AJAX виступають Java-аплети, JavaFX, технології ActionScript3, Flash Remoting, Adobe Flex, складові технологічну основу Rich Internet Applications від Macromedia, і Silverlight від корпорації Microsoft.

Таким чином, враховуючи усі особливості технології, будемо використовувати AJAX для організації роботи інформаційної системи з підбору вищого навчального закладу для абітурієнтів, в результаті чого увесь додаток матиме лише одну робочу сторінку – рис. 2.11.

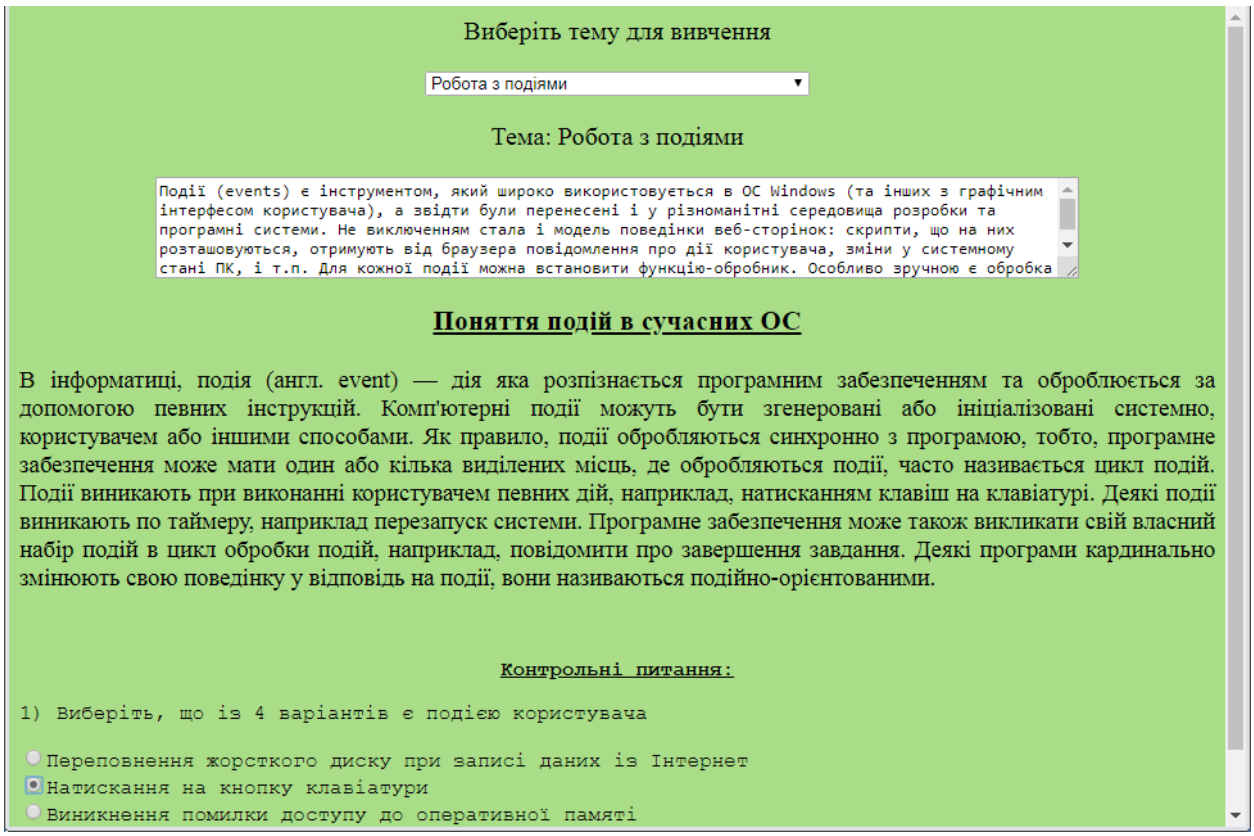


Рис. 2.11. Зовнішній вигляд вікна розробленого електронного курсу.

У верхній частині вікна розміщений елемент для вибору теми для вивчення. Нижче відображується її короткий опис. Далі розпочинаються розділи цієї теми, прописані у БД (для прикладу показано лише один розділ, щоби вмістити контрольні питання, які ідуть після теоретичного матеріалу).

Змінюючи елемент управління вибором теми, розміщений на цій сторінці, користувач буде ініціювати AJAX-запити, що швидко доставлятимуть назад усю необхідну інформацію без перезавантаження цілої сторінки. Тим самим досягається цілісність сприйняття системи, що реально видається для користувача одним додатком (на зразок Desktop-додатків для PC). Відсутність перезавантаження сторінок робить роботу в системі зручною та комфортною.

