МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка
до дипломної роботи
_____бакалавр _____
(освітньо-кваліфікаційний рівень)

на тему «Розробка Web-порталу по збереженню та обробці файлів відео-формату»

Виконав: студент 4 курсу, групи ІТ-651
напряму підготовки 6.050103 «Програмна інженерія»

_____ Зубенко Д.О.
(підпис)

Керівник,
доцент, к.т.н._____ Фесенко Т.М.
(підпис)

Рецензент,
Ст. викладач каф. П.М._____ Батурін О.І.
(підпис)

СЄВЄРОДОНЕЦЬК – 2019

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ
дипломної роботи студента гр. ІТ-651 Зубенко Д.О.

Науковий керівник

Доцент, к.т.н. _____ Фесенко Т.Н.

Оцінка наукового керівника: _____

Рецензент Батурін О.І. ст. викладач каф. ПМ СНУ ім В. Даля
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

Зав. кафедри ПМ
Д.т.н., доцент _____ Лифар В.О.
(підпис)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050103 «Програмна інженерія»
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри ПМ
д.т.н., доцент
_____ В.О. Лифар
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНУ **РОБОТУ СТУДЕНТУ**

Зубенко Дмитро Олегович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка Web-порталу по збереженню та обробці файлів відео-формату

керівник проекту (роботи) Фесенко Т.М. кандидат технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "23"04 2019 р. №68/14.04

2. Строк подання студентом роботи 10.06.2019

3. Вихідні дані до роботи Розробка Web-порталу: опис, документація, програмне забезпечення

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд обчислювальних мереж та постановка технічного завдання. Основні інструменти моніторінгу і аналізу мережі, неполадки й устаткування для їхнього пошуку. Діагностування каналів зв'язку обчислювальних мереж. Програмне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Дата видачі завдання 23.04.2019

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Складання плану роботи	23.04.19	Виконано
2	Аналіз літератури	22.05.19	Виконано
3	Вивчення і підбирання матеріалу	24.05.19	Виконано
4	Написання розділів	25.05.19	Виконано
5	Оформлення пояснювальної записки	02.06.19	Виконано
6	Оформлення графічного матеріалу	06.06.19	Виконано
7	Підготовка доповіді і слайдів для Презентації	10.06.19	Виконано

Студент _____ Зубенко Д.О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Фесенко Т.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка включає 63 сторінки, 22 джерел за переліком посилань.

У даній роботі досліджено і розглянуто засоби розробки порталу та відео-плеєру.

Ключові слова: Web-ПОРТАЛ, ВІДЕО-ПЛЕЄР, БУТСТРАП, ДАНІ

ЗМІСТ

ВСТУП

Актуальність розробки Web-порталу. На практиці цей ресурс є невід'ємною частиною «павутини» та суспільного використання для відпочинку, пошуку інформації для візуального й слухового сприймання, а ще для заробітку грошей. У кожного є свої вимоги до таких ресурсів, від переводу текстів та їх озвучення до вибору жанрів й інформації про ресурс. З точки зору програміста – розробка Web-порталу з використанням відео-файлів – це великий заробіток на маркетингу але він ,майже, самий найдовший, також за кожний перегляд відео-файлу, за кожен відгук портал становиться більш відомий й більш прибутковий. З використанням певних розділів та жанрів можливо підняти цільову аудиторію за лічені дні.

Об'єкт досліджень: Sibnet Video-player;

Предмет досліджень: розроблення порталу з використанням плеєру;

Мета дослідження: Розробка відео-плеєру, використання віртуального диску, підключення до хостингу;

Завдання дослідження:

а) Проаналізувати вихідну інформацію, визначити необхідність дослідження

б) Розробити відео-плеєр ;

в) Розробити програмне забезпечення

Методи дослідження: використання бібліотек програмування для отримання певного результату; використання літератури та технічної документації для написання програмного коду; аналіз сайтів на яких міститься інформація, огляд сайту на можливість автоматичної обробки інформації;

РОЗДІЛ 1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Web-вузол (Web-сайт) - це комплекс Web-сторінок та інших ресурсів, об'єднаних відповідно за змістом, розміщених в на одному домені. Web-сайт – це загальний інформативний джерело.

Мережа Internet складається з великого числа з'єднаних між собою комп'ютерів, маршрутизаторів та іншого, необхідного для правильної роботи.

Кожен елемент мережі Internet (вузол) має неповторний IP-адрес. Знаючи IP-адреса вузла, є можливість спробувати під'єднатися до нього, а маючи невеликі навички можна з'ясувати кому ця адреса належить і в якому знаходиться регіоні світу. IP-адреси прийнято записувати у вигляді чотирьох груп цифр, розділених крапками, наприклад

192.168.100.003 або 10.10.0.123

Запам'ятати адреси всіх часто відвідуваних сторінок практично неможливо, для цього в мережі Internet існують спеціальні сервера DNS (Domain Name System), на яких виконується зберігання всіх списків зіставлення IP-адресу і символьних імен. Завдяки цьому, до серверів, користувач завжди переходить по потрібному IP-адресу, набравши в браузері тільки ім'я сторінки.

Після цього, як ми ввели в рядок браузера ім'я необхідної сторінки, браузер автоматично отримує з DNS IP-адресу потрібного сервера і шле за цією адресою спеціальний запит на отримання сторінки (HTTP-запит). Працююча на сервері спеціальна програма (т.зв. Web-сервер) обробляє цей запит і відправляє назад в браузер необхідну сторінку.

Очевидно, що всі дії по відображенню сторінки можна розділити на дві категорії: що виконуються на стороні клієнта (код клієнтський або front-end) і на стороні сервера (код серверний або back-end). При цьому сервер зовсім нічого не знає про поточний стан клієнта, а клієнт - про стан сервера.

При розробці алгоритмів обміну потрібно завжди знати про це і своєчасно пересилати потрібні дані, що описують поточний стан або необхідну дію.

Залежно від середовища використання розрізняються і засоби реалізації частин. На стороні клієнта використовується зазвичай тільки HTML, JavaScript (AJAX), CSS, Flash.

1.1.1 Мова програмування JavaScript

Мова програмування JavaScript сконструйований компанією Netscape з метою формування інтерактивних HTML-документів. Це об'єктно мова розробки вбудованих додатків, які працюють як на стороні клієнта, так і сервера. Синтаксис мови дуже схожий на синтаксис мови Java - тому його часто називають Java-подібним. клієнтські програми виконуються браузером перегляду Web-документів на комп'ютері користувача, а серверні додатки виконуються на сервері.

При розробці обох типів додатків застосовується єдиний компонент мови, іменований ядром і містить визначення стандартних об'єктів і конструкцій (змінні, функції, основні об'єкти і засоби LiveConnect взаємодії з Java-апплетами), і відповідні компоненти доповнень мови, що мають характерні для кожного типу додатків визначення об'єктів.

Клієнтські програми безпосередньо вбудовуються в HTML-сторінки і інтерпретуються браузером згідно відображенням частин документа в його вікні.

Серверні додатки з метою підвищення продуктивності заздалегідь компілюються в проміжний byte-code.

Основні галузі використання мови JavaScript при створенні інтерактивних HTML-сторінок:

- динамічне формування документа за допомогою сценарію;
- оперативний контроль правдивості заповнюваних полів користувачем форм HTML аж до передачі їх на сервер;
- створення динамічних HTML-сторінок спільно з каскадними таблицями стилів і об'єктної моделлю документа;
- взаємодія з користувачем при вирішенні "локальних" завдань,

розв'язуваних додатком JavaScript, вмонтованому в HTML-сторінку.

1.2 Аналіз існуючих аналогів розробок

В процесі проектування були проаналізовані розробляючі і вже впроваджені системи, пов'язані з розробкою Web-порталу. Наприклад, youtube призначений для надання інформаційно-навчального матеріалу і підтримки зв'язку між користувачами в Web-порталу.

Web-портал youtube являє собою багатофункціональну систему і є засобом надання інформаційно-навчального матеріалу електронних ресурсів для використання в

навчальному процесі і підтримування зв'язку між користувачами.

Система функціонує в архітектурі клієнт-сервера. Робота здійснюється за аналогічною технологією, яка використовується при розробці Web-порталу. У всіх версіях здійснюється підтримка UNICODE.

Плюси замовлення на розробку Web-порталу у спеціалізованого розробника в тому, що в мінімально зазначений термін замовнику нададуть повністю робочий продукт, який відповідає всім заявленим вимогам, включаючи впровадження, технічну підтримку і супровід. Однак вартість Web-порталу становить 4 тис грв.

Переваги самостійної розробки очевидні: вартість витрат трохи нижче, повна підтримка і супровід без залучення сторонніх фахівців, можливість на етапі розробки проведення пробного впровадження і коригування результатів в ході тестування.

1.3 Вимоги до розробляємого порталу

Метою даної випускної кваліфікаційної роботи є розробка Web-порталу, що дозволяє отримувати інформацію в якості відео-матеріалу і підтримувати зв'язок між юзерами та адміністраторами.

Основними завданнями порталу є: отримувати інформацію в якості відео-матеріалу і підтримувати зв'язок між юзерами та адміністраторами и за рахунок забезпечення доступу через мережу Internet до Web-ресурсу.

На сьогоднішній день технології дозволяють створювати Web-портали високої складності з різним функціоналом. Переваги Web-порталів:

- програмний код Web-порталів виконується на віддаленому сервері, не використовуючи ресурсів машини користувача;
- відсутність необхідності установки на машину користувача будь яких компонентів додатка;
- відсутність проблем з оновленням і підтримкою старих версій програм;
- забезпечення мобільності користувачів.

Безумовно, у Web-порталів є істотні недоліки:

- необхідність підключення користувача до мережі (локальної або глобальної) для доступу до сервера;
- швидкість роботи програми залежить від швидкості передачі даних між клієнтом і сервером і завантаженості сервера, яка тим вище, чим більше користувачів одночасно звертаються до сервера.

Розробка Web-порталу буде здійснювати функції сполучної ланки між користувачем і БД.

Зберігання даних в базі даних (далі - БД) на сервері обумовлено наступними причинами:

- централізоване зберігання на сервері надійніше в порівнянні з зберіганням на локальних машинах: до сервера обмежений як фізичний, так і програмний доступ, постійно виконується резервне копіювання даних;
- реляційна структура БД забезпечує більш швидкий доступ до пов'язаних даними;
- виключається небажане дублювання даних;
- можливість вибирати тільки ті дані, які необхідні в даний момент.

Надійність програми повинна забезпечуватися на рівні використовуваних апаратних і програмних засобів. Це досягається за необхідне контролем вхідних даних, їх обробкою і зберіганням.

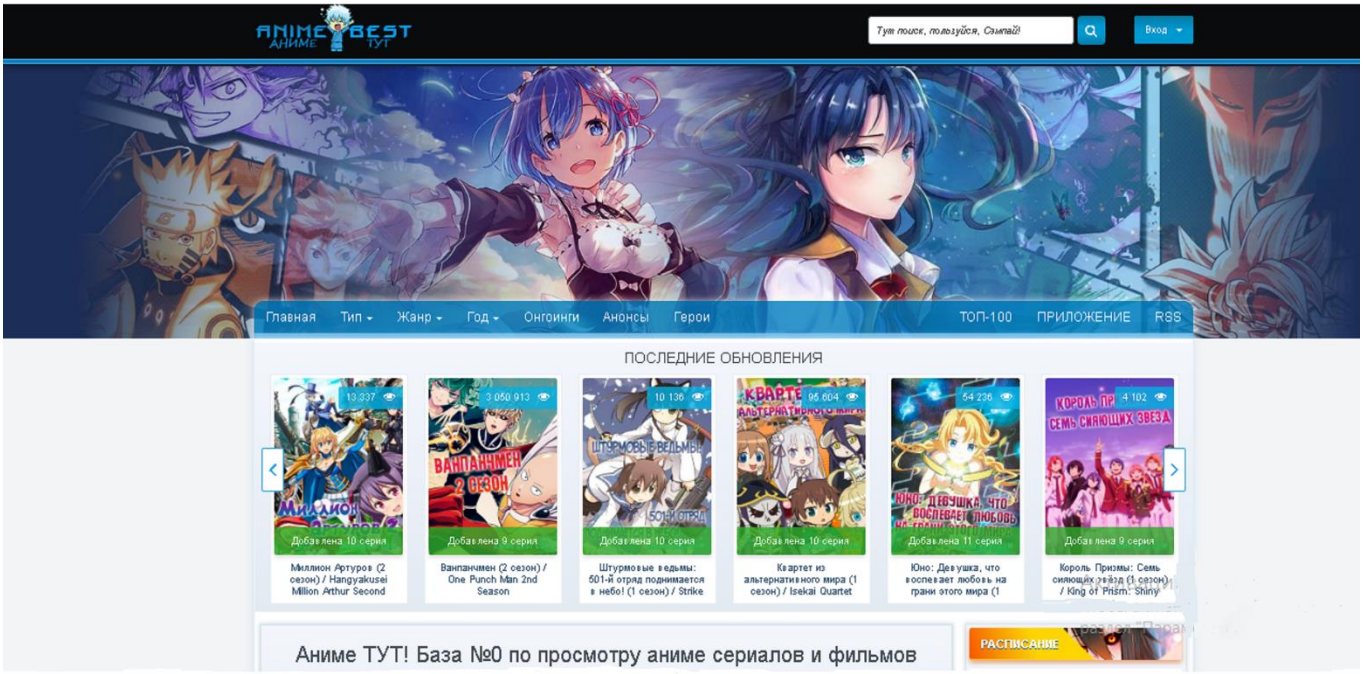
На обраному сервері необхідно встановити сервер Web-додатків, інтерпретатор, драйвер БД, а також встановити міжмережевий екран.

В такому випадку адміністратору мережі необхідно налаштувати відповідним чином DNS-сервер для правильної переадресації зовнішніх запитів до сервера Web-додатки. В такому випадку в якості клієнтських машин можуть виступати практично будь-які наявні ПК, за умови, що серед встановленого програмного забезпечення є браузер. модернізувати апаратну частину цих ПК немає необхідності.

Для клієнта програмний додаток може працювати не тільки на ПК, але і на мобільних пристроях з встановленим браузером (бажано Google Chrome або Opera з включеним JavaScript). Необхідна наявність справного мережевого адаптера для здійснення підключення до мережі і доступу до сервера, наявність справних засобів введення інформації.

Мінімальні системні вимоги ідентичні вимогам до рекомендованих браузерам.

1.4 Аналіз схожого Web-порталу



Потреба у web-порталах наразі дуже велика. Перш за усе, це відпочинок, адже всім хочеться щось подивитися, можливо це буде інформативний ресурс, можливо розважальний. Але на цьому етапі, портали й об’єднують людей, тому програмування порталу є одним за найприбутковішим заняттям. З кожної відвідуваної сторінки, з кожного перегляду відео-матеріалу та навіть з кожного непримусового кліку ми отримуємо копійчину. На разі web-портали знаходяться на сходинці біля соціальних мереж.

Як показує практика – ідеального нічого немає. Тому кожен сайт, портал має свої недоліки. Недоліки витікають з погано-відпрацьованного інтерфейсу, вибору поганого хостинга та з помилок у коді, котрі не помітно неозброєним оком.

Наразі, хочу проаналізувати один з найвідоміших порталів – animebest.org. Головно проблематикою порталу є інтерфейс – чесно кажучи, співвідношення чорного та блакитного кольорів, хоч і виглядає красиво, але це вже застаріло, та навіть деякі можуть подумати що макет – добре зплагіачен з інших сайтів. Якщо подивитися у правий верхній кут, там бачимо таб у вигляді входу на сайт, на мою думку, це також помилка, форма реєстрації повинна бути відокремлена від входу, або ж, цей таб повинні були назвати по іншому задля зручного користування. Можливо це такий хід для заробітку, при відкриття табу йде зайвий клік мишкою, що дуже вигідно для розробників. Третім недоліком є меню, дуже багато вікон для пошуку потрібного відео-

ресурсу, на цьому порталі неможливо вибрати і жанр, і тип, наприклад, вибір йде з вибором лише одного пункту. Як мінімум, чотири вкладки можна було зробити в одній. Також сайт на мобільній платформі виглядає дуже незручно, тому що він дуже довгий і потрібно багато часу для повної прокрутки до низу, так як якорів там немає, ще дуже багато витрачає трафіку не тільки в якості ресурса, а також в якості великої кількості реклами. Якщо зостатися на цьому порталі бодай на пів години, пів екрану протягом 20 хвилин буде займати реклама.

