МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Модифікований модуль обліку товарів в інтернет-магазині

Виконавець: Мельничук Д.А.

Керівник: Ткаченко В.Г.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)

Освітньо професійна програма «Системне програмування»
Форма навчання заочна

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Литвиненко О. Є.

«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломного проєкту

Мельничук Дар'ї Анатоліївни
(прізвище, ім'я, по батькові)

1.Тема роботи: "Модифікований модуль обліку товарів в інтернет-магазині"
затверджена наказом ректора від "21" грудня 2020 року № 2523 /ст.

2. Термін виконання роботи: з 11.01.2021 до 28.02.2021

3. Вихідні дані до роботи: 1) вимоги до модуля адміністратора сайту;
2) основні операції в управлінні контентом сайту торогового підприємства

4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз існуючих онлайнових систем підтримки роботи сайтів торгових підприємств;
2) структура програмного модуля адміністратора сайту в онлайновій системі;
3) модифікований модуль адміністратора сайту онлайнових замовлень.

5.Перелік обов'язкового графічного матеріалу:
1) діаграма прецедентів для формування замовлення;
2) зв'язки таблиць бази даних;
3) основні вікна модуля адміністратора сайту;
4) схема алгоритму обробки параметрів фільтру замовлень.

6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів
1	Провести аналіз літератури за темою дипломного проєкту та аналіз існуючих систем	11.01.21 12.01.21
2	Зробити вибір компонентів системи	13.01.21- 14.01.21
3	Розробити структуру програмних засобів для реалізації системи торгових операцій онлайн	15.01.21- 16.01.21
4	Розробити програмні засоби для впровадження модифікованих функцій адміністратора онлайн магазину	17.01.21- 28.01.21
5	Провести налаштування програмних засобів на сервері	29.01.21- 31.01.21
6	Написати пояснювальну записку	01.02.21- 12.02.21
7	Підготувати презентацію і захистити роботу	13.02.21- 23.02.21

7. Дата видачі завдання «_11_» січня 2021 р.

Керівник дипломного проєкту_____ Ткаченко В.Г.
(підпис)

Завдання прийняв до виконання_____ Мельничук Д.А.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту “Програмна система обліку роботи станції технічного обслуговування”: 86 с., 26 рис., 13 літературних джерел, 1 додаток.

СИСТЕМА ЗАМОВЛЕНЬ, СТАНЦІЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, БАЗИ ДАНИХ, ФІЛЬТРАЦІЯ ЗАМОВЛЕНЬ.

Об’єкт – облік робіт на станції технічного обслуговування.

Предмет – програмна система обліку роботи станції технічного обслуговування.

Мета дипломного проєкту – розробити програмну систему для адміністратора замовлень станції технічного обслуговування.

В межах системи компанії впроваджено модуль фільтрації з можливістю налаштування за рівнями доступу.

Практична значимість полягає у наданні можливості виконувати швидку фільтрацію в адміністративній частині онлайнової торгової системи, що дозволяє зменшити час на обробку кожного окремого замовлення.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ ОНЛАЙНОВИХ СИСТЕМ ПІДТРИМКИ РОБОТИ САЙТІВ ТОРГОВИХ ПІДПРИЄМСТВ 11
1.1. Огляд існуючих *ERP*-систем 11
1.2. Основні етапи розвитку інформаційних систем 16
1.3. Постановка завдання проєктування 19
1.4. Висновки до розділу 20
РОЗДІЛ 2 СТРУКТУРА ПРОГРАМНОГО МОДУЛЯ АДМІНІСТРАТОРА САЙТУ В ОНЛАЙНОВІЙ СИСТЕМІ 21
2.1. Проєктування логічної моделі програмного модуля фільтрації 21
2.2. Проєктування взаємодії модулів та реляційної бази даних 29
2.3. Висновки до розділу 32
РОЗДІЛ 3 МОДИФІКОВАНИЙ МОДУЛЬ АДМІНІСТРАТОРА САЙТУ ОНЛАЙН ЗАМОВЛЕНЬ 33
3.1. Розгортання бази даних для роботи системи 33
3.2. Опис віконних форм 39
3.3. Тестування обробки запитів до бази даних 61
3.4. Висновки до розділу 64
ВИСНОВКИ 65
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 68
ДОДАТОК А 70
ДОДАТОК Б 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

CRM	–	*Customer relationship management* (управління взаємовідносинами із замовниками)
ERP	–	*Enterprise Resource Planning* (управління ресурсами підприємств)
HR	–	*Human research* (Управління персоналом)
MRP	–	*Material Requirements Planning* (планування потреби в матеріалах)
MRPII	–	*Manufactory Resource Planning* (планування ресурсів підприємства)
PAP	–	*Password Authentication Protocol* (протокол простої перевірки автентифікації)
PS	–	*Project system* (система проєктів)
SCM	–	*Supply Chain Management* (управління ланцюгом поставок)
UML	–	*Unified Modeling Language*
ЕАМ	–	*Enterprise asset management* (управління майном підприємств)
ЄІП	–	єдиний інформаційний простір
КІС	–	корпоративна інформційна система
СІМ	–	*Computer Integrated Manufactoring* (комп'ютеризоване інтегроване виробництво)

ВСТУП

1. Веб-програма (веб-програма)- програма, в якій всі або деякі частини програмного забезпечення завантажуються з Інтернету кожного разу, коли вона запускається. Це може стосуватися додатків на основі браузера, що працюють у веб-браузері користувача, або додатків для робочих столів із «багатим клієнтом», які не використовують браузер, або мобільних додатків, які отримують доступ до Мережі для отримання додаткової інформації.

2. HTML- на основі мови розмітки тексту SGML, призначеної для розмітки документів, що містять текст, зображення, гіперпосилання тощо. Мова, якою написані веб-сторінки. Також відомі як гіпертекстові документи, веб-сторінки повинні відповідати правилам HTML, щоб правильно відображатися у веб-браузері. Синтаксис HTML базується на списку тегів, які описують формат сторінки та те, що відображається на веб-сторінці.

3. Каскадна таблиця стилів (CSS)- Таблиці стилів представляють великий прорив для дизайнерів веб-сторінок, розширюючи їх можливості покращувати зовнішній вигляд своїх сторінок. У науковому середовищі, в якому було створено Інтернет, людей більше турбує зміст їхніх документів, ніж презентація. Коли люди з більш широких верств суспільства відкривали Інтернет, обмеження HTML стали джерелом безперервного розчарування, і автори були змушені обійти стилістичні обмеження HTML.

4. Javascript - динамічна мова програмування. Найчастіше використовується як частина веб-браузерів, реалізації яких дозволяють клієнтським сценаріям взаємодіяти з користувачем, керувати браузером, асинхронно спілкуватися та змінювати відображуваний вміст документа. JavaScript (принаймні суворий підмножини asm.js) також вважається "мовою збірки Інтернету" - ціллю компіляції компіляторів від джерела до джерела - для створення веб-додатків на стороні клієнта з використанням інших мов програмування, що підтримуються основні браузери без плагінів. Він також використовується в серверному мережевому програмуванні із середовищами

виконання, такими як Node.js, розробкою ігор та створенням настільних та мобільних додатків.

5. Середовище розвитку -збір процедур та інструментів для розробки, тестування та налагодження програми чи програми. Зазвичай середовище розробки має три рівні серверів, які називаються розробкою, постановкою та виробництвом. Всі три рівні разом зазвичай називають DSP. Сервер розробки: тут розробник тестує код і перевіряє, чи успішно працює програма з цим кодом. Після того, як додаток буде протестовано, і розробник відчує, що код працює нормально, додаток переходить на проміжний сервер. Проміжний сервер: Це середовище створено так, щоб виглядати точно так, як середовище виробничого сервера. Програма тестується на проміжному сервері, щоб перевірити надійність та переконатися, що вона не виходить з ладу на реальному виробничому сервері. Цей тип тестування на проміжному сервері є завершальним етапом до того, як програму можна буде розгорнути на виробничому сервері. Додаток потрібно схвалити, щоб розгорнути його на робочому сервері.

6. Оптимізація пошукової системи (SEO) -процес впливу на видимість веб-сайту або веб-сторінки в неоплачених результатах пошукової системи - часто їх називають "природними", "органічними" або "зароблениими" результатами. Загалом, чим раніше і частіше сайт з’являється у списку результатів пошуку, тим більше відвідувачів він отримає від користувачів пошукової системи.

7. Розробка фронтенду - практика створення HTML, CSS та Javascript для веб-сайту або веб-програми, щоб користувач міг бачити їх і взаємодіяти з ними безпосередньо. Це поєднання навичок програмування (знання, яку програму вибрати) та естетики (розуміння розташування елементів на екрані, вибору кольору та шрифту). Проблеми, пов’язані з розробниками інтерфейсів, полягають у тому, що інструменти та методи, використовувані для створення інтерфейсу веб-сайту, постійно змінюються, і тому розробник повинен постійно бути в курсі того, як розвивається галузь.

8. Розробка бекенда -Внутрішня розробка в основному розробляє та підтримує основну функціональну логіку та операції програмного забезпечення

чи інформаційної системи. Зазвичай бек-енд розробник має навички програмування на C ++, C #, Java - іншій мові програмування високого рівня. Ключова робоча роль внутрішнього розробника полягає у забезпеченні того, що дані або послуги, що запитуються у внутрішній системі або програмному забезпеченні, доставляються за допомогою програмних засобів. Бек-енд-розробники також створюють і підтримують весь фоновий сервер, який складається з основної логіки додатків, баз даних, інтеграції даних та додатків, API та інших фонових процесів.

Інтернет - приємний засіб для зв'язку з усім світом. Люди використовують його як засіб для зв'язку з іншими людьми, обміну файлами, розвагами, інформацією та безліччю інших корисних та корисних дій у багатьох відношеннях. Під час перегляду Інтернету я знайшов багато веб-сайтів, корисних у багатьох відношеннях. Деякі з них - Google, Facebook, NYTimes, Reditt тощо. Інтернет змінив наше життя. Насправді, люди проводять все більше і більше годин в Інтернеті, чим довше вони користуються ним, а це означає багато з точки зору звикання та залежності.

Багато різних людей виграють від Інтернету. Споживачі отримують вигоду, оскільки вони можуть отримати доступ до інформації від користувачів продуктів, які вони планують придбати. Потім вони можуть прийняти рішення про придбання певного виду товару, виходячи з того, наскільки він працює для інших людей.

Вчителям це вигідно, оскільки вони можуть знайти ресурси для підтримки навчання студентів, а також веб-програми, за допомогою яких учні можуть виконувати цікаву роботу. Ділові люди отримують вигоду, оскільки вони можуть рекламувати свою продукцію у всьому світі за порівняно невеликі гроші. Сім'ї та друзі отримують вигоду, оскільки вони можуть підтримувати зв'язок між собою навіть на великі відстані через соціальні мережі. Шукачі роботи виграють, оскільки вони можуть знайти можливу роботу в Інтернеті та подати заявки через Інтернет, заощаджуючи поштові марки та час. Люди зі специфічними інтересами, такими як порівняння між давньою, середньою та сучасною англійською мовами,

можуть знайти ресурси та інші люди, які поділяють ці ж інтереси. Художники та майстри можуть знайти приклади мистецтв інших людей та інструкції щодо створення різноманітних проектів.

Люди з новими теоріями або наборами ідей можуть розміщувати їх в Інтернеті, не звертаючись до видавця паперу. Люди, які люблять дивитися новини, можуть знаходити звіти в режимі реального часу з багатьох джерел. Люди, яким потрібно знайти давно загублених членів сім'ї або друзів, часто можуть це зробити.

Інтернет значно полегшив розповсюдження інформації у всіх формах, прикладом чого є бібліотеки. Зараз багато людей мають доступ до всіх видів книг суто в Інтернеті. Люди всіх класів доходу мають можливість отримати доступ до будь-якої кількості інформації з бібліотек, якщо у них є бібліотечна картка. Навіть якщо люди самі не мають комп'ютера, більшість міст видають безкоштовні бібліотечні картки кожному, хто може довести місце проживання. Сучасні бібліотеки все ще мають книги, але вони також перетворюються на переважно безкоштовні точки доступу до Інтернету з комп'ютерними лабораторіями. Як результат, Інтернет швидко стає корисним способом для бідніших людей з меншим доступом до освіти покращити себе, навіть якщо шкільна система в їхньому районі погана. Це тим більше стосується людей за кордоном, які взагалі можуть мати невеликий доступ до навчання.

Інформація. Люди переглядають інформацію в Інтернеті. Вони люблять переглядати різні пошукові системи, такі як Google, Yahoo, щоб знати про будь-яку необхідну інформацію. Також люди люблять переглядати веб-сайти, такі як Вікіпедія, яка є повною енциклопедією в Інтернеті. Інтернет - це, мабуть, один із найуспішніших і корисних інструментів, які коли-небудь створювало людство. Насправді це найбільша бібліотека, коли-небудь створена, і щодня зростає. Хоча вам завжди потрібно бути обережними щодо своїх джерел, Інтернет - це сучасне джерело інформації, що постачається у багатьох засобах масової інформації: письмове слово, візуальна графіка та зображення, відео та аудіо змінили спосіб пошуку, пошуку інформації.

Соціальна мережа.Соціальні мережі є важливим засобом спілкування з друзями та членами сім'ї. Існує безліч веб-сайтів соціальних мереж, таких як Facebook, Orkut та Bebo, які користувачі споживають для спілкування з друзями. Люди використовують Інтернет для пошуку, підтримки або припинення стосунків. Але люди також можуть потрапити в залежність від соціальних мереж.

Спілкування.Спілкування - ще один спосіб використання Інтернету. Люди підключаються один до одного за допомогою різних служб обміну миттєвими повідомленнями, таких як Hangouts, Skype та Yahoo messenger. Є безліч інших служб, за допомогою яких люди надсилають повідомлення. Люди використовують Інтернет для спілкування один з одним. Програмне забезпечення дозволило передавати голос і відео по всьому світу з мінімальними затримками, а електронна пошта стала основним засобом спілкування для багатьох сучасних людей. Без Інтернету підтримувати особисті та професійні стосунки було б і дорожче, і повільніше.

Передача файлів.Від офісу до школи, від бізнесмена до студентів коледжу всі надсилають файли через Інтернет. Це важлива частина їхнього життя. Ці файли надсилаються через Інтернет. Люди використовують різні поштові служби, такі як Gmail, Yahoomail, AOL, Hotmail тощо для надсилання файлів. Сьогодні ми звикли до того, що ми можемо знайти будь-яку програму чи фільм, розміщений в Інтернеті. Існує безліч спеціально створених веб-сайтів, які дозволяють завантажувати безкоштовно або за певну плату. Інтернет дозволяв обмінюватися файлами з різним вмістом, тому зараз нам не потрібно їхати до Франції на національний кінофестиваль, щоб їх побачити. Ми можемо завантажити їх лише одним клацанням миші та протягом декількох хвилин (або протягом кількох годин) залежно від вашого з’єднання.

Розваги.Інтернет дуже тісно пов’язаний із розвагами. Це перегляд відео на YouTube, гра в живі ігри або завантаження фільмів; Інтернет доводить своє панування скрізь. Багато людей користуються Інтернетом, щоб насолоджуватися собою та займатися особистими інтересами. В останні роки багатокористувацькі ігри та віртуальні світи залучили час і гроші багатьох. Крім того, відео та музику

легко знайти, транслювати та завантажувати… плюс, засіб заохочує зворотний зв’язок! Дійсно, використання Інтернету може зайти занадто далеко. Але як ви можете знати, чи залежні ви від Інтернету? Ми перелічимо критерії та ознаки Інтернет-залежності за посиланням вище.

Інтернет-транзакції.Тепер Інтернет може заощадити час і гроші людей. Заклад відомий як Інтернет-банкінг, за допомогою якого люди можуть внести будь-який рахунок, переказати гроші через рахунки та вчасно зробити бронювання в Інтернеті з дому.

Самовираження.Люди не лише споживають інформацію в Інтернеті, вони її СТВОРЮЮТЬ. І цим люди можуть виражати себе політично, художньо, вокально, соціально і т. Д. І давати голос тому, що для них важливо. Інтернет - це найкращий форум, на якому ви можете обговорювати або монологізувати як завгодно.

Робити гроші.Люди можуть заробляти гроші в Інтернеті, використовуючи Інтернет. Люди можуть заробляти гроші в Інтернеті різними фактичними способами. Доступно безліч варіантів. Проектування, фріланс, консультації, постачальники програмного забезпечення, додатки завжди допомагають людям працювати простіше. Інтернет забезпечує альтернативу 9–5 робочим дням, оскільки все більше людей може працювати вдома або “їздити на роботу”. Плюс до цього, все більша кількість людей заробляє на життя самим Інтернетом, стаючи експертами в тому, як люди шукають в Інтернеті, надаючи IT-послуги або послуги з веб-розробки, або спеціалізуючись на Інтернет-маркетингу.

Маркетинг.Інтернет-маркетологи використовують Інтернет для продажу товарів. Є багато ентузіастів соціальних мереж, які просувають чужі товари в Інтернеті через різні сайти соціальних мереж. Доступно багато видавців, які рекламують інші продукти, рекламуючи на різних веб-сайтах, що ведуть блоги тощо. Люди використовують Інтернет для дослідження, пошуку та придбання послуг та продуктів. Або націлити та продати кінцевому споживачеві. По суті, Інтернет став НАЙКРАЩИМ способом купівлі та продажу товарів, оскільки Інтернет-магазини працюють цілодобово, 7 днів на тиждень.

Інтернет-освіта.Інтернет-освіта є дуже відомим засобом навчання у розвинених країнах та зростає в країнах, що розвиваються. Різні веб-сайти, такі як Академія Хана та FreeVideoLectures, пропонують онлайн-курси для вивчення різних речей, таких як проектування, програмування, інженерія, медицина, фінанси та інші предмети. Це дуже корисний засіб для просування освіти в тому місці, де курси недоступні. Все більше і більше початкових, середніх шкільних та університетських програм вимагає використання Інтернету для роботи в школі.

Незважаючи на те, що Інтернет був створений для полегшення спілкування в армії, він швидко став інструментом масового спілкування після випуску персонального комп'ютера. З моменту свого створення Інтернет з езотеричного походження перетворився на стандартну форму спілкування. За даними Miniwatts Marketing Group, понад два мільярди людей, або 34,3 відсотка населення світу, підключені до Інтернету в будь-який момент часу.