В той же час, використання технології AJAX вимагає залучення значної кількості сценаріїв, що обробляють запит (зазвичай POST), та надсилають назад певним чином сформовані дані. Для простих задач формат цих даних може визначатися програмістом, а при необхідності передачі великих обсягів даних слід використовувати формати представлення даних XML або JSON (останній став дозволеним для використання у стандарті HTML5). Підкреслимо, що для цілей даної роботи по мережі будуть передаватися порівняно малі обсяги інформації (орієнтовно кажучи, до десятків рядків SQL-запиту), тому доцільним є використання власного формату відповіді (розглянуто у наступному підрозділі).

2.5. Особливості програмної реалізації розроблюваного ресурсу

Беручи до уваги, що в роботі прийнято рішення про повне використання AJAX (і, отже, існування тільки однієї сторінки, що бачить користувач), може представляти інтерес питання організації всієї системи, з точки зору окремих файлів, яке наведемо на рис. 2.12.

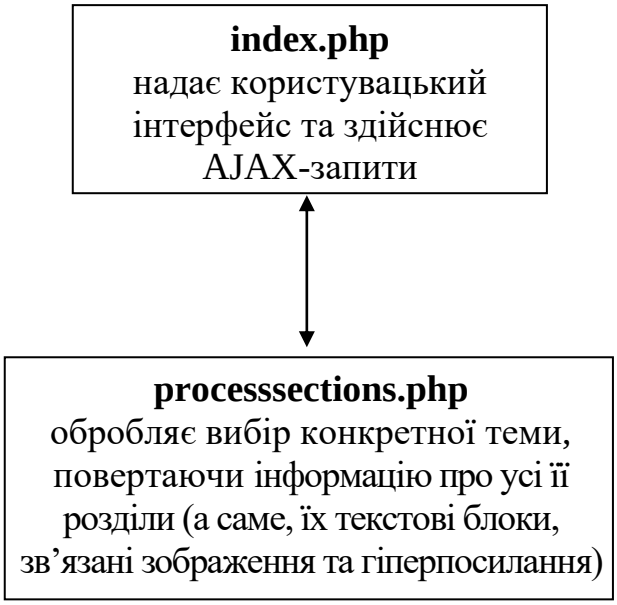


Рис. 2.12. Файлова структура електронного навчального курсу для вивчення бібліотеки jQuery.

Допоміжний файл processsections.php працює за наступною схемою:

- підключення до бази даних;

- формування та виконання SQL-запиту (на базі інформації із POST-запиту, здійсненого інтерфейсною частиною index.php);

- формування та друк у відповідь текстового повідомлення, що включає в себе усі необхідні результати пошуку в базі даних, причому яке записано за певними правилами.

Інформація у текстовому повідомленні повинна розділятися певними маркерами. У якості такого маркеру використано символ «*»: блок інформації про особливості обраної теми (ThemeInfo), відділяється від блоку інформації про її розділи саме зірочкою.

З точки зору структури проекту, інтерес також представляє набір функцій (користувача), що застосовані в роботі. Усі вони розміщуються у файлі index.php, який фактично являє собою основну програму проекту, і, відповідно до обраного вище структурного підходу до програмування складається із

окремих функцій. Розглянемо докладніше їх перелік та призначення кожної з них:

- fillsections(e) – функція, що обробляє відповідь на AJAX-запит, пов'язаний із отриманням інформації про обрану тему, а точніше, її опис та перелік розділів, які у ній містяться;

- ChangedThemeSelect() – обробник процесу змінення випадаючого списку з темами, надсилає AJAX-запит для отримання інформації про розділи цієї теми, текстові блоки, що до них входять, зображення, гіперпосилання та тестові питання;

- CheckAnswer(radio) – функція перевірки, чи є відповідь на тестове запитання правильною; аргументом є ідентифікатор тієї радіокнопки із групи, що відповідає правильній відповіді, а функція перевіряє, чи вибрана (checked) ця радіо кнопка – тоді видається вітання, або ні – видається попередження.