Інтерфейс – це не уся проблематика цього порталу. Іншим пунктом являє собою їх відео-плеєр. На перший погляд все дуже щільно зроблено, але не все так добре як здається, при перегляді відео йде не поступова загрузка контенту, а за один раз – все. Ця причина робить непрацездатним плеєр при поганому підключенні до інтернету або з обмеженим трафіком, іншими словами, - загрузка усього відео займає майже такий ж самий інтервал часу, яке займає все відео, також трафік використовується в два рази більше, наприклад, ніж на YouTube . Друга проблема цього плеєру, також з’єднання з інтернетом, при перегляді відео, якщо пропадає зв’язок або падає швидкість , то потрібно обновити сторінку та заново завантажувати відео, що й неоднократно впливає на трафік. Остання проблема плеєру не така значна як інші дві, але також дуже кидається в очі – це розміри відео-плеєру, розробники зробили його дуже великим, та при маленькому масштабі він займає більш ніж 90% екрану, а при використанні у весь екран, відео не перевертається, а по сторонам чорний екран.

Щоб вирішати ці проблеми, потрібно оновити кодинг плеєру, використати іншу сітку BootStrap (фреймворк css), змінити хостинг для покращення та пришвидчення роботи сайту й збільшить розмір віртуального диску для подальшого використання відео-плеєру.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-ПОРТАЛУ

2.1 Проектування структури Web-порталу

2.1.1 Архітектура Web-порталу

Подальший опис етапів створення кінцевого додатки буде грунтуватися на ітераційному підході, в основі якого лежить поступове ускладнення і деталізація первісної структури програми.

В основі розроблюваної системи лежить архітектура «клієнт-сервер», в якої мережева навантаження розподілені між постачальниками послуг (сервісів), званих серверами, і замовниками послуг, званих клієнтами.

Клієнтом є браузер користувача, де за допомогою DOM і JavaScript забезпечується взаємодія користувача з даними. Використання JavaScript само по собі не є обов'язковим, однак функціонування більшості Web-додатків без JS не представляється можливим. В якості середовища взаємодії клієнта з сервером використовується мережу Internet.

Також для вирішення поставленого завдання можна використовувати архітектуру «Трирівнева архітектура», проте в даному випадку подібний підхід буде надлишковий з причини відсутності великої фрагментованою бізнес-логіки додатка

Основними достоїнствами архітектури «клієнт-сервер» є:

- Можливість, в більшості випадків, розподілити функції обчислювальної системи між декількома незалежними комп'ютерами в мережі. це

дозволяє спростити обслуговування обчислювальної системи. Зокрема, заміна, ремонт, модернізація або переміщення сервера, не зачіпають клієнтів.

- Всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів. На сервері простіше забезпечити контроль повноважень, щоб вирішувати доступ до даних тільки клієнтам з відповідними правами доступу.
- Дозволяє об'єднати різні клієнти. Використовувати ресурси одного сервера часто можуть клієнти з різними апаратними платформами, операційними системами і т.п.

Основні недоліки архітектури «клієнт-сервер»:

- в разі використання централізованої системи, непрацездатність основного сервера може зробити непрацездатною весь Web-додаток;

- адміністрування даної системи вимагає кваліфікованого професіоналізму;

В ході вибору апаратної платформи будуть запропоновані і реалізовані рішення, що дозволяють мінімізувати ймовірність виходу з ладу серверної частини Web-додатку.

Основною метою програми є надання послуг за коштами мережі Internet, необхідно включити в його серверну частину Web-сервер - апаратно-програмний комплекс, призначений для обслуговування HTTP-запитів.

HTTP-запит - сформований відповідно до протоколу HTTP / 1.1 запит на сервер на заздалегідь визначений порт (за замовчуванням, порти 80 і 8080) з метою виконання будь-якого віддаленого дії (маніпуляції з інформацією, виконання певних команд і т.д.). Як правило, такі запити посилає браузер клієнта.

Оскільки Web-додаток передбачає велику кількість операцій по читанню, запису і зміни значного обсягу даних, найбільш зручним варіантом буде включення в серверну частину технологій БД.

Серверна частина вміщує себе Web-сервер і сервер БД.

До завдань Web-сервера входять:

- отримання і відповідь на HTTP-запити;
- перенаправлення запитів на необхідне Web-додаток (сайт), як правило, приписане до певного домену або субдоменів;
- надання Web-додатків доступу до необхідних модулів

(Наприклад, до модулю зв'язку з СУБД, модулю обробки php-програм та інші);

- авторизація та аутентифікація користувачів;
- реалізація функцій файл-сервера.

До завдань сервера БД входить:

- обслуговування запитів на маніпуляції з даними на основі мови

SQL;

- обслуговування БД;
- забезпечення цілісності даних;
- надання утиліт для адміністративного управління СУБД.

Клієнтська частина програми повинна підтримувати такі технології:

- доступ до мережі Internet;
- можливість роботи по протоколу HTTP / 1.1;
- висновок інформації;

- підтримка пристроїв введення даних.

2.1.2 Проектування Бази Даних

Вивчивши предметну область і спроектувавши структуру Web-додатки,

можна приступити до наступного кроку - проектування структури бази даних. В якості методу проектування БД обраний метод ER-діаграм.

2.1.3 Мови програмування

Web-додаток складається з двох рівноцінних частин: серверної і клієнтської. Можливо, використовувати одну й теж саму мову програмування у всьому додатку, проте це є самим неоптимальним варіантом.

Для серверної частини буде використовуватися Jquery- скриптова мова програмування загального призначення, інтенсивно застосовується для розробки Web-порталів. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов програмування, що застосовуються для створення динамічних Web-сайтів.

Для клієнтської частини будуть використовуватися такі мови:

HTML (від англ. HyperText Markup Language - «мова розмітки гіпертексту») - стандартна мова розмітки документів у Всесвітній павутині.

CSS (від англ. Cascading Style Sheets - каскадні таблиці стилів) - формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

JavaScript - об'єктно-орієнтована скриптова мова програмування. JavaScript підтримується всіма існуючими браузерами і є

стандартом де-факто для сучасних інтерактивних Web-додатків.

2.1.4 Середовище розробки

Для написання програмного коду і коду html-розмітки використовувався CMS WordPress - один з найпопулярніших систем для організації процесу проектування, редагування і керування вмістом. вона орієнтована на зручність використання, підтримка мережевих стандартів. WordPress безкоштовний движок і поширюється вільно. Мова написання - PHP, в якості бази даних використовується My SQL, має загальнодоступну ліцензію GNU. Розвинена система навігації (пошук по рядку, карта документа та ін.).

Контроль результатів роботи Web-додатки і інтерфейсу проводився на 2 браузерах: Google Chrome і Yandex.

2.1.5 Web-сервер

Web-сервер - це сервер, що приймає HTTP-запити від клієнтів, зазвичай Web-браузерів, і видає їм HTTP-відповіді, зазвичай разом з HTML сторінкой, зображенням, файлом, медіа-потоком або іншими даними.

2.1.5.1 Огляд сучасних Web-серверів

На даний момент існує всього два Web-сервера, які дозволяють реалізувати спільну роботу раніше обраних технологій:

Nginx - високопродуктивний HTTP-сервер, призначений в основному для роздачі клієнтам статичного контенту (зображень, javascriptфайлов, css-стилів і т.д.). В основі лежить технологія неблокіруючіх соединений, що при великій кількості одночасних підключень істотно економить ресурси сервера.

Apache HTTP-сервер - вільний Web-сервер. За статистикою на лютий 2019 року використовується в 45.8% всіх сайтів в Інтернеті, що робить його самим популярним Web-сервером. Версія 2.4.x володіє перевіреної роками і мільйонами користувачів стабільністю і надійністю. Web-сервер має велику кількість модулів для роботи з багатьма серверними технологіями.

2.1.5.2 Вибір Web-сервера відповідно до зазначених вимог

На підставі даної інформації в якості Web-сервера був обраний Apache HTTP-сервер так, як є кросбраузерності і безкоштовним.

ВИСНОВКИ

Сайти повинні бути доступні для будь-яких користувачів, навіть якщо вони зазнають труднощів при сприйнятті відео- та аудіо- інформації, використовують старі браузери або повільні підключення або переглядають сайт з мобільних пристроїв.

На етапі аналізу предметної області були виділені основні завдання, які має вирішувати Web-портал, і визначені технічні вимоги. Моделювання предметної області полегшує вибір інструментів і технологій, які будуть застосовуватися для реалізації програми.

Був проведен аналіз юзабіліті інших сайтів, знайшовши плюси та мінуси – було використано у своєму порталі. Щоб отримати максимальну аудиторію було використано матеріалу більш ніж з 5 порталів, а також тематичних сайтів.

РОЗДІЛ 2 ОПИС ІНФОРМАЦІЙНОЇ МОДЕЛІ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Що таке HTML та з чим його їдять

HTML (HyperText Markup Language - мова розмітки гіпертексту) є основною мовою програмування веб-сторінок. Описи веб-сторінок містяться в HTML-програмах (HTML-кодах), які зберігаються в звичайних текстових файлах з розширенням htm або html. Іноді ці програми називають HTML документами, але зазвичай HTML-документом вважається те, що можна бачити у вікні браузера. Програми на мові HTML містять інструкції (коди), звані тегами. Теги являють собою послідовності символів, укладені
в кутові дужки (наприклад, <P>).
Більшість сучасних браузерів допускають запис тегів в будь-якому регістрі, тобто як прописними, так і малими літерами. Всі ключові слова тегів, є зазвичай абревіатурами слів англійської мови, записуються буквами латинського алфавіту. Наприклад, IMG-скорочення слова image (зображення).

HTML-програма повинна починатися тегом <HTML> і закінчуватися тегом</ HTML>. Між ними знаходяться інші теги програми або текст, який ви хочете вивести у вікні браузера. Деякі теги використовуються тільки парами (наприклад, <HTML> і </ HTML>). При цьому перший з них називається відкриваючий, а другий - що закриває. Іноді парні теги називають контейнерними, тому що між ними можна розмістити інші теги. Таким чином, в контейнерні теги можна вкладати інші теги, в тому числі і контейнерні, тобто теги можуть бути вкладеними. Існують поодинокі теги, для них немає відповідних тегів для закриття. Прикладом одиночного тега є тег
 (кінець рядки).

Теги можуть містити параметри, звані атрибутами, які, в свою чергу, можуть мати значення - аргументи. Наприклад, для виведення на екран зображення, що зберігається в файлі використовується тег:

<img src=”picture.jpg”>

Тут IMG - назва тега, SRC - атрибут, а "picture.jpg" - аргумент атрибута SRC.

Отже, дял написання HTML-програми, потрібно включити в неї
два тега:

<HTML>
(Тут будуть інші теги програми)
</ HTML>

HTML-програми складаються з двох основних частин: заголовка і тіла. кожна з цих частин обмежується відповідною парою тегів. Так, заголовок обмежується парою тегів *<HEAD>* і *</ HEAD>*, а тіло - тегами *<BODY>* і *</ BODY>*. В результаті HTML-програма виглядає наступним чином:

<HTHL>
<HEAD>
(Тут буде заголовок)
</ HEAD>
<BODY>

(Тут будуть теги тіла програми)

</ BODY>

</ HTML>

Зауважимо, що писати кожен тег з нового рядка або робити відступи зовсім не обов'язково, проте завдяки цьому програма простіша для читання.

Між тегами *<HEAD>* і *</ HEAD>*, що обрамляють заголовок програми (HTML файла), пишемо ще два тега: *<TITLE>* і *</ TITLE>*. За допомогою цих тегів обрамляється текст, який поміщається в заголовок браузера, тобто в саму верхню смужку його вікна. Нехай текст заголовка буде, наприклад, таким: «Основні елементи HTML». Тоді програма прийме наступний вигляд:

<HTML>

<HEAD>

<TITLE> Основні елементи HTML *</ TITLE>*

</ HEAD>

<BODY>

(Тут будуть розташовані теги тіла програми)

</ BODY>

</ HTML>

Зауважимо, що тег *<TITLE>* вкладений в тег *<HEAD>*, а той вкладений в тег *<HTML>*.

Багато користувачів звикли працювати з солідними текстовими редакторами (Текстовими процесорами), наприклад MS Word. Для написання текстів HTML-програм можна використовувати будь-які редактори. Однак при цьому можуть виникнути проблеми. Наприклад, MS Word розпізнає (не тільки по розширенню файлу, але і за змістом), що файл містить HTML-програму, і показує не її текст (код), а результат роботи програми, тобто те, як буде виглядати цей HTML-документ у вікні браузера. Тому для використання MS Word, потрібно зберігати текст HTML-програми як текстовий файл у форматі txt, а потім перейменовати його в файл з розширенням htm або html. Всього цього можна уникнути, якщо відразу використовувати простий текстовий редактор Блокнот (Notepad). Використовуючи Блокнот, а не MS Word, можна бути впевненим, що всі недоречності відображення HTML-документа в браузері визначаються вашої HTML-програмою, а не «інтелектом» текстового процесора. При вивченні основи мови HTML, MS Word, а також інших засобів автоматизації створення HTML документів (наприклад, FrontPage) дійсно полегшить програмування. Замість простого текстового редактора можна використовувати спеціальні редактори веб-сторінок, наприклад Microsoft FrontPage або Macromedia Dreamweaver.

Як FrontPage, так і MS Word при створенні HTML-програм вставляють в заголовок (тобто між *<HEAD>* і *</ HEAD>)* теги *<META>* з різними атрибутами.

Приклад:

<HTML>

<HEAD>

<META http-equiv = "Content-Language" content = "ru">

<META http-equiv = "Content-Type" content = "text / html"; charset = "windows1251">

<META name = "GENERATOR" content = "Microsoft FrontPage 4.0">

<META name = "ProgID" con t en t = "Front Page. Editor. Document">

</ HEAD>
<BODY>
</ BODY>
</ HTML>

Інформація, що міститься в тегах *<META>*, не відображається браузером і служить спеціальним цілям. Наприклад, в ній вказується мова, на якій написаний документ, кодова сторінка, ключові слова для пошуку та ін. Ці дані дуже важливі для настройки браузера і роботи пошукових серверів. Теги *<META>* можна вставити в HTML-програму на заключному етапі розробки веб-сторінки, званому публікацією.

При написанні HTML-програм, особливо якщо в них міститься багато тегів, виникає необхідність вставки коментарів - пояснюють текст, які не видно при завантаженні документа в браузер. Коментарі необхідні розробнику HTML-документа. Для цієї мети служить тег *<!>*. Все, що укладено між символами *<! i>*, вважається коментарем і не відображається браузером на екрані.

Звернемо увагу на те, що перенесення слів тексту у вікні браузера відбувається в залежності від ширини вікна. Якщо потрібно, щоб текст розташовувався по центру, застосовуємо тег *<CENTER>*. Якщо користувач змінить розміри вікна браузера, відбудеться нове вирівнювання тексту. Зауважимо, що тег *<CENTER>* можна застосовувати не тільки до тексту, але і до графіку, таблиць та інших елементів. Тег *<CENTER>* є контейнерним, тобто йому відповідає тег для закриття - *</ CENTER>*.

Деяким тегам в HTML-програмі відповідають цілком певні елементи NTM-документа, що розміщуються на сторінці. Наприклад, можна написати теги, які виведуть на екран такі елементи, як текст, малюнок, відеокліп та ін. Інші ж теги нічого не виводять. Вони призначені для спеціальних цілей. Типовими прикладами таких тегів є контейнерні теги (або просто контейнери): *<HEAD>*, *<BODY>* і *<CENTER>*. Також тут ще один контейнерний тег *<DIV>*, за допомогою якого можна виділити частину (або розділ) HTML-документа.

У HTML-програму можна вставляти фрагменти програм, написаних на мові JavaScript або Visual Basic Script (тільки для браузера Internet Explorer). Такі фрагменти називаються сценаріями і обрамляються тегами *<SCRIPT>* і *</ SCRIPT>*. У сценаріях зазвичай описується обробка таких подій, як клацання клавішею миші на кнопці або зображенні.

Сторінка зазвичай містить текст - простий і найбільш поширений спосіб представлення інформації, хоча далеко не єдиний.

Якщо ви працюєте в спеціальній програмі, призначеної для створення веб-сторінок (наприклад, FrontPage), то у вас є безліч варіантів, подібних

форматування в звичайних текстових процесорах. Експериментуйте, спостерігаючи при цьому, які теги створює програма FrontPage (див. Вкладку HTML).