Люди використовують Інтернет для швидкого доступу до інформації та спілкування з іншими за допомогою соціальних мереж, таких як Facebook, Twitter та Skype. Інтернет також служить джерелом розваги для людей завдяки відносній легкості завантаження музики та потокового відео в Інтернеті. Підключення до Інтернету також дає людям можливість обмінюватися файлами та дізнаватися про місцеві події. Незважаючи на те, що вміст в Інтернеті доступний кожному, хто має зв’язок, деякі країни встановлюють обмеження на веб-пошук, щоб обмежити обсяг інформації, до якої можуть отримати доступ громадяни.

Інтернет зростає з величезною швидкістю. Це стало однією з важливих частин життя. Життя без Інтернету неможливо уявити сучасним людям. Всі ці сервіси стали можливими за допомогою веб-додатків, які сформували їх основи.

Метою цього проекту є розгляд сучасних методів розробки веб-додатків високого рівня, оцінка технологій та інструментів, які ці методи використовують для визначення найбільш ефективних з них.

Для досягнення зазначеної вище мети проекту було визначено головне завдання:

- визначити методи розробки та їх деталі;
- порівняти ці методи, надаючи статистику використання та актуальності, а також враховуючи переваги та недоліки кожного з них;
- розглянути вимоги до сучасних веб-додатків та характеристики SEO;
- розглянути сучасні середовища розробки та вимоги до обладнання та програмного забезпечення робочого місця розробника;
- визначити найефективніші з сучасних методів розробки веб-додатків високого рівня.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ОНЛАЙНОВИХ СИСТЕМ ПІДТРИМКИ РОБОТИ САЙТІВ ТОРГОВИХ ПІДПРИЄМСТВ

Коли ви новачок у веб-розробці або коли довго не займаєтесь веб-розробкою, ви, ймовірно, загубитесь у цих джунглях веб-технологій. Не хвилюйтеся, якщо вас це пригнічує, адже це навмисно. Ця хмара тегів містить багато мов та веб-фреймворків, з яких складаються сучасні веб-програми. Але він також містить ряд засобів розвитку, які використовуються під час розробки. Мета цієї публікації - провести вас через ці джунглі, пояснивши вам, які технології ви можете використовувати для створення сучасного веб-додатку та які інструменти вам потрібні протягом повного життєвого циклу розробки.

Щоб краще зрозуміти, які технології та інструменти нам слід використовувати, спершу слід трохи розглянути еволюцію ряду важливих технологій. Тоді ми зрозуміємо, чому веб-розробка сьогодні є досить складною і чому нам потрібні інструменти. Як тільки це стане зрозумілим, ми можемо поглянути на наявні інструменти.

1.1.Проблеми та недоліки методологій, інструментів та методів проектування веб-додатків

1.1.1. Накладні витрати методології проти параметрів розвитку

Формальні методології, як правило, важкі в документації та трудомісткі. Розробники повинні оцінити, чи можливо з точки зору часу та бюджету застосовувати певну методологію. Для малих та середніх проектів, які не є критично важливими, вартість помилок, як правило, низька. У цих сценаріях, коли дотримання графіку для маркетингових цілей важливіше 100% без помилок, методологічні процедури, такі як перевірка функціональної специфікації, документації та контролю якості, зазвичай жертвують.

Коли є конкуруючі цілі, методологія має низький пріоритет. Більше того, через відсутність розуміння щодо складних методологій розробники не можуть належним чином застосовувати їх для отримання бажаного результату. Коли один крок методології застосовується неправильно, ефект доміно може призвести до непослідовності та помилкових результатів.

1.1.2. Якість розробника проти методології

Відгук одного з респондентів: "Незалежно від того, яку методологію ви використовуєте, вам все одно потрібні добрі люди, хороший зв’язок, стислі вимоги, розумні терміни, відповідне фінансування, загальногрупова дисципліна та багато удачі". Щоб усвідомити переваги, що надаються методологією, нам потрібні розробники, які розуміють, як це працює, можуть повідомляти про це всій команді та володіти дисципліною, щоб підтримувати послідовність та поверхневі проблеми. У той час як компетентні розробники можуть досягти успіху без офіційної методології, методологія не може світити в руках посередніх розробників. Однак відповідна методологія може полегшити та підтримати планування, встановлення цілей, придбання ресурсів та ведення переговорів.

1.1.3. Зверху вниз проти знизу вгору

Деякі організації не застосовують формальну методологію, оскільки топ-менеджмент їх не вимагає. Коли топ-менеджмент не встановлює стандартів, керівникам проектів важко застосовувати методологію різних розробників у проекті. У середовищі, яке не є стандартним, розробники, як правило, використовують те, що вони знають, для виконання покладених на них завдань за найкоротший проміжок часу, без особливої уваги щодо функціональності високого рівня та довгострокової ремонтопридатності. На додаток до стандартного забезпечення, топ-менеджмент повинен виділити фінансування, щоб забезпечити навчання для розробників, щоб усі могли знаходитись на одній сторінці для спілкування та вирішення проблем.

1.1.4. Швидкість, точність та ефективність методологій

Скарга номер один від розробників - це неоднозначне твердження вимог та постійні зміни вимог з боку кінцевих користувачів. Існуюча методологія не забезпечує інтерактивного інструменту, який може визначити функціональність у

зрозумілому форматі для перегляду, перегляду та затвердження кінцевими користувачами. Кінцеві користувачі вимагають функціональності швидше, ніж будь-яка методологія може ефективно документувати, перевіряти та прототипувати. Повзання обсягу стало основним фактором відмови для розвитку системи через відсутність методології її управління.

Вкрай важливо, щоб топ-менеджмент мав політику блокування специфікацій вимог після розумного періоду ітеративного проектування та перевірки. Аналіз впливу на зміни вимог необхідний для оцінки того, чи слід приймати зміни вимог від кінцевих користувачів. Зрештою, це компроміс між швидкістю та якістю. Вище керівництво повинно керувати пріоритетами та вирішувати боротьбу за владу серед ключових гравців проекту. Щоб методологія була реально застосовною та корисною, на додаток до технологічної підтримки, методологія повинна передбачати процедуру та методи, які реально спрощують природжену складність розробки системи, забезпечують ефективне спілкування між членами команди, вирішують конфлікти між вимогами кінцевого користувача та обмеженнями розробника, і вирівняти всю діяльність до тих самих цілей організації.

1.1.5. Нові методології веб-додатків

Веб-програми піддаються змінам вимог частіше, ніж інші системи. Цикл розробки веб-додатків також коротший і більш мінливий, ніж інші. Чи потрібна методологія розробки веб-додатків? Судячи з коментарів респондентів з цього опитування, відповідь однозначно так. Методологія потрібна для ремонтопридатності, масштабованості та модернізації веб-додатків, особливо для великомасштабних та критично важливих проектів. Що стосується цих проектів, то, здається, розробники повертаються до життєвого циклу розвитку водоспаду за його надійність та послідовність, доповнену спеціальними швидкими прототипами, як того вимагають ситуації, щоб надати оглядовий документ для кінцевих користувачів. Як запропонував один з респондентів,

1.1.6. Нові етапи проектування та інструменти для веб-додатків

Враховуючи, що веб-програми часто використовують сторонні веб-служби та компоненти, покладаються на офшорні підряди та складаються з проектів, що складаються з декількох компаній, існує потреба у етапах розробки та інструментах для перевірки багаторівневої сумісності та інтеграції, а також для забезпечення безпека між усіма частинами системи на всіх можливих платформах усіма можливими користувачами. Експертна оцінка та оцінка кінцевих споживачів можуть сприяти якнайшвидшому виникненню проблем. Відгуки користувачів веб-додатків постійні та швидкі завдяки характеристикам роботи Мережі. Етап обслуговування веб-додатків є постійним та стиснутим, що вимагає інструментів для швидкої діагностики та доставки для вирішення проблем.

1.1.7. Необхідні знання та навички для успішного проектування веб-додатків

Одним з вражаючих коментарів щодо необхідних знань та навичок для успішної розробки веб-додатків респондентів є м'які навички розробників, включаючи слухання, етику, критичне / аналітичне / логічне мислення, усне / письмове спілкування, міжособистісний / дипломатичний маневр, лідерство, бажання вчитися та розуміти, управління часом, розуміння та адаптація до різних зацікавлених сторін, а також здатність узгоджуватися та розвиватися з організаційними мандатами та цілями. З точки зору технічних навичок та знань, розробники повинні знати інтерфейс, інтерфейс, базу даних та концептуальність архітектури. Експертний досвід включає програмування (захисне, екстремальне), сценарії, прототипування, дизайн макета інтерфейсу користувача, методології, схематичні інструменти та фреймворк. Досвід роботи включає програмування на стороні сервера,

Оскільки веб-сайти сьогодні керуються базами даних, знання бази даних та їх зв’язок, маніпулювання та обслуговування є критично важливими. Архітектура та інфраструктурні знання для мультиплатформенних, мультисистемних та мультиорганізаційних систем, що складаються із сторонніх компонентів та веб-служб, також мають важливе значення. Оскільки методології,

інструменти та техніки змінюються та змінюються, розробникам важливіше зрозуміти їх функції та цілі, аніж вивчати їх усі.

Оскільки шляхів досягнення мети може бути декілька, розуміння цілі може надати розробникам гнучкість для оцінки нових методологій та інструментів та їх цінностей. Тоді як технічна експертиза може зробити роботу, м'які та навички людей роблять її успішною. Довіра, довіра та придбання від кінцевих користувачів, особливо потужних кінцевих користувачів, є критично важливими для успішної розробки та впровадження веб-додатків. Це дуже гарне нагадування про те, що слід включити до навчальних програм для навчання веб-розробників. Зазвичай це не методика чи інструмент, а люди, які зазнають невдачі.

1.2 Вимоги до сучасного процесу проектування веб-додатків

Односторінкові програми відрізняються своєю здатністю перемальовувати будь-яку частину інтерфейсу користувача, не вимагаючи серверного зворотного зв'язку для отримання HTML. Це досягається відокремленням даних від подання даних за допомогою рівня моделі, який обробляє дані, і рівня подання, який зчитує з моделей. Більшість проектів починаються з високих амбіцій та недосконалого розуміння суті проблеми. Наші реалізації, як правило, випереджають наше розуміння. Можна написати код, не розуміючи проблеми повністю; цей код просто складніший, ніж повинен бути, через нашу нерозуміння.

Хороший код походить від вирішення однієї і тієї ж проблеми кілька разів або рефакторингу. Зазвичай це відбувається шляхом помічення повторюваних шаблонів і заміни їх на механізм, який послідовно робить одне і те ж - замінюючи велику кількість коду, що стосується конкретного випадку, який насправді був саме там, оскільки ми не бачили, що простіший механізм міг досягти того самого. Архітектури, що використовуються в односторінкових додатках, представляють результат цього процесу: там, де ви робите щось спеціально, використовуючи jQuery, ви тепер пишете код, який використовує переваги стандартних механізмів (наприклад, для оновлення інтерфейсу тощо). Програмісти одержимі легкістю, а не простотою (спасибі Річу Хікі за те, що це сказав); або, що таке досвід програмування замість того, як виглядає отримана

програма. Це призводить до марних розмов про крапку з комою та те, чи потрібен нам препроцесор, який усуває фігурні дужки. Ми все ще говоримо про програмування так, ніби введення коду було найскладнішою частиною. Це не так - важка частина - це підтримка коду.

Для написання коду, який можна підтримувати, нам потрібно зробити все простішим. Це постійна боротьба; легко додати складності (взаємопов'язаність / залежності) для вирішення нікчемної проблеми; і вирішити проблему легко, не зменшуючи складності. Прикладом останнього є простори імен.

Маючи це на увазі, давайте розглянемо, як сучасна веб-програма структурована з трьох різних точок зору:

- **Архітектура**: з яких (концептуальних) частин складається наш додаток? Як різні частини спілкуються між собою? Як вони залежать один від одного?
- **Упаковка активів**: як наша програма структурована на файли, а файли на логічні модулі? Як ці модулі будуються та завантажуються у браузер? Як можна завантажити модулі для модульного тестування?
- **Стан виконання**: які частини програми при завантаженні в браузер знаходяться в пам'яті? Як ми виконуємо переходи між станами та отримуємо видимість до поточного стану для усунення несправностей?

1.2.1. Сучасна архітектура веб-додатків

DOM лише для запису.З DOM не зчитується стан / дані. Додаток виводить HTML та операції з елементами, але з DOM ніколи нічого не читається. Зберіганням стану в DOM стає важко керувати дуже швидко: набагато краще мати одне місце, де живуть дані, і відображати інтерфейс користувача з даних, особливо коли одні й ті самі дані повинні відображатися в декількох місцях в інтерфейсі.

Моделі як єдине джерело істини. Замість того, щоб зберігати дані в DOM або в випадкових об'єктах, існує набір моделей в пам'яті, які представляють весь стан / дані в програмі.

Сучасні односторінкові програми загалом структуровані, як показано на малюнку 1:

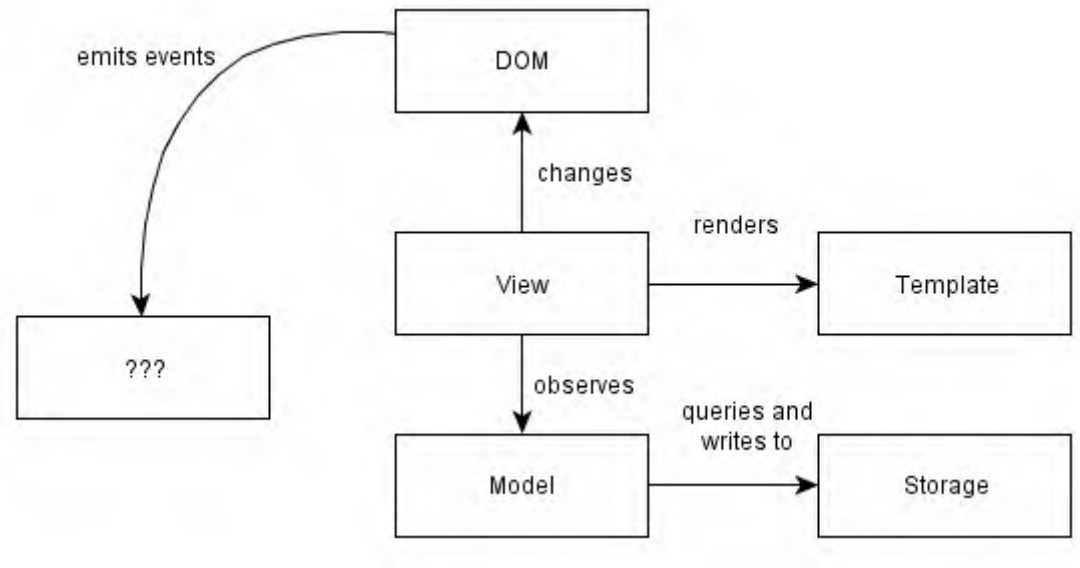

Рис. 1. Структура сучасного односторінкового веб-додатку

Моделі як єдине джерело істини. Замість того, щоб зберігати дані в DOM або в випадкових об'єктах, існує набір моделей в пам'яті, які представляють весь стан / дані в програмі.

Погляди спостерігають зміни моделі.Ми хочемо, щоб погляди відображали зміст моделей. Коли кілька подань залежать від однієї моделі (наприклад, коли модель змінюється, перемальовуйте ці подання), ми не хочемо відстежувати вручну кожен залежний вигляд. Замість того, щоб відстежувати речі вручну, існує система подій змін, за допомогою якої подання отримують сповіщення про зміни від моделей і самі обробляють перемальовування.

Незв'язані модулі, що оголюють невеликі зовнішні поверхні. Замість того, щоб робити речі глобальними, нам слід намагатися створити невеликі підсистеми, які не взаємозалежні. Залежності ускладнюють налаштування коду для тестування. Невеликі зовнішні поверхні полегшують рефакторинг внутрішніх елементів, оскільки більшість речей можуть змінюватися, якщо зовнішній інтерфейс залишається незмінним.

Мінімізація DOM-залежного коду.Будь-який код, який залежить від DOM, повинен бути перевірений на сумісність між браузерами. Написавши код таким чином, щоб ізолювати ці неприємні частини, потрібно перевірити набагато обмежену поверхню на сумісність між браузерами. Несумісність між браузерами таким чином набагато більш керована. Несумісність полягає у реалізаціях DOM,

а не в реалізаціях Javascript, тому має сенс мінімізувати та ізолювати DOM-залежний код.

1.2.2. Упаковка активів

Упаковка активів - це місце, де ви берете код програми JS і створюєте один або кілька файлів (пакетів), які браузер може завантажити за допомогою тегів сценаріїв. Здається, ніхто не підкреслює, наскільки важливим є це право! Упаковка активів не полягає в прискоренні часу завантаження - це в тому, щоб зробити вашу програму модульною та переконатися, що вона не перетворюється на неперевірений безлад. Проте люди думають, що мова йде про продуктивність, а отже, необов'язкова.

Якщо є одна частина, яка впливає на те, наскільки перевіряється та наскільки реконструйований ваш код, це наскільки добре ви розділяєте свій код на модулі та застосовуєте модульну структуру. І ось у чому полягає "упаковка активів": розподіл речей на модулі та переконання, що стан виконання не перетворюється на безлад. За замовчуванням ("кинути кожен файл JS у глобальний простір імен і сподіватися, що результат спрацює") жахливий, тому що це робить модульне тестування - і, за розширенням, рефакторинг - важким. Це пояснюється тим, що погана модуляризація призводить до залежності від глобального стану та глобальних назв, що ускладнює налаштування тестів.

Крім того, неявні залежності дуже ускладнюють знання, які модулі залежать від коду, який ви рефакторингуєте; ви в основному покладаєтесь на інших людей, які послідовно дотримуються належної практики (не залежать від речей, які я розглядаю як внутрішні деталі). Явні залежності забезпечують загальнодоступний інтерфейс, а це означає, що рефакторинг стає набагато меншим болем, оскільки інші можуть залежати лише від того, що ви виставляєте. Це також заохочує більше думати про публічний інтерфейс.

Подробиці того, як це зробити, містяться в розділах про ремонтопридатність та модульність. Крім того, неявні залежності дуже ускладнюють знання, які модулі залежать від коду, який ви рефакторингуєте; ви в основному покладаєтесь на інших людей, які послідовно дотримуються

належної практики (не залежать від речей, які я розглядаю як внутрішні деталі). Явні залежності забезпечують загальнодоступний інтерфейс, а це означає, що рефакторинг стає набагато меншим болем, оскільки інші можуть залежати лише від того, що ви виставляєте. Це також заохочує більше думати про публічний інтерфейс. Подробиці того, як це зробити, містяться в розділах про ремонтопридатність та модульність. Порівняння підходів показано нижче в таблиці 1:

Таблиця 1.