Важливим елементом при розробці ПЗ є також набір змінних, за допомогою яких і функціонує програма. Аналізуючи файл index.php зверху-вниз, випишемо усі його глобальні змінні, відмічаючи мову програмування, у якій використовується та, чи інша змінна, а також даючи пояснення щодо її суті та призначення:

- ThemeSelected (змінна JavaScript) – обрана користувачем системи тема із курсу jQuery, яку він хоче вивчати у даний момент;

- $hostname (змінна PHP) – змінна, яка містить адресу сервера, на якому є встановленою СУБД MySQL;

- $username (змінна PHP) – містить ім'я користувача, що використовується при підключенні до СУБД MySQL;

- $password (змінна PHP) – містить пароль користувача, що використовується при підключенні до СУБД MySQL;

- $dbName (змінна PHP) – містить назву бази даних, що використовується всією системою;

- $link (змінна PHP) – містить змінну, яка використовуватиметься для ідентифікації зв'язку з БД що використовується програмою;

- $query (змінна PHP) – містить текст запиту SQL, який передаватиметься та виконуватиметься СУБД;

- $result (змінна PHP) – використовується для отримання результату виконання запиту SQL;

- $num_rows (змінна PHP) – містить кількість рядків, що повернуті СУБД після виконання останнього запиту SQL;

- $i (змінна PHP) – традиційна назва для лічильника циклу;

- $row (змінна PHP) – змінна, що містить черговий рядок із великої їх кількості, що повернуті в результаті виконання останнього запиту SQL.

Також розглянемо локальні змінні по окремим функціям.

1) Для функції fillsections(e):

- blocks (змінна JavaScript) – масив із двох елементів, що являють собою великі блоки інформації, на які розділяється (зірочкою «*») весь текст, що повертається на веб-сторінку у результаті AJAX-запиту;

2) Для функції ChangedThemeSelect():

- index (змінна JavaScript) – допоміжна змінна для швидкого звертання до вибраного пункту випадаючого списку вибору теми;

- data (змінна JavaScript) – змінна, необхідна для імітації даних, прикріплених до HTTP-запиту типу POST, який виконується за технологією AJAX;

- req (змінна JavaScript) – змінна, що є об’єктом типу XMLHttpRequest і за допомогою якої здійснюється AJAX-запит.

З алгоритмічної точки зору інтерес може представляти структура файлу processsections.php. Послідовність виконання запитів до бази даних у ньому є наступною:

а) першим виконується запит інформації про тему, обрану у випадаючому списку:

```
$query="SELECT ThemeInfo,ThemeID FROM themes WHERE ThemeName='".$_POST["theme"]."';";
```

б) наступний запит повертає усі розділи, які існують у даній обраній темі:

```
$query="SELECT SectionID, SectionName FROM sections WHERE SectionTheme='".$row['ThemeID']."';";
```

в) розпочинається зовнішній великий цикл for з лічильником $i , у якому виконуються наступні операції для всіх розділів, знайдених на попередньому кроці:

- запитуються усі текстові блоки, з яких утворений поточний розділ:

```
$query2="SELECT TextBlockID, TextBlockText, TextBlockFontName, TextBlockFontSize FROM textblocks WHERE TextBlockSection='".$row['SectionID']."';";
```

- розпочинається вкладений цикл for з лічильником $i2, у якому виконуються наступні операції для всіх текстових блоків, знайдених на попередньому кроці:

1) запитується, чи є у базі гіперпосилання на поточний текстовий блок, що розглядається на черговій ітерації циклу:

```
$query3="SELECT AnchorLink FROM anchors WHERE AnchorTextBlock='".$row2['TextBlockID']."';";
```

2) якщо на попередньому кроці у базі знайдено гіперпосилання для поточного текстового блоку, то він обрамляється відповідним тегом …;

3) запитується, чи закріплене за даним текстовим блоком зображення:

```
$query4="SELECT ImageID,ImageURL FROM images WHERE ImageTextBlock='".$row2['TextBlockID']."';";
```

4) якщо на попередньому кроці знайдено зображення, закріплене за поточним текстовим блоком, то запитується, чи немає у базі гіперпосилання для цього зображення:

```
$query5="SELECT AnchorLink FROM anchors WHERE AnchorImage='".$row4['ImageID']."';";
```

5) якщо на попередньому кроці у базі знайдено гіперпосилання для поточного зображення, то воно обрамляється відповідним тегом ….

- виконується запит про питання, закріплені за даним розділом:

```
$query6="SELECT QuestionText, QuestionVar1, QuestionVar2, QuestionVar3, QuestionVar4,QuestionAnswer FROM questions WHERE QuestionSection='".$row['SectionID']."';";
```

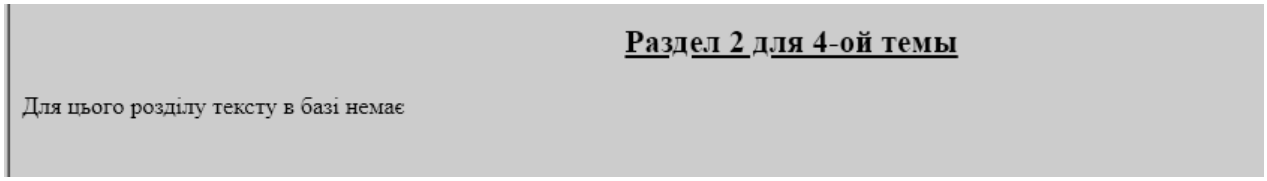
- розпочинається цикл for з лічильником $i6, у якому виконується формування і запис у веб-сторінку тестових питань, знайдених на попередньому кроці.

