**Анализ на методите за оптимизация**

Развитието на глобалната мрежа води след себе си множество промени както в процеса на изграждане на уеб страниците, така и в този на оптимизация. Създават се нови методи, стари променят ефекта си, а някои отпадат, в резултат на излизането на дадена технология от употреба. Анализът на съществуващите методи помага за тяхното разбиране и ефективно прилагане. За да се оцени ефектът от прилагането на методите за оптимизация е необходимо познаване на факторите, определящи ефективността на уеб сайтовете. Тази глава разглежда различните методи, използвани в процеса на разработка и оптимизация на съвременни уеб страници и приложения, както и характеристиките, описващи техния успех.

1. **Метрични и неметрични характеристики на уеб сайта.**

Преди да се разгледа подробно оптимизацията на уеб сайтовете, трябва да се дефинира един или повече показатели, които определят ефективността на дадена страница и съответно ефектът на извършената оптимизация. Тези характеристики могат да се разделят на два основни вида - метрични и неметрични. Примери за метрични характеристики са брой на уникалните посетители, брой на заредените страници, време за зареждане и позициониране в търсачките. Неметричните показатели са свързани с това дали страниците са достъпни за хора с увреждания, дали сайтът може да се разглежда на мобилни устройства, цветова гама, възприемане на дизайна от потребителите, потребителско преживяване и др. Двата типа характеристики са взаимно свързани и промяната на характеристики от единия тип може да доведе до промяна на тези от другия тип. Пример за това е, че подобряването на скоростта на зарежданетоподобрява потребителското преживяване, но един не добре обмислен дизайн или информационна архитектура на същия сайт може да отблъсне потребителите и да доведе до спад в броя на посещенията.

Тъй като стойността на един сайт се определя от неговите посетители, като основни метрични показатели за ефективността използвани в практиката са приети:

- Брой уникални посетители - отразява броя на уникалните посетители на сайта

- Брой страници на посещение - дава информация за количеството страници, които потребителите преглеждат в рамките на едно посещение

- Степен на напускане - показва процента потребители, напуснали сайта без навигация към други страници

- Продължителност на престоя – времето, което посетителите прекарват на сайта

- Степен на конверсия - показва отношението на постигнатите цели(продажби, регистрация на нови потребители, извършване на определени действия и др.) към общия брой уникални посещения.

В по-ранните години на развитие на Мрежата на броя посетители се гледа като основен фактор за успеха на сайтовете. Погледнато по-общо ефективността на сайтовете нараства с броя на посетителите и продължителността на посещенията, които от своя страна увеличават броя на постигнатите цели. В зависимост от водещата метрика тук могат да се разглеждат два типа цели наоптимизация - повишаване на трафика към сайта иповишаване на степента на конверсия.

Разпространена практика е фокусът да пада първо върху подобряването на конверсията и след това да се работи върху увеличаването на броя посетители. Особеното в случая е, че при сайтове с недостатъчен трафик подходът е обратен, като се цели повишаване на броя посетители с цел получаване на по-ясни резултати за ефективността на оптимизацията на конверсията чрез АБ тестване.

1. **Оптимизация на търсещите машини (SEO)**
   1. **История**

Процесът на оптимизация за търсещи машини възниква в средата на 90-те години на 20-ти век като резултат от създаването на първите интернет търсачки. Скоро след тяхното създаване администраторите на интернет сайтовете започват да осъзнават ползите от доброто позициониране в страниците с резултати и започват работа в тази насока. Същевременно възникват и множество интернет каталози и регистрацията на сайтовете в тях бързо се превръща в част от процеса на оптимизация. В този момент търсачката на Yahoo доминира пазара и става основна цел при оптимизация. Заедно с развитието на търсачките започват да се появяват все по-големи количества SPAM (масови количества нежелани съобщения), което значително влошава качеството на информацията.