При оформленні текстів використовуються спеціальні теги. Розглянемо деякі з них.

• Тег *
* наказує перехід на новий рядок.

• Тег *<P>* є тегом абзацу. Після нього текст буде виводитися з нового рядка і, крім того, один рядок буде пропущен. Якщо не використовувати ці теги, то розбиття тексту на рядки буде визначатися шириною вікна браузера, так що вид тексту може виявитися зовсім не таким, яким ми його собі уявляли.

• Якщо ви хочете, щоб текст вирівнювався по центру вікна браузера, використовуйте тег <*CENTER*>, який згадувався вище.

УВАГА

Теги <*BR*> і <*P*> діють по відношенню не тільки до текстів, але і до інших елементів сторінки. Наприклад, якщо ви хочете, щоб малюнок розміщувався нижче тексту, то поставте між текстом і тегом малюнка <*BR*> або <*P*>.

Стандартні логічні стилі

Щоб змінити розмір шрифту можна використовувати теги так званих логічних стилів. Їх всього шість, зазвичай вони використовуються для визначення заголовків різного рівня. При переході від першого стилю до шостого поступово зменшується розмір і товщина букв шрифту. Теги логічних стилів записуються як <*H1*>, <*H2*>, ... <*Hn*>. Кожен з них має відповідний закриваючий тег. Наприклад, тегу <*H1*> відповідає закриває тег </ *H1*>.

приклад

<*H1*> Заголовок 1-го рівня </ *H1*>

задає виведення тексту Заголовок 1-го рівня шрифтом, відповідним першому логічного стилю.

Зауважимо, що логічний стиль визначає стиль тексту згідно налаштувань браузера. При цьому стиль <*H2*> завжди буде «менше», ніж <*H1*>, якщо, звичайно, автор сторінки не перевизначив його на свій розсуд. Справа в тому, що є можливість перевизначити установки за замовчуванням. Для цього використовуються кошти каскадних таблиць стилів (CSS).

<*HTHL*>
<*HEAD*> <*TITLE*> *Основні елементи HTML* </ *TITLE*> </ *HEAD*>
<*BODY*>
<*H1*> *Заголовок 1-го рівня* </ *H1*>
<*H2*> *Заголовок 2-го рівня* </ *H2*>
<*H3*> *Заголовок*
<*H4*> *Заголовок*
<*H5*> *Заголовок*
<*H6*> *Заголовок*
<*H7*> *Тема*
</ *BODY*>
</ *HTML*>

Управління шрифтом

Крім використання стандартних розмірів і накреслень (гарнітури) шрифтів можна визначати шрифти для кожного фрагмента тексту за допомогою спеціальних тегів.

Найпростіший спосіб - використання так званих фізичних стилів

- Напівжирний (Bold) <B>
- Курсив (Italic) <I>
- Підкреслений (Underscore) <U>
- Викреслений (Strike Out) <S>
- Друкарська машинка (Typewriter) <TT>
- Мерехтливий (тільки для браузера Netscape Navigator) <BLINK>

Для кожного тега фізичного стилю існує відповідний закриває

тег, який скасовує стиль, встановлений раніше. Наприклад, для тега <*B*> закриває тегом є <*/ B*>.

Усередині тега заголовка можна вставити тег фізичного стилю, щоб модифікувати весь заголовок або тільки деяку його частину. Наприклад, щоб виділити курсивом частину тексту, певного як заголовок, можна використовувати наступну конструкцію:

<*HTML*>

<*HEAD*> <*TITLE*> *Фізичні і логічні стилі* <*/ TITLE*> <*/ HEAD*>

За допомогою спеціального тега <*FONT*> можна налаштувати шрифт для зображення тексту: задати гарнітуру, розмір і колір. Перш за все ви можете встановити розмір основного шрифту, який використовується в документі за замовчуванням. Тег основного шрифту має формат <*BASEFONT SIZE = размер_шріфта*>.

Розмір основного шрифту можна встановити від 1 до 7. Якщо не використовувати цей тег, то розмір основного шрифта встановлюється рівним 3.

Тег <*FONT SIZE = размер_шріфта*> використовується для установки розміру поточного шрифту в окремих фрагментах тексту. На стилі цей тег не впливає. Діапазон можливих значень - від 1 до 7. Даний тег дозволяє також керувати розміром поточного шрифту щодо основного. Для цього використовуються символ «+» (Щоб збільшити) і символ «-» (щоб зменшити розмір шрифту на задану величину). Наприклад, якщо розмір основного шрифту встановлено рівним 3, то тег <FONT SIZE = + 2> встановлює розмір поточного шрифту рівним 5.

Для завдання гарнітури шрифту використовується тег <FONT FACE = "імя_шріфта">.

приклад

<FONT FACE = "Arial">

Якщо цей тег не використовується у нашому HTML-документі, то браузер буде застосовувати шрифт, встановлений в настройках. Тому текст на екрані користувача може виглядати зовсім не так, як ми собі його уявляли. слід також мати на увазі, що якщо призначений вами шрифт не встановлений на комп'ютері користувача, то браузер буде зображувати текст шрифтом, встановленим по замовчуванням. У тезі <*FONT*> ви можете вказати через кому перелік шрифтів. В цьому випадку браузер буде використовувати перший знайдений шрифт.

приклад

<*FONT FAC E = "AriaI, Sans Serif, Courier"*>

Зазвичай в переліку задають схожі шрифти. Рекомендується призначати найбільш популярні шрифти. При розміщенні на сторінці текстової інформації краще взагалі не призначати назву шрифта, покладаючись на стандартні настройки браузера. Але тоді при розробці сторінки слід також використовувати стандартні настройки браузера, щоб синхронізувати своє сприйняття тексту з можливим сприйняттям користувача.

За допомогою атрибута COLOR в тезі <*FONT*> можна задати колір шрифту:

<*FONT COLOR = "колір"*>

Аргумент атрибута COLOR є шістнадцятирічний запис коду кольору (червоного, зеленого і синього складової, або, інакше кажучи, RGB-складової)

Зауважимо, що в тезі <*FONT*> можна використовувати кілька його

атрибутів або все можливі.

приклад

<FONT FACE = "Arial" SIZE = 5 COLOR = "BLUE">

У математичних формулах, а також для підрядкових зауважень часто застосовуються індекси, які відрізняються від основного тексту становищем щодо рядка (трохи вище або нижче) і розміром. Для цієї мети служать теги *<SUP> і <SUB>* - відповідно для верхніх і нижніх індексів.

Крім розглянутих вище, є додаткові теги форматування текстів:

• <ADDRES> - виділення адрес електронної пошти, поштових адрес і номерів телефонів;

• <C1TE> - виділення цитат;

• <CODE>, <SAMP> -запис текстів програм, символьних констант;

• <KBD> - введення текстів з клавіатури.

В останніх трьох стилях використовується моноширинний шрифт (зазвичай Courier).

Наприклад, символ «I» даного шрифту займає стільки ж місця, скільки і буква «Ж». Використання таких шрифтів обумовлено простою можливістю вирівнювання тексту за допомогою символу пропуску.

Відзначимо ще один момент. В теги управління шрифтом, як і в теги логічних стилів, можна вставляти атрибут TITLE = "рядок", що дозволяє прив'язати до тексту всередині цього тега підказку. Аргументом атрибута TITLE є рядок підказки. При зупинці курсору миші на виділеному слові або фразі близько курсору з'явиться підказка. За допомогою цього прийому можна розшифровувати абревіатури, давати додаткові пояснення і рекомендації користувачеві.

1.1.1 колір

За замовчуванням браузери заповнюють фон суцільним кольором, визначеним налаштуванням браузера: сірим, білим або чорним. Користувачі по-різному налаштовують кольори, тому іноді має сенс примусово зафіксувати колір фону або створити фонове зображення. Якщо не подбати про це, в гіршому випадку користувач не зможе прочитати синій текст на чорному тілі. Фоновий колір задається в тезі *<BODY>*.

Колір фону визначається атрибутом BGCOLOR тега *<BODY>*. Наприклад, тег, що задає колір фону "# FF1230", має вигляд:

<BODY BGCOLOR = "# FF1230">

а жовтий колір фону - такий вигляд*:*

<BODY BGCOLOR = "YELLOW">

Можна задати і колір тексту. Для цього служить атрибут TEXT тега *<BODY>*. Тег, наведений нижче, задає зелений колір фону і синій колір для тексту:

<BODY BGCOLOR = "GREEN" TEXT = "BLUE">

Текст заданого формату

Браузер зазвичай перетворює текст HTML-файлу при виведенні його на екран, тобто ігнорує зайві прогалини, символи табуляції і символи кінця рядка. Якщо потрібно, щоб текст на екрані виглядав так, як його ввели в HTML-документ, то потрібно скористатися тегом попереднього форматування <PRE>. Текст повинен знаходитися між тегами <PRE> і </ PRE>.

1.1.2 Списки

Досить часто потрібно розмістити на сторінці списки (переліки елементів).

Списки бувають неупорядкованими і впорядкованими (за алфавітом або по цифрам). При відображенні списків браузер виділяє їх відступом від краю сторінки. Крім того, списки можуть бути вкладеними.

Впорядковані списки задаються тегом <*OL*>, а невпорядковані - тегом <*UL*>. Обидва ці тега парні, тобто контейнерні.

Для впорядкованих списків можна вибрати спосіб індексації. Це робиться за допомогою атрибута TYPE з аргументами: 1 (арабські цифри), A (великі літери), a (малі літери), I (римські цифри). Можна задати номер, з якого починається нумерація елементів списку. Для цього служить атрибут START всередині тега <*OL*>.

Перед елементами списків слід поставити тег <*LI*>, щоб індексація відбувалася автоматично. У цьому тезі можна використовувати і вищеописаний атрибут TYPE.

1.1.3 Рухомий рядок

Internet Explorer підтримує тег <MARQUEE>, який дозволяє створити так-званий біжучий рядок, тобто ефект прокручування тексту в заданому полі. Характеристики рядка, що біжить задаються наступними атрибутами:

• WIDTH - ширина поля рядка, що біжить в пікселах або відсотках від ширини вікна;

• HEIGHT - висота поля рядка, що біжить (в пікселах);

• HSPACE, VSPACE - інтервали по горизонталі і вертикалі між текстом рядки і краями її поля (в пікселах);

• ALIGN - визначає положення тексту рядка, що біжить в її поле; можливі аргументи:

• TOP (вгорі); BOTTOM (внизу); MIDDLE (посередині);

• DIRECTION - визначає напрямок руху; можливі аргументи:

• LEFT (справа наліво); RIGHT (зліва направо);

• BEHAVIOR - характер руху рядка; можливі аргументи:

• SCROLL - текст з'являється від одного краю і ховається за іншим; Про SLIDE - рядок витягується з одного краю поля і зупиняється в іншім краї;

• ALTERNATE - задає змінну напрямок руху, від одного краю до іншого, а потім назад;

• LOOP - кількість повторень тексту в рухомому рядку, що задається числом;

Якщо необхідно «нескінченне» кількість повторень, то слід задати аргумент у вигляді ключового слова INFINITY. Атрибут LOOP не впливає на поведінку біжучого рядка, якщо для атрибуту BEHAVIOR задані аргументи ALTERNATE або SLIDE;

• SCROLLAMOUNT - встановлює довжину в пікселях «стрибка» тексту за один такт; при великому значенні цього параметра текст рухається ривками, а при малому - уповільнено;

• 5 - визначає величину паузи між тактами переміщення тексту в мілісекундах;

BGCOLOR - встановлює колір поля рядка, що біжить, що задається шістнадцятковим числом або ім'ям.

Співвідношення між довжиною тексту, розміром шрифту і швидкістю переміщення, при яких біжучий рядок виглядає прийнятно, підбираються дослідним шляхом.

2 Графіка на веб-сторінках

У більшості веб-сторінок зустрічається графіка. Вона дозволяє барвисто і наочно представити інформацію. У багатьох випадках краще показати зображення, ніж давати довге текстовий опис.

Існує два способи розміщення зображень на сторінці:

-вставка окремих зображень;

-заповнення фону малюнком. »

У будь-якому випадку графічний об'єкт береться з файлу.

вставка зображень

Вставка на сторінку зображення з файлу графічного формату проводиться за допомогою тега *<IMG>* (від англ: image - зображення) із зазначенням адреси файлу в якості аргументу атрибута SRC:

<IMG SRC = "адрес_графіческого_файла">

Адреса графічного файлу - це або URL-адресу, або ім'я файлу, можливо, із зазначенням шляху.

Наприклад, для показу графічного файлу logotip.jpg слід написати тег

<IMG SRC = "logotip. Jpg">

Для збільшення швидкості передачі зображення в тезі *<IMG>* можна використовувати атрибут (додатковий параметр) LOWSRC, який приймає в якості аргументу адресу графічного файлу. Можна створити два графічних файлу: один (наприклад, нехай це файл logotip.jpg) містить зображення, отримане з високою роздільною здатністю, а другий (наприклад, Logotip.gif) - малюнок, отриманий з низьким дозволом. Тоді тег

<IMG SRC = "logotip.jpg" LOWSRC = "logotip. Gif">

накаже браузеру спочатку завантажити файл logotip.gif, а потім у міру завантаження сторінки замінити його файлом logotip.jpg.

Інший спосіб прискорення завантаження графіки полягає в завданні розмірів зображення за допомогою атрибутів WIDTH (ширина) і HEIGHT (висота), вимірюваних в пікселах. Якщо вказати ці атрибути, то браузер спочатку виділить місце під графіком, підготує макет документа, відобразить текст і тільки потім завантажить графіком. Зауважимо, що браузер стискає або розтягує зображення, вставляючи його в рамки зазначеного розміру. Приклад вказівки розмірів зображення:

<IMG SRC = "logotip.gif" WIDTH = 40 HEIGHT = 20>

Тут атрибут WIDTH задає ширину прямокутної області, в якій буде розміщено графічне зображення, а HEIGHT - висоту.

Графіка зазвичай використовується разом з текстом, тому виникає задача вирівнювання тексту і зображення. Це завдання вирішується за допомогою атрибута ALIGN тега *<IMG>* із застосуванням аргументів. Наприклад, ми можемо побажати, щоб текст обтікав малюнок праворуч або ліворуч. Зазвичай зображення вбудовується впритул з текстом, що може бути некрасиво. Щоб цього уникнути, можна задати порожні поля навколо ілюстрації. Поля створюються за допомогою атрибутів VSPACE для верхнього і нижнього полів і HSPACE для бічних полів в тезі *<IMG>*.

Аргументи цих атрибутів вказуються у вигляді чисел, що визначають розміри полів в пікселах. Для скасування обтікання графіки текстом служить тег <BR CLEAR = ...>.

2.1 посилання

Посилання (або гіперпосилання) дозволяють клацанням кнопкою миші на виділеному тексті або зображенні перейти до іншого файлу або фрагменту сторінки. Посилання застосовуються в більшості існуючих сторінок. Вони можуть бути текстовими і графічними. Текстові посилання є виділене слово або ціла фраза. Виділення посилання виробляється кольором або підкресленням, в залежності від настройки браузера.

У мові HTML структури текстових і графічних посилань подібні один одному. Всі вони задаються тегом *<A HREF>*, якому відповідав би закриває тег *</ A>*. На засланні спочатку вказується ім'я файлу, на який вона посилається, а потім текст або ім'я графічного файлу, що містить зображення посилання. Крім простих графічних посилань можна створити так звану графічну карту посилань: зображення з областями, клацання на яких призводить до спрацьовування відповідних посилань.

Структура текстового посилання має такий вигляд:

<A HREF = "адрес_ссилкі"> текст_ссилкі </ A>

Відзначимо, що браузер не виводить на екран ім'я файлу, до якого потрібно перейти за посиланням, а лише показує текст, укладений в тезі між кутовими дужками > і <. Якщо ж потрібно, щоб зовні посилання виглядала як ім'я файлу, на який вона посилається, то просто пишемо його ім'я замість тексту.

приклад

<A HREF = "докум2.html"> докум2.html </A>

Можна посилатися не тільки на інші файли, але і на свій власний файл. Оскільки налаштування кольору в браузері у різних користувачів можуть відрізнятися, виникає завдання примусово задавати кольори, щоб посилання були добре видні. Вище ми вже розглядали, як задати колір фону і тексту. Це робиться в тезі *<BODY>* за допомогою атрибута LINK для непрочитаного посилання і VLINK - для прочитаного посилання.