Як видно із опису алгоритму роботи файлу processsections.php, процес формування сторінки із навчальним матеріалом, що відповідає одній обраній темі, є досить складним і оригінальним, тому можна сказати, що розроблено новий «движок» навчальної платформи, що відрізняється простотою використовуваних елементів та, отже, доброю швидкістю роботи.

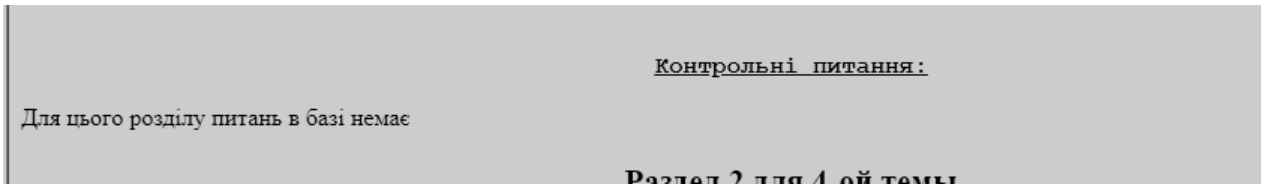
2.6. Тестування та результати роботи спроектованої системи

Головне вікно електронного навчального курсу уже було наведено на рис. 2.11. Проведено тестування роботи системи для усіх запланованих тем, які наповнені відповідними розділами, для кожного з яких вказано необхідні текстові блоки, зображення та гіперпосилання. Встановлено, що робота системи відбувається адекватно поставленій задачі та не викликає помилок.

Система містить різні елементи захисту від некоректних ситуацій. Наприклад, якщо у базі даних відсутня якась інформація, необхідна для заповнення текстових блоків, або контрольних питань, то користувачеві ресурсу видається про це одне із попереджувальних повідомлень – рис. 2.13.



а)



б)

Рис. 2.13. Попереджувальні повідомлення у випадку, якщо для обраних користувачем розділів відсутнє наповнення у базі даних: *а* – немає текстових блоків; *б* – немає контрольних питань.

Після тривалої роботи з електронним навчальним курсом для вивчення бібліотеки jQuery мови JavaScript встановлено, що система працює у штатному режимі без виникнення аварійних ситуацій, що в першу чергу обумовлене чіткою взаємодією серверної частини (файлу processsections.php) та (в основному) клієнтської частини index.php (хоча в ній є невелика частина коду доступу до БД), а також використанням надійної системи управління базами даних.

Після тестування та відлагоджування системи у локальному режимі була виконана публікація сайту на хостингу в мережі Інтернет. Слід відмітити, що задача пошуку відповідного серверу (тобто, фактично, компанії), що відповідає необхідним вимогам не є простою задачею, а саме хостинг-провайдер повинен відповідати наступним умовам:

- надавати хостинг безкоштовно, оскільки весь розроблений ресурс присвячено процесу навчання, причому на безкомерційній основі;

- вбудовувати мінімум сторонньої реклами у сторінки ресурсу (або в ідеалі не використовувати її взагалі);

- надавати можливість завантаження власних готових веб-сторінок, причому з їх обробкою PHP-інтерпретатором;

- надавати можливість використання системи управління базами даних MySQL (оскільки саме ця СУБД використана у даній роботі; слід пам’ятати про ребрендинг цієї марки та її нову назву MariaDB);

- бути доступним цілодобово (деякі хостинги мають час «відпочинку» для безплатних ресурсів, що може сягати кількох годин протягом доби).

Усім цим вимогам відповідає сервер 000webhost.com, який і було обрано у якості базового для розміщення розробленого ресурсу в мережі Інтернет. Адреса сайту https://jquery2019.000webhostapp.com/, за якою він є доступним для будь-кого в мережі Інтернет. Для адміністративного входу було зареєстровано спеціальну поштову скриньку на порталі Gmail і вхід до адмінпанелі виконувався за соцмережею Google на основі цієї скриньки.