Порівняння модульних та немодульних активів розробки коду

Брудний і випадковий (немодульний)	Пакети та модулі (модульні)
• Кожен фрагмент коду за замовчуванням робиться глобальним • Імена загальносвітові • Повністю прохідні простори імен • Порядок завантаження має значення, оскільки будь-що може перезаписати або змінити що-небудь інше • Неявна залежність від чогось глобального • Файли та модулі не мають жодного значущого зв’язку • Запускається лише у браузері, оскільки залежності не ізольовані	• Пакети мають єдиний загальнодоступний інтерфейс • Імена місцеві для пакунка • Деталі реалізації недоступні поза пакетом • Порядок завантаження не має значення завдяки упаковці • Явно оголошені залежності • Кожен файл містить один модуль • Запуск з командного рядка без браузера без голови

Крім того, неявні залежності дуже ускладнюють знання, які модулі залежать від коду, який ви рефакторингуєте; ви в основному покладаєтесь на інших людей, які послідовно дотримуються належної практики (не залежать від речей, які я розглядаю як внутрішні деталі). Явні залежності забезпечують загальнодоступний інтерфейс, а це означає, що рефакторинг стає набагато меншим болем, оскільки інші можуть залежати лише від того, що ви виставляєте. Це також заохочує більше думати про публічний інтерфейс. Подробиці того, як це зробити, містяться в розділах про ремонтопридатність та модульність.

1.2.3. Стан виконання

Третій спосіб розглянути сучасний односторінковий додаток - поглянути на стан його роботи. Стан часу роботи стосується того, як виглядає програма, коли вона запущена у вашому браузері - такі речі, як, які змінні містять, яку інформацію та які кроки беруть участь у переході від однієї діяльності (наприклад, сторінки) до іншої.

Тут є три цікаві стосунки:

URL <-> станОдносторінкові програми мають шизофренічне відношення до URL-адрес. З одного боку, існують односторінкові програми, завдяки яким користувачі можуть багатше взаємодіяти з додатком. Багатіші дії означають, що існує більше стану перегляду, ніж може обґрунтовано вміститися всередині URL-адреси. З іншого боку, ми також хотіли б мати можливість зробити URL-адресу закладкою та повернутися до тієї ж активності. Для підтримки закладок нам, мабуть, потрібно дещо зменшити рівень деталізації, який ми підтримуємо в URL-адресах. Якщо кожна сторінка має одну основну діяльність (яка з певним рівнем деталізації представлена в URL-адресі), то кожна сторінка може бути відновлена з закладки в достатній мірі. Вторинні дії (як, скажімо, чат у програмі веб-пошти) при перезавантаженні повертаються до стану за замовчуванням, оскільки зберігати їх у URL-адресі, до якої можна зробити закладку, безглуздо.

Визначення <-> ініціалізаціяДеякі люди все ще змішують це двоє разом, що погана ідея. Компоненти багаторазового використання слід визначати без фактичного створення / активації, оскільки це дозволяє повторно використовувати та тестувати. Але як тільки ми це робимо, як насправді ми

виконуємо ініціалізацію / інстанціювання різних станів програми? Я думаю, що є три загальних підходи: один - мати невелику функцію для кожного модуля, яка приймає певні вхідні дані (наприклад, ідентифікатори) та створює екземпляри відповідних подань та об’єктів. Інший - мати глобальний файл завантажувального файлу, за яким слід маршрутизатор, який завантажує правильний стан з-поміж глобальних станів. Останній - обернути все цукром, що робить порядок створення випадків невидимим.

Мені подобається перший; другий здебільшого спостерігається в додатках, які органічно виросли до того моменту, коли речі починають заплутуватися; третій видно в деяких рамках, особливо щодо шару перегляду. Причиною, що мені подобається перша, є те, що я вважаю стан (наприклад, екземпляри об’єктів та змінних) огидним і вартим виділення в одному файлі (для кожної підсистеми - стан повинен бути локальним, а не глобальним, але про це пізніше). Чисті дані прості, так само як і визначення. Це коли ми маємо багато взаємозалежних та / або важко помітних станів, що все ускладнюється; важко міркувати і взагалі неприємний.

Інша перевага першого підходу полягає в тому, що він не вимагає завантаження повного додатка при кожному перезавантаженні сторінки. Оскільки кожну діяльність можна ініціалізувати самостійно, ви можете протестувати окрему частину програми, не завантажуючи повну програму. Подібним чином, ви маєте більшу гнучкість у попередньому завантаженні решти програми після активного початкового перегляду (порівняно з початком); це також означає, що початковий час завантаження не збільшиться пропорційно кількості модулів, які має ваш додаток.

Елементи HTML <-> переглядати об'єкти та події HTML <-> перегляд змін

Нарешті, виникає питання про те, скільки видимості ми можемо отримати у стані часу роботи фреймворку, який ми використовуємо. Я не бачив, щоб фреймворки чітко зверталися до цього (хоча, звичайно, є хитрощі): коли я запускаю свою програму, як я можу визначити, що відбувається, вибравши

певний елемент HTML? І коли я переглядаю певний елемент HTML, як я можу сказати, що станеться, коли я клацну на ньому або виконаю якусь іншу дію?

Простіші реалізації, як правило, проходять краще, оскільки відстань від елемента HTML / події до вашого об'єкта перегляду об'єкта / обробки подій значно менша. Я сподіваюся, що фреймворки будуть приділяти більше уваги виведенню цієї інформації.

1.3. Сучасні вимоги до SEO

Двадцять років тому найпоширенішим способом пошуку бізнесу було перегортання Жовтих Сторінок та здійснення численних телефонних дзвінків різним постачальникам. Зараз, коли Інтернет та пошукові системи широко використовуються та доступні, набагато простіше - і швидше - знайти саме те, що ви шукаєте. Коли з'явилися перші пошукові системи, такі як Yahoo! та AltaVista були створені, вони розділили бізнес категорично. Це означає, що ви зайшли б, потрапили в ту категорію, яка, на вашу думку, найбільше підходить для вашої компанії, і людина, яка шукала вас, отримала б уточнений список.

Коли Google вийшов, вони розробили алгоритмічну систему, яка в основному передбачала: "Нам байдуже, яка категорія вас цікавить, просто введіть, що хочете, і все інше зробить наша технологія". З огляду на це, загальними характеристиками, які Google або пошукові системи шукають на веб-сайті, є те, що ми називаємо основами оптимізації пошукових систем або основами SEO. Протягом мого п'ятирічного досвіду роботи з основ SEO було помічено багато значних покращень, але загалом основи SEO, як правило, дуже узгоджені.

Перше, на що звертає увагу Google, - це те, як оформлено ваш сайт і чи все там, де повинно бути. Ви повинні переконатися, що у вас є правильний заголовок веб-сторінки, мета-теги та мета-описи. Google досить добре розповідає, що потрібно виправити, і якщо щось не там, де повинно бути. Ось чому важливо, щоб хтось зосередився на основах SEO вашої веб-сторінки. Правильне розміщення вашої сторінки полегшує Google пошуку вас.

За останні кілька років фокус на вмісті зріс в геометричній прогресії, але все одно його слід розміщувати після посилань. Посилання все ще мають

сильніше зважене значення всередині вмісту алгоритму, але Google фокусується на вмісті вашої веб-сторінки, щоб переконатись, що інформація оновлюється. Якщо ви не зміните свій вміст або якщо ваш вміст дублюється по всьому Інтернету, ваш веб-сайт втрачає довіру і стає важко знайти.

Рушійною силою масового оновлення системи Google є Google Caffeine. Програма подрібнює величезні обсяги даних та вмісту, щоб визначити, хто що оновлює та як часто вони складаються. Google кофеїн є позитивним підкріпленням, коли ви намагаєтеся збільшити свою експозицію, але гарантує, що ви оновлюєте свій сайт.

Рейтинг 30 пошукових термінів, які глибоко містять вміст вашої домашньої сторінки, дуже складний. Замість того, щоб прагнути до цього, ви хочете класифікувати приблизно п’ять пошукових термінів на домашній сторінці та решту на своїх підсторінках. Компанія Google чітко дала зрозуміти, що вони не хочуть, щоб хтось натискав тричі, щоб отримати інформацію, яку вони шукають. Через це вони зазвичай не спрямовують когось на домашню сторінку. Підсторінка, що має кращий вміст, теги та основи, частіше отримує звернення від Google.

Бути номером один - абсолютна мета будь-якого конкурентного пошукового терміна, але не завжди найкращий варіант. Ви повинні визначити, чи вам краще поставити рейтинг на першій сторінці за загальним пошуковим терміном, чи ви хочете класифікувати на другій сторінці за довшими ключовими словами, які легше перетворити. Це справді те, що розділяє людей, які просто шукають оптимізацію пошукової системи, і людей, які шукають збільшення експозиції. Альтернативою виклику рейтингу в топ-10 Google для конкуруючих ключових слів із широкою відповідністю є розуміння вашого трафіку, поведінки користувачів та тестування менш конкурентоспроможних ключових слів. Коли ми часто працюємо з нашими клієнтами, дані свідчать про те, що вигідніше покращувати рейтинг ключових слів із довшими хвостами та відповідних підсторінок. Допоможіть відвідувачеві швидше знайти бажане. Найкращий спосіб зробити це - допомогти Google дізнатися, де саме знаходиться ваш веб-сайт. Нарешті, переконайтеся, що взаємодія з користувачем є оптимальною, і

ваші коефіцієнти конверсії виправдають маркетингові витрати. Основні вимоги до SEO наведені на малюнку нижче на малюнку 2:

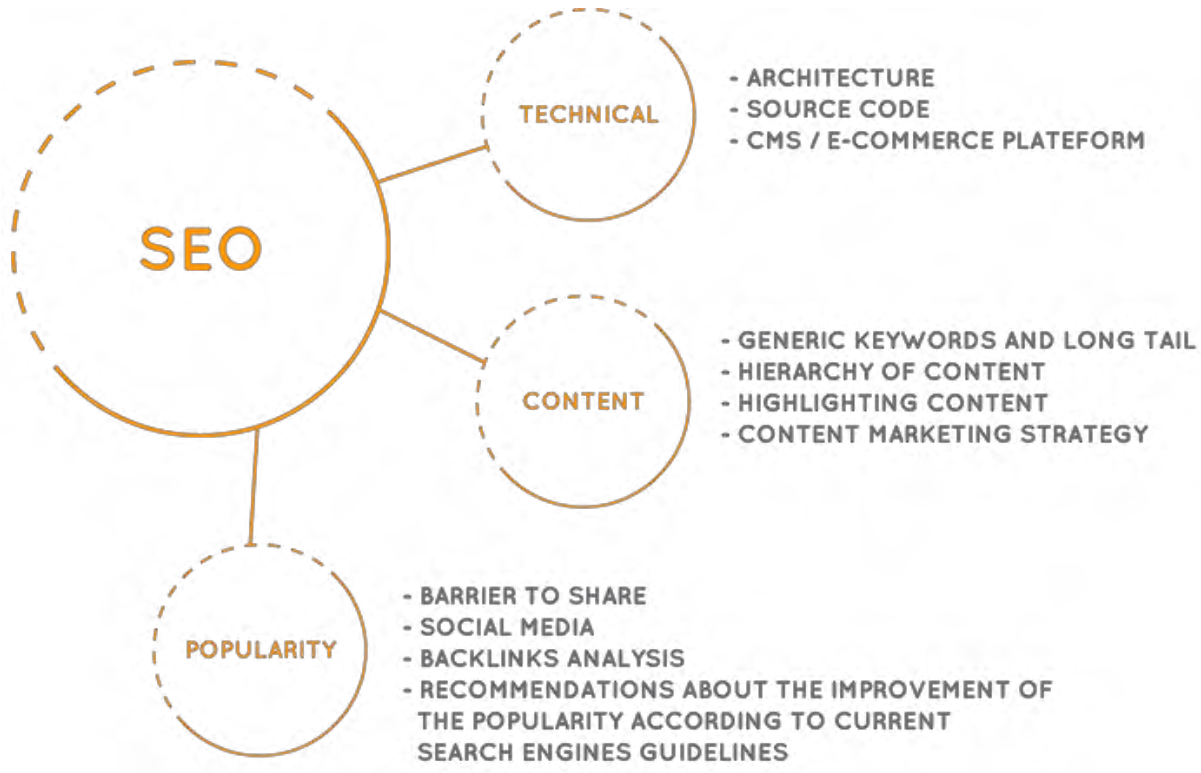

Рис. 2. Основні вимоги до SEO

1.3.1. Вимоги до платформи для SEO

Платформа диктує, як можна створити ваш сайт, що, в свою чергу, має значний вплив на продуктивність SEO нестандартно. Вибір платформи відразу обмежує параметри SEO, з якими вам доведеться працювати, набором, який підтримує платформа.

Завдяки необмеженому бюджету будь-яку платформу можна змінити, щоб вона працювала ефективніше для SEO, але доповнення та користувацький код є дорогими і мають власний ризик. Модифікації післяпродажного обслуговування можуть негативно вплинути на інші напрямки роботи сайту, і їх необхідно підтримувати та відновлювати в міру розвитку платформи. Наступні вимоги допоможуть вам вибрати найсильнішу платформу для вашого сайту електронної комерції.

- Кожна унікальна сторінка вмісту має одну унікальну URL-адресу. Сторінка визначається як унікальний показ вмісту, який відвідувач визнав би

таким. Наприклад, сторінка, що містить категорію товарів, відрізняється від сторінки, що містить менший сегмент цих товарів;

- URL-адреса кожної унікальної сторінки однакова при кожному відвідуванні цієї сторінки, незалежно від шляху відвідувача до сторінки;
- Фільтри категорій повинні розглядатися як унікальні сторінки з унікальними URL-адресами та доступною навігацією - за замовчуванням;
- Кожна унікальна сторінка містить канонічний тег, що ідентифікує одну канонічну URL-адресу для цієї сторінки, щоб нейтралізувати можливі дублікати URL-адрес.

1.3.2. Міжнародні вимоги до SEO

На додаток до перерахованих вимог до платформи, міжнародні компанії електронної комерції мають додаткові технічні вимоги.

- Вибір країни та мови повинен бути відображений у звичайному тексті HTML та посилання для користувачів із вимкненими JavaScript, CSS та файлами cookie;
- Якщо планується геолокація, (a) вибір селектора країни повинен «замінити» будь-яку доставку геолокації на основі доставки IP, (b) «заміщення» має базуватися на URL-адресі, а не на основі файлів cookie, (c) геолокація лише на першій сторінці входу , не кожен клік після цього, (d) здійснювати геолокацію лише на .com та інших загальних TLD, а не на ccTLD;
- На кожній сторінці для кожної країни та мови використовуйте теги hreflang та метадані мови, щоб вказати країну та позначення мови.

1.3.3. Вимоги до управління вмістом для SEO

Вимоги до вмісту починаються з системи управління вмістом. Якщо система управління вмістом не дозволяє редагувати елемент на сторінці, для оптимізації потрібно втручання розробника.

- Кожна окрема сторінка повинна бути унікально редагованою та оптимізованою в системі управління вмістом;
- Дочірні сторінки не повинні успадковувати оптимізацію батьківської сторінки;

- Увімкніть оптимізоване відображення назви сторінки проти назви сторінки за замовчуванням, щоб покращити SEO для категорій та продуктів, які не мають розпізнаваних імен за замовчуванням у вашій внутрішній базі даних;
- Кожен елемент, який впливає на продуктивність SEO, повинен редагуватися в системі управління вмістом, не вимагаючи втручання розробника, включаючи (a) ім'я сторінки відображення, (b) тег заголовка, (c) метаопис, (d) заголовок H1, (e) основну копію та опис ;
- Переконайтеся, що жорстко закодовані посилання в текстовій копії та описах автоматично оновлюються, якщо змінюється URL-адреса сторінки, на яку посилається;
- Увімкнути масове завантаження вмісту з файлу CSV.

1.3.4. Вимоги до впровадження для SEO

Після вибору платформи впровадження сайту на цій платформі є критичним для успіху SEO. Дизайн та архітектура важливі, але вибір розробки та впровадження може заперечити будь-яку користь від SEO, яку б інакша архітектура. Пам’ятайте про ці вимоги, розробляючи та впроваджуючи веб-сайт електронної комерції на новій платформі.

- Вміст на сторінках, які ви хочете класифікувати та стимулювати звичайний пошуковий трафік, повинен відображатися у звичайному тексті HTML та посиланнях для користувачів із вимкненими JavaScript, CSS та файлами cookie.
- Дизайн вмісту - тобто карусель, навігаційна навігація, селектори країн, локатори магазинів, відповідні модулі товарів, огляди - повинен перейти до простого тексту HTML та посилань на сторінці.
- Навігація особливо важлива. Він повинен відображатися у звичайному тексті HTML та посиланнях для користувачів із відключеними JavaScript, CSS та файлами cookie.
- Використовуйте формули, щоб замінити більш оптимальні версії за замовчуванням, створені системою URL-адреси, теги заголовків та мета-описи.
- Застосовуйте структуровані дані, такі як ціни, відео та сукупні рейтинги, щоб увімкнути розширені описи в результатах пошуку.

- Завжди виділяйте час на стратегію та реалізацію 301 перенаправлення.

Так, перші три з цих вимог були в основному однаковими, але сформульовані трьома різними способами. Це тому, що важливо сприймати вміст як щось більше, ніж текстове поле на сторінці статті. Вміст - це будь-який комунікабельний або навігаційний елемент на сторінці. Весь вміст може бути представлений у спосіб, який може допомогти SEO. Його також можна зробити невидимим для пошукових систем. Перегляньте кожен елемент веб-сайту в кожному шаблоні та прийміть усвідомлене рішення, чи застосовувати його таким чином, що приносить користь SEO.

1.4. SEO методи

1.4.1. Довгий хвіст пошук ключових слів.

Ключові слова з довгим хвостом дають можливість надати користувачам вміст, який є більш конкретним на основі їх пошукових запитів. Запити користувачів стають набагато точнішими та цілеспрямованішими. Наприклад, те, що раніше було пошуком "печива", тепер стало "печивом без алергенів поблизу Х'юстона".

Ви можете знайти ключові слова з довгим хвостом, які найбільш підходять для вашої галузі чи ніші, використовуючи інструмент добору ключових слів Google Adwords та переглядаючи дані в Інструментах для веб-майстрів та Google Analytics. Регулярно контролюйте вміст форуму, соціальні медіа та спілкування з клієнтами, а також знаходьте відповідні ключові слова з довгим хвостом, рекомендує Moz.