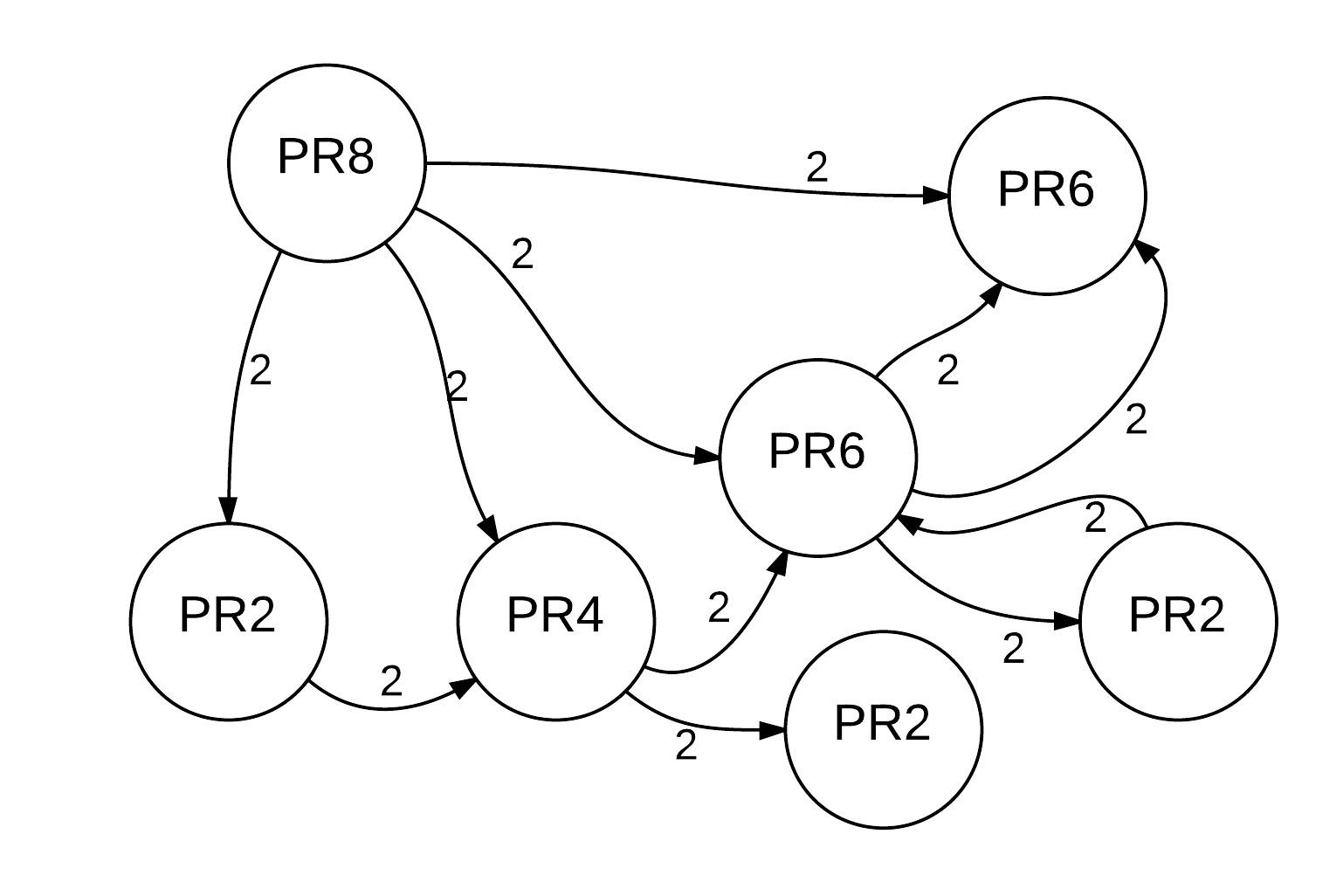
Пред 1998 година биват стартирани търсачката на Гугъл, както и интернет директорията DMOZ. През следващите години оптимизацията за търсещи машини претърпява значително развитие.

През 2001 година Гугъл представят PageRank - нов алгоритъм, който използват при позиционирането на резултатите от търсене. При този алгоритъм към всеки линк се присвоява число показващо относителната важност на сайта. Това число се изменя в граници от 0 до 10, като важността нараства в увеличаването на числото. За определяне на ранга на страницата се използва формула:

(2.1),

Изразено с думи рангът на страницата„**u**“[PR(u)] е сума от отношенията на ранга[PR(v)] и броя линкове[L(v)] за всяка страница v, съдържаща връзка към u. Пример, страница А има PR равен на 6 и 3 линка, от които един към страница С. Страница В има PR равен на 4 и 2 линка от които един към страница: **PR(C) = PR(A) / L(A) + PR(B) / L(B) = 6/3 + 4/2 = 4**

Графично формулата може да бъде илюстрирана със следната графика.

***Фиг. 2.1.*** *Графично представяне на преноса на ранг между страниците*

Тук трябва да се отбележи, че рангът на страниците, който Гугъл определя е приблизителен и е резултат от повтарящи се изчисления. За всяка страница се задава дадена начална стойност и след определен брой итерации нейният ранг може да се определи приблизително точно. Според авторите на PageRank алгоритъма - Лари Пейдж и Сергей Брин, 100 итерации са достатъчни да се определи приблизително точно рангът на всички страници в глобалната мрежа.