Оскільки в цілому робота спроектованого ресурсу не відрізняється у локальному та публічному режимах, то тут можна лише коротко зауважити, що функціонування сайту є стабільним та виконано повністю у рамках завдання на дипломну роботу.

Підсумовуючи даний розділ, можна сказати, що було успішно спроектовано та реалізовано конкретний програмний продукт – електронний навчальний курс для вивчення бібліотеки jQuery мови програмування JavaScript. При цьому створено оригінальний «движок» на базі мови PHP та СУБД MySQL, який потім наповнено корисною навчальною інформацією, довівши увесь продукт до завершеного практично корисного стану.

ВИСНОВКИ

В роботі виконано розробку електронного навчального курсу для вивчення бібліотеки jQuery, написаної для мови JavaScript. Рішення виконано у вигляді односторінкового веб-додатку та реалізоване на основі сучасної актуальної зв’язки наступних мов програмування, засобів та технологій:

- JavaScript;
- CSS;
- PHP;
- MySQL;
- AJAX.

Інтерфейс сайту зручний для користувача, не вимагає постійного перезавантаження робочої сторінки, має деякі динамічні елементи управління. Усі ці фактори (беручи до уваги, що сайт належним чином наповнений корисною навчальною інформацією) підвищують імовірність реального практичного використання сайту сторонніми особами при вивченні окремого курсу jQuery або відповідного модуля в рамках курсу веб-програмування.

В процесі створення програмної складової дипломної роботи було виконано аналіз сучасних засобів та інструментів для веб-розробки (причому як фронт-енд частини, так і бек-енд) та обґрунтовано використання саме серверної мови програмування PHP разом із клієнтською програмною компонентою, реалізованою на JavaScript.

Також спроектовано структуру бази даних, у якій мають зберігатися відомості, необхідні для функціонування електронного навчального курсу. У її структурі виділено 6 таблиць, причому всі вони несуть інформаційний характер. Весь матеріал пропонується ділити на окремі укрупнені теми, а теми – на розділи. Кожен розділ може складатися із значної кількості текстових блоків, причому до деяких із них, за необхідності, можна кріпити зображення. Окремі текстові блоки та окремі зображення можуть бути оформлені гіперпосиланнями для організації зв’язку частин електронного навчального

курсу між собою та з іншими доступними, зокрема в мережі Інтернет, матеріалами. Ресурс розміщено у мережі Інтернет для загального доступу.

В цілому, можна сказати, що задача магістерської кваліфікаційної роботи є виконаною, а її мету можна вважати досягнутою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Вишня В. Б. Інформаційні системи та технології: підруч. Запоріжжя: ЗНУ, 2021.
2. Вишня В. Б., Ісмайлов К. Ю., Краснобрижий І. В. Інформаційні технології: підруч. Дніпро: ДДУВС, 2021.
3. Гуржій А. М., Возненко Л. І., Поворознюк Н. І., Самсонов В. В. Основи інформаційних технологій: навч. посіб. К.: Літера ЛТД, 2023.
4. Кравченко І. В., Микитенко В. І. Інформаційні технології: підручник. Київ: КПІ ім. Ігоря Сікорського, 2022.
5. Пасічник В. В., Пасічник О. В., Угрин Д. І. Веб-технології: підруч. Львів: Магнолія-2006, 2018. 336 с.
6. Bear Bibeault. jQuery in Action. N.-Y.: Manning Publications, 2008. 300 p.
7. Turban E., Pollard C., Wood G. Information Technology for Management: Driving Digital Transformation to Increase Local and Global Performance, Growth and Sustainability. Hoboken: Wiley, 2020.
8. Stair R., Reynolds G. Principles of Information Systems. 13th ed. Boston: Cengage Learning, 2018.
9. Gallaugher J. Information Systems: A Manager’s Guide to Harnessing Technology. 9th ed. Boston: FlatWorld, 2021.
10. Yates S. J., Rice R. E. (eds.) The Oxford Handbook of Digital Technology and Society. Oxford: Oxford University Press, 2020.
11. Khan W. Z., Ahmed E., Hakak S., Yaqoob I., Din I. U. Edge computing: A survey. Future Generation Computer Systems. 2019. 97:219-235.
12. Matheu S. N., García E., Hernández-Ramos J. L., Skarmeta A., Baldini G. A Survey of Cybersecurity Certification for the Internet of Things. ACM Computing Surveys. 2020. 53(6):131. DOI:10.1145/3410160.
13. Bertolino A., De Angelis G., Di Nitto E., Ferreira M. A Systematic Review on Cloud Testing. ACM Computing Surveys. 2019. 52(5):87. DOI:10.1145/3331447.