Веб-сайт, який надає конкретну інформацію, використовуючи варіанти ключових слів з довгим хвостом, буде розглядатися Google (і користувачами) як більший авторитет і більш авторитетний, ніж веб-сайт, який просто повторює стандартні загальні ключові слова в галузі.

Слова та фрази з найбільшою щільністю ключових слів позначаються і використовуються для визначення того, про що йдеться у вмісті. Якщо довге хвостове ключове слово "без алергенів поблизу Х'юстона" позначено, пошукова

система очікує побачити в цьому вмісті також інші відповідні терміни (терміни LSI), такі як шоколад, арахіс, інгредієнти, сейф, Техас та інші.

1.4.2. Цитати та згадування брендів

Експрес-посилання - це URL-адреси, які ведуть назад до веб-сторінки; маються на увазі посилання, які згадують бренд чи сайт, фактично не посилаючись на нього. Згадування брендів можна знайти в публікаціях блогу, коментарях та інших місцях на всьому сайті, де конкретно згадується бренд.

Через широкомасштабне зловживання та зловживання створенням посилань у минулому, Google більше, ніж у минулі роки, звертав свою увагу на посилання, що маються на увазі. Цитатами з брендом рідше маніпулювати з єдиною метою досягнення вищого рейтингу в пошукових системах, а також важче маніпулювати ними. У минулі роки побудова посилань як тактика означала, що більше посилань дорівнює вищим авторитетам. З тих пір, звичайно, Google звик до цієї техніки SEO «чорної капелюхи» - тепер ці старі методи побудови посилань завдадуть шкоди авторитету сайту. Конкретні поради щодо успішного використання згаданих брендів та прямих посилань пропонує Джейсон Демерс.

З огляду на це, оскільки згадування брендів стають дуже важливими для авторитету сайту, це не означає, що традиційні експрес-посилання зараз застаріли. Для ефективної оптимізації пошукової системи важливо, щоб веб-сайти включали в стратегії як прямі, так і прямі посилання.

1.4.3. Мобільна зручність використання

Все більше стає очевидним, що користувачі мобільних пристроїв вимагають сайтів, оптимізованих для цих мобільних пристроїв. Крім того, компанія також додала розділ зручності користування мобільними пристроями в облікових записах Google Webmaster, щоб допомогти власникам веб-сайтів побачити, як ефективні їх сайти на мобільних пристроях. Також, схоже, Google надає великого значення в зручності використання мобільних пристроїв та реагуванні на них - ми вже бачили, як Google тестує використання піктограм

поруч із погано оптимізованими сайтами, попереджаючи, що ці сайти не є зручними для мобільних пристроїв.

Власники також можуть отримати конкретні попередження, вказуючи, де вони можуть вдосконалити свої сайти. За минулий рік Google покарала веб-сайти, які містять помилки для мобільних пристроїв. Веб-сайти, що містять відео, які не можна відтворити на мобільних пристроях, а також веб-сайти, які мають несправні перенаправлення, караються з пріоритетом. Хоча в даний час зручність користування мобільними пристроями офіційно не є чинником ранжування, це лише питання часу, коли це стане. У нещодавній статті Google для веб-майстрів Google заявив, що оскільки глобальний веб-трафік із мобільних пристроїв збільшується, відвідувачі мобільних телефонів частіше відвідуватимуть веб-сайти, зручні для мобільних пристроїв. "Мобільна зручність користування зараз актуальна для оптимальних результатів пошуку", - йдеться в статті.

1.4.4. Сигнали соціальних мереж

Facebook, Google+, Twitter та інші сайти в соціальних мережах є об’єктами посилення присутності компаній та корпорацій. Експерти з питань SEO та соціальних медіа наполягають на тому, щоб налаштовувати профілі та підтримувати їх за допомогою коментарів, твітів, оновлень, фотографій тощо. 2015 рік стане банерним для маркетингу в соціальних мережах.

Експерти також просували авторство Google у 2014 році, вважаючи, що це буде важливо для досягнення високого рейтингу в пошукових системах. У серпні минулого року Google закінчив програму авторства, однак після того, як вона була визнана не такою "корисною для наших користувачів, як ми сподівались", за словами Джона Мюллера.

Зараз багато експертів вважають, що Google буде наполягати на посиленні акценту на Facebook та Twitter, хоча раніше це було небажано. Завжди існували здорові суперечки щодо того, чи використовує в даний час Google соціальні сигнали в своєму алгоритмі ранжирування. Хоча Google заперечує, що в його алгоритмі використовуються соціальні сигнали, багато досліджень показують сильну кореляцію між рейтингами та сигналами.

1.4.5. Інтегрований SEO

Натомість він повністю інтегрується з іншими маркетинговими аспектами. Є ще багато підприємств, які мають власні відділи SEO або найняли спеціалістів, що займаються SEO, для цього аспекту свого маркетингу. Однак об'єднання зусиль із соціальними медіа разом із маркетингом, що стосується конкретного контенту, розгониться повним ходом для досягнення цілей вищого рейтингу пошукових систем. Щоб веб-сайти мали успіх, їм доведеться інтегрувати та співвідносити як частини повного пакету оптимізації пошукової системи.

Це не означає, що звичайний SEO зникає або поступово відміняється - натомість вважайте це просто зміною. Замість того, щоб SEO було окремою дисципліною, воно частіше використовуватиметься як основний компонент, на який впливають (і впливають) інші аспекти маркетингу (соціальні медіа та контент).

Окрім розуміння спеціалізованих навичок SEO та дотримання актуальних тактик, змін та методів SEO, консультантам SEO також потрібно зрозуміти створення та просування контенту на сайті, в усьому Інтернеті та на сайтах соціальних медіа.

Як і в 2015 році, він уже вносить багато захоплюючих змін у всі речі, пов'язані з SEO; на додаток до згаданих раніше п'яти змін до SEO у 2015 році, протягом року відбуватиметься багато інших - деякі невеликі, а деякі змінюють ігри. Великі чи маленькі, як завжди, професіоналам потрібно не тримати очей і вух назовні та бути готовими до відповідних налаштувань, щоб не відставати.

Висновки

Респонденти виявляють повзучість, мінливі вимоги користувачів, відсутність стандартів розробки, підтримуваних вищим керівництвом, виправлених та складних методологій, помилок користувачів, відсутність людей та дипломатичних навичок, відсутність ефективних засобів комунікації для взаємодії з користувачами та членами команди розробників, відсутність загальних цілі та брак часу як загальні проблеми розвитку. Є кілька порад, які допоможуть вирішити проблеми з розробкою веб-додатків.

По-перше, практикуючим потрібні вказівки щодо визначення того, коли використовувати які методології та інструменти. Оскільки різні методології та засоби мають свої переваги та недоліки, знання того, коли використовувати, що зменшить час та витрати на розробку. Відповідно до гібридної моделі, фахівці також повинні знати, коли слід перейти на іншу методологію для досягнення оптимальних показників.

По-друге, для спілкування з різними зацікавленими сторонами, практики повинні мати ефективні засоби комунікації. Їм особливо потрібен інструмент, який може враховувати вимоги кінцевого користувача в зрозумілому, правильному та технічно обґрунтованому форматі. Цей інструмент комунікації може дозволити кінцевим споживачам переглянути функціональні вимоги, а також побачити відповідні витрати. По-третє, топ-менеджмент повинен встановлювати та виконувати цілі, врегулювати конфлікти та виконувати стандарти. По-четверте, нам потрібно навчити розробників не лише технічним навичкам, а й навичкам людей. По-п'яте, інтеграція сторонніх компонентів та веб-служб, архітектурна підтримка та стандарти даних вимагають ретельного планування та консолідації.

Аналіз існуючих сервісів і фреймворків показав, що при розробці і модифікації програмних систем доцільно використовувати перевірені методи проєктування, уважно вивчати рекомендації месенджерів і проводити тестування на невеликій аудиторії.

РОЗДІЛ 2
СТРУКТУРА ПРОГРАМНОГО МОДУЛЯ АДМІНІСТРАТОРА САЙТУ В ОНЛАЙНОВІЙ СИСТЕМІ

Дизайн веб-сайтів означає планування, створення та оновлення веб-сайтів. Він також включає архітектуру інформації, структуру веб-сайту, користувальницький інтерфейс, ергономіку навігації, макет веб-сайту, кольори, контрасти, шрифти та зображення (фотографії), а також дизайн піктограм.

Всі ці елементи веб-сайту об'єднані у формі веб-сайтів. Часто значення "дизайн" сприймається виключно як візуальний аспект. В Інтернеті першою справою бізнесу є проектування роботи сайту. Перш ніж розробляти веб-сайт, важливо визначити цілі сайту, як він буде використовуватися та як відвідувачі рухаються по ньому. Ці завдання підпадають під дисципліни Дизайн взаємодії, Дизайн інтерфейсу користувача та Дизайн досвіду користування. Насправді дизайн веб-сайту включає елементи, які є більш абстрактними. Вони включають зручність використання, ергономіку, традиції компонування, звички користувачів, логіку навігації та інші речі, які спрощують використання веб-сайтів та допомагають швидше знаходити інформацію.

Популярність в Інтернеті зростає понад 20 років, постійно розвиваючись і набуваючи економічного значення. Більше того, Інтернет потрапив на такі пристрої, як смартфони, планшети, телевізори тощо. Тому веб-сайт повинен бути адаптований до нових гаджетів. Область дизайну веб-сайтів швидко розвивається; таким чином, воно вимагає постійного вдосконалення існуючого розвитку відповідно до нових вимог. Тому підтримка хорошого веб-сайту повинна здійснюватися протягом усього періоду його роботи, щоб відповідати сучасним вимогам та залучати нових користувачів.

2.1. Об'єкт веб-дизайну та аспекти

Об’єктом веб-дизайну є веб-сторінка, яка повинна відповідати сучасним вимогам юзабіліті та дизайну та бути простою для розуміння новим користувачам.

Основні аспекти веб-дизайну охоплюють п’ять областей:

• Зміст. Сюди входять форма та організація вмісту на сайті. Можливий діапазон - від того, як написати текст до того, як він буде впорядкований, структурований та представлений за допомогою технологічної розмітки, наприклад HTML.

• Візуальні зображення. Цей аспект стосується розташування екранного простору на сайті. Зазвичай ця композиція створюється з використанням HTML, CSS або навіть Flash і може містити графічні елементи, які виступають в ролі прикраси або навігації. Візуальна сторона сайту - найочевидніший аспект веб-дизайну, але не єдина та найважливіша частина дисципліни.

• Технологія. Хоча використання різноманітних основних веб-технологій, таких як HTML або CSS, підпадає під цю категорію, технологія в цьому контексті часто посилається на різні інтерактивні елементи веб-сайту, зокрема за допомогою програмних методів. Ці елементи можуть бути в діапазоні мов сценаріїв, що працюють на стороні клієнта, таких як JavaScript, до серверних додатків, таких як JavaScript та PHP-скрипти.

• Доставка. Це означає швидкість та надійність сайту доставки в Інтернеті чи інтрамережі до прийняття апаратного, програмного забезпечення та ввімкненої архітектури мережі.

• Призначення. Причина існування сайту часто пов'язана з економічними проблемами і є, мабуть, найважливішою частиною веб-дизайну. Цей елемент слід враховувати в будь-якому рішенні, яке зачіпає інші сфери. Звичайно, ступінь впливу кожної сторони веб-дизайну на сайт може відрізнятися залежно від типу створеного сайту.

2.2. Основні цілі та вимоги до сучасного дизайну веб-сайту

Щоб створити успішний і популярний веб-сайт, потрібно визначити бізнес-цілі та визначити аудиторію відвідувачів. Це допоможе вибрати дизайн веб-сторінки та знайти найкращі технологічні рішення. Крім того, дизайн веб-сайту повинен відповідати аудиторії. Наприклад, яскраві кольори не можна використовувати для веб-сайту про фінанси, а суворі відтінки не підходять молодіжному веб-сайту.

Під час проектування веб-сайтів слід враховувати наступні вимоги:

• Дотримуючись веб-стандартів. Корисним початком є дотримання стандартів, зафіксованих Консорціумом Всесвітньої павутини. Дотримання веб-стандартів - це інструмент забезпечення максимально узгодженого веб-сайту з усіма браузерами, що відповідають стандартам (вони складають приблизно 99% браузерів, що використовуються в даний час). Це також допомагає зробити ваш вміст сумісним у міру розвитку веб-технологій та можливостей браузера. Ще одна перевага полягає в тому, що ви можете сказати своїм клієнтам, що ви створюєте сайти, що відповідають стандартам; отже, ви будете в основному апелювати до них.

• Прогресивне вдосконалення. Безліч браузерів є основою для безлічі рівнів підтримки веб-стандартів. Насправді жоден браузер не впровадив усіх стандартних 100%, і завжди є нові технології, які повільно набирають обертів. Прогресивне вдосконалення - одна із стратегій боротьби з невідомими можливостями браузера. При розробці з поступовим вдосконаленням розробник починає з базового досвіду, який робить вміст або функціональність доступними навіть для самих елементарних браузерів або допоміжних пристроїв. Прогресивне вдосконалення - це підхід, який інформує про всі аспекти дизайну та виготовлення сторінок, включаючи HTML, CSS та JavaScript.

• Адаптивний веб-дизайн. Вона була зосереджена на тому, як адаптивна розробка дозволяє створювати сайти, які добре працюють на мобільних пристроях.

• Юзабіліті. Цілями зручності є подання інформації та вибору чітким і стислим способом, відсутність двозначності та розміщення важливих предметів у відповідних областях. Одним з важливих елементів веб-зручності є забезпечення

того, щоб вміст працював на різних пристроях та браузерах. Ще однією проблемою зручності є забезпечення відповідності веб-сайту будь-якому віку та статі.

• Доступність. Люди отримують доступ до Інтернету різними способами - за допомогою зчитувачів з екрана, виведення шрифтом Брайля, луп, джойстиків, педалей тощо. Веб-дизайнери повинні створювати сторінки таким чином, щоб створювати якомога менше бар'єрів для отримання інформації,

можливостей користувача та пристрою, що використовується для доступу до Інтернету. Іншими словами, веб-сайт повинен бути спроектований для доступності.

2.3. Юзабіліті

Юзабіліті - це атрибут якості, який оцінює, наскільки простими користувальницькі інтерфейси. Слово "зручність використання" також відноситься до методів поліпшення простоти використання під час проектування. Юзабіліті - це ефективність, ефективність та задоволеність, якими зазначені користувачі досягають визначених цілей у конкретних середовищах.

Міжнародна організація зі стандартизації згадує два найважливіші критерії для суб'єктивної та об'єктивної оцінки юзабіліті: ефективності та задоволеності. І те, і інше стосується відчуття користувача при експлуатації певного виробу чи пристрою. Проте об'єктивний результат та вкладений час також цікавлять. Однак задоволення користувачів, як правило, стоїть на передньому плані, оскільки воно може зробити різницю між успішним та прибутковим продуктом та таким, який не є.

Юзабіліті, яка є виміром практики користувальницького досвіду, - це підхід до розробки продукту, який включає прямі відгуки користувачів протягом циклу розробки з метою зменшення витрат та створення продуктів та інструментів, що відповідають потребам користувачів.

Два міжнародні стандарти визначають зручність використання та орієнтований на людину (або орієнтований на користувача) дизайн:

• Юзабіліті означає, наскільки продукт може бути використаний зазначеними користувачами для досягнення визначених цілей з ефективністю, ефективністю та задоволенням у певному контексті використання.

• Орієнтований на людину дизайн характеризується: активним залученням користувачів та чітким розумінням вимог користувача та завдань; відповідний розподіл функцій між користувачами та технологіями; ітерація дизайнерських рішень; мультидисциплінарний дизайн.

3.1. Досвід користувача

Перша вимога до зразкового користувацького досвіду полягає у тому, щоб продукція відповідала точним потребам замовника, без суєти та клопоту. Далі йде простота та елегантність продуктів, якими радість володіти, радістю користуватися. Справжній досвід користувачів виходить далеко за рамки надання клієнтам того, що вони хочуть, або надання функцій контрольного списку. Для досягнення якісного користувацького досвіду в пропозиціях компанії повинно бути безперервне злиття послуг різних дисциплін, включаючи інженерію, маркетинг, графічний та промисловий дизайн та дизайн інтерфейсів. Таким чином, досвід користувачів охоплює всі аспекти взаємодії кінцевого користувача із послугами та продуктами.

Поодинокі враження впливають на загальну взаємодію користувача: досвід натискання клавіші впливає на досвід введення текстового повідомлення, досвід введення повідомлення впливає на досвід текстових повідомлень, а досвід текстових повідомлень впливає на загальний досвід користування телефоном . Загальний досвід користувачів - це не просто сума менших взаємодій, оскільки деякі враження є більш помітними, ніж інші.

Одна з галузей у дослідженні досвіду користувачів зосереджена на емоціях, що включає короткочасні переживання під час взаємодії: проектування афективної взаємодії та оцінка емоцій. Інша галузь зацікавлена в розумінні довгострокового співвідношення між досвідом користувача та оцінкою товару. Галузь бачить хороший загальний досвід користувачів із продуктами компанії як критично важливий для забезпечення лояльності до бренду та посилення зростання клієнтської бази. Усі часові рівні користувальницького досвіду

(короткочасні, епізодичні та довгострокові) важливі, але методи проектування та оцінки цих рівнів можуть бути дуже різними. Тому веб-сайти можуть залучати користувачів новими цікавими або добре розробленими функціями.

2.3.2. Якісні компоненти

За словами Якоба Нільсена, зручність використання визначається наступними п’ятьма компонентами якості:

- Навчання: Наскільки швидко користувачі можуть навчитися працювати з певним інтерфейсом і як легко їм виконувати більшу чи меншу кількість складних завдань при першій роботі з інтерфейсом.
- Ефективність: наскільки швидко користувачі виконують різні завдання після того, як вони інтегрують функціональність дизайну інтерфейсу.
- Запам’ятовуваність: коли користувачі тривалий час не працювали з інтерфейсом, скільки його функціональних можливостей вони пам’ятають і як швидко вони можуть повернути свої знання.
- Помилки: скільки помилок роблять користувачі під час роботи з інтерфейсом.

2.3.3 Важливість юзабіліті

В Інтернеті юзабіліті є необхідною умовою виживання. Люди залишають сайти в ряді

випадків:

- Якщо веб-сайтом важко користуватися;
- Якщо на домашній сторінці не вдається чітко вказати, що пропонує компанія та що можуть зробити користувачі на сайті;
- Якщо користувачі втрачаються на веб-сайті;
- Якщо інформацію веб-сайту важко прочитати або не відповідає на ключові запитання користувачів.