Благодарение на PageRank и новите алгоритми за позициониране много от използваните до момента SPAM техники биват неутрализирани и търсачката на Гугъл започва да придобива все по-голяма популярност. Оптимизацията за търсачки започва сигурно да се развива, при което се оформят два основни клона от техники - бял и черен. В годините след въвеждането на PageRank алгоритъма множество оптимизатори започват активни кампании, изцяло насочени върху изграждането на връзки с цел повишаване на PageRank-а на страниците, често без оглед на фактори като качество на връзките и др..

Поради множеството опити за манипулация на PageRank през 2004-та година Золтан Гйонгуй, Хектор Гарсия-Молина и Ян Педерсен предлагат алтернативен алгоритъм наречен TrustRank. Тъй като е практически невъзможно всички страници да бъдат преглеждани от хора, които да определят тяхната репутация, се предлага експерти да проверяват малък набор от страници и ръчно да определят репутацията им. След това автоматично се обхождат останалите страници, като тяхната репутация се определя на база приликата и близостта до вече оценените страници. Авторите твърдят, че чрез използване на първоначален набор от 200 страници значителна част от SPAM страниците могат да бъдат филтрирани.

Ариндам Чакрабарти от университета в Бъркли предлага модел, при който връзките между страници имат както положителна, така и отрицателна стойност. Според представеното изследване, чрез използването на този метод се постига по-висока ефективност при определянето на ранга на страниците и филтрирането на тези с „лошо“ съдържание.

През 2009 г. е предложена алтернативна система за определяне на важността на страниците, основана на поведението на потребителите, наречена BrowserRank. Част от параметрите, използвани в новия алгоритъм са броят на посещенията и продължителността на престоя на потребителите.

През 2010 г. в свое видео Мат Кътс съветва оптимизаторите и техните клиенти да не се фокусират върху стойността на PageRank показателя, а върху качеството на съдържанието и потребителското преживяване.

През Април 2013-та година Борис Демария публикува статия, в която разглежда доколко ранга на страниците все още е важен за доброто позициониране. Той започва с твърдението, че страница с PR2 може да се класира на първата страница в резултатите от търсенето, докато такава с PR6 изобщо да не фигурира в тях. Този факт е потвърден и от други експерти в областта като Нейл Пател и Дани Съливан.

Бялата оптимизация (White-hat SEO) представлява съвкупност от методи и техники, които се фокусират върху човешкия фактор и удобството на потребителя, като следват условията и политиките на търсещите машини. Процесът е по-продължителен, но с по-добър дълготраен ефект.

Противно на нея, черната оптимизация (Black-hat SEO) използва методи в разрез с правилата на търсещите машини и води до по-лошо потребителско преживяване. Примери за такива техники са използването на скрити текстове или множество ключови думи в META-тага с цел заблуждаване на търсачките и по-високо класиране на сайта в резултатите от търсене. Макар и с доста по-бърз видим ефект подобни техники са непрепоръчителни не само от етична гледна точка, но и защото често могат да доведат до наказания от страна на търсещите машини и деиндексация на целия сайт. Като допълнителен фактор срещу употребата на забранени техники Мат Кътс изтъква и факта, че Гугъл налагат по-строги наказания за потребители, които повторно нарушават правилата на търсачката.

През 2005 година Мат Кътс от Гугъл и Джейсън Шелен от Блогър представят идея за нов атрибут към линковете, който да ограничи спама, генериран чрез блог коментари. Това е процес, при който в множество блогове изкуствено се публикуват коментари с цел повишаване на PageRank на определен сайт, който е даден като линк към коментара. Представения атрибут се казва **nofollow**, в буквален превод означава „не-следвай“ и указва на търсещите машини, че не трябва да се прехвърля PageRank към страницата, към която сочи линка.

Четири години по-късно през 2009-та година Гугъл представят нов търсещ алгоритъм наречен Caffeine (Кофеин), като подобренията са в две основни насоки:

* + Скорост - времето за връщане на резултати е подобрено близо 2 пъти
  + Точност - залага се на използването на цели изрази, а не само на отделните ключови думи

Заедно с това се използва и анализ на допълнителни информационни източници (прес - съобщения, изображения, видео файлове, новини и др.) при генерирането на резултатите от търсенето.