14. Xu J., Xiong H., Chen X., Li J., Yang J. A Survey of Blockchain Consensus Protocols: Taxonomy, Performance, and Security. ACM Computing Surveys. 2023. 55(13s):269. DOI:10.1145/3579845.
15. Hassan H. B., Bahsoon R., Kazman R. Survey on serverless computing. arXiv preprint. 2021. arXiv:2106.11773.
16. Dizdarevic J., Carpio F., Jukan A., Masip-Bruin X. A Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration // ACM Computing Surveys. 2019. 51(6):116.
17. Winkler F., Dörнер R., Winner K. A Systematic Literature Review of DevOps Success Factors. Proceedings of the International Conference on Software and Systems Process. ACM, 2023.
18. Ozdogan M., Sen B., Ozkan S. Digital Transformation Maturity Models: A Systematic Literature Review. Journal of Enterprise Information Management. 2021. 34(6):1521-1547.
19. Bogoviz A. V., Ragulina J. V., Alekseev A. N. Industry 5.0: Theory and Practice of Future Management. Cham: Springer Nature, 2021.
20. Fountaine T., McCarthy B., Saleh T. Building the AI-powered Organization. Harvard Business Review. 2019. Vol. 97, No. 4:62-73.
21. Sommerville I. Software Engineering. 11th ed. Harlow: Pearson Education, 2020.
22. Schwalbe K. Information Technology Project Management. 9th ed. Boston: Cengage Learning, 2018.

Додаток А. Технічне завдання на розробку.

Розробці підлягає електронний навчальний курс для вивчення бібліотеки jQuery, створеної для мови JavaScript.

Інтерфейс користувача – графічний.

Змістовну інформацію зберігати у реляційній базі даних.

Програмний продукт слід реалізувати на базі веб-технологій:

- фронт-складова – засобами HTML+CSS+JavaScript;

- бек-складова – засобами PHP+MySQL.

Методика програмування – структурне програмування.

В роботі застосовувати технологію AJAX з метою мінімізації перезавантажень веб-сторінок. Для взаємодії клієнтської та серверної частин використовувати простий текстовий формат.

Передбачити реєстрацію користувачів в окремій таблиці БД.

Готовий продукт піддати тестуванню, зробити висновки про результати роботи.