URL-адреси посилань

У розглянутих вище прикладах як адресу посилання використовувалося ім'я файлу. У загальному випадку можна застосовувати URL-адресу (Uniform Resource Locator - уніфікований покажчик ресурсу). Формат URL-адреси включає в себе тип мережевий служби (протокол зв'язку), адреса сервера, шлях пошуку і ім'я файлу.

Нижче перераховані URL найбільш популярних служб Інтернету.

URL найбільш популярних служб Інтернету

-World Wide Web

-FTP

-Mail mailto:

-Gopher

-Телеконференції UseNet news:

Якщо ви вказуєте адресу, що починається з http, тим самим ви звертаєтеся до ресурсів, доступ до яких здійснюється по протоколу HTTP (Hyper Text Transfer Protocol - протокол передачі гіпертексту). Цей протокол використовується в якості основного в Інтернеті при передачі інформації, що

знаходиться в HTML-документах.

Префікс адреси ftp означає, що слід використовувати протокол передачі файлів (File Transfer Protocol - FTP). Цей протокол використовується при передачі файлів-програм, що мають розширення exe. Він може використовуватися при переміщенні будь-яких файлів з одного комп'ютера на інший. Зокрема, при перекачуванні файлів вашої сторінки на сервер використовується саме цей протокол.

Протокол FTP забезпечує високу надійність передачі файлів. Наприклад, якщо втрату до 10% звичайної текстової інформації ще можна пережити, то при передачі програми взагалі не допускається втрата - неточно передана програма просто не буде працювати.

Якщо перед адресою посилання вказується mailto, це означає, що слід використовувати протокол передачі повідомлень по електронній пошті.

Gopher - служба (а значить, і протокол), призначена, в першу чергу, для роботи неграфічних браузерів. Вона надає систему доступу до інформації, засновану на меню.

News - служба забезпечення телеконференцій; це система типу дошки оголошень, на яку ви можете помістити своє повідомлення і на якій можна прочитати те, що там розмістили інші учасники телеконференції.

Нижче наведені приклади посилань із застосуванням URL-адрес:

<A HREF = "ftp: //ftp.admi rat.ru/PUB/index. Прим"> Безкоштовний доступ </ A>

<A HREF="mailto:zubenko@maiI.admiral.гі"> 0тправіть пошту </ A>

Шляхи пошуку можуть бути абсолютними і відносними. Абсолютний шлях описує місце розташування файлу починаючи з найвищого рівня і включає імена всіх каталогів, що ведуть до файлу. Помилка в записі абсолютного шляху (адреси) файлу призводить до того, що файл не буде знайдений. Відносний шлях (Адреса) описує місце розташування файлу щодо місця розташування поточного документа. Так, якщо вказати просто ім'я файлу myfile.htm, це означає, що вказуємо відносну адресу. В даному випадку браузер буде шукати його в тому ж каталозі, де знаходиться поточний документ. Якщо перед ім'ям файлу поставити ../ (наприклад ,. ./myfile.htm), то браузер буде шукати файл в каталозі, що знаходиться на один рівень вище, ніж той, в якому знаходиться поточний документ. Аналогічно, якщо перед ім'ям файлу поставити ../../../ .. /myfHe.htm), то браузер буде шукати файл в каталозі на два рівні вище, ніж поточний.

При створенні посилань можна вказувати не тільки на конкретні документи і програми (тобто конкретні файли, використовуючи шлях до них), а й на папки (Каталоги). Іншими словами, адреса - це опис місця розташування ресурсу (Одиниці зберігання інформації). Він може бути точним або «приблизними» (неповним). Можна посилатися на папку, HTML-документ, документ, створений будь-яким додатком (наприклад, MS Word, MS Excel), або просто на текстовий файл з розширенням .txt. Нарешті, можна послатися на файл програми з розширенням exe. Однак в останньому випадку (значною мірою, в Internet Explorer) спрацює захист вашого комп'ютера. З'явиться вікно з попередженням і пропозицією можливих варіантів: запустити файл програми або зберегти його на диску - спосіб захисту від потенційних вірусів. Можна відразу запустити файл програми, а можна зберегти його на локальному диску, перевірити за допомогою антивірусної програми і тільки потім запустити, вилікувати або знищити.

3 Платформа web-розробки

HTML5 та CSS3 – не просто два нові стандарти запропонованих комітетом W3C (World Wide Web Consortium) і його робочими групами. Це наступне покоління щодня використовуваних технологій, створене для того, щоб було простіше і зручніше будувати сучасні веб-програми. Перш ніж занурюватися в подробиці HTMT5 і CSS3, трохи інформації про переваги, а також про деякі проблеми, з якими зіткнемося при використанні цих технологій.

Багато нових можливостей HTML, спрямовані на вдосконалення платформи для побудови веб-додатків. HTML5 надає в розпорядження розробника багато нових інструментів для покращення призначеного для користувача інтерфейсу, від більш змістовних тегів і поліпшених засобів між-сайтових і між-віконних комунікацій до анімації і поліпшеною мультимедійної підтримки. У кожній версії HTML з'являється нова розмітка, але ще ніколи не було стільки доповнень, безпосередньо пов'язаних з описом контенту.

У HTML5 включена підтримка технології Web-Sockets, що реалізує довгострокове підключення до сервера. Замість постійного опитування виконавчої підсистеми для отримання інформації про хід виконання web-сторінка підключається до сокета, а виконавча підсистема доставляє оповіщення користувачам.

Зазвичай HTML5 розглядається як web-технологія, але з появою прикладних інтерфейсів (API) Web Storage і Web SQL Database з’явилась можливість створення браузерних додатків, які зберігають всі дані на машині клієнта.

Інтерфейс є важливою частиною веб-додатків. Доводиться щодня йти на всілякі хитрощі, щоб змусити браузери працювати так, як ми хочемо. Щоб визначити стильове оформлення таблиці або закруглити кути, доводиться використовувати або бібліотеки JavaScript, або додавати значну кількість додаткової розмітки для використання стилів. Завдяки HTML5 і CSS3 ця практика стає справою минулого.

HTML5 надає в розпорядження розробника вдосконалені елементи призначеного для користувача інтерфейсу. Протягом багатьох років використовувалось JavaScript і CSS3 для побудови повзунків, календарів для вибору дати і палітр для вибору кольору. У HTML5 вони стали звичайними елементами - такими ж, як відкриті списки, прапори і перемикачі. І хоча підтримка нових елементів реалізована ще не у всіх браузерах, про неї не варто забувати, особливо якщо розробляються web-додатки. Крім зручності використовування, що не потребує бібліотек JavaScript, існує і інша перевага - поліпшена доступність для користувачів з фізичними недоліками. Екранні диктори та інші спеціалізовані браузери можуть реалізувати ці елементи особливим чином, щоб з ними було зручно працювати людям з обмеженими можливостями.

Використання нових елементів HTML5 спрощує роботу з контентом в

таких спеціалізованих програмах, як екранні диктори. Наприклад, область навігації по сайту набагато простіше знайти, якщо ви шукаєте тег *nav* замість конкретного тега *div* або невпорядкованого списку. Завершувач, бічні панелі і інші структурні елементи легко переміщуються або зовсім виключаються з перегляду. Спрощення розбору сторінок в цілому також спростить роботу зі сторінкою для людей, що використовують допоміжні технології. Крім того, можливість визначення ролей в нових атрибутах елементів допоможе екранним дикторам працювати з ними.

Селектори CSS3 дозволяють визначити непарні і парні рядки в таблиці, всі встановлені прапорці і навіть останній абзац в групі. Більш складні завдання вирішуються з меншим об'ємом коду та розмітки.

Тіні, що відкидаються текстом і зображеннями, надають веб-сторінці візуальну глибину, а градієнти створюють ілюзію обсягу. CSS3 дозволяє додавати тіні і градієнти без використання фонової графіки і додаткової розмітки. Трансформації використовуються для закруглення кутів, деформації та повороту елементів.

3.1 Зворотня сумісність

Одна з найбільш вагомих причин для негайного переходу на HTML полягає у тому, що розмітка працює в більшості існуючих браузерів. *Doctype* повідомляє програмам валідації і редакторам, які теги і атрибути будуть використовуватися в документі і як повинен бути сформований документ. Оголошення *doctype* також використовується багатьма браузерами для визначення того, як браузер повинен відтворювати сторінку. Дійсне оголошення *doctype* зазвичай змушує браузер відтворювати сторінку у «режимі відповідності стандартам». У порівнянні з досить розлогим оголошенням XHTML1.0 Transitional, використовуваним на багатьох сайтах:

<!DOCTYPE html PUBLICK “-//W3C/DTD XHTML 1.0 Transitional//EN”

http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd>

оголошення doctype в HTML5 виглядає дуже просто:

<!DOCTYPE html>

Помістивши його в початок документа і з цього моменту використовується HTML5.

Нажаль, не усі нові елементи HTML5 використовуються у всіх браузерах, але документ буде проходить перевірку на валідність як розмітка HTML5.

3.2 Нова модель

Сайти повинні бути доступні для будь-яких користувачів, навіть якщо вони зазнають труднощів при сприйнятті відео- та аудіо- інформації, використовують старі браузери або повільні підключення або переглядають сайт з мобільних пристроїв. У HTML5 з'явився ряд нових елементів - таких як *audio, video* або *canvs*. У аудіо-і відео контенту з доступністю були деякі утруднення, але елемент *canvas* створює нові проблеми. Він призначений для створення векторних зображень в документах HTML на JavaScript. Це сприяє проблемам не тільки для

лиць з обмеженими зоровими можливостями, але і для 5% які вимикають JavaScript у своїх браузерах .

У HTML5 з'явилося багато нових елементів, але специфікація також оголошує застарілими деякі стандартні елементи, які можуть бути присутніми у web-сторінках. Перш за все зникли деякі елементи уявлення. Насамперед, їх заміняють семантично правильними елементами і придають потрібний вигляд засобами CSS:

- basefont;
- big;
- center;
- font;
- s;
- strike;
- tt;
- u.

Деякі з цих тегів зустрічаються відносно рідко, але теги *font* і *center* присутні в багатьох сторінках, для роботи з якими використовуються візуальні редактори Dreamweaver.

Крім елементів уявлення, була вимкнена підтримка фреймов. Фрейми завжди користувалися популярністю в корпоративних веб додатках: PeopleSoft, Microsoft Outlook Web Access і навіть спеціалізованих порталах. Незважаючи на широке використання, фрейми створювали стільки проблем з доступністю і зручністю використання, що від них було вирішено відмовитися. Таким чином, зникли такі елементи:

- frame;
- frameset;
- noframes.

Ще деякі елементи зникають через появу більш досконалих замінників:

- acronym змінюється на abbr;
- applet змінюється на object;
- dir змінюється на ul.

Крім застарілих елементів, недійсними стали багато

атрибути. До їх числа відносяться такі атрибути уявлення:

- align;
- атрибути link, vlink, alink та text тега body;
- bgcolor;

- height та width;
- атрибут scrolling елементу iframe;
- valign;
- hspace та vspace;
- атрибути cellpading, cellspacing та border тега table.

Але не варто забувати, що змінюються також і браузери; це сприяє до появленню непрацездатних сайтів.

4 JavaScript та JQuery

4.1 Введення і виведення даних

В JavaScript передбачені досить мізерні кошти для введення і виведення даних. Це цілком виправдано, оскільки JavaScript створювався в першу чергу як мова сценаріїв для веб-сторінок. Основою веб-сторінок є код, написаний на мові HTML, який спеціально розрахований на форматування інформації і створення призначеного для користувача інтерфейсу. Оскільки сценарії на JavaScript добре інтегруються з HTML-кодом, остільки для введення і виведення даних цілком підійдуть засоби HTML Якщо писати програму на JavaScript, яка буде виконуватися веб-браузером Internet Explorer, то можна скористатися трьома стандартними методами для введення і виведення даних: alertQ, promptQ і confirmQ. Розглянемо ці методи браузера докладніше.

alert

Даний метод дозволяє виводити діалогове вікно з заданим повідомленням і кнопкою ОК. Синтаксис відповідного виразу має наступний вигляд: alert ("повідомлення")

Якщо повідомлення конкретно, тобто являє собою цілком певний набір символів, то його необхідно укласти в подвійні або одинарні лапки. Взагалі кажучи, повідомлення являє собою дані будь-якого типу: послідовність символів, укладену в лапки, число (в лапках або без них), змінну або вираз. Діалогове вікно, виведене на екран методом alert (), можна прибрати, клацнувши на кнопці ОК. До тих пір поки ви не зробите цього, перехід до раніше відкритих вікон неможливий. Вікна, що володіють властивістю зупиняти всі наступні дії користувача і програм, називаються модальними. Таким чином, вікно, створюване за допомогою alertQ, є модальним.

confirm

Метод confirm дозволяє вивести діалогове вікно з повідомленням і двома кнопками - ОК і Відміна (Cancel). На відміну від методу alert цей метод повертає логічну величину, значення якої залежить від того, на якій з двох кнопок клацнув користувач. Якщо він клацнув на кнопці ОК, то повертається значення true (Істина, так); якщо ж клацнути на кнопці Скасування, то повертається значення false (Брехня, немає). Значення, що повертається можна обробити в програмі і, отже, створити ефект інтерактивності, тобто діалогової взаємодії програми з користувачем. Якщо повідомлення конкретно, тобто являє собою цілком певний набір символів, то його необхідно взяти в лапки, подвійні або одинарні.

Наприклад, confirm ("Ви дійсно хочете завершити роботу?"). Взагалі кажучи, повідомлення являє собою дані будь-якого типу: послідовність символів, укладену в лапки, число (в лапках або без них), змінну або вираз. Діалогове вікно, виведене на екран методом confirm (), можна прибрати клацанням на будь-який з двох кнопок - ОК або Скасувати. До тих пір поки не зробити цього, перехід до раніше відкритих вікон неможливий. Вікна, що володіють властивістю зупиняти всі наступні дії користувача і програм, називаються модальними. Таким чином, вікно, створюване за допомогою confirmQ, є модальним. Якщо користувач клацне на кнопці ОК, то метод поверне логічне значення true (істину, так), а якщо він клацне на кнопці Скасування, то повертається логічне значення false (брехня, немає). Значення, що повертається можна потім обробити в програмі і, отже, створити ефект інтерактивності, то - є діалогової взаємодії програми з користувачем.

prompt

Метод prompt дозволяє вивести на екран діалогове вікно з повідомленням, а також з текстовим полем, в яке користувач може ввести дані. Крім того, в цьому вікні передбачені дві кнопки: ОК і Відміна (Cancel). На відміну від методів alertQ і confirm () даний метод приймає два параметри: повідомлення і значення, яке має з'явитися в текстовому полі введення даних за замовчуванням. Якщо користувач клацне на кнопці Про До, то метод поверне вміст поля введення даних, а якщо він клацне на кнопці Скасування, то повертається логічне значення false (брехня, немає). Значення, що повертається можна потім обробити в програмі і, отже, створити ефект інтерактивності, тобто діалогової взаємодії програми з користувачем. Синтаксис застосування методу prompt має наступний вигляд: prompt (повідомлення, значеніе_поля_ввода_данних) Параметри методу prompt () не є обов'язковими. Якщо їх не вказати, то буде виведено вікно без повідомлення, а в поле введення даних підставлено значення за замовчуванням - undefined (не визначене).

4.2 Типи даних

У будь-якій мові програмування дуже важливо поняття типу даних. Якщо не усвідомити його з самого початку, то потім доведеться часто стикатися з дивною поведінкою створеної вами програми. Дані, які зберігаються в пам'яті комп'ютера і піддаються обробці, можна віднести до різних типів. Поняття типу даних виникає природним чином, коли необхідно застосувати до них операції обробки. Наприклад, операція множення застосовується до чисел, тобто до даних числового типу. Це відомо ще з початкової школи. А що вийде, якщо помножити слово «Вася» на число 5? Оскільки важко дати зрозумілу відповідь на це питання, напрошується висновок: деякі операції не слід застосовувати до різнотипним даними. Я також не знаю, що повинно вийти в результаті множення слів «Вася» і «Маня», тому роблю висновок, що певні операції взагалі не стосовно відповідних деяких типів. З іншого боку, існують операції, результат яких залежить від типу даних. Наприклад, операція додавання, позначається символом « + ». Може застосовуватися і до двох числах, і до двох рядках, що складається з довільних слів. У першому випадку результатом застосування цієї операції буде деяке число, а в другому - рядок, що виходить шляхом приписування другого рядка до кінця першої. У разі рядків операцію складання ще називають склеюванням або

конкатенацией. Операції, що застосовуються до різних типів даних, але позначаються одним і тим же символом, називають також перевантаженими. Так, операція, що позначається символом t, є перевантаженою: стосовно числах вона виконує арифметичне додавання цих чисел, а стосовно до рядків символів - склейку (приписування, конкатенацію).