Следващите големи промени в търсачката на Гугъл са така наречените Панда и Пингвин обновявания. Според Огнян Младенов от СЕОМ близо 40% от сайтовете в Топ 10 за България са пострадали от въпросните ъпдейти в периода Април - Юли 2012 година. Панда ъпдейтът цели намаляване на ранга на сайтове с лошо качество и подобряване при новинарски сайтове и социални мрежи. Смята се, че промяната е дала отражение върху 12% от всички резултати от търсене, като основен негативен ефект има върху сайтове, съдържащи големи количества реклама.

След Панда ъпдейта следва Пингвин, първоначално наричан “Алгоритъма Уебспам” (The Webspam Algorithm). Той от своя страна цели понижаване на ранга на сайтове, нарушаващи правилата, дефинирани в „Инструментите на Гугъл за администратори на уеб сайтове“ (Google Webmaster Tools). По изчисления на Гугъл обновлението в алгоритъма е засегнало “3,1% от всички търсения на английски, около 3 процента при други езици, като немски, китайски и арабски и дори по-висок процент при езици с по-голямо количество SPAM”. За пример дават промените в търсенето на полски, където 5% от резултатите са променени до степен това да бъде забелязано от потребителите на търсачката. Тъй като тези алгоритми са фирмена тайна на Гугъл, точният им принцип на работа не е известен. Данните за техния ефект и факторите, които отчитат са резултат на наблюдения и информация, подадена от самата компания.

Заедно с тези ъпдейти са появяват и наказанията за т. нар. „неестествени“ връзки. Неестествените връзки са изкуствени, заблуждаващи или манипулиращи такива, сочещи от или към даден уебсайт. Те влошават потребителското преживяване и водят до увреждане на интернет средата. Примери за такива линкове са линкове, закупени от други сайтове, както и линкове от статии с „лошо“ или плагиатствано съдържание. Под „лошо“ съдържание тук се има предвид статии с ниско качество на съдържанието, включващо неприемливи и обидни изрази, а също така и съдържание несвързано с темата.

В свое видео Мат Кътс от Гугъл дава насоки за работата на алгоритъма, който Гугъл използват при представяне на резултатите от дадено търсене. При определяне на позицията на дадена страница в резултатите алгоритъмът взима под внимание повече от 200 различни фактори, сред които:

* брой на срещанията на търсените ключови думи
* дали думите се срещат в заглавието на страницата
* дали думите се срещат в URL-a на страницата
* при търсене на няколко ключови думи се взима под внимание дали документът съдържа точния израз
* дали страницата е от сайт, предлагащ качествено съдържание или не
* PageRank - фактора на страницата

В своя презентация Ранд Фишкин представя промените в тежестта на позициониращите фактори, сравнявайки 2 свои изследвания, направени през 2009-та и 2011-та година.

През 2009-та година петте най-важни позициониращи фактора са:

* ключови думи в текста на външни връзки
* количество и качество на външните връзки
* разнообразие на външните връзки - брой на уникалните домейни
* използване на ключовите думи в заглавния таг на страницата
* авторитет на домейна

Влиянието на отделните групи фактори в алгоритъма е представено в следните отношения:

* 43% Връзки към страницата - брой на връзките и разнообразие на домейните
* 24% Авторитет на домейна и външните линкове
* 15% Използване на ключовите думи в страницата
* 7% Фактори, несвързани с ключовите думи на ниво страница
* 6% Социални фактори за съответната страница
* 5% Фактори, несвързани с ключовите думи на ниво домейн – уникалност, възраст на съдържанието и др.

Изследването направено през 2011-та година, показва еволюцията на алгоритъма за позициониране и промяната на важността на отделните фактори:

* 22% Връзки към страницата - брой на връзките и разнообразие на домейните
* 21% Авторитет на домейна и външните линкове
* 14% Използване на ключовите думи в страницата
* 11% Използване на ключовите думи в сайта
* 7% Социални фактори за съответната страница
* 7% Фактори, свързани с марката - цялостен трафик и авторитет
* 7% Фактори, несвързани с ключовите думи на ниво страница
* 6% Трафик към съответната страница
* 5% Фактори, несвързани с ключовите думи на ниво домейн - уникалност и възраст на съдържанието и др.

Наблюдава се тенденция за увеличаване на влиянието на социалните фактори, разпознаваемостта и авторитета на марката (сайта), както и цялостното използване на ключовите думи в сайта, а не само в специфичната страница. От своя страна броят и качеството на външните връзки намаляват значението си близо двойно.

Повишаването на значимостта на трафика към сайта за неговото позициониране показва, че посетителите не само трябва да достигнат сайта, но и да желаят да го посетят отново. Тук важен фактор е степента на преобразуване - отношението между посетителите и тези от тях извършили определено действие: регистрация, закупуване на продукт, коментиране към статия и т.н.