Додаток Б. Висхідний текст розробленого програмного забезпечення: файл index.php.

```
<?php header('Content-Type: text/html; charset=UTF-8');
?>
<html>
<head>
<title>
Електронний навчальний курс з основ jQuery</title>
</head>
<body bgcolor=#BBCCFF style="background-image:
url(images/background.png)">
<script language="JavaScript">
var ThemeSelected="";

<?php
$hostname="localhost";
$username="id11116003_root";
$password="000000";
$dbName="id11116003_jquery";
$link=mysqli_connect($hostname,$username,$password,$dbNam
e) or die("No connect to DB!");
mysqli_set_charset($link,'utf8');
$query="SELECT ThemeName FROM themes ORDER BY
ThemeOrder;";
$result=mysqli_query($link,$query);
$num_rows=mysqli_num_rows($result);
print "var text=new Array(";
for($i=0;$i<$num_rows;$i++)
{
    if($i)print ",";
    $row=mysqli_fetch_array($result);
    print    "\"{$row['ThemeName']}\"";
```

```
    }
    print ");";
    ?>

    function fillsections(e)
    {
        var blocks=e.target.responseText.split('*');
        document.getElementById("themeinfo").innerHTML=block
s[0];
        document.getElementById("maindiv").innerHTML=blocks[
1];
    }

    function reportadding(e)
    {
        alert(e.target.responseText);
    }

    function ChangedThemeSelect()
    {
        var
index=document.getElementById("themeselect").selectedIndex;
        ThemeSelected=document.getElementById("themeselect")
.options[index].text;
        document.getElementById("themefrom").innerText="Тема
: "+ThemeSelected;
        var data=new FormData();
        data.append('theme',ThemeSelected);
        var req=new XMLHttpRequest();
        req.addEventListener('load',fillsections,false);
        req.open("POST","processsections.php",true);
        req.send(data);
    }
```

```
    function CheckAnswer(radio)
    {
        if(!document.getElementById(radio).checked)
            alert('Нажаль, Ви помилилися!')
        else
            alert('Молодець, правильно!');
    }

    function ShowOrHideForm()
    {
        if(document.getElementById("reglayer").style.display
=="none")
        {

    document.getElementById("reglayer").style.display="block"
;

    //document.getElementById("CancelButton").style.display="
block";

    document.getElementById("ShowOrHideButton").value="Зберег
ти";
        }
        else
        {

    if(!document.getElementById("FirstName").value||!document
.getElementById("LastName").value)
            {
                alert("Введіть і прізвище і ім'я!");
                return;
            }
```

```
    document.getElementById("reglayer").style.display="none";

    document.getElementById("ShowOrHideButton").value="Зареєс
труватися";
            var data=new FormData();

    data.append('FirstName',document.getElementById("FirstNam
e").value);

    data.append('LastName',document.getElementById("LastName"
).value);

    data.append('Patronymic',document.getElementById("Patrony
mic").value);
            var req=new XMLHttpRequest();
            req.addEventListener('load',reportadding,false);
            req.open("POST","addreguser.php",true);
            req.send(data);
        }
    }

    function HideForm()
    {
        document.getElementById("reglayer").style.display="n
one";
        document.getElementById("ShowOrHideButton").value="З
areєструватися";
    }
    </script>
    <div      style="width:100%;top:-3px;text-align:center;z-
index:10">
    <form>
```

```
    <p style="font-size:20">Виберіть тему для вивчення</p>
    <select                style="width:300px;position:center"
id="themeselect" onchange="ChangedThemeSelect();">
    <option>Оберіть тему...</option>
    <script>
    for(i=0;i<text.length;i++)
    {
        document.write("<option
value=\""+text[i]+"\">"+text[i]+"</option>");
    }
    </script>
    </select><br>
    <p                 style="font-size:20;text-align:center"
id="themefrom">Тема: оберіть тему...</p>

    <textarea rows=5 cols=100 id="themeinfo">
    </textarea>

    </form>
    </div>
    <div   style="position:absolute;top:0px;width:100%;align-
content:right;align-items:right;text-align:right;z-index:0"
id="regdiv">
    <form>
    <input     type="button"     onclick="ShowOrHideForm();"
value="Зареєструватися..."              id="ShowOrHideButton"
style="width:120">
    <div style="display:none" id="reglayer">
    <input         type="button"         onclick="HideForm();"
value="Відмінити" id="CancelButton" style="width:120"><br>
    Ім'я<input type="text" id="FirstName"><br>
    Прізвище<input type="text" id="LastName"><br>
    По-батькові<input type="text" id="Patronymic"><br>
```

```
    </div>
    </form>
    </div>
    <div                    style="top:200px;width:80%;align-
content:center;align-items:center;text-align:center;z-
index:0" id="maindiv">
    </div>
    </body>
    </html>
```