Типи даних Приклади Строковий або символьний (string), Числовий (number), логічний (Булевский, boolean), Null, Об'єкт (object), Функція (function) "Привіт" Послідовність символів, укладена "Д.т.н. 123-4567" в лапки, подвійні або одинарні 3.14 -567 +2.5 Число, послідовність цифр, перед якою може бути вказаний знак числа (+ або перед позитивними числами не обов'язково ставити знак «+»; ціла і дробова частини чисел розділяються крапкою. Число записується без лапок true false true (істина, так) або false (брехня, немає); можливі тільки два значення Відсутність будь-якого було значення. Програмний об'єкт, який визначається своїми властивостями. Зокрема, масив також є об'єктом. На даному етапі слід звернути особливу увагу на відмінності в поданні числових і строкових (символьних) даних. Числа як дані числового типу завжди представляються без лапок. Для їх написання використовуємо цифри і, при необхідності, знак числа і розділову точку. Строкові дані полягають в лапки, подвійні «" » або одинарні Наприклад, запис -345.12 представляє число, а запис - рядок символів. Це дані різних типів, хоча ми і можемо сказати, що їх змістом є одне і те ж число. Але це не що інше, як буденна інтерпретація даних або, як ще кажуть, семантика (сенс) даних. З точки зору мови програмування число (точніше, дані числового типу) можна коректно використовувати в арифметичних операціях, а рядки - в строкових операціях. Що означає «коректно використовувати »? Те, що використання даних в мові програмування має відповідати нашим традиціям, не пов'язаним з програмуванням. Наприклад, звернення з числами коректно, якщо воно підпорядковується правилам математики; звернення з рядками коректно, якщо не суперечить правилам редакторської правки тексту (вставка, видалення, склейка фрагментів і т. п.). Мова програмування покликаний забезпечити в тій чи іншій мірі виконання операцій, що мають аналоги у звичайній людській діяльності. Цій меті служить, зокрема, і поняття типів даних.

Зауважимо, що рядок не містить жодного символу (навіть пробілу), називається порожньою. При цьому рядок, що містить хоча б один пробіл не порожній. Дані логічного типу можуть мати одне з двох значень: true або false. Ці значення записуються без лапок. Значення true означає істину (та), а false - брехня (немає). Зазвичай ці значення виходять в результаті обчислення виразів з використанням операцій порівняння двох даних, а також логічних операцій (I, I Л I, Н Е). Наприклад, результатом обчислення виразу 2 <3 є, очевидно, значення true (дійсно, число 2 менше числа 3). Логічний тип даних називають ще булевих (boolean) на честь англійського математика Джона Буля, який придумав алгебру для логічних величин.

При створенні програм на JavaScript за типами даних стежить сам програміст. Якщо він переплутає типи, то інтерпретатор НЕ зафіксує помилки, а спробує привести дані до рандомного типу, щоб виконати зазначену операцію. Багато мови програмування, в тому числі С і Pascal, не володіють цим типом, вони вимагають явного вказівки типу даних. Наприклад, якщо написати вираз 5+

"Вася", то результатом його обчислення буде рядок символів "5Вася". Таким чином, інтерпретатор, зіткнувшись з виразом складання числа і рядки символів, переводить число в рядок, що містить це число, а потім виконує операцію складання двох рядків. Додавання двох рядків в JavaScript дає в результаті рядок, яка утворюється шляхом приписування другого рядка до кінця першої. Результатом обчислення виразу 2 + 3 буде число 5, а вираження 2+ "3" - рядок "23", що складається з двох цифрових символів. Як неважко помітити, в разі застосування операції складання до числа і рядку символів інтерпретатор перетворює число в рядок символів і повертає результат обчислення виразу теж у вигляді рядка символів. Інакше кажучи, в разі смисловий неузгодженості типів даних інтерпретатор використовує деякі правила їх узгодження, прийняті за замовчуванням. В результаті можуть з'явитися важко виявляються помилки. З іншого боку, цю особливість мови можна використовувати для написання витончених і компактних кодів, дотримуючись дуже обережним.

Для перетворення рядків у числа в JavaScript передбачені вбудовані функції parselntQ і parseFloatQ. В результаті обчислення функції виходить деяке значення. Функція parseInt (cipOKa, підстава) перетворює зазначену в параметрі рядок в ціле число в системі числення на зазначених підставах (8, 10 або 16). Якщо основа не вказано, то передбачається 10, тобто десятирічна система числення. Звернемо увагу, що при перетворенні в ціле число округлення не відбувається: дрібна частина просто відкидається. Функція parseFloat (cipoKa) перетворює зазначену рядок в число з плаваючою розділової (десяткового, підстава) точкою.

Завдання перетворення чисел в рядки виникає рідше, ніж зворотне перетворення. Щоб перетворити число в рядок, досить до порожньому рядку додати це число, тобто скористатися оператором складання «+». наприклад, обчислення виразу "" +3.14 дасть в результаті рядок "3.14". Для визначення того, чи є значення виразу числом, служить вбудована функція 1 $ (значення). Обчислення цієї функції дає результат логічного типу. Якщо вказане значення не є числом, функція повертає true, інакше - false. Однак тут поняття «число» не збігається з поняттям «значення числового типу». Функції isNaNQ вважає числом і дані числового

Змінні і оператор привласнення

Що станеться, якщо в програмі просто написати дані будь-якого типу, наприклад число 314? Інтерпретатор виконає цю запис, розмістивши її у внутрішньому форматі десь в пам'яті комп'ютера. От і все. Щоб зберігати дані в пам'яті і в той же час залишати їх доступними для подальшого використання, в програмах використовуються змінні.

Імена змінних

Змінну можна вважати контейнером для зберігання даних. Дані, які зберігаються в змінної, називають значеннями цієї змінної. Мінлива має ім'я - послідовність літер, цифр і символу підкреслення без пробілів і розділових знаків, що починається обов'язково з букви або символу підкреслення. Приклади правильних імен змінних: myFamily, my_adress, _x, tel412_3456. Приклади неправильних імен змінних: 512group, myadress, tel: 412 3456. При виборі імен

змінних можна використовувати ключові слова, тобто слова, які використовуються у визначеннях конструкцій мови. Наприклад, не можна вибирати слова var, if, else, const, true, false, function, super, switch і ряд інших. Ім'я повинно відображати зміст змінної. Якщо ім'я складається з декількох слів, то між ними можна вводити символ підкреслення або писати їх разом, починаючи кожне слово з великої літери. Ось приклади: my_first_adress, myFirstAdress. Іноді в якості першого символу імені використовують букву, що вказує на тип даних (значень) цієї змінної: з - строковий (character), n - числовий (number), b-логічний (boolean), про - об'єкт (object), а - масив (array). Наприклад cAdress, nCena, aMonth. JavaScript є чутливі до регістру мовою. Це означає, що дає змогу змінювати регістр символів (з прописних на рядкові і навпаки) в імені змінної призводить до іншої змінної. Наприклад: variable, Variable і vaRiabLe – різні змінні.

Створення змінних

Створити змінну в програмі можна декількома способами: За допомогою оператора присвоєння значень в форматі: ім'я_змінної = значення. Оператор присвоєння позначається символом рівності «=».

приклад:

За допомогою ключового слова var (від variable - змінна) в форматі: var ім'я_змінної. В цьому випадку створена змінна не має ніякого значення, але може його отримати в подальшому за допомогою оператора присвоєння.

приклад:

```
var myName
myName = "Іван"
```

За допомогою ключового слова var і оператора присвоєння в форматі:

```
var ім'я_змінної = значення
```

приклад

```
var myName = "Іван"
```

Рядок коду програми з ключовим словом var задає так звану 'ініціалізацію змінної і для кожної змінної використовується один раз. Тип змінної визначається типом значення, яке вона має. На відміну від багатьох інших мов програмування, при ініціалізації змінної не потрібно описувати її тип. Змінна може мати значення різних типів і неодноразово їх змінювати. Наприклад, можна написати наступний код програми:

```
var x = "Іван"
// деякий код
x = "Анна"
// деякий код
x = 2.5
```

Одне ключове слово var можна використовувати для ініціалізації відразу декількох змінних, як з оператором присвоєння, так і без нього. При цьому змінні та вирази з операторами присвоєння розділяються комами, наприклад:

var name = "Вася", address, x = 3.14

Якщо змінна в даний момент має значення числового типу, то кажуть, що це числова змінна. Аналогічно можна говорити про строкових, логічних (булевих), невизначених (в разі типу null) змінних. Вище ми використовували оператор присвоєння значення змінної, що позначається символом рівності - • Не слід плутати цей оператор з відношенням рівності і відповідної операцією порівняння. Вираз з оператором «=» інтерпретатор обчислює наступним чином: змінної зліва від нього присвоюється значення, розташоване праворуч від нього. Якщо x і y - дві змінні, то вираз x = y інтерпретується так: змінної x присвоюється значення змінної y. Інакше кажучи, змінної можна привласнити значення іншої змінної, вказавши праворуч від оператора «=» її ім'я. Таким чином, до значень можна отримувати доступ опосередковано - через імена

змінних.

Область дії змінних

Змінні, які створені в програмі за допомогою оператора присвоєння з використанням ключового слова var або без нього, є глобальними. Це означає, що вони доступні скрізь в цій же програмі, а також в викликаються програмах з інших файлів. Ці змінні також доступні всередині коду функцій. Крім глобальних існують і локальні змінні. Вони можуть бути створені всередині коду функцій. Можена визначити змінні з однаковими назвами і у зовнішній програмі, і в коді функції. У цьому випадку змінні, визначені в коді функції за допомогою ключового слова var, будуть локальними: зміни їх значень всередині функції ніяк не позначаться на змінних з такими ж іменами, визначених у зовнішній програмі. При цьому значення локальних змінних не доступні з зовнішньої програми. Нерідко область дії називають областю видимості. Змінна може бути видна або хоч всередині програмної одиниці (функції, підпрограми). Область видимості, доступності або дії змінної - еквівалентні терміни. Крім них ще використовують поняття часу життя змінної. Час життя змінних в JavaScript визначається інтервалом часу від завантаження до вивантаження програми з пам'яті комп'ютера. Так, якщо програма (сценарій) записані в HTML-коді веб-сторінки, то після його вивантаження весь сценарій разом з певними в ньому змінними припиняє активну існування.

Оператори

Оператори призначені для складання виразів. Оператор застосовується до одного або двох даними, які в цьому випадку називаються операндами. Наприклад, оператор додавання застосовується до двох операндам, а оператор логічного заперечення - до одного операнду. Елементарне вираз, що складається з операндів і оператора, обчислюється інтерпретатором і, отже, має деяке значення. У цьому випадку говорять, що оператор повертає значення. Наприклад, оператор додавання, застосований до числам 2 і 3, повертає значення 5. Оператор має тип, що співпадає з типом, що повертається їм значення. Оскільки

елементарне вираз з оператором і операндами повертає значення, цей вислів можна привласнити змінної. Один оператор ми вже розглянули в попередньому розділі. Це оператор присвоєння «=». Однак слід зауважити, що існує ще п'ять різновидів оператора присвоєння, що поєднують в собі дії звичайного оператора «=» і операторів додавання, віднімання, множення, ділення і ділення по модулю.

1.6. функції

Функція представляє собою підпрограму, яку можна викликати для виконання, звернувшись до неї по імені. Взаємодія функції з зовнішньою програмою, з якою вона була викликана, відбувається шляхом передачі функції параметрів і прийому від неї результату обчислень. Втім, функція в JavaScript може і не вимагати параметрів, а також нічого не повертати. В JavaScript є вбудовані функції, які можна використовувати в програмах, але код яких не можна редагувати або подивитися. Все, що ми можемо дізнатися про них, - це опис іхдействія, параметрів і значення, що повертається. Крім використання вбудованих функцій можна створити свої власні, так звані призначені для користувача функції. Часто використовувані фрагменти програмного коду доцільно оформляти у вигляді функцій. Такий фрагмент коду полягає в фігурні дужки, а перед ним пишеться ключове слово function, за яким слідують круглі дужки, що обрамляють список параметрів.

Щоб викликати відповідну функцію в режимі, слід написати вираз в наступному форматі: імя_функциі (параметри). Якщо потрібні параметри, то вони вказуються в круглих дужках через кому. Функція може і не мати параметрів. В цьому випадку в круглих дужках нічого не вказується.

Вбудовані об'єкти

Об'єкти представляють собою програмні одиниці, що володіють деякими властивостями. Про об'єкт ми можемо судити за значеннями його властивостей і опису того, як він функціонує. Програмний код вбудованих в JavaScript об'єктів нам недоступний. Головне для нас зараз - засвоїти, як користуватися об'єктами. Це зовсім просто, потрібно тільки дотримуватися нехитрий синтаксис. Використовувати об'єкти не важче, ніж функції. Управління веб-сторінками за допомогою сценаріїв, написаних на JavaScript, полягає в використанні і зміні властивостей об'єктів HTML-документа і самого браузера. Вбудовані об'єкти мають фіксовані назви і властивості. всі властивості цих об'єктів поділяють на два види: просто властивості і методи. Властивості аналогічні звичайним змінним. Вони мають імена і значення. деякі властивості об'єктів доступні тільки для читання. Це означає, що їх значення не можна змінювати. Інші властивості доступні і для запису - їх значення можна змінювати за допомогою оператора присвоєння. Методи аналогічні функціям, вони можуть мати параметри або не мати їх. Щоб дізнатися значення властивості об'єкта, необхідно вказати ім'я цього об'єкта і ім'я властивості, відокремивши їх один від одного крапкою. Зауважу, що об'єкт може і не мати властивостей. Ми можемо змусити об'єкт виконати той чи інший властивий йому метод. В цьому випадку також кажуть про застосування методу до об'єкта. Синтаксис відповідного вираження такої: імя_об'єкта.метод (параметри). Зауважимо, що об'єкт може не мати методів.

Отже, по синтаксису властивості відрізняються від звичайних змінних тим,

що мають складові імена, а також тим, що значення деяких властивостей не можна змінити. Методи відрізняються з точки зору синтаксису від звичайних функцій тільки тим, що мають складові імена. Ми можемо впливати на об'єкт тільки за допомогою властивостей і методів. У популярних книгах часто порівнюють об'єкт з чорним ящиком, у якого є входи і виходи, доступні для спостереження і, можливо, для управління.

В JavaScript математичні обчислення, складна обробка рядків і дат, а також створення масивів виробляються за допомогою відповідних вбудованих об'єктів. Для розробників веб-сайтів особливо важливі об'єкти String (обробка рядків), Array (масиви), Math (математичні формули і константи) і Date (робота з датами). Вбудовані об'єкти, як уже зазначалося, мають фіксовані назви. Об'єкти з іменами, що збігаються з їх фіксованими назвами, називаються статичними. Однак можна створити екземпляри (копії) статичних об'єктів, присвоївши їм свої власні імена. Примірники статичних об'єктів є об'єктами у програмі, які успадковують від перших все їх властивості та методи. Примірники об'єктів - це деякі приватні втілення в програмі відповідних статичних об'єктів. Разом з тим ви можете використовувати і статичні об'єкти в чистому вигляді, не створюючи жодних їх копій. Наприклад, для формульних обчислень використовується статичний об'єкт Math, а в разі масивів створюються екземпляри статичного об'єкта Array, утримуючи конкретні дані, до яких можна застосувати всі загальні методи і властивості статичного об'єкта Array. Вбудовані об'єкти мають, серед інших, властивість prototype (прототип), за допомогою якого можна додавати нові властивості і методи до вже існуючих екземплярів об'єктів. Ці нові властивості і методи, зрозуміло, ви повинні попередньо самі продумати і втілити в вигляді програмних кодів. При розробці сценаріїв для веб-сторінок таке завдання рідко виникає, але JavaScript призначений не тільки для створення сценаріїв.