Перший закон електронної комерції полягає в тому, що якщо користувачі не можуть знайти товар, вони також не можуть його придбати.

Переваги планування зручності використання у вашому проекті з самого початку:

• Підвищення рівня задоволеності користувачів, що безпосередньо означає довіру та лояльність до торгової марки;

• Підвищення продуктивності, успішності та завершення роботи користувачів (і продажів);

• Знижені довгострокові витрати на розробку (витрати, понесені внаслідок виправлення помилок проекту);

• Зниження витрат на навчання та підтримку;

• Вищий рівень повторних клієнтів, що покращує вашу конкурентоспроможність.

2.3.4. Тестування юзабіліті

Існує безліч методів перевірки зручності використання, але найосновнішим та найкориснішим є тестування користувачів, яке базується на спостереженні за переходом веб-сайту користувача. Важливо протестувати користувачів індивідуально і дозволити їм вирішити будь-які проблеми самостійно. Щоб визначити найважливіші проблеми юзабіліті дизайну, зазвичай достатньо протестувати п'ять користувачів. Замість того, щоб проводити велике, дороге дослідження, краще використовувати ресурси, щоб запустити безліч невеликих тестів і переглянути дизайн між ними. Тому розробник може виправити недоліки юзабіліті. Ітераційний дизайн - найкращий спосіб підвищити якість користувацького досвіду.

Тестування користувачів відрізняється від фокус-груп, що є поганим способом оцінки зручності використання. Фокусні групи мають місце у дослідженні ринку. Однак для оцінки конструкцій взаємодії розробники повинні уважно спостерігати за окремими користувачами, коли вони виконують завдання за допомогою інтерфейсу користувача. Слухати, що люди говорять, вводить в оману: розробники повинні стежити за тим, що вони насправді роблять.

2.4. Графіка у веб-дизайні

Оскільки веб-сторінка публікується через мережу, її потрібно архівувати через рядки у вигляді невеликих пакетів даних, щоб досягти кінцевого користувача. Тож інтуїтивно зрозуміло, що для отримання більших обсягів даних потрібно більше часу.

Таким чином, народжуються конфліктні стосунки з графікою в Інтернеті. З одного боку, зображення роблять веб-сторінку цікавішою, ніж сам текст, і здатність відображати графіку є одним із факторів, що сприяють успіху Мережі. З іншого боку, графіка також випробовує терпіння користувачів із повільним з'єднанням з Інтернетом.

2.4.1. Векторна графіка

Векторна графіка складається із шляхів, які визначаються початковою та кінцевою точками, а також іншими кривими та кутами на шляху. Шлях може бути лінією, квадратом, трикутником або криволінійною фігурою. Ці шляхи можна використовувати для створення простих креслень або складних діаграм. Шляхи використовуються навіть для визначення символів певних гарнітур. Векторні зображення можна масштабувати до більшого розміру і не втрачати жодної якості зображення, оскільки вони складаються з певної кількості точок.

Об'єктно-орієнтована графіка відноситься до програмного та апаратного забезпечення, що використовує геометричні формули для подання зображень. Інший метод представлення графічних зображень - це растрові зображення, в яких зображення складається з малюнка крапок. Його іноді називають растровою графікою.

Векторно-орієнтовані зображення є більш гнучкими, ніж растрові зображення, оскільки їх можна змінювати і розтягувати. Крім того, зображення, що зберігаються у вигляді векторів, виглядають краще на пристроях (моніторах і принтерах) з більш високою роздільною здатністю, тоді як бітові зображення завжди виглядають однаковими, незалежно від роздільної здатності пристрою. Ще однією перевагою векторної графіки є те, що для подання зображень часто потрібно менше пам'яті, ніж для бітових зображень.

2.4.2. Загальні стратегії оптимізації зображення

Графіка на веб-сторінці повинна бути добре оптимізована, щоб не займати багато трафіку та не збільшувати швидкість завантаження сторінки. Тому такі стратегії оптимізації зображень мають бути реалізовані:

• Граничні розміри. Найпростіший спосіб зменшити розмір файлу - це обмежити розміри самого зображення. Це означає, що ніхто просто не робить зображення більшими, ніж вони повинні бути. Наприклад, не потрібно використовувати зображення із розмірами, які перевищують розмір найвищої роздільної здатності в наші дні.

• Повторне використання та переробка. Якщо одне і те ж зображення використовується неодноразово на сайті, краще створити лише один файл із зображенням і вказувати на нього кілька разів, де це потрібно. Це дозволяє браузеру скористатися кешованим зображенням та уникнути додаткових завантажень.

• Дизайн для стиснення. Однією з найкращих стратегій зменшення розміру файлів є розробка ефективного стиснення. Наприклад, стиснення GIF любить однорідні кольори; отже, немає необхідності розробляти зображення GIF із поєднанням градієнтних кольорів, коли буде достатньо однотонного кольору. Подібним чином, зображення JPEG, як м’які переходи та відсутність твердих країв, тому їх можна використовувати для стратегічного розмиття зображень, які будуть збережені у форматі JPEG.

2.4.3. Оптимізація GIF-файлів

Стиснення GIF працює зі смугами стиснення кольорів пікселів, що повторюються. Різні стратегії оптимізації працюють, створюючи більше областей суцільного кольору для схеми стиснення.

Загальними методами контролю розмірів файлів GIF є:

• Зменшення кількості кольорів (бітової глибини) зображення. Це найефективніший спосіб зменшити розмір файлу GIF. Тому першим кроком на шляху оптимізації є зменшення кількості кольорів на зображенні. Хоча GIF-файли можуть містити до 256 кольорів, немає правила, яке б стверджувало, що вони повинні. Насправді, зменшуючи кількість кольорів (глибину бітів), можна значно зменшити розмір файлу зображення. Однією з причин цього є те, що файли з меншою глибиною бітів містять менше даних. Іншим побічним продуктом зменшення кольорів є те, що більша частина плоских кольорів створюється поєднанням подібних, стикаються піксельних кольорів. Більш рівні

площі кольорів означають стиснення, яке є більш ефективним. Майже всі графічні програми, що дозволяють зберігати або експортувати у формат GIF, також дозволять вказати кількість кольорів або глибину бітів.

• Зменшення розмивання зображення. Коли кольори на зображенні зменшуються до певної палітри, кольори, яких немає в цій палітрі, апроксимуються шляхом дизерингу. Дизеринг - це крапчастий малюнок, який виникає в результаті змішування кольорів палітри для імітації недоступного кольору. На фотографічних зображеннях розмивання не є проблемою і може навіть бути корисним. Однак розмивання на плоских кольорових ділянках, як правило, відволікає увагу та небажане. З точки зору оптимізації, розмивання є невигідним, оскільки цяточки порушують інакше гладкі ділянки кольору. Ці блукаючі вкраплення заважають стисненню GIF і створюють великі файли. Знову ж таки, майже всі інструменти для створення GIF дозволяють включати та вимикати затухання. На зображеннях із плавними кольоровими градієнтами вимкнення згладжування призводить до неприйнятних смуг та плям.

• Фільтр втрат. Цей параметр дозволяє програмі викидати дані для того, щоб

зменшити розмір файлу. Чим вище значення, тим більше даних відкидається. Залежно від зображення, ви можете застосувати значення втрат від 5% до 20%, не погіршуючи його серйозно. Цей прийом найкраще працює для мистецтва безперервного тону.

• Дизайн з однотонними кольорами. Формат GIF пропонує численні можливості для оптимізації. В першу чергу дизайн плоскими кольорами - це хороша стратегія для створення невеликих GIF-файлів. Наступна тактика - зберегти GIF із якомога меншою кількістю кольорів, щоб зберегти зображення цілим. Регулювання величини стирання та застосування фільтра втрат - це додаткові способи видалити ще більше байтів.

2.4.4. Оптимізація JPEG

Схема стиснення JPEG розроблена для зображень із тонкими градаціями, невеликою кількістю деталей та відсутністю жорстких країв. Основним інструментом для оптимізації JPEG є налаштування якості. JPEG стискає області

гладких, змішаних кольорів ефективніше, ніж області з високою контрастністю та чіткими деталями. Насправді, чим розмитіше зображення, тим меншим є отриманий JPEG. Дуже плоскі кольори погано виходять у форматі JPEG, оскільки кольори, як правило, зміщуються і стають строкатими через стиснення, особливо при більш високих швидкостях. Загалом, плоскі графічні зображення слід зберігати у форматі GIF, оскільки якість зображення буде кращою, а розмір файлу меншим. Налаштування якості дозволяє встановити швидкість стиснення; нижча якість означає вищу стиск і менші файли. У разі використання стиснення 50% або менше, це дозволяє зменшити розмір файлу JPEG, не знижуючи при цьому якість зображення. На рисунку 4.4 показано якість зображення JPEG із різними значеннями стиснення.

Мал. 4.4. Порівняння стиснення JPEG.

2.5. Ефект паралаксу

Прокрутка паралаксу стає все більш популярною стратегією веб-дизайну. На додаток до здатності залучати користувачів до веб-сайту, прихильники цієї техніки стверджують, що вона також покращує загальний досвід користувачів.

Веб-дизайнери почали включати паралаксну прокрутку в 2011 році, використовуючи HTML5 та CSS3. Веб-сайти з паралаксним фоном стають дедалі популярнішою стратегією, оскільки прихильники стверджують, що це простий спосіб охопити плинність Інтернету. Крім того, прихильники використовують фони паралаксу як інструмент для кращого залучення користувачів та покращення загального досвіду веб-сайту. Однак дослідження Університету Пердью, опубліковане в 2013 році, показало наступні висновки: "... хоча паралаксне прокручування покращує певні аспекти взаємодії з користувачем, це не обов'язково покращує загальний досвід користувачів".

2.5.1. Концепція і методи реалізації паралаксу

Паралакс-прокрутка - це спеціальний прийом прокрутки в комп'ютерній графіці, коли фонові зображення рухаються камерою повільніше, ніж зображення переднього плану, створюючи ілюзію глибини у 2D-відеоіграх і додаючи занурення.

На веб-сайтах використовуються три основні методи прокрутки паралаксу:

• Метод шару. Деякі системи відображення підтримують безліч фонових шарів, які можна прокручувати незалежно в горизонтальному та вертикальному напрямках та компонувати один на одного, імітуючи багатоплоскостну камеру. У такій системі відображення зображення може створювати паралакс, просто змінюючи положення кожного шару на різну величину в одному напрямку. Шар, який рухається швидше, сприймається ближче до віртуальної камери. Шари можна розміщувати перед веб-сторінкою, що містить об'єкти, з якими взаємодіє користувач, - з різних причин, таких як збільшення розміру, затемнення деяких дій сторінки або відволікання користувача;

• Метод повторення шаблону / анімації. Екрани прокрутки, побудовані з окремих плиток, можна зробити так, щоб вони «плавали» над фоном, що повторюється, анімуючи растрові зображення окремих плиток, щоб зобразити ефект паралакса. Кольоровий цикл можна використовувати для швидкої анімації плиток на всьому екрані. Цей програмний ефект дає ілюзію іншого шару;

2.5.2.Прокрутка Parallax та взаємодія з користувачем

Прокрутка паралаксу стала дуже популярною технікою веб-дизайну за останні кілька років. Ефект прокрутки паралакса дозволяє кільком фонам веб-сторінки рухатися одночасно з різною швидкістю, створюючи сприйняття 3D, тим самим збагачуючи досвід перегляду. На додаток до своєї естетичної привабливості, паралаксна прокрутка пропонує веб-дизайнам можливість безпосередньо привернути увагу користувачів та направити їх на їхні продукти або „закликати до дії”. Наприклад, дизайнери можуть створювати історії, які можуть бути відкриті для користувачів під час прокрутки веб-сторінки або в інших випадках, анімовано показувати користувачам функціональність продукту. Багато веб-дизайнерів схвалюють паралаксну прокрутку як інструмент, який підходить для поліпшення взаємодії з користувачем, але завдяки його продуктивності інші стверджують інакше.

2.5.3. Переваги та недоліки ефекту Паралакс

Ефект Parallax - відносно нова технологія, тому його вплив на взаємодію з користувачем веб-сайтів не вивчений до кінця. Але це справді відзначає кілька основних його переваг та недоліків.

Переваги:

- Прокрутка паралакса дозволяє створити динамічний, інтерактивний досвід користувача, який може покращити візуальну привабливість.
- Це дозволяє видавцю вмісту відображати вміст на одній сторінці, таким чином зменшуючи навігацію та кліки, необхідні кінцевому користувачеві для споживання вмісту.

Недоліки:

- Оскільки прокрутка паралакса передбачає використання інтенсивної анімації, завантаження веб-сторінок може тривати довше.
- Прокрутка Parallax не працює гладко у всіх браузерах.
- Паралакс-прокрутка може ускладнити або засмутити кінцевого користувача легким споживанням вмісту через надмірне прокручування.

Зі зростанням кількості мобільних користувачів, які користуються Інтернетом, дизайнери повинні перекласти вплив досвіду на мобільні пристрої, щоб уникнути проблем сумісності.

2.6. Чуйний дизайн

Зі зростанням популярності смартфонів і планшетів люди мають доступ до Інтернету в будь-який момент. З огляду на це, сама ідея розробки веб-сайту, який не оптимізований для хорошої роботи на мобільному пристрої, абсурдна.

Ринок смартфонів вже не є нішевою частиною індустрії мобільних телефонів, а натомість бурхливо розвивається. Це одна з найважливіших частин Інтернет-пристроїв. Тому сучасні веб-сайти повинні розроблятися для зручного користування мобільними пристроями. Термін адаптивний дизайн походить від способу реагування браузера на навколишнє середовище. Адаптивний дизайн - це підхід до розробки веб-сайту, який має на меті забезпечити користувачам веб-сайту хороший досвід незалежно від використовуваного браузера, пристрою чи розміру екрана. Веб-сайти, розроблені з адаптивним дизайнерським підходом, адаптують свій макет за допомогою рідинних сіток, текучого вмісту (наприклад,

зображень, відео та тексту). Адаптивний дизайн відходить від використання фіксованих одиниць, таких як пікселі.

За замовчуванням більшість браузерів на невеликих пристроях, таких як смартфони та планшети, зменшують веб-сторінку вниз, щоб вона відповідала екрану та забезпечувала механізми збільшення та переміщення по сторінці. Незважаючи на те, що це технічно працює, це не великий досвід. Текст замалий для читання, посилання занадто малі, щоб натискати, і все це масштабування та панорамування навколо відволікає.

2.6.1. Концепція чуйного дизайну

Адаптивний веб-дизайн - це підхід до веб-дизайну, спрямований на створення веб-сайтів для забезпечення оптимального перегляду - легке читання та навігація з мінімальним зміною розміру, панорамуванням та прокруткою на широкому діапазоні пристроїв (від моніторів настільних комп’ютерів до мобільних телефонів) веб-дизайн - це стратегія надання користувацьким макетам пристроїв залежно від розміру області перегляду (вікна браузера). Фокус у адаптивному веб-дизайні - це подання одного документа HTML на всі пристрої, але застосування різних таблиць стилів залежно від розміру екрану, щоб забезпечити найбільш оптимізований макет для цього пристрою. Наприклад, при перегляді сторінки на смартфоні вона відображається в одному стовпці з великими посиланнями для зручного натискання. Однак, коли та сама сторінка переглядається у великому браузері настільного комп'ютера,

Адаптивний веб-дизайн був зосереджений на тому, як адаптивна розробка дозволяє створювати сайти, які добре працюють на мобільних пристроях. Крім того, настільні або ноутбучні комп’ютери з вищою роздільною здатністю та більшими дисплеями стають все більш звичним явищем. Історично, ширина веб-сайтів була побудована з метою екранів з роздільною здатністю 1024 на 800. Однак у березні 2012 року 1366 на 768 екранів стали найпоширенішою роздільною здатністю. Завдяки чуйним методам є перевага цього додаткового простору - це великі поля на веб-сайті. Зображення можуть бути більшими, вміст може бути більш рознесеним, і більше вмісту може бути видимим для користувача ще до того, як він навіть почав прокручувати.

2.6.2. Прогресивне вдосконалення та витончена деградація

Оскільки веб-браузери існують стільки ж, скільки і Інтернет, клієнти можуть переглядати ваші веб-сторінки у браузерах, які є надзвичайно старими та не мають функцій сучасних браузерів. Прогресивне вдосконалення - це стратегія управління дизайном веб-сторінок для різних браузерів.

Прогресивне вдосконалення існує принаймні з 2003 року. Тоді Інтернет все ще переходив до використання CSS замість табличних макетів. Прогресивне вдосконалення забезпечило підхід, який дозволив веб-професіоналам доставляти вміст скрізь і все ще користуватися новими функціями, де це можливо.

Прогресивне вдосконалення - це стратегія веб-дизайну, яка підкреслює доступність, семантичну розмітку HTML, а також зовнішні таблиці стилів та технології сценаріїв. Прогресивне вдосконалення використовує веб-технології пошарово, що дозволяє кожному отримати доступ до основного вмісту та функціональності веб-сторінки, використовуючи будь-який браузер або підключення до Інтернету, а також надаючи розширену версію сторінки тим, хто має більш досконале програмне забезпечення браузера або більшу пропускну здатність. .

Прогресивне вдосконалення - це спосіб проектування веб-сторінок, так що чим більше функцій підтримує користувальницький агент, тим більше функцій буде мати веб-сторінка. Це протилежність витонченої деградації стратегії дизайну, яка спочатку створює сторінки для найсучасніших браузерів, а потім перетворює їх на роботу з менш функціональними браузерами.

Прогресивне вдосконалення - це багаторівневий підхід до створення веб-сайтів із використанням HTML для вмісту, CSS для презентації та JavaScript для інтерактивності. Якщо з якихось причин JavaScript ламається, сайт все одно повинен працювати і виглядати добре. Якщо CSS завантажується неправильно, вміст HTML все одно повинен бути там із значущими гіперпосиланнями. І навіть якщо HTML неправильно сформований, браузери досить розумні, щоб продовжувати відображати те, що вони можуть.

Прогресивне вдосконалення зазвичай дає надійний результат, стійкий до несправностей і доступний. Люди, які мають найновіші пристрої,

отримуватимуть найпрогресивніший досвід, а люди, які мають повільні зв'язки або менш спроможні пристрої, все одно матимуть доступ до інформації. Веб-сайти не повинні виглядати однаково в кожному браузері, вони просто повинні надавати основний вміст та функціональність.