Додаток В. Висхідний текст розробленого програмного забезпечення: файл processsections.php.

```
<?php
header('Content-Type: text/html; charset=UTF-8');
$hostname="localhost";
$username="id11116003_root";
$password="000000";
$dbName="id11116003_jquery";
$link=mysqli_connect($hostname,$username,$password,$dbName)
or die("No connect to DB!");
mysqli_set_charset($link,'utf8');
$query="SELECT ThemeInfo,ThemeID FROM themes WHERE
ThemeName='".$_POST["theme"]."' ORDER BY ThemeOrder;";
$result=mysqli_query($link,$query);
$num_rows=mysqli_num_rows($result);
for($i=0;$i<$num_rows;$i++)
{
     $row=mysqli_fetch_array($result);
     print $row['ThemeInfo'];
}

print "*";
$query="SELECT SectionID, SectionName FROM sections WHERE
SectionTheme='".$row['ThemeID']."';";
$result=mysqli_query($link,$query);
$num_rows=mysqli_num_rows($result);
for($i=0;$i<$num_rows;$i++)//прохід по всім розділам обраної
теми
{
     $row=mysqli_fetch_array($result);
```

```
    print "<p style=\"font-size:16pt;text-
decoration:underline;font-
weight:bold\">".$row['SectionName']."</p>";
    $query2="SELECT
TextBlockID,TextBlockText,TextBlockFontName,TextBlockFontSize
FROM textblocks WHERE
TextBlockSection='".$row['SectionID']."';";
    $result2=mysqli_query($link,$query2);
    $num_rows2=mysqli_num_rows($result2);
    for($i2=0;$i2<$num_rows2;$i2++) //прохід по всім
текстовим блокам даного розділу
    {
        $row2=mysqli_fetch_array($result2);
        $query3="SELECT AnchorLink FROM anchors WHERE
AnchorTextBlock='".$row2['TextBlockID']."';";
        $result3=mysqli_query($link,$query3);
        $num_rows3=mysqli_num_rows($result3);
        if($num_rows3>0)
        {
            $row3=mysqli_fetch_array($result3);
            print "<a href=\"".$row3['AnchorLink']."\">";
        }
        print "<p style=\"font-
family:".$row2['TextBlockFontName'].";font-
size:".$row2['TextBlockFontSize']."pt;text-
align:justify\">".$row2['TextBlockText']."</p>";
        if($num_rows3>0)
        {
            print "</a>";
        }
        $query4="SELECT ImageID,ImageURL FROM images WHERE
ImageTextBlock='".$row2['TextBlockID']."';";
        $result4=mysqli_query($link,$query4);
```

```
          $num_rows4=mysqli_num_rows($result4);
          if($num_rows4>0)
          {
               $row4=mysqli_fetch_array($result4);
               $query5="SELECT AnchorLink FROM anchors WHERE
AnchorImage='".$row4['ImageID']."';";
               $result5=mysqli_query($link,$query5);
               $num_rows5=mysqli_num_rows($result5);
               if($num_rows5>0)
               {
                    $row5=mysqli_fetch_array($result5);
                    print "<a
href=\"".$row5['AnchorLink']."\">";
               }
               print "<img
src=\"images/".$row4['ImageURL']."\">";
               if($num_rows5>0)
               {
                    print "</a>";
               }
          }
     }
     if(!$num_rows2)
          print "<p style=\"text-align:justify\">Для цього
розділу тексту в базі немає</p>";
     print "<br><p style=\"font-weight:bold;font-
family:Courier;font-size:12pt;text-
decoration:underline\">Контрольні питання:</p><div
style=\"text-align:left\">";
     $query6="SELECT
QuestionText,QuestionVar1,QuestionVar2,QuestionVar3,QuestionV
ar4,QuestionAnswer FROM questions WHERE
QuestionSection='".$row['SectionID']."';";
```

```
    $result6=mysqli_query($link,$query6);
    $num_rows6=mysqli_num_rows($result6);
    for($i6=0;$i6<$num_rows6;$i6++)
    {
        $row6=mysqli_fetch_array($result6);
        print      "<p      style=\"font-family:Courier;font-
size:12pt\">".($i6+1).") ".$row6['QuestionText']."</p>";
        print              "<input             type=\"radio\"
name=\"question".$i.$i6."\"     id=\"radio".$i.$i6."1\"><label
for=\"radio".$i.$i6."1\"     style=\"font-family:Courier;font-
size:12pt\">".$row6['QuestionVar1']."</label>";
        print           "<br><input           type=\"radio\"
name=\"question".$i.$i6."\"     id=\"radio".$i.$i6."2\"><label
for=\"radio".$i.$i6."2\"     style=\"font-family:Courier;font-
size:12pt\">".$row6['QuestionVar2']."</label>";
        print           "<br><input           type=\"radio\"
name=\"question".$i.$i6."\"     id=\"radio".$i.$i6."3\"><label
for=\"radio".$i.$i6."3\"     style=\"font-family:Courier;font-
size:12pt\">".$row6['QuestionVar3']."</label>";
        print           "<br><input           type=\"radio\"
name=\"question".$i.$i6."\"     id=\"radio".$i.$i6."4\"><label
for=\"radio".$i.$i6."4\"     style=\"font-family:Courier;font-
size:12pt\">".$row6['QuestionVar4']."</label>";
        print          "<br><input          type=\"button\"
value=\"Відповісти\"
onclick=\"CheckAnswer('radio".$i.$i6.$row6['QuestionAnswer'].
"');\"><br>";
    }
    if(!$num_rows6)
        print   "<p   style=\"text-align:justify\">Для   цього
розділу питань в базі немає</p>";
    print "</div>";
}?>
```

Додаток Г. Висхідний текст розробленого програмного забезпечення: файл addreguser.php.

```
<?php
header('Content-Type: text/html; charset=UTF-8');
$hostname="localhost";
$username="id11116003_root";
$password="000000";
$dbName="id11116003_jquery";
$link=mysqli_connect($hostname,$username,$password,$dbName)
or die("No connect to DB!");
mysqli_set_charset($link,'utf8');
$query="INSERT INTO regusers
(UserName,UserLastname,UserPatronymic) VALUES
('".$_POST["FirstName"]."','".$_POST["LastName"]."','".$_POST
["Patronymic"]."');";
$result=mysqli_query($link,$query);
if($result)
    print "Вітаємо, Ваші дані успішно додано в загальний перелік!";
else
    print "Під час запису на курс виникла помилка. Спробуйте внести свої дані пізніше";
?>
```