4.3 Призначені для користувача об'єкти

У попередньому розділі ми розглянули вбудовані об'єкти, тобто заздалегідь визначені в JavaScript і часто використовувані в програмах. За допомогою виразів з ключовим словом new можна створювати екземпляри цих об'єктів, тобто їх конкретні втілення. Більш того, завдяки властивості prototype є можливість додавати до об'єктів нових властивостей і методи, придумані користувачем і відсутні у вихідних вбудованих об'єктах. У більшості випадків, зокрема при створенні сценаріїв для веб-сторінок, всього цього більш ніж достатньо. Однак не можна оминути увагою можливість створення власних об'єктів. Навіщо потрібні власні об'єкти? Вони не є необхідними для вирішення практичних завдань. З точки зору програміста, об'єкт являє собою просто зручний засіб організації даних і функцій їх обробки. Щоб якось організувати дані і функції в програмі, далеко не завжди є потреба вдаватися до такої конструкції, як об'єкт. Адже навіть при значній кількості даних програміст не завжди організовує їх у вигляді масиву (об'єкта Array): буває достатнім обмежитися простими змінними.

Зараз концепція об'єктно-орієнтованого програмування є провідною в технологіях створення великих додатків, тому буде корисно познайомитися з нею ближче.

5 JQuery

Написання коду на JavaScript, який би чітко і однозначно працював у всіх основних браузерах, - дуже непросте завдання. Існує багато бібліотек, які спрощують цей процес; однією з найбільш популярних серед них є JQuery. Вона проста у використанні, має велику базу готового коду і добре підходить для створення простих обхідних рішень.

5.1 Завантаження JQuery

Можна завантажити бібліотеку з сайту JQuery і пов'язати його зі сценарієм JQuery безпосередньо, але я завантажую з серверів Google:

<script type=”text/javascript”

Charset=”utf-8”

Src=”http://ajax.googLeapis.com/ajax/libs/jquery/1.4.2/jquery.min.js>

</script>

Кількість одночасних підключень до сервера з браузера має обмеження. Якщо розподілити графіку і сценарії по декількох серверах, це прискорить завантаження сторінок користувачами. Використання мережі доставки контенту Google також має додаткову перевагу - так як інші сайти зв'язуються з бібліотекою JQuery в Google, вона може вже перебувати в кеші їх браузерів (як вам, мабуть, відомо, для ідентифікації кешованої копії браузер використовує повний URL-адреса файлу). З іншого боку, якщо збираємося працювати з JQuery на портативному або настільному комп'ютері, що не має постійного підключення до Інтернету, використовуємо локальну копію.

Після того як бібліотека JQuery буде завантажена сторінкою, можна починати працювати з елементами. JQuery містить функцію з ім'ям $(); ця функція є «ядром» бібліотеки. Розглянемо кілька прикладів її використання. Для пошуку тега 11 на сторінці використовується запис виду

$ ("H1 ");

Якщо буде потрібно знайти всі елементи з класом important, використовується запис $ (". important ");

Ще раз подивимося на ці два приклади. Вони розрізняються тільки селектором CSS. Функція JQuery повертає об'єкт JQuery - спеці ний об'єкт JavaScript з масивом елементів DOM, відповідних селектору. Об'єкт визначає багато корисних методів для виконання операцій з відібраними елементами.

Методи зміни контенту

Методи hide() і show() приховують і відображають елементи інтерфейсу. Щоб приховати один або кілька елементів на сторінці, використовується запис наступного виду: $ ("h1").hide ();

Для відображення прихованих елементів досить викликати метод $show(). Метод hide() часто використовується в книзі для приховування розділів сторінок, які повинні відображатися тільки при відключенні JavaScript (наприклад, розшифровок аудіороликів або іншого обхідного контенту).

Методи html,var і attr

Метод html() використовується для читання і запису внутрішнього контенту заданого елемента.

$("message") .html("Hello world!");

У цьому прикладі між тегами що відкриває і закриває записується текст «Hello world».

Метод va1() призначений для читання і запису значень полів форм. Він gрацює практично так само, як метод html(). Метод а attr() призначений для читання і запису атрибутів елементів.

Методи append, prepend і wrap

Метод append() додає новий дочірній елемент після існуючих елементів. Припустимо, є проста форма і порожній неупорядкований список.

```
<form id=”add”>
    <label for=”task”>Task</label>
    <input type=”text” id=”task”>
    <input type=”submit” value=”add”>
</form>
<ul id=”links”> </ul>
```

Нові елементи списку можна створювати, приєднуючи їх при відправці форми

```
$(function() {
    $ ( "#add"). submit(function(event) {
    event.preventDefault();
    var new_element = $(“<li>” + $(“#email”). val() + “</li>”);
  });
});
```

Метод prepend() працює за аналогією з методом append(), але нові елементи вставляються НЕ після існуючих, а перед ними. Метод wrap() «упаковує» обраний елемент в елемент, представлений заданим об'єктом JQuery.

```
var wrapper = $ ( "#message") .wrap ( "<div><h2>message</h2></div>"). parent();
```

Використовуючи ці методи, можна на програмному рівні створювати достаточно складні структури.

CSS і класи

Метод css() використовується для визначення стилів елементів.

```
$( “label”).css( “color”, "#f00");
```

Також можна скористатися синтаксисом хешу JavaScript для примінення до елементу відразу декількох правил css:

$ ("h1").css ({ "color": "red",

"tetxt-decoration": "underline"});

Втім, подібне змішання стильового оформлення з сценарним кодом небажано. Методи JQuery addClass() і removeClass() можуть використовуватися для додавання і видалення класів при виникненні певних подій; стилі краще асоціювати з цими класами. Наприклад, події JQuery в поєднанні з класами дозволяють змінювати фон полів форми при отриманні і втрати ними фокуса:

$ ("input") .focus $ (function(event) {

$(this).addClass(“focused”);

});

$ ("input") .blur (function(event) {

$ (this) .removeClass("focused");

});

Цей тривіальний приклад можна замінити псевдокласом css3:focus, але в деяких браузерах цей псевдоклас не підтримується.

Зчеплені виклики

Методи об'єктів JQuery повертають об'єкти JQuery; отже, виклики методів можна зчіплювати до довільної довжини.

$ ("h2") .addClass ("hidden") .removeClass("visible");

Однак зловживати зчепленими викликами не рекомендується, тому що вони ускладнюють читання коду.

Створення елементів

Час від часу потрібно створити новий елемент HTML для його вставки в документ. Для створення елементів можна скористатися методом jquery().

var input = $ ("input");

Хоча того ж результату можна добитися викликомdocument.creaet-Element ("input") ;, використання функції JQuery спрощує додатково тільrі виклики методів.

Var element = $ ("<p> Hello world </ p>");

Element.css ("color", "#f00") .insertAfter ("# header");

Цей приклад в черговий раз показує, як зчеплені виклики Jquery допомагають створювати структури і виконувати різні операції з ними.

Події

Часто при взаємодії користувача зі сторінкою повинні ініціюватися різні події. У JQuery багато поширених події являють собою звичайні методи об'єкта

JQuery з параметром - функцією. Наприклад, щоб всі посилання на сторінці з класом popup відкривалися в новому вікні, виконуємо наступний фрагмент:

```
Var links = $(“#links a”);
Links.click(function(event) {
    Var address = $(this).attr(‘href’);
    Event.preventDefault();
    Window.open(address);
});
```

У обробнику події JQuery для звернення до поточного елементу використовується ключове слово this. У рядку 3 воно передається функції JQuery, де для нього викликається метод attr() з метою швидкого получення адреси посилання.

Функція preventDefault() запобігає ініціювання вихідного події, щоб воно не заважало нашій обробці.

Прив'язка

Деякі події не підтримуються JQuery безпосередньо; для їх обробки можна скористатися методом bind(). Наприклад, при реалізації перетягування з специфікації HTML5 необхідно скасувати подія ondragover. Метод bind() використовується наступним чином.

```
target = $ ( “#droparea”)
target.bind(‘dragover’,function(event) {
    if (event.preventDefault) event.preventDefault();
    return false;
});
```

Для оброблюваного події префікс on не вказується.

початкове подія

При використанні подієвих функцій JQuery (таких, як bind() або click()) JQuery інкапсулює подію JavaScript в окремому об'єкті, копіюючи в нього лише частину вихідних властивостей. Іноді потрібно надати доступ до вихідної події для звернення до властивостей, що не були скопійовані. Події JQuery надають доступ до начальній події через властивість originalEvent. Звернення до властивості data події onmessage виглядає наступним чином:

```
$(window).bind(“message”,function(event) {
    Var message_data = event.originalEvent.data;
});
```

Властивість orginalEvent може використовуватися для звернення до любих властивостях і методах вихідної події.

5.2 функція document.ready

Виявом «ненав'язливий JavaScript» позначається код JavaScript, повністю відділений від контенту. Замість включення атрибутів onclick в елементи HTMГL використовуємо обробники подій. Поведінка додається в документ без зміни самого документа, а розмітка HTML не залежить від того, чи включена призначена для користувача підтримка JavaScript.

Недолік такого підходу полягає в тому, що JavaScript «не бачить» ніякі елементи документа до їх оголошення. Код JavaScript можна було б включити в блок script в кінці сторінки, але таке рішення перешкоджає повторному використанню коду між сторінками. Код можна було б включити в обробник події JavaScript window.onLoad(), але подія ініціюється після завантаження всього контенту. Затримка означає, що користувачі будуть взаємодіяти з елементами до підключення до них подій.

Функція JQuery document.ready вирішує саме це завдання, до того ж це рішення працює у всіх браузерах. Приклад її використання:

```
$(document).ready(function() {
    Alert(“HI”);
});
```

Було розглянуто лише мала частина того, що можна зробити завдяки JQuery. Крім маніпуляцій з документом, JQuery надає методи серіалізації форм, створення запитів Ajax, а також ряд допоміжних функцій, що спрощують перебір і переміщення по моделі DOM. Безсумнівно, у міри освоєння JQuery знайдемо багато інших можливостей для використання цієї бібліотеки в проектах.

6 Графика та відео

Впровадження відео і аудіо

Аудіо- та відеоматеріали стали важливою частиною сучасного Інтерне- та. Подкасти, аудіо-коментарі і навіть відео-інструкції з'явилися повсюдно, і до теперішнього часу для їх відтворення вимагалися спеціальні браузерні плагіни. У HTML5 з'явилися нові механізми вбудовування аудіо- та відео-файлів в сторінку.

6.1. Трішки історії

Спроби використання аудіо та відео в веб-сторінках почалися давно. Спочатку розробники впроваджували Midi-файли в свої домашні сторінки, і використовували тег embed для посилань на них.

```
<embed src=”muzika.mp3” autostart=”true”
    Loop=”true” controller=”true”></embed>
```

Тег embed так і не став стандартним, тому замість нього стали користуватися тегом object, прийнятий в якості стандарту W3C. Для старих браузерів, які не підтримують тег object, часто використовується конструкція з вкладенням тега embed в object.

Однак не кожен браузер міг здійснювати потокове відображення контенту,

і не кожен сервер був правильно налаштований для роботи з ним. Ситуація ще сильніше ускладнилася з ростом популярності відео в Web. Робота з аудіо-та відео-контентом в пройшла багато ітерацій, від Real Player до Windows Media і Quick Time. Кожна компанія мала свою стратегію роботи з відео; часом здавалося, що в Інтернеті не знайти два сайти з однаковими методами і форматами кодування відео.

Фірма MacroMedia (нині Adobe) досить швидко усвідомила, що її Flash Player може стати ідеальним крос-платформних засобом відтворення аудіо- і відео-контенту. Технологію Flash підтримує приблизно 97% браузерів. Коли виробники контенту зрозуміли, що контент після однократного кодування можна відтворювати де завгодно, тисячі сайтів перейшли на потокові технології Flash для роботи з аудіо і відео.

Потім в 2007 році фірма Apple випустила iPhone і iPod Touch. Було вирішено, що Apple не підтримуватиме flash на цих пристроях. Постачальники контенту відреагували на цю новину створенням відео- потоків, нормально відтворюваних в браузері Mobile Safari. Ці відео-потоки, що використовують кодек h>264, також могли відтворюватися звичайним програвачем Flash Player, що дозволило постачальникам контенту, як і раніше, забезпечувати відтворення на різних платформах при одноразовому кодуванні.

Творці специфікації HTML5 вважають, що браузер повинен мати вбудовану підтримку аудіо і відео, замість використання плагінів, що вимагають великого обсягу шаблонного коду HTML. Саме тут проявляється основна стратегія роботи з аудіо і відео в HTML5: аудіо і відео стали повноправними видами веб-контенту.

6.2 Контейнери та кодеки

Під час обговорення роботи з відео Web часто використовуються терміни «контейнер» і «кодек». Контейнер можна порівняти з конвертом, содержащім аудіо- і відео-потоки, а також іноді додаткові метаданні (наприклад, субтитри). Ці аудіо- і відео-потоки повинні бути якось закодовані; це завдання вирішується за допомогою кодеків. Існують сотні різних способів кодування аудіо і відео, але в контексті відео HTML5 важливі лише кілька з них.

Відеокодеки

При перегляді відео програвач повинен декодувати його. На превеликий жаль, може виявитися, що використовується програвач котрий не здатний декодувати той відео-потік, який хочемо подивитися. Деякі програвачі використовують програмне декодування відео-потоку, яке може виконуватися більш повільно або сильніше витрачати ресурси процесора. Інші програвачі використовують апаратне декодування, а отже, їх можливості відтворення більш обмежені.

Кодеки і їх підтримка браузерами

H.264 - високоякісний кодек, створений групою MPEG і став стандартом в 2003 році. Для забезпечення підтримки пристроїв з мінімальними можливостями (наприклад, телефонів) паралельно з підтримкою пристроїв високої чіткості специфікація H.264 ділиться на профілі. Вони володіють деякими загальними

можливостями, але профілі більш високого рівня надають додаткові можливості для поліпшення якості. Наприклад, iPhone і Flash Player можуть відтворювати відео-потік, закодований за стандартом H.264, але iPhone підтримує тільки низькоякісний «базовий» профіль, тоді як Flash Player підтримує потоки більш високої якості. Відео-потік можна закодувати один раз з внедренням декількох профілів - в цьому випадку він буде добре виглядати на різних платформах.

H.264 є фактичним стандартом завдяки підтримці фірм MicroSoft і Apple, які є власниками ліцензій. Крім того, відео-сервіс Google YouTube перетворив свої відеоролики з використанням кодека H.264, щоб вони могли відтворюватися наіPhone; крім того, H.264 підтримується програмою Flash Player фірми Adobe. Проте технологія не є відкритою. Кодек запатентован, а його використання визначається умовами ліцензування. Виробники контенту повинні платити ліцензійні відрахування за кодування відео з використанням кодека H.264, однак ці відрахування не поширюються на контент, безкоштовно надається кінцевому користувачу.

Прихильники вільного поширення ПО турбуються про те, що з часом правовласники почнуть вимагати високих відрахувань від виробників контенту. Це призвело до створення і поширення альтернативних кодеків.

Theora

Theora - безкоштовний кодек, розроблений Xiph.Org Foundation. Хоча виробники контенту можуть використовувати Theora для створення відео якості, порівнянного з H.264, виробники пристроїв не поспішають з його підтримкою. Firefox, Chrome і Opera можуть відтворювати відео в форматі Theora на будь-якій платформі без додаткового програмного забезпечення, але Internet Explorer, Safari і пристрої iOs такої підтримки не надають. Apple і Microsoft побоюються «прихованих патентів» - цим терміном позначається навмисна затримка публікації і оформлення патенту для збереження скритності, поки інші реалізують технологію. Коли приходить відповідний момент, право-володар патенту «заявляє про себе» і починає вимагати ліцензіонні відрахування з нічого не підозрюющего ринку.

VP8

VP8 фірми Google - повністю відкритий безкоштовний кодек, по якості схожий з H.264. Він підтримується в Mozilla, Google Chrome і Opera, а Microsoft обіцяє забезпечити підтримку VP8 в Internet Explorer 9 за умови, що кодек вже встановлений у користувача. Крім того, кодек підтримується в Adobe Flash Player, що робить його цікавою альтернативою. З іншого боку, кодек не підтримує в Safari і пристроях iOs; таким чином, не дивлячись на безкоштовність кодека, виробникам відео-контенту для iPhone або iPod все одно приходиться використовувати кодек H.264.

Аудіокодеки

AAC (Advance Audio Coding)

Фірма Apple використовує цей аудіо-формат в своєму магазині iTunes Store. Кодек забезпечує більш високу якість, ніж MP3, при сходном розмірі файлу, а також підтримує кілька аудіо-профелів за аналогією з H.264. Крім того, як і у

випадку з H.264, використання кодека вимагає ліцензійних відрахувань. Файли AAC відтворюються всіма продуктами Apple, а також Adobe Flash Player і програвачем з відкритим кодом VAC.