Прогресивне вдосконалення складається з таких основних принципів:

- базовий вміст повинен бути доступним для всіх веб-браузерів;
- базова функціональність повинна бути доступною для всіх веб-браузерів;
- розріджена семантична розмітка містить весь вміст;
- розширений макет забезпечується зовнішнім CSS;
- покращена поведінка забезпечується ненав'язливим зовнішнім JavaScript-кодом;
- налаштування веб-браузера для кінцевих користувачів дотримуються.

Витончена деградація - це практика побудови вашої веб-функціональності таким чином, щоб вона забезпечувала певний рівень користувальницької роботи в більш сучасних браузерах, але вона також витончено знижувалась до нижчого рівня користувацького досвіду у старих браузерах. Цей нижчий рівень не настільки приємний у використанні для відвідувачів сайту, але він все одно надає їм основну функціональність, яку вони прийшли на ваш сайт для використання.

У наш час прогресивне вдосконалення стає все більш популярним, але витончена деградація стає життєздатною в декількох ситуаціях:

- Розробники не мають часу закінчити продукт із повним поступовим вдосконаленням (часто ознакою поганого планування або закінчення бюджету).
- Продукт за визначенням настільки залежить від сценаріїв, що має сенс більше

підтримувати «базову» версію, а не вдосконалювати її (карти, поштові клієнти, читачі стрічок). У таблиці 6.3 коротко пояснено відмінності між прогресивним покращенням та витонченою деградацією.

Таблиця 6.3. Порівняльне прогресивне покращення та витончена деградація

Прогресивне вдосконалення Витончена деградація

Додаткові функції на веб-сторінці. Використовуючи лише функції, сумісні з

найпопулярніші браузери.

Основний вміст повинен бути доступним для всіх Весь вміст повинен бути доступним для всіх

веб-браузери. веб-браузери.

Будує сторінки для найсучасніших Будує сторінки для старих браузерів і спочатку браузери, а потім перетворює їх чим створити новий функціонал для сучасних для роботи з менш функціональними браузерами браузери.

Забезпечує найкращий досвід користувачів для Працює на сумісності зі старими бробсерами. сучасні браузери

2.6.3. Рідкі сітки

В адаптивних сітках розробники визначають розміри на основі пікселів. Тому розробникам доведеться регулювати ширину та висоту вручну у певних вікнах перегляду пристрою. Оскільки сітки для рідини протікають природним чином у межах розмірів батьківського контейнера, для різних розмірів екранів та пристроїв будуть потрібні обмежені налаштування.

Мобільні пристрої стають все меншими, і люди вважають за краще використовувати їх у своїй особистій роботі. З іншого боку, настільні монітори стають ширшими з вищою роздільною здатністю. Перевага рідинної сітки полягає в тому, що її можна регулювати максимальну ширину, і вона все одно працюватиме на більших екранах завдяки розрахунку на основі відсотків. Перша ключова ідея адаптивного дизайну - використання того, що відоме як рідинна сітка. В останній пам’яті створення «рідкого макета», який розширюється разом

із сторінкою, не було настільки популярним, як створення макетів фіксованої ширини; дизайни сторінок, які мають фіксовану кількість пікселів, а потім центрируються на сторінці. Однак, якщо взяти до уваги величезну кількість роздільних здатностей екранів, присутніх на сучасному ринку, вигода від рідинних макетів занадто велика, щоб її ігнорувати.

Рідкі сітки виходять на кілька кроків за межі традиційного компонування рідини. Замість того, щоб розробляти макет на основі жорстких пікселів або довільних процентних значень, сітка рідини більш ретельно розробляється з точки зору пропорцій. Таким чином, коли макет стискається на крихітний мобільний пристрій або розтягується на величезному екрані, всі елементи макета змінюватимуть свою ширину по відношенню один до одного.

Висновки другої частини

Веб-дизайн включає архітектуру інформації, структуру веб-сайту, користувальницький інтерфейс, ергономіку навігації, макет веб-сайту, кольори, контрасти, шрифти, зображення тощо. Оскільки з ростом веб-технологій розробка та підтримка хорошого веб-сайту повинна здійснюватися протягом усього періоду його робота щодо відповідності сучасним вимогам та залучення нових користувачів.

Головною метою створення веб-сайту є розробка успішного та популярного веб-сайту з великою аудиторією. Для досягнення цієї мети веб-сайт повинен відповідати сучасним вимогам, таким як унікальний вміст, придатні та інформативні візуальні дані, використання сучасних технологій, швидка швидкість доставки вмісту веб-сайту, високомотивоване та зрозуміле пояснення призначення веб-сайту.

Для утримання нового відвідувача та залучення нових користувачів веб-сайт повинен відповідати вимогам юзабіліті, наприклад, привласнювати його для будь-якого віку та статі та стабільну роботу на всіх пристроях із простим та зручним інтерфейсом.

Щоб створити зручний і зрозумілий веб-сайт, дизайнери та розробники повинні взяти

зручність користування обліковим записом. Юзабіліті - це атрибут якості, який оцінює наскільки легкий користувач

використовувати інтерфейси. Слово "зручність використання" також відноситься до методів поліпшення простоти використання під час проектування.

Графіка на сучасному веб-сайті дійсно потрібна, але повинна бути сильно оптимізована. Векторна графіка підходить для основного дизайну, наприклад, піктограм навігації, логотипів тощо. Зображення вмісту повинні бути оптимізовані, щоб не витрачати величезну кількість трафіку даних. Для цього можуть бути використані обмежувальні розміри, масштабування зображень, якісна техніка стиснення.

Зі зростанням популярності смартфонів, планшетів та інших мобільних пристроїв з’явилася концепція адаптивного веб-дизайну. Адаптивний веб-дизайн - це стратегія надання користувацьким макетам пристроїв залежно від розміру вікна браузера. Це підхід до розробки веб-сайту, який має на меті надати користувачам веб-сайту хороший досвід незалежно від використовуваного браузера, пристрою чи розміру екрана. Для використання нових функцій браузера розробники повинні використовувати принципи поступового вдосконалення або витонченої деградації. Прогресивне вдосконалення - це спосіб проектування веб-сторінок, так що чим більше функцій підтримує користувальницький агент, тим більше функцій буде мати веб-сторінка. А спочатку створюючи сторінки для найсучасніших браузерів, а потім перетворюючи їх на роботу з менш функціональними браузерами.

У результаті аналізу оформлено словник предметної області, який містить текстовий опис термінів, сутностей, користувачів тощо.

РОЗДІЛ 3
МОДИФІКОВАНИЙ МОДУЛЬ АДМІНІСТРАТОРА САЙТУ ОНЛАЙН ЗАМОВЛЕНЬ

Оптимізація веб-сайтів означає оптимізацію означає оптимізацію вмісту, URL-адрес, дизайну, стилів та програмного коду для отримання швидкого та якісного веб-сайту для залучення відвідувачів.

3.1. Пошукова оптимізація

Оптимізація пошукової системи (SEO) - це процес впливу на видимість веб-сайту або веб-сторінки в неоплачених результатах пошукової системи - їх часто називають «природними», «органічними» або «зароблениими» результатами. Загалом, чим раніше (або вище рейтинг на сторінці результатів пошуку), і чим частіше сайт з'являється у списку результатів пошуку, тим більше відвідувачів він отримає від користувачів пошукової системи. SEO може націлювати різні типи пошуку, включаючи пошук зображень, локальний пошук, пошук відео, академічний пошук, пошук новин та галузеві вертикальні пошукові системи.

Як стратегія маркетингу в Інтернеті, SEO розглядає, як працюють пошукові системи, що люди шукають, фактичні пошукові терміни або ключові слова, введені в пошукові системи, і яким пошуковим системам надає перевагу їх цільова аудиторія. Оптимізація веб-сайту може включати редагування його вмісту, HTML та відповідного кодування, щоб як підвищити його відповідність певним ключовим словам, так і усунути перешкоди для діяльності з індексування пошукових систем. Просування сайту для збільшення кількості зворотних посилань або вхідних посилань - ще одна тактика SEO.

3.1.1. Типи факторів успіху пошукової системи

Існує дві основні групи факторів успіху пошукової системи:

• SEO на сторінці фактори рейтингу пошуку - це ті, які майже повністю перебувають під власним контролем видавця. Він включає тип вмісту, що публікується на веб-сайті, надаючи важливі підказки HTML, які допомагають пошуковим системам (і користувачам) визначати актуальність.

• Позасторінковий SEOфактори рейтингу - це ті, які видавництва безпосередньо не контролюють. Пошукові системи використовують їх, оскільки на початку зрозуміли, що покладання лише на контрольовані видавцями сигнали не завжди дає найкращі результати. Наприклад, деякі видавці можуть намагатися зробити себе більш актуальним, ніж вони є насправді.

3.1.2. Фактори успіху вмісту та пошукової системи

Вміст є основною частиною веб-сайту, оскільки без високоякісного вмісту отримати високопоставлений сайт неможливо.

Якість вмісту. Ви більше за все інше створюєте якісний контент? Якщо веб-сайт мав на меті щось продати, чи це виходить за рамки простої брошури з тією ж інформацією, яку можна знайти на сотнях інших сайтів. Це повинно бути причиною того, що люди витрачають більше кількох секунд на читання ваших сторінок. Крім того, він повинен пропонувати відвідувачам справжню цінність, щось суттєве, унікальне, різне, корисне, чого вони не знайдуть деінде.

Дослідження вмісту / Дослідження ключових слів. Мабуть, найважливішим фактором SEO після створення хорошого контенту є хороше дослідження ключових слів. Існує безліч інструментів, які дозволяють виявити конкретні способи, за якими люди можуть шукати ваш вміст. SEO повинен створювати вміст із використанням цих ключових слів, фактичних пошукових термінів, які люди використовують, щоб ви могли створювати вміст, який ефективно “відповідає” на цей запит.

Свіжість вмісту. Пошукові системи люблять новий вміст. Тож ми не можемо щодня оновлювати свої сторінки (або дату публікації), думаючи, що це зробить їх «новими» та з більшою ймовірністю їх рейтингу. Ми не можемо просто додавати нові сторінки постійно, лише заради того, щоб мати нові сторінки, і думати, що це дає нашому веб-сайту свіжість. Найкращий спосіб пояснити це таким терміном, як "ураган". Якщо активного урагану немає, тоді результати пошуку, швидше за все, будуть містити списки державних та довідкових сайтів. Але якщо буде активний ураган, результати будуть змінюватися і можуть відображати історії, новини та інформацію про активний ураган. Сайти можуть скористатися цим підвищенням свіжості, виробляючи відповідний вміст, який відповідає пульсу в реальному часі їхньої галузі.

3.1.3. Архітектура сайту та фактори успіху пошукової системи

Сканування сайту. Пошукові системи "сканують" веб-сайти, неймовірно швидко переходячи з однієї сторінки на іншу, діючи як гіперактивні швидкісні зчитувачі. Вони роблять копії ваших сторінок, які зберігаються в так званому «покажчику», який нагадує величезну книгу в Інтернеті.

Коли хтось шукає, пошукова машина гортає цю велику книгу, знаходить усі відповідні сторінки, а потім вибирає, що, на її думку, найкраще показати першим. Щоб вас знайшли, ви повинні бути в книзі. Щоб бути в книзі, веб-сайт повинен бути просканований. На більшості сайтів зазвичай немає проблем сканування, але є речі, які можуть спричинити проблеми. Наприклад, JavaScript або Flash можуть потенційно приховувати посилання; зробити сторінки, на які ведуть ці посилання, прихованими від пошукових систем. І те, і інше потенційно може спричинити приховання справжніх слів на сторінках.

Кожному сайту надається бюджет сканування, приблизний час або сторінки, які пошукова система буде сканувати щодня, виходячи з відносної довіри та повноважень сайту. Більші сайти можуть прагнути покращити

ефективність сканування, щоб забезпечити частіше сканування "правильних" сторінок. Використання robots.txt, внутрішніх структур посилань, а також спеціально вказівка пошуковим системам не сканувати сторінки з певними параметрами URL-адреси, може покращити ефективність сканування.

Мобільний Більше пошукових запитів у Google відбувається на мобільних пристроях, ніж на настільних. Враховуючи це, не дивно, що Google винагороджує веб-сайти, придатні для мобільних пристроїв, з шансом отримати кращі рейтинги за мобільними пошуками, тоді як тим, які не є, може бути важче виглядати. Бінг теж робить те саме. Тому ми повинні отримати веб-сайт, зручний для мобільних пристроїв. Це збільшить шанси на успіх із пошуковим рейтингом, оскільки це порадує відвідувачів мобільних пристроїв.

Швидкість сайту.Google хоче зробити Інтернет швидшим і заявив, що швидкі сайти отримують невелику перевагу в рейтингу над повільними. Однак швидке створення пухирців на веб-сайті не є гарантованою швидкою поїздкою до вершини результатів пошуку. Швидкість - незначний фактор, який впливає лише на 1 із 100 запитів за даними Google. Але швидкість може посилити інші фактори, а насправді може покращити інші. У наш час ми нетерпляча група людей, особливо коли ми користуємось мобільними пристроями. Тож залучення (і перетворення) на сайті може покращитися залежно від швидкого часу завантаження.

HTTPS / захищений сайт. Google хотів би бачити всю мережу, на якій працюють сервери HTTPS, з метою забезпечення кращого захисту веб-серферів. Щоб допомогти в цьому, це винагороджує сайти, які використовують HTTPS, з невеликим підвищенням рейтингу. Як і при підвищенні швидкості веб-сайту, це лише один із багатьох факторів, які Google використовує, приймаючи рішення про те, чи повинна веб-сторінка мати високий рейтинг. Це одне не гарантує досягнення найкращих результатів.

3.1.4. HTML-код та фактори успіху пошукової системи

Тег заголовка HTML.Назви HTML завжди були і залишаються найважливішим сигналом HTML, який використовують пошукові системи, щоб зрозуміти, про що йдеться на сторінці. Погані заголовки на сторінках - це як мати погані назви книг у прикладах вище. Насправді, якщо заголовки HTML вважаються поганими або не мають опису, Google змінює їх.

Тег мета-опису. Тег мета-опису, один із найдавніших підтримуваних елементів HTML, дозволяє запропонувати, як сторінки описуватись у списках пошуку. Якщо заголовок HTML еквівалентний назві книги, метаопис подібний до розмиття на звороті, що описує книгу.

Тег заголовка.Теги заголовка є формальним способом ідентифікації ключових розділів веб-сторінки. Пошукові системи здавна використовують їх як підказки щодо того, про що йдеться на сторінці. Якщо пошукові слова знаходяться в тегах заголовка, це може трохи збільшити ймовірність появи в пошуках цих слів. Теги заголовка корисні, коли вони відображають логічну структуру (або контур) сторінки. Якщо у вас є головний заголовок, використовуйте тег H1. Відповідні підзаголовки повинні використовувати тег H2. Потрібно використовувати заголовки, оскільки вони мають сенс, і вони можуть посилити інші фактори ранжування.

3.1.4. Історія сайту

Історія. Оскільки пошукові системи постійно відвідують веб-сайти, вони можуть зрозуміти, що є “нормальним” або як це поводилося з часом. Веб-сайту з історією, що порушує вказівки та отримує численні штрафи, може бути важче повернутися до пошуку популярності. Хороший загальний послужний список може допомогти. Старий, встановлений сайт може виявити, що він може продовжувати курсувати разом із успіхом у пошуку, тоді як новому сайту, можливо, доведеться «сплачувати внесок», так би мовити, тижнями, місяцями чи навіть довше, щоб завоювати повагу.

Заручини. Якісний сайт повинен створювати значущі взаємодії з користувачами. Пошукові системи можуть спробувати виміряти цю взаємодію -

залучення - різними способами. Наприклад, як довго користувачі залишаються на вашій сторінці? Вони шукали, переходили до вашого списку, але потім негайно поверталися до результатів, щоб спробувати щось інше?

І навпаки, чи надсилають люди відносно тривалий час перегляд вашого вмісту стосовно подібного вмісту на інших сайтах? Цей показник "час на сайті" або "довгий клік" - це ще один тип взаємодії, який пошукові системи можуть виміряти та використовувати для оцінки відносної цінності вмісту. Соціальні жести, такі як коментарі, поділитися і "подобається", є ще одним способом вимірювання залученості. Ми розглянемо їх більш докладно в соціальному розділі цього посібника.

Як правило, пошукові системи сумнівно ставляться до використання метрик залучення, а тим більше до специфіки цих метрик. Однак я впевнений, що залучення вимірюється та використовується для інформування результатів пошуку.

3.1.5 Персоналізація.

Роками тому всі бачили абсолютно однакові результати пошуку. Сьогодні ніхто не бачить абсолютно однакових результатів пошуку ні в Google, ні в Bing. Кожен отримує персоналізований досвід до певної міри, навіть у приватних вікнах перегляду. Звичайно, є ще багато спільного. Це не те, що всі бачать абсолютно різні результати. Натомість усі бачать багато однакових “загальних” списків. Але також з’являтимуться деякі списки через те, де хтось перебуває, кого він знає або як вони користуються Інтернетом.

Країна. Одним з найпростіших для розуміння факторів ранжирування персоналізації є те, що людям показують результати, що відповідають країні, в якій вони перебувають. Хтось у США, який шукає „футбол”, отримає результати про американський футбол; хтось у Великобританії отримає результати про тип футболу, який американці називали б футболом.

Місцевість.Пошукові системи не зупиняють персоналізацію на рівні країни. Вони пристосовують результати відповідно до міста чи столичного

району залежно від місцезнаходження користувача. Як і у випадку з персоналізацією країни, якщо ми хочемо з'являтися, коли хтось отримує результати для конкретного міста, нам потрібно переконатися, що ваш сайт відповідає цьому місту.