Оптимизацията може да бъде разделената на вътрешна, при която промените се извършва директно върху сайта и външна. Един от основните методи за външна оптимизация на страниците е включването на сайта в различни списъци или интернет директории. Регистрирането на карта на сайта в търсачките спомага за по-бързото намиране и по-пълно обхождане на страниците. Създаване на общества в социалните мрежи и взаимодействието с тях създава по-добра разпознаваемост както на марката, така и на самия сайт. Друг ефективен подход е създаването на блог и популяризирането на марката или продуктите на компанията през него. Представянето чрез сайта на важна информация за индустрията или сферата на работа на компанията спомага както за неговото популяризирането, така и за това на марката като цяло. Използването на различни рекламни кампании за промоциране на сайта е друг ефективен метод за достигане до конкретна група потребители.

От друга страна оптимизацията, която се извършва върху самия сайт може да се раздели на глобална за сайта (site-wide) и специфична за отделните страници (on-page).

1. **Глобална оптимизация**

Глобалната оптимизация включва методи рефлектиращи върху цялостната структура и работа на сайта. Пример за такива методи са:

* Структура на URL адресите
* Достъпност
* Карти на сайта
* Използване на RSS feed

Структурата на URL адресите включва елементи като използване на www представка, описателни изрази в адреса, вид на разделителя между думите, домейн и др.

Използването на www представка няма реален SEO ефект, т.е. изборът на вариант е изцяло в ръцете на администратора на сайта. Все пак се препоръчва да се избере предпочитан домейн (example.com или [www.example.com](http://www.example.com)) и другият такъв да се пренасочи с помощта на постоянно пренасочване (301 Redirect). Алтернатива на използването на пренасочване е използването на канонични адреси. Чрез добавяне на rel=”canonical” към заглавната част на страницата, можем да укажем на търсещите машини, че дадена група страници съдържат приблизително идентично съдържание и да се укаже коя от тях е с максимална стойност. По този начин се цели минимизиране на шанса търсещите машини да интерпретират двете версии като отделни сайтове, а това може до доведе до наказание за копирано съдържание или по-лошо позициониране в резултатите от търсене. Това може да се случи, защото при използването на двете версии на адресите рангът на страниците (PageRank) се разпределя между тях. При използване на постоянно пренасочване то се интерпретира от търсещите машини, като указание, че страницата е преместена на друг адрес и той трябва да се интерпретира като основен.

Използване на описателни URL адреси е друг метод за SEO оптимизация. Гугъл препоръчват използването на URL адреси, които са прости, но и описват съдържанието на уеб страницата. “A site's URL structure should be as simple as possible. Consider organizing your content so that URLs are constructed logically and in a manner that is most intelligible to humans (when possible, readable words rather than long ID numbers).”.

В превод - “Структурата на URL адресите трябва да е възможно най-проста. Обмислете организацията на съдържанието си, така че адресите на страниците да са логически структурирани и възможно най-разбираеми за хората (при възможност използвайте четливи думи вместо идентификационни номера). Пример за това е заменянето на адреси като *“/index.php?id=9DddF4453BC05d80DF11D3ACB60090271E26A8&c=abc”* със по-краткия и информативен *“/blog/decorators-in-python-3”*.

Също така се препоръчва използването на тирета като разделители на думите в URL-адреса, тъй като някои търсещи машини интерпретират използване на долна черта като знак за свързване на думите.

Използване на локален домейн от най-високо ниво (TopLevelDomain) при таргетиране в конкретен регион е важен фактор за търсещите машини за създаване на връзка между отделен сайт и дадена географска локация. Още през 2005-та година Гугъл добавят в списъка с насоки към уеб разработчиците, препоръката за използване на локални домейни от най-високо ниво. Част от алгоритмите на търсещите машини се грижи за това потребителите да получават не само информация, отговаряща на ключовите думи в търсенето им, но и по-възможност свързана с географското местоположение на потребителя. Уикипедия предоставя пълен списък на всички домейни от най-високо ниво, съдържащи кода на страната, която представляват. В случай на използване на домейн от най-високо ниво, който не носи географска информация, Гугъл препоръчват използването на техния набор инструменти за администратори на уеб сайтове, където може да се зададе географска област, която да се използва за съответния домейн. Примери за такива домейна са .bg, .co.uk, .in и т.н.