Vorbis (OGG)

Безкоштовний формат з відкритим кодом, що не вимагає ліцензійних відрахувань; підтримується в FireFox, Opera і Chrome. Може використовуватися в поєднанні з відеокодеками Theora і VP8. Файли Vorbis забезпечують хорошу якість звуку, але недостатньо широко підтримуються апаратними програвачами.

MP3

Формат MP3, незважаючи на свою виняткову популярність, що не підтримується в Firefox і Opera через патентні ускладнення. З іншого боку, він підтримується в Safari і Google Chrome.

Для відтворення і поширення відео-контенту відео- і аудіо-потоки упаковуються в один контейнер. Отже, що ж таке «Відео-контейнер»?

Контейнери та кодеки: спільна робота

Контейнер являє собою файл метаданих, який описує і об'єднує аудіо- і відео-файли. Контейнер не містить інформації про те, яким чином закодована міститься в ньому інформація. По суті, контейнер є «обгорткою» для аудіо- та відео-потоків. У загальному випадку контейнер може містити довільну комбінацію закодованих потоків, але при роботі з відео в зазвичай використовуються наступні комбінації:

- Контейнер OGG з відео Theora і аудіо Vorbis - працює в Firefox, Chrome і Opera.
- Контейнер MP4 з відео H.264 і аудіо AAC - працює в Safari і Chrome. Також відтворюється вAdobe Flash Player, на пристроях iPhone, iPod і iPad.
- Контейнер WebM з відео VP8 і аудіо Vorbis - працює в Firefox, Chrome, Opera і Adobe Flash Player.

Google і Mozila в своєму розвитку орієнтуються на VP8 і, так що Theora згодом можна буде виключити з розгляду. Однак поточний стан справ такий, що ми все одно стикаємося з необхідністю дворазового кодування відео - для користувачів Apple (які займають відносно малу нішу в секторі комп'ютерів, але є власниками значної частини мобільних пристроїв в США) і для користувачів Firefox і Opera, так як обидва цих браузера відмовляються відтворювати H.2641.

РОЗДІЛ 3 СПОСІБ ВИРІШЕННЯ ЗАДАЧ, ВИБІР СИСТЕМ

1 БУТСТРАП

Bootstrap- це CSS-фреймворк. По-перше, Bootstrap є найпопулярнішим фреймворком, у його найближчого конкурента в 3-5 разів менше співтовариство. По-друге, це не тільки css, а й js-фреймворк. Тобто в Bootstrap написані готові стилі і скрипти, для застосування яких достатньо всього лише прописати необхідні стильові класи і атрибути html-елементів. Завдяки сітці Bootstrap, можливо легко адаптувати будь-який сайт і добре відображати його на будь-яких пристроях.

Css-фреймворк - це файл або декілька файлів з готовим написаним кодом, які підключаються в сайту в секції head, після чого стає можливим використання можливостей цього фреймворка. Фреймворки створюють для того, щоб іншим веб-розробникам було легше верстати сайти.

Якщо робити розробку сайту з нуля, то доведеться подбати про дуже багато речей. Все css-стилі, все веб-сценарії доведеться писати з нуля, але ж це можуть бути сотні і тисячі рядків коду. Причому можливо зробити масу помилок у верстці. Наприклад, просто шаблон буде по-різному виглядати в основних браузерах або він буде не адаптивен. Взагалі то, якраз заради адаптивної верстки і варто використовувати Bootstrap, тому що якщо говорити про фіксовані макетах, то їх легко зробити навіть з нуля. Просто створюємо блоки, задаємо їм фіксовану ширину і працюємо по макету. Але у випадку з адаптивною версткою все в рази складніше. Потрібно буде зробити так, щоб на будь-яких дозволах екранів сайт відображався добре. Для цього доведеться використовувати медіа-запити. Для великих шаблонів таких ось запитів може знадобитися дуже багато, крім того, ще потрібно їх правильно писати.

А що ж bootstrap? Якщо вивчити цей фреймворк, то він сильно спростить верстку. По-перше, фреймворк бере на себе кросбраузерность і адаптивність, а це основні речі, про які повинен подбати розробник. Але з bootstrap реалізувати їх дуже просто. Це дозволяє створити html-шаблон навіть людині, який раніше дуже мало займався версткою і особливо не знайомий з css.

По-друге, фреймворк ідеально підходить при роботі в команді. Верстка на bootstrap при належному вмінні і розумінні відбувається в 3-5 разів швидше, а однаковість коду дозволить будь-якому колезі внести правки. Якщо ж говорити про верстку без фреймворку, то тут і кожного розробника може бути свій стиль і іншій людині доведеться витратити час на вивчення його коду.

Недоліки Bootstrap

По суті, їх усього два. Перший - коду зазвичай в бібліотеці написано більше, ніж якщо б писати при розробці з нуля. Тому що коли при роботі самостійно, йде реалізація тільки необхідного функціоналу і все. У Bootstrap ж є все на всі випадки життя. Навіть те, що може не знадобитися. Але знову ж таки, ця проблема дуже легко вирішується тим, що Bootstrap надає змогу самому

вибирати, які компоненти фреймворка завантажити в css-файл. Наприклад, взагалі можна завантажити тільки сітку, а все інше робити самостійно.

Другий недолік - шаблонний дизайн. Так, дійсно, дуже часто при відвідуванні різних сайтів можна побачити однакові кнопки, вони зроблені в Bootstrap, тому що аж надто це очевидно. Але і ця проблема легко вирішується, тому що вона буде існувати тільки в тому випадку, якщо використовувати тільки готові компоненти фреймворка і нічого ніколи не кастомінизувати під себе.

А якщо, наприклад, підключіть тільки сітку Bootstrap, тоце надає змогу відтворити будь-який дизайн, при цьому користуючись дуже зручною гнучкою сіткою фреймворка.

Компоненти фреймворка

Bootstrap є всеосяжним фреймворком. Це означає, що в нього закладено багато компонентів. По суті, все, що може знадобитися при розробці типових сайтів. Це, наприклад, меню, що випадає, кнопки, АЛЕРТ, таби, індикатори стану, хлібні крихти, списки, заголовки і т.д. У Bootstrap ж досить вивчити трохи сам фреймворк і всі ці речі можна використовувати дуже швидко.

Розглянемо приклад коду написання кнопок:

<Button type = "button" class = "btn btn-default"> За замовчуванням </ button>

<Button type = "button" class = "btn btn-primary"> Основний </ button>

<Button type = "button" class = "btn btn-success"> Успіх </ button>

<Button type = "button" class = "btn btn-info"> Інормірованіе </ button>

<Button type = "button" class = "btn btn-warning"> Провал </ button>

<Button type = "button" class = "btn btn-danger"> Попередження </ button>

Зрозуміло, щоб все спрацювало, bootstrap вже повинен бути підключений до html-документів. Зауважимо, як все працює. По-перше, є універсальний клас btn, який визначає загальні стилі для всіх кнопок. По-друге, вже безпосередньо для додаткової стилізації використовується інший стильовий клас. Таким чином в Bootstrap виконується робота і з усіма іншими компонентами.

Окремо хотів би виділити такий компонент, як іконочний шрифт. Він дуже сильно полегшує роботу з іконками. А саме, не буде потрібно завантажувати ці іконки у вигляді зображень. За замовчуванням в Bootstrap є близько 200 ікон, вставляти їх на веб-сторінки дуже просто, до прикладу можна ознайомитися в офіційній документації.

Bootstrap: з чого почати роботу з фреймворком?

Почати потрібно, звичайно ж, з відвідування офіційного сайту getbootstrap.com. Далі потрібно перейти на сторінку "З чого почати" або Getting Started. На ній буде надана можливість завантажити фреймворк одним із способів, дуже добре підійде для початку найперший, тобто просте завантаження

повного фреймворка. Крім цього, можна підключити потрібні файли через CDN. Це означає, що їх не доведеться завантажувати собі на комп'ютер, копіювати в папку з проектом, а всього лише потрібно прописати в секції head сайту підключення тих файлів з cdn-сховища.

За допомогою такого підходу можна позбутися зайвого коду і залишити тільки те, що потрібно. Зазначу, що сторінка кастомізації дуже довга, на ній можна і навіть потрібно налаштувати дуже багато параметрів, наприклад, кольору за замовчуванням для різних компонентів і т.д. На цій сторінці можна провести дуже багато часу, але зате в результаті отримати унікальну версію Bootstrap, заточену під потрібний проект. Далі потрібно лише нажати на кнопку, щоб завантажити фреймворк.

2 Видео-плеєр

Вставка відео на веб-сторінку

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<video width="450" height="450" controls>
    <source src="vidosik.mp4 type="video/mp4">
</video>
</body>
</html>
```

Це базовий кістяк HTML5 майбутнього відео-плеєра. Він використовує 10 основних рядків коду, які дозволять відео відображатися на будь-який веб-сторінці з основними кнопками управління. Почнемо з розмітки HTML, в ній використовується універсальне оголошення doctype <! DOCTYPE html>. Це перше, з чого починається будь-який HTML-документ. Воно потрібне для того, щоб браузер був в курсі, який документ ви використовуєте. Тепер перейдемо до елементів, які потрібно включити в HTML: <head> і <body>. Зараз ми повинні зосередитися на тому, що відбувається в body. Неможливо створити відео без тега <video>. Усередині <head> вставляємо <video>. Тепер в тезі <video> потрібно вказати, які розміри повинен мати плеєр (рекомендується встановити розміри плеєра, щоб уникнути мерехтіння). Джерело відео, яке потрібно відтворити в плеєрі, і зображення обкладинки. Це буде презентацією відео, яке глядачі побачать, перш ніж натиснуть кнопку «Play».

Тепер розглянемо доступні атрибути та подивимося, як вони працюють. Атрибут poster - він потрібен для створення зображення-презентації відео. У ньому необхідно вказати папку із зображенням і назва файлу. Потім потрібно вибрати ширину і висоту плеєра. Я вирішив вибрати симетричну форму. Щоб зібрати плеєр для сайту, важливо вставити атрибут «controls». Без нього ви

можете управляти своїм відео тільки правою кнопкою миші, а потім вибрати «Відтворити» або інші основні функції. Тег <controls> відображає основний масив елементів управління: кнопки «Відтворити», «Пауза», «Гучність» і кнопку повноекранного режиму для більш зручного використання функцій. Далі йде тег <source>, в якому необхідно вказати атрибут src з джерелом відео. Оскільки тег <video> підтримує три формату відео (MP4, WebM і Ogg) необхідно вказати в атрибуті type, який з них використовується. Для зручності користувачів рекомендується використовувати якомога більше версій відео. Тому, якщо користуємося лише одним форматом .ogg-версією відео, потрібно відкрити ще один тег <source>. Наприклад: <source src = "videoexample.ogg" type = video / ogg>.

Позиціонування відео-плеєра за допомогою CSS

Створюваний плеєр для сайту буде перебувати в , який в свою чергу буде містити два інших :

```
<!DOCTYPE html>
<html>
<head></head>
<body>
    <div id=”video-player”>
    <div id=”video-tree”>
        <video width=”450” height=”450” controls>
            <source src=”vidosik.mp4 type=”video/mp4”>
    </div>
    <div id=”progress-tree”>
    </div>
    <div id=”button-tree”>
    </div>
</body>
</html>
```

Потім потрібно «побудувати майданчик» для CSS-коду. Для цього було створено три ідентифікатора всередині великого тега div з ім'ям video-player, оскільки - це мета порталу. Перший div-контейнер відповідає за скелет відео. Сюди перенесено початкові рядки тега <video>, який було створено раніше на декілька сторінок вище. Другий div-контейнер містить індикатор перегляду, а третій - кнопки відео-плеєра. Кожен тег повинен мати унікальний ідентифікатор:

```
<!DOCTYPE html>
<html>
<head></head>
<body>
    <div id=”video-player”>
    <div id=”video-tree”>
        <video width=”450” height=”450” controls>
            <source src=”vidosik.mp4 type=”video/mp4”>
    </div>
    <div id=”progress-tree”>
        <div id=”progress”></div>
    </div>
    <div id=”button-tree”>
        <img id=”play-button” width=”100 height=”100”>
            <div id=”time-factor>
                0:00 / 0:00
            </div>
        <img id=”backward-button” width=”100 height=”100”>
        <div id=”soundbar-tree”>
            <div id=”soundbar” width=”100 height=”100”></div>
        </div>
    <img id=”forward-button width=”100 height=”100”></div>
    </div>
</body>
</html>
```

Далі задаємо кожному необхідні атрибути. Таким чином, у div video-tree є video теги. <Div> progress-tree відповідає за індикатор виконання, тому має ідентифікатор «progress». <Div> button-tree вимагає більше уваги. Було вставлено три кнопки: play (відтворити), back (назад) і next (вперед). Таким чином, кожна кнопка укладена в свій власний тег , має власний ідентифікатор («play-button», «backward-button» і «forward-button») і розміри (100 на 100 пікселів для кожної кнопки). У кнопки відтворення є своя шкала часу, яку встановлено в з ідентифікатором «time-factor». Не забуваємо також використовувати обмеження часу «0: 00/0: 00», які являють собою час початку і момент часу, якого

досягло відео.

Стилі відеоплеєра

Наступний етап потребує файлу з розширенням .css, тому у тій же папці що і знаходиться .html було створено ще один файл. Він допоможе зробити плеєр більш комфортабельним та інноваційним. Тепер повернемося в файл html і додаємо в тег <head> атрибути, які зв'яжуть файл html з css-файлом: <link rel = "stylesheet" type = "text / CSS" href = "video -player.css»>. Незалежно від структури, яку хочемо використовувати в файлі css, просто вказуємо елемент з id, який відзначили в html-файлі, вказавши на початку #. Так йде повідомлення редактору коду, яку частину необхідно стилізувати першої:

```
#video-player {
    Background-color: aquamarine;
    Display: inline-block;
}
Img {
    Background-color: aqua;
}
#video-tree {
    Width: 640 px;
    Height: 640 px;
}
Video {
    Height: 100% ;
    Width: 100% ;
}
#progress-tree {
    Background-color: CadetBlue;
}
#progress {
    Height: 5 px;
    Width: 50% ;
    Background-color: black;
}
#button-tree {
    Height: 50 px;
```

```
}
#button-tree > * {
    Display: inline-block;
    Height: 50 px;
    Width: 50 px;
    Vertical-align: middle;
}
```

У відеоплеєра синій фон, він обмежений розмірами дисплея плеєра, так як функція display має значення inline-block. Тому веб-сторінка не стане повністю синьою, так як синій фон буде обмежений розмірами відеоплеєра. Наступний елемент проектування - це video-tree, для якого я вибрав потрібні розміри, і вказав, щоб відео виводилося на весь екран. Для progress-tree вибрано тільки колір, і більше зосередився на гілці «progress», що визначає індикатор перегляду. Для button-tree було створено дві різні записи. Перший запис фокусує свою увагу виключно на ширині кнопок. Другий запис управляє кнопками при горизонтальній перебудові за допомогою команди «display: inline-block» і центрується атрибутом «vertical-align: middle».

Створення функціональності за допомогою JavaScript

Потім потрібно зв'язати файл JavaScript з вихідним файлом HTML5 рядком між тегом <link> і закриває тегом <head>. Наприклад: <script type = "text / javascript" src = "video-player.js"> </ script>:

```
Window.addEventListener('load', function() {
    Video = document.getElementById ('video');
    Play-button = document. getElementById ('play-button');
    Play-button. addEventListener ('click', playOrPause, false);
}, false;
Function playOrPause () {
    If (video.paused) {
        Video.play();
    } else {
        Video.pause ();
    }
}
```

Спочатку ми вводимо ідентифікатор елемента, з яким хочемо працювати в першу чергу. У нашому випадку це ідентифікатор «play-button». Потім необхідно прописати форму кнопці через GetElementbyID. Далі, коли глядач натискає на кнопку відтворення, бробляємо «Click» за допомогою методу addEventListener.

Функція «playOrPause» змушує кнопку «Відтворити» працювати, як звичайну кнопку відтворення, а також як кнопку «Пауза». Потім в коді створення плеєра для сайту описуємо функцію playOrPause. Якщо відео призупинено, натискання кнопки активує відтворення. Якщо не призупинено (блок «else»), натискання кнопки «Відтворити» зупинить відтворення.

3 Хостинг

Хостинг - це послуга надання ресурсів на сервері для розміщення інформації (найчастіше файлів сайту) в мережі Інтернет. Зазвичай під хостингом мають на увазі послугу розміщення файлів сайту на заздалегідь налаштованому сервері з програмним забезпеченням, необхідним для обробки запитів до цих файлів і супутніми сервісами (веб-сервер, сервер баз даних, DNS-сервер, PHP-обробник, FTP-сервер, поштовий сервер) . Є й інші способи використання хостингу. На ньому можна розміщувати програми, що вимагають виходу в інтернет (1С Бухгалтерія або MetaTrader), створити власний сервер (наприклад, ігровий), організувати файло-обмінник або просто зберігати файли для особистого користування. Хостинг провайдер - це компанія, що надає послуги хостингу.

види хостингу

- Віртуальний хостинг. Його особливістю є те, що всі ресурси сервера розділені між обліковими записами хостингу, в кожній з яких може бути розміщено кілька сайтів. На такому хостингу вже є необхідна для роботи сайтів програмне забезпечення - веб-сервер, сервер баз даних, FTP-сервер, PHP-обробник.
- Реселлер-хостинг. Готове рішення для перепродажу хостингу. Купуючи реселлер-хостинг, ви стаєте посередником між хостинговим провайдером і кінцевим користувачем.
- Віртуальний сервер (VPS-сервер або VDS-сервер). Ця послуга має на увазі, що на одному фізичному сервері може бути запущено кілька віртуальних машин, кожна з яких повністю ізольована від решти. Орендар такого сервера отримує повний доступ до управління ВПС (на рівні адміністратора / root-користувача).
- Виділений сервер. Окремий фізичний носій, всі ресурси якого знаходяться у вашому розпорядженні. Такий варіант необхідний для розміщення ресурсо-ємкого проекту з великою кількістю відвідувачів, наприклад, середнього або великого інтернет-магазину.
- Колокація. Розміщення власного виділеного сервера в дата-центрі хостінгового провайдера.

Вибір хостингу: технічні параметри

Вибір правильного хостинг-пакету з технічного боку важливий, тому що саме це в подальшому визначить, наскільки стабільно буде працювати сайт або ж додаток. Самі параметри потрібно вибирати виходячи з потреб і вимог

розробників програмного забезпечення і скриптів, які будемо розміщувати. Слід звернути увагу на наступні технічні параметри хостінгового пакета:

- операційна система (Linux або Windows);
- підтримка програмного забезпечення (PHP, MySQL, Java);
- Об'єм оперативної пам'яті;
- ресурси процесора;
- дисковий простір (а також тип дисків - HDD або SSD);
- обсяг місячного трафіку;
- швидкість передачі даних;
- розташування серверів (рекомендується вибирати дата-центр, близький до аудиторії проекту);
- наявність обмежень щодо використання послуги (кількість розміщуваних сайтів, файлів, баз даних).

Вибір хостингу: організаційні моменти компанії

Вибір провайдера так само важливий, як і вибір правильного пакету. Саме цій компанії будемо довіряти дані, тому необхідно переконатися, що дані нікуди не зникнуть, а на будь-які питання завжди зможе відповісти співробітник компанії:

- наявність компетентної цілодобової підтримки;
- зручні методи оплати;
- відгуки про хостинговому провайдера;
- наявність повних юридичних реквізитів компанії;
- наявність тестового періоду;
- можливість повернення коштів.

Зазначені вище параметри є не єдиними, але одними з найбільш важливих критеріїв вибору хостингу.

Віртуальний хостинг

Віртуальний хостинг (shared-хостинг) - це такий вид хостингу, при якому сервер поділений на велику кількість акаунтів, що поділяють одні й ті ж IP-адреси. Віртуальний хостинг сайтів найпоширеніший і економічний, але і має ряд обмежень. Варто розуміти, що деякі ресурси (оперативна пам'ять, трафік, процесорний час) всі користувачі на одному сервері будуть ділити між собою. Проте, якщо невеликий сайт, ресурс з невисокою відвідуваністю, то віртуальний хостинг підійде як не можна до речі.

Кількість компаній, що надають такий тип хостингу, величезна і тому для

того, щоб обійти конкурентів, вони пропонують безлімітний трафік або дисковий простір за дуже низьку ціну. Але за привабливою дешевизною часто ховається і ряд неприємних проблем: низька швидкість завантаження сторінок, погана продуктивність, часті збої в роботі серверів, некомпетентна служба підтримки, перевантажені сервери з величезною кількістю акаунтів і так далі. Також існує ризик зіткнутися з хостингових компаній-шахраями: сьогодні ці компанії існують і пропонують вам дешеві пакети, а завтра вони закриваються і ми не отримуєте ні хостингу, ні повернення грошей.

На які параметри потрібно звернути увагу у першу чергу:

- дисковий простір;
- обмеження трафіку;
- характеристики баз даних;
- кількість веб-сайтів (доменів), які можемо розмістити в одному акаунті;
- кількість піддоменів і паркованих доменів;
- можливість покупки власного IP-адреси;
- кількість можливих FTP-акаунтів;
- кількість поштових акаунтів (ящиків), які зможемо створити в одному акаунті.

Це одні з найважливіших параметрів, які розрізняють можливі пакети віртуального хостингу. А ось при виборі хостинг-провайдера важливо звертати увагу на наявність і тривалість тестового періоду. Можливість тестування послуги перед її покупкою є практично у всіх хостинг-провайдерів, однак, тривалість тестування всі пропонують різну. У кого-то це 3 дні, у кого-то 7, а хтось дасть, навіть, безкоштовно користуватися послугами протягом цілого місяця.

Важливо згадати і про контрольну панелі управління, за допомогою якої зможемо управляти хостингом. Контрольних панелей існує багато, але у випадку з віртуальним хостингом на лідируючій позиції знаходиться cPanel - універсальна і в той же час інтуїтивно зрозуміла навіть для недосвідченого користувача система.

Для чого підійде віртуальний хостинг: для простих сайтів-візиток, інформаційних сайтів, среднепосещаемих блогів і форумів, а також інтернет-магазинів з мінімальними вимогами.

переваги:

- низька вартість;
- зрозуміле навіть новачкам управління своїм хостинговим акаунтом через зручну панель;
- повна підтримка хостінгового провайдера при проблемах, пов'язаних з роботою сервера.

недоліки:

- обмежені адміністративні права;
- неможливість визначати програмну конфігурацію;
- необхідність ділити деякі серверні ресурси з іншими акаунтами, розташованими на тому ж сервері, що і наш .

Віртуальні виділені сервери (VPS / VDS)

Віртуальний виділений сервер передбачає, що фізичний сервер розділений на кілька віртуальних серверів, кожен з яких абсолютно ізольований від інших, має свою операційну систему і може бути перезавантажений незалежно від інших серверів. Чим відрізняється VDS від VPS? Нічим, ці дві абревіатури означають одне і те ж. VDS від англ. - virtual dedicated server, а VPS - virtual private server. Dedicated і private можна сприймати як слова-синоніми. Кожен віртуальний сервер ділить між собою ресурси фізичного сервера, але при цьому вони відокремлені один від одного.

Кожен VPS ділить між собою ресурси фізичного сервера, але при цьому вони відокремлені один від одного. Якщо на одному VPS-сервері виникають які-небудь проблеми або він споживає занадто велику кількість ресурсів, то на інші VPS, розташовані на цьому ж фізичному носії, це ніяк не вплине. Послуга VPS є як би проміжною між віртуальним хостингом і повноцінним виділеним сервером.

Є безліч позитивних сторін вибору VPS-сервера: він значно дешевше повністю виділеного сервера, але має практично ті ж можливості; дає більше свободи і гнучкості в порівнянні з лімітами і обмеженнями віртуального хостингу. Якщо порівнювати VPS з віртуальним хостингом, то на останньому маємо вкрай обмежені адміністративні права і не можемо визначати програмну конфігурацію. Можливість VPS включають в себе виконання будь-яких дій з нашої контрольної панелі і немає необхідності звертатися в технічну підтримку кожного разу, коли необхідно щось оновити або налаштувати.

З VPS обмежені лише можливостями обраного сервера, тому вибираємо досить потужний для потреб веб-сайту. У різних хостингових провайдерів набір послуг, що входять в базове адміністрування (тобто, те, яке вже входить у вартість самого VPS) і в повне адміністрування (за додаткову плату) може відрізнятися. VPS-сервер з повним адмініструванням дозволить управляти сервером з тією ж легкістю, як і віртуальним хостингом. Так, в такому випадку не буде рутовий (адмінській) доступ до сервера, але все ще будемо приймати рішення про функціонування сервера.

VPS-сервер - це ідеальне рішення по співвідношенню ціни і якості для більшості великих сайтів, а також веб-проектів, яким потрібно налаштування операційної системи.

Важливо відзначити, що на відміну від віртуального хостингу, не отримаємо графічну панель управління типу cPanel за замовчуванням. Управління хостингом здійснюється за допомогою SSH або можемо самостійно встановити панель управління.

Функціональна частина віртуального виділеного сервера в більшості випадків практично нічим не поступається реальним серверам. Повний root-доступ дозволяє встановлювати будь-які доповнення та тонко підлаштованої конфігурації сервера під потреби.

Для чого потрібен VPS: для великих веб-проектів, що вимагають настройки операційної системи і самостійного визначення різних програмних конфігурацій.

переваги:

- необмежені адміністраторські права і root-доступ;
- можливість визначати програмну конфігурацію;
- незалежність від інших користувачів, розташованих на цій же фізичній машині;
- виділені ресурси віртуального сервера.

недоліки:

- ціна вище, ніж на віртуальний хостинг;
- потужність VPS все одно нижче, ніж у виділеного сервера;
- труднощі в адмініструванні при відсутності необхідних знань.

виділені сервери

Виділений сервер - це послуга, при якій отримуємо фізичну машину в своє повне розпорядження і платимо виключно за її оренду. Це найпотужніший варіант хостингу і підходить для веб-сайтів з найвищими вимогами. Як і у випадку з VPS-серверами, відсутність специфічних технічних знань не є перешкодою в управлінні виділеним сервером, оскільки цей тип хостингу також в більшості випадків передбачає можливість повного адміністрування за додаткову плату.

Одне з основних переваг виділеного сервера - це можливість докупити певні ресурси без переїзду на новий сервер. Якщо така необхідність виникає на віртуальному хостингу, то будемо змушені поміняти свій тариф на більш високий. З VPS-серверами все трохи простіше, тому що, зазвичай є можливість докупити деякі ресурси, але при цьому все одно вони будуть досить обмежені. У той же час для виділеного сервера можна легко докупити додатковий трафік, гігабайти RAM, збільшити швидкість порту і додати додаткові жорсткі диски.

Для чого підійде виділений сервер: для сайтів з великою відвідуваністю і високими ресурсними вимогами, розміщення додатків, роботи з великими базами даних.

переваги:

- повний root-доступ до сервера;
- можливість повністю визначати програмну конфігурацію;
- можливість докупити певні ресурси без переїзду на новий сервер;

- можливість забезпечити максимальну безпеку і потужність.

недоліки:

- висока вартість оренди;
- труднощі в адмініструванні при відсутності необхідних знань.

Хостинг для реселерів

Реселінг хостингу - цей вид хостингу дозволяє перепродувати хостингові послуги. По суті отримаємо від хостинг-провайдера аккаунт, який можна розділити на більш дрібні хостингові акаунти і продавати їх під своїм брендом. При реселлінгу хостинга отримуємо більш розширений доступ до сервера, ніж пересічний користувач віртуального хостингу, з можливістю керувати іншими хостингових акаунтами. Незважаючи на це, даний тип хостингу також відноситься до віртуального хостингу, просто з іншими умовами.

Колокейшен хостинг

Такий тип хостингу передбачає розміщення фізичної машини в дата-центрі хостінгового провайдера. По суті, йде оренда місця, де розміщується фізичний сервер, а також послуги підключення до інтернету і систем енергопостачання і охолодження. За все інше, що відбувається на фізичному сервері, відповідаємо тільки ми.

ВИСНОВКИ

Розробка порталу є поширеною практикою у програмістів. Проаналізувавши інші портали, їх недоліки та переваги, був розроблен власний портал з висновками та влиянням інших порталів.

Мною було вибрано розробку з HTML5, тому що , наразі, його підтримують усі браузери, css3 по тій же причині, а також JavaScript і Jquery, для полегшення перегляду портала на інших приладах.

Недоліком порталу лишається тільки піар, та мала база матеріалів відео-формату. Перевагою порталу – інтерактивний і зручний у використанні інтерфейс, використання трафику зменшено до мінімума у відео-плеєрі.

Із завданнями справився на відмінно, усі поставлені цілі було виконано.

СПИСОК ЛІТЕРАТУРИ

1 Методичні вказівки для виконання випускних кваліфікаційних робіт бакалавра (для здобувачів вищої освіти за напрямами підготовки 6.050103 «Програмна інженерія», 6.040302 «Інформатика»)/Укл.: Іванов В. Г., Ковальов Ю.Г., Лифар В.О. –Сєвєродонецьк: Вид-во СНУ ім. В. Даля, 2017.– 75 с.

2 Вильям Столлингс Компьютерные системы передачи даннях. Шестое издание. Издательский дом «Вильямс», 2002.

3 Иванов А.Б. Волоконная оптика. Компоненты, системы передачи, измерения, Москва, 1999

4 Бакланов И.Г. Методы измерений в системах связи. -М.: Из-во "Эко- Трендз", 1999 г., стр. 88.

5 Семенов А.Б. Волоконная оптика в локальных и корпоративных сетях связи. - М.: Из-во "Компьютер пресс", 1998г.

6 Олифер В.Г., Олифер Н.А. Компьютерные сети: принципы, технологии, протоколы. -С.Пб.: Из-во "Питер", 2001г.

7 Потапов Т.В. Измерение потерь мощности излучения в ВОЛС. Бюллетень "Фотон - экспресс" Август, 2000 г., стр. 13.

8 http://www.o-link.ru/content/view/630/98/

9 ДСТУ 3008 – 95. Документація. звіти у сфері науки і техніки Структура і правила оформлення. Утвержден и введен в действие приказом Госстандарта Украины № 58 от 23 февраля 1995 г.

10 Ю. А. Кравцов, А. Н. Сахаров, «xDSL: диагностика кабельных линий», журнал «Вестник связи», N 8/2002 г.

11 Игорь Иванцов, Цикл статей по рефлектометрии, журнал «LAN» 2005 г.

12 Макфарланд Д.С. Большая книга CSS. – СПБ.: БХВ-Петербург, 2009.

13 Устин В.Б. Учебник дизайна. Композиция, методика, практика. – М.: АСТ: Астрель, 2009. – 254 с.: ил.

14 www.fortress-design.com

15 Вишнякова, С.М. Профессиональное образование: Словарь. Ключевые понятия, термины, актуальная лексика/С.М. Вишнякова. - М.: Речь, 1999. – 197 с.

16 Гитман, Е.К. Проектирование содержания специальных дисциплин/Е.К. Гитман //Специалист. – 1997 - № 12 – С. 29.

17 Данилец, Н. А. Ускоренная профессиональная подготовка/ Н.А. Данилец, Т.Е. Колкова, А.Е. Хрупало ; Акад. проф. образования. Журн "Проф. образование". - М. : Изд. центр АПО, 2002. - 42 с.

18 Жуков, Г.Н Основы общей и профессиональной педагогики : Учебное пособие / Под общ ред проф Г.П. Скамницкой. - М.: Гардарики, 2005. - 382 с.

19 Инновации в системе начального профессионального образования: актуал. пробл. проф. подготовки: IX Обл. науч.-практ. конф. (10-11 дек. 2002г.,: Тез. докл. и сообщ. - Челябинск :, 2003 - 151 с.

20 Концепция модернизации российского образования на период до 2010 г//Профессиональное образование. - №4. – 2006. -19 с.

21 Лисовец, А. В. Методы и алгоритмы мониторинга знаний студентов в учебном процессе профессионального образования: 05.13.10: Автореф. дис. на соиск. учен. степ. канд. техн. наук / А.В. Лисовец. - Барнаул : [б. и.], 2002. - 18 с.

22 Макиенко, Н.И. Педагогический процесс в училищах профессионально-технического образования/Н.И. Макиенко. – Минск, Высш. школа, 1977. – 256 с.

23 Никитина, Н.Н., Железнякова, О.М., Петухов, М.А. Основы профессионально-педагогической деятельности/Н.Н. Никитина и др. – М.: Мастерство, 2002 – 288 с.

24 Оконь. В. Введение в общую дидактику/В. Оконь. – М.: Высш. шк., 1990. -340 с.