Особиста історія. Що хтось шукав і натискав у результатах пошуку? Які сайти вони регулярно відвідують? Вони "сподобалися" сайту за допомогою Facebook, поділились ним через Twitter чи, можливо, поставили +1? Цей тип особистої історії використовується різним чином і способами як Google, так і Bing впливати на результати пошуку. На відміну від персоналізації в країні чи місті, немає простого способу спробувати зробити себе більш актуальним. Натомість це надає більше значення першим враженням та лояльності до бренду. Коли користувач натискає "звичайний" результат пошуку, ви хочете переконатись, що представляєте чудовий досвід, щоб вони знову прийшли. З часом вони можуть шукати ваш бренд у результатах пошуку, натискаючи його, незважаючи на те, що він знаходиться нижче інших списків. Ця поведінка посилює ваш сайт як такий, що його слід частіше показувати цьому користувачеві. Ще краще, якщо вони ініціюють соціальний жест, такий як "Подобається", "+1" або "Твіт", який вказує на більшу прихильність до вашого сайту або бренду. Історія ще важливіша в нових пошукових інтерфейсах, таких як Google Now, який активно надаватиме «картки» користувачам на основі чітких уподобань (тобто - які спортивні команди чи акції ви відстежуєте) та історії пошуку. Що чиїсь друзі думають про веб-сайт? Це один із нових факторів ранжування, що впливає на результати пошуку. Соціальні зв'язки когось можуть впливати на те, що вони бачать у Google та Bing. Ці зв'язки є справді важливими, оскільки пошукові системи розглядають ці зв'язки як особистий набір радників користувача. Офлайн, ви можете довіряти і просити своїх друзів порадити вам про ресторан чи садівництво. Все частіше, коли ви шукаєте сьогодні, пошукові системи намагаються наслідувати цей офлайн-сценарій. Отже, якщо користувач зв'язаний із другом, і цей друг переглянув ресторан або поділився статтею про

вирощування помідорів, тоді цей ресторан та стаття можуть отримати вищу оцінку для цього користувача. Якщо хтось може слідувати за вами або легко ділитися вашим вмістом, це допомагає потрапити на ваш сайт у коло довіри та збільшує шанси, що інші, кого вони знають, знайдуть вас. Ніде це не настільки трансформативно, як Google+, де об'їзд сторінки Google+ у веб-сайті змінить персоналізовані результати пошуку для цього користувача. це допомагає потрапити на ваш сайт у коло довіри та збільшує шанси, що інші, кого вони знають, знайдуть вас. Ніде це не настільки трансформативно, як Google+, де об'їзд сторінки Google+ у веб-сайті змінить персоналізовані результати пошуку для цього користувача. це допомагає потрапити на ваш сайт у коло довіри та збільшує шанси, що інші, кого вони знають, знайдуть вас. Ніде це не так трансформативно, як Google+, де об'їзд сторінки Google+ у веб-сайті змінить персоналізовані результати пошуку для цього користувача.

3.1.6. Соц.медіа

Використання посилань як фактора ранжування поза сторінками стало великим стрибком вперед для пошукових систем. Але з часом посилання втратили частину своєї цінності з різних причин. Деякі сайти скупі на посилання. Інші блокують посилання для боротьби зі спамом. А посилання купують і продають, роблячи їх менш надійними. Якби посилання були способом, яким люди могли "проголосувати" на користь сайтів, спільний доступ у соціальних мережах є способом продовження такої поведінки при голосуванні. Соціальні сигнали з'являються як фактори ранжування, оскільки пошукові системи визначають, як використовувати нашу соціальну взаємодію та поведінку.

Соціальна репутація.Подібно до того, як пошукові системи не враховують усі посилання однаково, вони не розглядають усі соціальні акаунти як однакові. Це має сенс, оскільки кожен може створити новий обліковий запис у соціальній мережі. Що заважає комусь робити 100 різних облікових записів, щоб виробляти фальшиві казки? Насправді нічого, крім підроблених облікових записів, подібних до цих, часто буває легко знайти. У них може бути лише кілька "якісних" друзів

у своїй мережі, і мало хто може передавати матеріали, якими вони діляться. В ідеалі ви хочете отримати довідки з соціальних акаунтів з хорошою репутацією. Важливо мати власну соціальну присутність, яку добре розглядають. Тож беріть участь у відповідних соціальних платформах по-справжньому, справжньо, як і на своєму веб-сайті, або з клієнтами в режимі офлайн.

Соціальні акції. Подібно до посилань, отримання якісних соціальних акцій є ідеальним, але широке поширення в соціальних мережах все одно корисно. Хороші речі трапляються, коли більше людей бачать сайт чи бренд. Тож ми можемо брати участь у соціальних сайтах і обмінюватися вмістом, розміщувати свій бренд на великій кількості соціальних груп.

3.2. Оптимізація коду.

Мініфікація (також мінімізація або мінімізація) в мовах комп'ютерного програмування і особливо JavaScript - це процес видалення всіх непотрібних символів із вихідного коду без зміни його функціональних можливостей. Ці непотрібні символи зазвичай включають пробіли, символи нового рядка, коментарі, а іноді і роздільники блоків, які використовуються для додавання читабельності коду, але не потрібні для його виконання.

Зменшений вихідний код особливо корисний для інтерпретованих мов, розгорнутих та переданих в Інтернеті (наприклад, JavaScript), оскільки зменшує обсяг даних, які потрібно передати. Зменшений вихідний код також може використовуватися як різновид затуманення, хоча термін затухання може бути розрізнений як форма помилкової криптографії, тоді як примірник мініфікованого коду може бути зворотним за допомогою гарненького принтера.

3.2.1 Мініфікація JavaScript

Що стосується створення сторінки або запуску сценарію, веб-браузери не турбуються про читаність коду. Мініфікація позбавляє файл коду всіх даних, які не потрібні для виконання файлу. На відміну від традиційних методів стиснення, зменшені файли не потрібно розпаковувати, перш ніж їх можна прочитати, змінити або виконати.

Мініфікація виконується після написання коду веб-програми, але до розгортання програми. Коли користувач запитує веб-сторінку, замість повної версії надсилається мінімізована версія, що призводить до швидшого часу відгуку та менших витрат на пропускну здатність. Мініфікація використовується на веб-сайтах, починаючи від невеликих особистих блогів і закінчуючи багатомільйонними послугами для користувачів.

Мініфікація працює, аналізуючи та переписуючи текстові частини веб-сайту, щоб зменшити загальний розмір файлу. Мініфікація поширюється на сценарії, таблиці стилів та інші компоненти, які веб-браузер використовує для візуалізації сайту.

Мініфікація виконується на веб-сервері перед надсиланням відповіді. Після мініфікації веб-сервер використовує зменшені ресурси замість вихідних для швидшого розподілу серед користувачів.

Ось покроковий опис того, як працює мініфікація:

1) Веб-розробник створює файл JavaScript або CSS для використання у веб-програмі. Ці файли відформатовані для зручності розробника, а це означає, що вони використовують пробіли, коментарі, довгі імена змінних та інші методи читання.

2) Розробник застосовує техніку мініфікації (див. Нижче) для перетворення файлу в той, який є більш оптимізованим, але важчим для читання. Поширені методи мініфікації включають видалення пробілів, скорочення назв змінних та заміну багатослівних функцій на більш короткі, стислі функції.

3) Веб-сервер використовує мініфікований файл під час відповіді на веб-запити, що призводить до зменшення використання пропускної здатності без шкоди для функціональності.

Перевага мініфікації полягає в тому, що її потрібно виконувати лише при зміні вихідного файлу. У поєднанні з іншими методами стиснення мініфікація може значно зменшити використання смуги пропускання як для підприємства, так і для користувача.

Переваги мініфікації:

1) Мініфікація має всі переваги традиційних методів стиснення з меншою кількістю недоліків.

2) Користувачі завантажують вміст швидше, оскільки потрібно завантажувати менше непотрібних даних. Користувачі користуються ідентичним сервісом без додаткових накладних витрат.

3) Підприємства бачать нижчі витрати на пропускну здатність, оскільки менше даних передається по мережі. Додатковий вміст, про який піклуються лише розробники, більше не надсилається користувачам. Підприємства також бачать менший рівень використання ресурсів, оскільки для кожного запиту потрібно обробляти менше даних. Зменшений вміст, який потрібно генерувати лише один раз, можна використовувати для необмеженої кількості запитів.

3.2.2. Мініфікаційний Javascript від YUI Compressor

Метою YUI Compressor було зменшити файли JavaScript, застосовуючи розумні оптимізації до вихідного коду. Окрім видалення коментарів, пробілів та вкладок, YUI Compressor також безпечно видаляє розриви рядків, додатково зменшуючи загальний розмір файлу. Однак найбільша економія байтів відбувається за рахунок заміни імен локальних змінних на імена з одним або двома символами. Наприклад, припустимо, у вас є така функція:

```
сума функції (число1, число2) {
повернути num1 + num2;
}
```

YUI Compressor змінює це на:

```
сума функції (A, B) {повернути A + B;}
```

Зверніть увагу, що дві локальні змінні, num1 та num2, були замінені на A та B відповідно. Оскільки YUI Compressor насправді аналізує весь вхід JavaScript, він може безпечно замінити імена локальних змінних, не вводячи помилок коду. Загальна функція продовжує працювати так само, як і спочатку, оскільки імена

змінних не мають відношення до функціональності. У середньому YUI Compressor може стискати файли на 18% більше, ніж JSMin.

У наші дні зазвичай використовується інструмент мініфікації та стиснення HTTP для подальшого зменшення розміру файлу JavaScript. Це призводить до ще більшої економії, ніж використання одного з цих методів.

3.2.3. Шаблони кодування, які можуть зменшити продуктивність мініфікації.

EVAL ()

Завдання оператора eval () полягає в тому, щоб взяти рядок і інтерпретувати його як код JavaScript. Наприклад:

eval ("alert ('Привіт світ!');");

Хитра частина eval () полягає в тому, що він має доступ до всіх змінних та функцій, які існують навколо нього. Ось більш складний приклад:

var message = "Привіт світ!";

функція doSomething () {

eval ("попередження (повідомлення)");

}

Коли ви телефонуєте doSomething (), відображається попередження із повідомленням: «Привіт, світ!». Це тому, що рядок, переданий у eval (), отримує доступ до повідомлення глобальної змінної та відображає його. А тепер подумайте, що сталося б, якщо ви автоматично замінили повідомлення імені змінної:

var A = "Привіт світ!";

функція doSomething () {

eval ("попередження (повідомлення)");

}

Зверніть увагу, що зміна імені змінної на A призводить до помилки при виконанні doSomething () (оскільки повідомлення не визначено). Першим завданням YUI Compressor є збереження функціональності вашого сценарію, і

тому, коли він бачить eval (), він перестає замінювати змінні. Це може здатися не такою поганою ідеєю, поки ви не зрозумієте всіх наслідків: Заміна імені змінної запобігається не тільки в локальному контексті, де викликається eval (), але і у всіх контекстах, що містять. У попередньому прикладі це означає, що як контекст всередині doSomething (), так і глобальний контекст не можуть замінювати імена змінних.

Використання eval () в будь-якому місці вашого коду означає, що імена глобальних змінних ніколи не будуть змінені. Розглянемо наступний приклад:

функція handleJSONP (об'єкт) {

повернути об'єкт;

}

інтерпретація функціїJSONP (код) {

дані var = eval (код); // обробка даних

}

У цьому коді зробіть вигляд, що handleJSONP () та інтерпретаціяJSONP () визначені посеред інших функцій. JSONP- це широко використовуваний формат спілкування Ajax, який вимагає інтерпретації відповіді механізмом JavaScript. У цьому прикладі зразок відповіді JSONP може виглядати так:

handleJSONP ({повідомлення: "Привіт світ!"});

Якщо ви отримали цей код назад із сервера за допомогою виклику XMLHttpRequest, наступним кроком є його оцінка, після чого eval () стає дуже корисним. Але наявність у коді коду EVAL () означає, що жоден із глобальних ідентифікаторів не може замінити їх імена. Найкращий варіант - обмежити кількість введених вами глобальних змінних.

Часто можна врятуватися, створивши самовиконуючуся анонімну функцію, таку як:

(функція () {

функція handleJSONP (об'єкт) {

повернути об'єкт;

```
}
інтерпретація функціїJSONP (код) {
дані var = eval (код); // обробка даних
}
}) ();
```

Цей код не вводить нових глобальних змінних, але оскільки використовується eval (), жодне з імен змінних не буде замінено. Фактичний результат (110 байт):

```
(функція () {
функція handleJSONP (об'єкт) {
повернути об'єкт
}
функція »interpretJSONP (код) {
var data = eval (код)
}
}) ();
```

Приємним у JSONP є те, що він покладається на існування лише одного глобального ідентифікатора, функції, якій повинен бути переданий результат (у цьому випадку handleJSONP ()). Це означає, що йому не потрібен доступ до будь-яких локальних змінних або функцій і дає вам можливість секвеструвати функцію eval () у своїй власній глобальній функції. Зверніть увагу, що ви також повинні перемістити handleJSONP () назовні, щоб бути глобальним, щоб його ім'я не замінювалось:

```
функція myEval (код) {
повернення eval (код);
}
функція handleJSONP (об'єкт) {
повернути об'єкт;
}
```

```
(функція () {
інтерпретація функціїJSONP (код) {
дані var = myEval (код); // обробка даних
}
}) ();
```

Функція myEval () тепер діє як eval (), за винятком того, що вона не може отримати доступ до локальних змінних. Однак він може отримати доступ до всіх глобальних змінних та функцій. Якщо код, який виконується eval (), ніколи не потребуватиме доступу до локальних змінних, тоді такий підхід є найкращим. Зберігаючи єдине посилання на eval () поза анонімною функцією, ви дозволяєте замінювати кожне ім'я змінної всередині цієї функції. Ось результат:

```
функція myEval (код) {
повернути eval (код)
}
функція handleJSONP »(a) {
повернути a
}
(функція () {
функція a (b) {
var c = myEval (b)
}
}) ();
```

Ви бачите, що і interpretJSON (), і код, і дані були замінені (на a, b та c відповідно). Результат - 120 байт, які, як ви зауважите, більші за приклад без секвестру eval (). Це не означає, що підхід хибний, просто цей приклад коду занадто малий, щоб побачити вплив. Якби ви застосували цю зміну до 100 КБ коду JavaScript, ви побачили б, що отриманий код набагато менший, ніж залишення eval () на місці.

Звичайно, найкращий варіант - взагалі не використовувати eval (), оскільки ви уникнете багато стрибків в обручі, щоб зробити YUI Compressor щасливим. Однак, якщо потрібно, тоді секвестр функції eval () - найкращий вибір для оптимальної мініфікації.

З твердженням

Оператор with - це друга зла характеристика, яка заважає техніці змінної заміни YUI Compressor. Для тих, хто не знайомий, withstatement призначений (теоретично) для зменшення розміру коду, усуваючи необхідність писати однакові імена змінних знову і знову. Розглянемо наступне:

```
var об'єкт = {
повідомлення: "Привіт",
messageSuffix: "і ласкаво просимо."
};
object.message + = "світ" + object.messageSuffix; оповіщення (object.message);
```

Оператор with дозволяє вам переписати цей код як:

```
var об'єкт = {
message: "Привіт", messageSuffix: "і ласкаво просимо."
};
with (object) {
повідомлення + = "світ" + messageSuffix;
попередження (повідомлення);
}
```

Фактично, оператор with уникає необхідності повторювати "об'єкт" кілька разів у коді. Але ці заощадження мають свою ціну. По-перше, єнаслідки для продуктивностівід використання оператора with, оскільки місцеві змінні стають повільнішими для доступу. Це трапляється тому, що змінні всередині оператора with є неоднозначними до часу виконання: вони можуть бути властивостями об'єкта контексту оператора with або можуть бути змінними функції або іншого

контексту виконання. Щоб краще зрозуміти цю двозначність, подивіться на код, коли додається повідомлення про локальну змінну та видаляється визначення об’єкта:

```
var message = "Йо,";
with (object) {
повідомлення + = "світ" + messageSuffix;
попередження (повідомлення);
}
```

Коли повідомлення-ідентифікатор використовується всередині оператора with, воно може посилатися на повідомлення локальної змінної або посилатися на властивість з ім'ям message на об'єкті. Оскільки JavaScript є мовою пізнього прив'язки, неможливо дізнатися справжнє посилання на повідомлення, не виконавши повністю код і не визначивши, чи має об'єкт властивість з іменем message. Свідчіть, як пізнє прив’язування впливає на цей код:

```
функція displayMessage (об'єкт) {
var message = "Йо,";
with (object) {
повідомлення + = "світ" + messageSuffix;
попередження (повідомлення);
}
}
displayMessage ({
```

message: "Привіт", messageSuffix: "і ласкаво просимо." }); displayMessage ({messageSuffix: ", і ласкаво просимо."});

Перший раз, коли викликається displayMessage (), переданий об’єкт має властивість з іменем message. Коли оператор with виконується, посилання на повідомлення прив'язується до властивості об'єкта, і, таким чином, відображається повідомлення "Привіт, світ, і ласкаво просимо". Вдруге переданий об’єкт має лише властивість messageSuffix, тобто посилання на

повідомлення всередині оператора with посилається на локальну змінну, і, таким чином, відображається повідомлення: «Yo, world, and welcome».

Оскільки YUI Compressor насправді не виконує код JavaScript, він ніяк не може дізнатися, чи є ідентифікатори в операторі with властивостями об'єкта (у такому випадку небезпечно їх замінювати) або посиланнями на локальні змінні (у такому випадку, їх можна безпечно замінити). YUI Compressor розглядає оператор with те саме, що і eval (): коли він присутній, він не буде виконувати заміну змінної функції або будь-якого контексту, що містить виконання.

На відміну від eval (), немає способу секвеструвати оператор with таким чином, щоб він не впливав на більшість коду. Рекомендація полягає в тому, щоб взагалі уникати використання атрибуту. Навіть незважаючи на те, що на момент написання коду це зберігає байти, ви фактично втрачаєте байти, втрачаючи функцію заміни змінної YUI Compressor. Функція displayMessage () зменшується таким чином:

```
функція displayMessage (об'єкт) {
var message = "Йо,";
з (об’єктом) »{
повідомлення + = "світ" + messageSuffix; попередження (повідомлення)}
};
```

Цей результат - 112 байт. Якщо функцію переписано, щоб уникнути оператора with, displayMessage () виглядає так:

```
функція displayMessage (об'єкт) {
var message = "Йо,";
object.message + = "світ" + object.messageSuffix;
оповіщення (object.message);
}
```

Після зменшення ця нова версія функції стає:

```
функція displayMessage (a) {var b = "Yo,";
a.message + = "world" + »a.messageSuffix; alert (a.message)};
```

Розмір цього результату становить 93 байти, тож навіть незважаючи на те, що вихідний код більший. Зменшений вихідний код стає меншим, оскільки ми використовували заміну змінної.

Заміна імені змінної

Заміна імен локальних змінних на короткі (один або два символи) варіанти є найбільшою економією байтів, яку пропонує YUI Compressor. Часто іменування змінних викликає страх у кодуванні, але принципово імена змінних мають значення лише для людей, які намагаються зрозуміти код. Переконавшись, що людині не потрібно інтерпретувати код, змінні просто стають загальними заповнювачами значень. Розгортаючи JavaScript у виробництві, вашим єдиним споживачем є браузер, і браузер не міг дбати про імена змінних. З цієї причини YUI Compressor замінює ці імена змінних на коротші версії.