Под-домейни и/или под-папки се използват за отделяне на части от съдържанието на уеб сайта - например могат да се използват различни секции за блог, клиентски профили и т.н. Според Мат Кътс от Гугъл, алгоритъмът за индексиране третира под-домейните (например: *blog.example.com*) и под-папките (например: *example.com/blog*) по еднакъв начин. Поради това изборът между двете се свежда не до SEO ефекта, а до по-удобния за уеб сайт администратора вариант. За разлика от създаването на под-домейни, където се изисква допълнителна работа по създаване на нов DNS запис, използването на под-папки е по-простият вариант от гледна точка на изпълнение, но има своите недостатъци. Например при този тип разделение не може да се използва друг сървър с цел разпределение на натоварването, както и може да има технологични ограничения, т.е. сървърът да не поддържа платформата, използвана за новото съдържание.

Когато се говори за оптимизация за търсещи машини под термина достъпност се разбира осигуряване на лесен и удобен достъп на търсещите машини до страниците на сайта. За целта администраторите на уеб страници могат да използва файла robots.txt, който служи като универсален идентификатор за търсещите машини. В него се поставят инструкции, които указват дали и кои части от сайта да бъдат индексирани. Забраната за индексиране се използва с цел отделни страници да не попадат в резултатите от търсене. Примери за такива страница са административни панели, секции от сайта с ограничен достъп или такива в процес на обработка. Важен е фактът, че наличието на robots.txt не ограничава достъпа на търсачките до сайта. Макар инструкциите, зададени в robots.txt да се спазват от основните търсещи машини, съществуват и такива, които не зачитат тези указания. Допълнителен елемент от достъпността на страниците е използването на указатели за страниране. Тъй като роботите, които обхождат сайтовете невинаги могат коректно да идентифицират връзките, е възможно те да не бъдат обходени или съдържанието в тях да се интерпретира като дублиращо това на други страници. За да се избегнат тези проблеми, през 2011 година, Гугъл представят два нови атрибута - “rel=”next” и “rel=”prev”,които указват съответно следващата и предишната страница при използване на страниране. В своя статия Васил Тошков разглежда внимателно тези атрибути и дава няколко съвета относно тяхното използване. Един от тях е да се използва метатаг robots със стойност noindex, nofollow, като по този начин се инструктират търсещите машини да не индексират преходните страници. С това се цели потребителите, идващи от търсещите машини, винаги да попадат на началната страницата, където са показани най-новите или търсени статии и по този начин да се увеличи степента на конверсия.

Картите на сайта (sitemaps) представляват списъци с връзки към отделните части на страниците. Те бива няколко вида, като най-разпространени са HTML и XML картите, но съществуват и такива за изображения и видео.

HTML и XML картите са страници, представящи структурата на сайта съответно в HTML или XML формат. HTML форматът използва различни графични и стилови елементи, с цел представяне на информацията в по-удобен за потребителите начин, но може да се използва и за търсещите машини. Недостатък тук е, че поради естетически и свързани с удобството на потребителя причини, за сайтове с множество страници е неподходящо представянето на връзки към всички тях в HTML картата. От своя страна XML форматът се използва основно за търсещите машини и поради тази причина обикновено се включват връзки към всички страници от сайта. Стандартният формат за XML карти на сайта е представен на фигура 3.1:

<?xml version="1.0" encoding="utf-8"?>  
<urlset xmlns="<http://www.sitemaps.org/schemas/sitemap/0.9>">  
<url>  
<loc><http://example.com/></loc>  
<lastmod>2006-11-18</lastmod>  
<changefreq>daily</changefreq>  
<priority>0.8</priority>  
</url>  
</urlset>

***Фиг.3.1*** *Пример за XML карта на сайт*

Където :

* **urlset** е основният елемент и означава контейнера на всички връзки в картата
* **url** елементът се използва за отделяне на връзките една от друга
* **loc** съдържа пълния път до страницата, включително и протокола, по който тя да се зареди (http или https)
* **lastmod** е незадължителен таг, който съдържа датата на последната промяна на страница в ISO формат
* **changefreq** - незадължителен таг, показващ колко често страницата може да бъде променяна, валидни стойности са: always, hourly, daily, weekly, monthly, yearly и never
* **priority** - отново незадължителен таг, показващ важността на дадена страница спрямо останалите от сайта. Използва се, за да може да се подскаже на обхождащите машини кои страници са с по-висока важност. Валидните стойности са от 0.0 до 1.0, като 1.0 указва най-висок приоритет.

Тъй като съществува ограничение, че XML картите на сайта не могат да надвишават 10MB или 50,000 връзки, се създават така наречените “индекси на карти на сайта” (sitemap index) – Фиг.3.2. Те съдържат връзки не към страници от сайта, а към отделните карти, всяка съдържаща линковете за определена част от сайта. Тяхната структура изглежда по следния начин:

<sitemapindexxmlns="<http://www.sitemaps.org/schemas/sitemap/0.9>">  
<sitemap>  
<loc>https://www.example.com/sitemap-blog.xml</loc>  
</sitemap>  
<sitemap>  
<loc>https://www.example.com/sitemap-general.xml</loc>  
</sitemap>  
</sitemapindex>

***Фиг.3.2*** *Пример за XML индекс на карти на сайт*

Където:

* **sitemapindex** е основният елемент и означава контейнера на всички връзки към отделните XML карти
* **sitemap** елемента се използва за отделяне на картите на сайта една от друга
* **loc** съдържа пълния път до картата, включително и протокола по който тя да се зареди (http или https)

Индексите на карти се използват не само, когато броят линкове надвишава 50,000, но и за индексиране на страниците на сайтове, включващи множество под-домейни. Причината за ограничението в броя и размера на картите е да се ограничи натоварването на уеб сървърите при изтегляне на картите от търсещите машини.

В свое видео от Октомври 2009-та година, Мат Кътс казва, че в случай на използване на XML карти, Гугъл не гарантират, че всички адреси, описани в картата ще бъдат обходени, но картите им помагат за откриването на нови страници. При използване на HTML карта на сайта, към която има връзка от основния сайт, обхождащата машина ще използва връзките в картата за обхождане на отделните адреси. В случай, че е невъзможно използването и на двата типа карти той препоръчва създаването на HTML такава.

Картите за изображения и видео (image sitemaps и video sitemaps) биват въведени от Гугъл с цел разширяване възможностите на стандартните XML карти и по-добро представяне на видео файловете и изображенията в сайтовете. Въвеждат се нови xml-тагове, които могат да се използват както за създаването на нови карти, така и в съществуващите. Пример за използване на видео тагове е даден на фигура 3.3.

<urlset xmlns=<http://www.sitemaps.org/schemas/sitemap/0.9>  
 xmlns:video="http://www.google.com/schemas/sitemap-video/1.1">  
<url>  
<loc>http://www.example.com/video.html</loc>  
<video:video>  
<video:thumbnail\_loc>http://example.com/th.jpg</video:thumbnail\_loc>  
<video:title>Video Title</video:title>  
<video:description>Video Description</video:description>  
<video:content\_loc>http://example.com/video.flv</video:content\_loc>  
<video:duration>600</video:duration>  
</video:video>  
</url>  
</urlset>

***Фиг.3.3*** *Пример за използване на видео тагове*

Където:

* **loc** съдържа пълния път до страницата, на която е разположено видеото
* **video:video** тагът е контейнер за информацията, свързана с видеото
* **video:thumbnail\_loc** – указва пътя до изображението(thumbnail) на видеото
* **video:title** – съдържа заглавието на видеото
* **video:description** – съдържа кратко описание на видеото
* **video:content\_loc** – съдържа пътя към самия видео ресурс
* **video:duration** – продължителността на видеото в секунди

За включване на изображения в картите се използват следните тагове:

* **image:image -** тагът е контейнер за информацията, свързана с изображението
* **image:loc** – указва пътя до изображението
* **image:title** – заглавието на изображението
* **image:caption** – кратък описателен текст към изображението
* **image:geo\_location** – географското местоположение, например къде е заснета снимката <image:geo\_location>Sofia,Bulgaria</image:geo\_location>
* **image:license**– URL адрес към лиценза за използване на изображението

Използването на RSS Feеd е метод за оптимизация, най-често използван в блогове, новинарски сайтове или такива включващи множество статии, които периодично се обновяват. С помощта на XML се описват последните няколко страници, като се включва техният адрес, дата на публикуване, заглавие и кратък отрязък от съдържанието. Потребители могат да се абонират за тези feed-ове чрез специализирани приложения и по този начин да бъдат известявани за новите статии в съответния сайт. Съществуват, и така наречените RSS агрегатори – сайтове които комбинират информацията от множество различни feed-ове с цел предоставяне на обединение от статии с определена насоченост. Съществуват различни версии на RSS формата, както и алтернативи като Atom. Пример за RSS feed eпоказан на фигура 3.4.