Перш ніж говорити про методи оптимального заміщення змінних, важливо зрозуміти частини коду, які не можна замінити. Наступні часто вважаються замінними:

- імена властивостей (object.property),
- ключові слова (if, var, return тощо) та
- глобальні змінні.

Багато розробників не зупиняються, думаючи про те, скільки разів доступ до властивості здійснюється або скільки ключових слів JavaScript вони використовують під час написання коду. Врахування імен властивостей, ключових слів та глобальних змінних у структурі коду може призвести до набагато більших коефіцієнтів мініфікації.

Назви властивостей

Хоча YUI Compressor може виконувати заміну змінних на локальні змінні, він нічого не може зробити щодо властивостей. Оскільки імена властивостей визначаються на самому об'єкті, немає можливості безпечно їх виявити або замінити. Це означає, що код, який отримує доступ до багатьох властивостей, в кінцевому підсумку буде більшим за код, який цього не робить. Наприклад:

функція toggleImage (id) {

якщо (document.getElementById (id) .src.indexOf ("1.png")> -1)

{document.getElementById (id) .src = document.getElementById »(id) .src.replace (" 1.png "," 2.png ");

}

}

Ця функція перемикає відображення зображення, змінюючи його з одного зображення на інше, маніпулюючи властивістю src. На жаль, отримана мінімізована функція майже така ж велика (168 байт проти 205 байт):

функція toggleImage (a) {

якщо (document.getElementById (a) .src.indexOf »(" 1.png ")> - 1) {

document.getElementById (a) .src = document.getElementBy »Ідентифікатор (a) .src.replace (" 1.png "," 2.png ")

}

};

Зверніть увагу, що document.getElementById (a) .src повторюється тричі, що майже точно відповідає оригіналу (за винятком імені змінної). Коли одне і те ж властивість об’єкта використовується часто, найкращою стратегією є зберігання значення властивості в локальній змінній, а потім використання локальної змінної. Попередній код можна переписати як:

функція toggleImage (id) {

var image = document.getElementById (id);

якщо (image.src.indexOf ("1.png")> -1) {

image.src = image.src.replace ("1.png", "2.png");

}

}

Ця версія коду не тільки менша за попередню (185 байт), але і зменшена версія ще менше:

функція toggleImage (b) {

var a = document.getElementById (b);

if (a.src. »indexOf (" 1.png ")> - 1) {

a.src = a.src.replace ("1.png", "2.png")

}

};

Цей код має лише 126 байт - набагато менше вихідного коду і все ще менший за вихідний мініфікований код. Ця економія відбувається без будь-яких змін у функціональних можливостях, лише трохи рефакторингу. Варто зазначити, що ця зміна також покращує продуктивність роботитому що document.getElementById () не потрібно телефонувати тричі.

Ключові слова

Переглядаючи будь-який великий файл JavaScript, ви помітите, що використовується багато ключових слів. Такі ключові слова, як var, if, for та return, регулярно використовуються для створення бажаної функціональності. Проблема полягає в тому, що цими ключовими словами можна зловживати і як такі впливати на зменшений розмір файлу. Двома найбільш вживаними ключовими словами є var і return.

Оператор var визначає одну або кілька змінних. Іноді ви побачите код, який виглядає так:

var image = document.getElementById ("myImage");

var div = document.getElementById ("myDiv");

Цей код визначає дві змінні, одну за одною. Я часто бачу цей шаблон для 10 або більше змінних поспіль. Це штучно завищує розмір вашого коду, оскільки кілька операторів var можна об’єднати в один за допомогою оператора коми:

var image = document.getElementById ("myImage"),

div = document.getElementById ("myDiv");

Цей код також визначає дві змінні та забезпечує однакову ініціалізацію. Однак ви зберегли три байти, які коштував би інший var. Три байти можуть здатися не великою справою, але якщо ви зможете знайти десятки місць, де в даний час є додаткова заява var, економія може насправді збільшитися. Найкраща порада - використовувати оператор onevar на початку кожної функції, щоб визначити всі змінні, що використовуються у цій функції.

Друге твердження, return, зазвичай отримує зловживання таким чином:

```
функція getValueFor (дані) {
if (firstCondition (data)) {
повернути 1;
}
else if (secondCondition (data)) {
повернення 2; }
else if (thirdCondition (data)) {
повернення 3;
}
ще {
повернення 4;
}
}
```

Зрештою, ця функція повертає значення на основі оцінки деяких умов. Зменшена версія - 146 байт. Ви можете зберегти деякі байти, використовуючи один оператор return:

```
функція getValueFor (дані) {
значення var; if (firstCondition (data)) {
значення = 1;
}
else if (secondCondition (data)) {
значення = 2;
```

```
}
else if (thirdCondition (data)) {
значення = 3;
}
ще {
значення = 4;
}
повернене значення;
                    }
```

Переписаний код замінює більшість випадків повернення локальною змінною, яку можна зменшити. Зменшена версія цього коду - 140 байт. Код може бути реконструйований ще більше:

```
функція getValueFor (дані) {
значення var = 4;
if (firstCondition (data)) {
значення = 1;
}
else if (secondCondition (data)) {
значення = 2;
}
else if (thirdCondition (data)) {
значення = 3;
}
повернене значення;
}
```

Цей код зменшується до 133 байт і дає той самий результат. Зверніть увагу, що ця версія також видаляє додаткове ключове слово, інакше, що допомагає зробити його ще меншим.

Використання одного оператора return для кожної функції зменшує розмір функцій при зменшенні.

Глобальні змінні

Як вже згадувалося раніше, заміна імені змінної відбувається лише на локальні змінні. Спроба замінити імена глобальних змінних, включаючи глобальні імена функцій, може призвести до непрацездатного коду, оскільки YUI Compressor не може знати, де ще ці змінні та функції можуть використовуватися. Це стосується як попередньо визначених глобальних систем, таких як вікно та документ, так і глобальних змінних, які ви створюєте.

Якщо вам потрібно більше разів використовувати глобальну змінну у функції, найкраще зберегти цю глобальну змінну в локальній змінній. Це має дві важливі переваги: Швидший доступ до локальних змінних під час виконання та після збереження в локальній змінній, YUI Compressor може замінити ім'я змінної. Наприклад:

```
функція createMessageElement (повідомлення) {
var div = document.createElement ("div");
div.innerHTML = повідомлення;
document.body.appendChild (div);
}
```

Ця функція має дві локальні змінні, message і div, а також одну глобальну змінну, документ. Подивіться, що відбувається, коли код зменшується:

```
функція createMessageElement (a) {
var b = document.createElement ("div"); »B.innerHTML = a;
document.body.appendChild (b)
};
```

Видно, що документ двічі відображається у зменшеному коді. Ви зберегли 43 байти (158 для оригіналу, 115 зменшених), що непогано, але може бути і краще. Зберігаючи документ у локальній змінній, ви можете заощадити ще більше. Ось переписаний код:

функція createMessageElement (повідомлення) {

var doc = документ,

div = doc.createElement ("div");

div.innerHTML = повідомлення;

doc.body.appendChild (div);

}

Зменшена версія цього коду є такою:

функція createMessageElement (a) {

varb = документ, c = b.createElement »(" div ");

c.innerHTML = a;

b.body.appendChild (c)

};

Знову ж таки, навіть незважаючи на те, що вихідний код трохи більший (175 байт), зменшений код насправді менший (110 байт). Заощадження в цьому випадку становлять додаткові п’ять байт, що може здатися не дуже великим, але ця функція використовує документ лише двічі; якби його застосовували частіше, економія була б більшою.

Глобальні змінні

YUI Compressor не може замінити жодних глобальних систем, включаючи ті, які ви визначили самі. З цієї причини найкраще мінімізувати кількість глобальних змінних та функцій, які ви вводите (це також вважається найкращою практикою дляремонтопридатність, так само). Розглянемо наступний приклад:

var helloMessage = "Привіт світ!";

функція displayMessage () {

попередження (helloMessage);

}

displayMessage ();

У цьому коді і message, і displayMessage () є глобальними, тому їхні імена не можуть бути замінені. Отриманий мініфікований код такий:

```
var helloMessage = "Привіт світ!";
функція displayMessage () {
попередження »(helloMessage)
}
displayMessage ();
```

Оскільки вихідний код - 113 байт, стислий - 95 байт; не величезна економія. У більшості випадків для виконання завдання вам насправді не потрібні глобальні змінні. Наприклад, цей код працює однаково добре, коли і helloMessage, idisplayMessage () є локальними змінними. Ви можете створити всі змінні та функції в заданому блоці коду, побудувавши навколо нього самовиконуючуся анонімну функцію.

Цей код створює функцію і виконує її негайно. Це схоже на це:

```
функція doSomething () {
// код тут
}
робити щось();
```

Оскільки функція буде викликана лише один раз, ви можете зберегти байти, виключивши ім'я. Самовиконання - це найшвидший спосіб створити функцію та виконати її рівно один раз. Також дуже легко включити вже існуючий код всередину самовиконуючоїся функції. Ось як:

```
(функція () {
var helloMessage = "Привіт світ!";
функція displayMessage () {
попередження (helloMessage);
}
displayMessage ();}) ();
```

Усередині функції самовиконання helloMessage та displayMessage () є локальними. Зменшена версія цього коду є такою:

```
(function () {var b = "Привіт світ!";
```

функція a () {alert (b)} a ()}) ();

Зверніть увагу, що навіть незважаючи на те, що оригінальний код дещо більший, зважуючи 154 байти, зменшена версія складає лише 63 байти (на 32 байти менше, ніж оригінальний мініфікований код).

Висновки третьої частини

Оптимізація веб-сайтів означає оптимізацію означає оптимізацію вмісту, URL-адрес, дизайну, стилів та програмного коду для отримання швидкого та якісного веб-сайту для залучення відвідувачів.

Оптимізація пошукової системи (SEO) - це процес впливу на видимість веб-сайту або веб-сторінки в неоплачених результатах пошукової системи - їх часто називають «природними», «органічними» або «заробленими» результатами. Загалом, чим раніше (або вище рейтинг на сторінці результатів пошуку), і чим частіше сайт з'являється у списку результатів пошуку, тим більше відвідувачів він отримає від користувачів пошукової системи. SEO може націлювати різні типи пошуку, включаючи пошук зображень, локальний пошук, пошук відео, академічний пошук, пошук новин та галузеві вертикальні пошукові системи.

SEO є важливою та важливою частиною веб-розробки, і це потрібно для того, щоб врахувати це під час створення веб-сайту.

Мініфікація (також мінімізація або мінімізація) в мовах комп'ютерного програмування і особливо JavaScript - це процес видалення всіх непотрібних символів із вихідного коду без зміни його функціональних можливостей. Ці непотрібні символи зазвичай включають пробіли, символи нового рядка, коментарі, а іноді і роздільники блоків, які використовуються для додавання читабельності коду, але не потрібні для його виконання.

Зменшений вихідний код особливо корисний для інтерпретованих мов, розгорнутих та переданих в Інтернеті (наприклад, JavaScript), оскільки зменшує обсяг даних, які потрібно передати.

Метою YUI Compressor було зменшити файли JavaScript, застосовуючи розумні оптимізації до вихідного коду. Окрім видалення коментарів, пробілів та вкладок, YUI Compressor також безпечно видаляє розриви рядків, додатково зменшуючи загальний розмір файлу. Однак найбільша економія байтів відбувається за рахунок заміни імен локальних змінних на імена з одним або двома символами.

Мініфікація працює, аналізуючи та переписуючи текстові частини веб-сайту, щоб зменшити загальний розмір файлу. Мініфікація поширюється на сценарії, таблиці стилів та інші компоненти, які веб-браузер використовує для візуалізації сайту.

ВИСНОВКИ

Веб-технологія - це розробка механізму, який дозволяє комунікаціям ще двох комп'ютерних пристроїв через мережу. Веб-технології зробили революцію в методах спілкування та зробили ці операції набагато ефективнішими. Головною перевагою веб-технологій є те, що вона пропонує зручність та високу швидкість спілкування в комп'ютерному світі.

Веб-браузери та веб-сервери разом функціонують як система клієнт-сервер. У комп'ютерних мережах клієнт-сервер є стандартним методом для розробки додатків, де дані зберігаються в центральних місцях та ефективно обмінюються з будь-якою кількістю інших комп'ютерів (клієнтів) за запитом. Веб-браузери функціонують як клієнти, які запитують інформацію з веб-сайтів.

Протокол передачі гіпертексту (HTTP) - це прикладний протокол для розподілених спільних інформаційних систем гіпермедіа. Це основа передачі даних для Всесвітньої павутини (тобто Інтернету) з 1990 року. HTTP - це загальний протокол без громадянства, який може використовуватися для інших цілей, а також за допомогою розширень методів запитів, кодів помилок та заголовків. HTTP є базовим протоколом у розробці Інтернету.

Браузер - це програма, яка надає можливість переглянути та взаємодіяти з усією інформацією у Всесвітній павутині. Технічно веб-браузер використовує HTTP для надсилання запитів веб-серверів в Інтернеті від імені користувача браузера. Отже, веб-браузер - це програмний додаток, що дозволяє переглядати сторінки у Всесвітній павутині.

Клієнтські сценарії, як правило, відносяться до класу комп'ютерних програм в Інтернеті, які виконуються на стороні клієнта веб-браузером користувача, а не на стороні сервера (на веб-сервері).

Серверні сценарії - це техніка, що використовується при веб-розробці, яка передбачає використання сценаріїв на веб-сервері, які дають відповідь,

налаштовану на запит кожного користувача (клієнта) до веб-сайту. Альтернатива - веб-сервер сам доставляє статичну веб-сторінку. Сценарії можна писати будь-якою з багатьох доступних мов сценаріїв на стороні сервера.

Веб-дизайн включає архітектуру інформації, структуру веб-сайту, користувальницький інтерфейс, ергономіку навігації, макет веб-сайту, кольори, контрасти, шрифти, зображення тощо. Оскільки з ростом веб-технологій розробка та підтримка хорошого веб-сайту повинна здійснюватися протягом усього періоду його робота щодо відповідності сучасним вимогам та залучення нових користувачів.

Головною метою створення веб-сайту є розробка успішного та популярного веб-сайту з великою аудиторією. Для досягнення цієї мети веб-сайт повинен відповідати сучасним вимогам, таким як унікальний вміст, придатні та інформативні візуальні дані, використання сучасних технологій, швидка швидкість доставки вмісту веб-сайту, високомотивоване та зрозуміле пояснення призначення веб-сайту.

Для утримання нового відвідувача та залучення нових користувачів веб-сайт повинен відповідати вимогам юзабіліті, наприклад, привласнювати його для будь-якого віку та статі та стабільну роботу на всіх пристроях із простим та зручним інтерфейсом.

Щоб створити зручний і зрозумілий веб-сайт, дизайнери та розробники повинні взяти

зручність користування обліковим записом. Юзабіліті - це атрибут якості, який оцінює наскільки легкий користувач

використовувати інтерфейси. Слово "зручність використання" також відноситься до методів поліпшення простоти використання під час проектування.

Графіка на сучасному веб-сайті дійсно потрібна, але повинна бути сильно оптимізована. Векторна графіка підходить для основного дизайну, наприклад, піктограм навігації, логотипів тощо. Зображення вмісту повинні бути оптимізовані, щоб не витрачати величезну кількість трафіку даних. Для цього

можуть бути використані обмежувальні розміри, масштабування зображень, якісна техніка стиснення.

Зі зростанням популярності смартфонів, планшетів та інших мобільних пристроїв з'явилася концепція адаптивного веб-дизайну. Адаптивний веб-дизайн - це стратегія надання користувацьким макетам пристроїв залежно від розміру вікна браузера. Це підхід до розробки веб-сайту, який має на меті надати користувачам веб-сайту хороший досвід незалежно від використовуваного браузера, пристрою чи розміру екрана. Для використання нових функцій браузера розробники повинні використовувати принципи поступового вдосконалення або витонченої деградації. Прогресивне вдосконалення - це спосіб проектування веб-сторінок, так що чим більше функцій підтримує користувальницький агент, тим більше функцій буде мати веб-сторінка. А спочатку створюючи сторінки для найсучасніших браузерів, а потім перетворюючи їх на роботу з менш функціональними браузерами.

Оптимізація веб-сайтів означає оптимізацію означає оптимізацію вмісту, URL-адрес, дизайну, стилів та програмного коду для отримання швидкого та якісного веб-сайту для залучення відвідувачів.

Оптимізація пошукової системи (SEO) - це процес впливу на видимість веб-сайту або веб-сторінки в неоплачених результатах пошукової системи - їх часто називають «природними», «органічними» або «зароблениими» результатами. Загалом, чим раніше (або вище рейтинг на сторінці результатів пошуку), і чим частіше сайт з'являється у списку результатів пошуку, тим більше відвідувачів він отримає від користувачів пошукової системи. SEO може націлювати різні типи пошуку, включаючи пошук зображень, локальний пошук, пошук відео, академічний пошук, пошук новин та галузеві вертикальні пошукові системи.

SEO є важливою та важливою частиною веб-розробки, і це потрібно для того, щоб врахувати це під час створення веб-сайту.

Мініфікація (також мінімізація або мінімізація) в мовах комп’ютерного програмування і особливо JavaScript - це процес видалення всіх непотрібних символів із вихідного коду без зміни його функціональних можливостей. Ці непотрібні символи зазвичай включають пробіли, символи нового рядка, коментарі, а іноді і роздільники блоків, які використовуються для додавання читабельності коду, але не потрібні для його виконання.

Зменшений вихідний код особливо корисний для інтерпретованих мов, розгорнутих та переданих в Інтернеті (наприклад, JavaScript), оскільки зменшує обсяг даних, які потрібно передати.

Метою YUI Compressor було зменшити файли JavaScript, застосовуючи розумні оптимізації до вихідного коду. Окрім видалення коментарів, пробілів та вкладок, YUI Compressor також безпечно видаляє розриви рядків, додатково зменшуючи загальний розмір файлу. Однак найбільша економія байтів відбувається за рахунок заміни імен локальних змінних на імена з одним або двома символами.

Мініфікація працює, аналізуючи та переписуючи текстові частини веб-сайту, щоб зменшити загальний розмір файлу. Мініфікація поширюється на сценарії, таблиці стилів та інші компоненти, які веб-браузер використовує для візуалізації сайту.