<?xmlversion="1.0" encoding="UTF-8" ?>  
<rssversion="2.0">  
<channel>  
<title>Заглавие на сайта/фийда</title>  
<description>Описание</description>  
<link><http://www.example.com/></link>  
<item>  
<title>Заглавие на статия</title>  
<description>Описание на статията</description>  
<link><http://www.example.org/new-article></link>  
<pubDate>Mon, 06 Sep 2009 16:45:00 +0000 </pubDate>  
</item>  
</channel>  
</rss>

***Фиг.3.4*** *Пример за използване на RSS*

Където:

* **channel -** е основният елемент и означава контейнера на всички статии
* **title -** означава заглавието на фийда, обикновено се използва това на сайта. Изключения се правят при използване на няколко фийда за един сайт, например отделни такива за всяка категория
* **description -** описание на фийда/сайта
* **link** - URL адреса на началната страница на сайта/категорията
* **item** - елемента се използва за отделяне на статиите една от друга
* **title** - заглавието на статията
* **description** - кратко описание или откъс от съдържанието на статията
* **link** - съдържа пълния път до статията
* **pubDate** - съдържа датата на публикуване в ISO формат

1. **Специфична за отделните страници оптимизация**

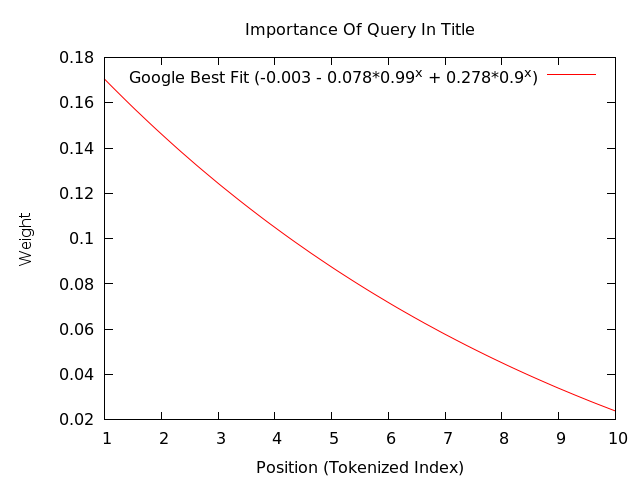
Този тип оптимизация включва методи, променящи специфичните за всяка една страница от уеб сайта параметри. Пример за такива методи са:

* използването на ключови думи
* съдържанието на страницата
* използване на изображения с цел подобряване качеството на съдържанието
* преглед на страниците в социалните мрежи
* представяне на страницата в резултатите от търсене
* оптимизиране за локално търсене

Между експертите в областта съществува разделение по въпроса дали дадена страница трябва да се оптимизира за една или за повече ключови думи. При използване на една ключова дума се улеснява създаването на съдържанието, авторът по-лесно запазва фокуса върху съответната ключова дума или израз и съдържанието става по-релевантно и по-добро като резултат от търсенето на потребителя. Недостатъци тук са липсата или ниският ранг на страницата при търсене за други ключови думи, както и фактът, че конкуренцията при използване на второстепенни изрази може да е значително по-ниска. Също така, за да се постигне цялостна оптимизация на сайта за множество ключови думи трябва да се създадат статии за всяка една от тях. При използване на няколко ключови думи се улеснява създаването на съдържанието и се постига по-добро общо позициониране за всички ключови думи. Недостатък тук е както по-лошото позициониране при търсене на основната ключова дума, така и възможността качеството на съдържанието да намалее заради употребата на множество ключови думи. Експертите препоръчват да се използва една ключова дума, когато се създава съдържание, подлежащо на редакция или подновяване веднъж или повече пъти в седмицата. В противен случай е по-добре използването на няколко ключови думи (изрази), но не повече от 3 за страница. Друг важен етап при избора на ключови думи е проучването, което се провежда преди да се пристъпи към създаване или оптимизация на съществуващи текстове. Продукти като инструмента за планиране на ключови думи на Гугъл наречен KeywordPlanner

имат за цел да изберат думи с най-добро съотношение между трафик и конкуренция. Използване на ключовите думи в заглавието на страницата, URL адреса, заглавните елементи и съдържанието. Препоръчва се използването на ключовата дума в заглавния таг на страницата, като търсещите машини интерпретират позицията и като означение за важността. Например при оптимизация на страница за “потребителски кредити” се препоръчва заглавието да изглежда като “Потребителски кредити от банка АБВ”, а не “Банка АБВ - Потребителски кредити”. Същото важи и за използването на ключовата дума в заглавните тагове(h1, h2, h3 и т.н.) в съдържанието на страницата, както и използването ѝ в първия параграф на текста. Важен фактор за доброто позициониране на страницата е използването на ключовата дума в URL адреса на страницата, пример <http://abc-bank.bg/potrebitelski-krediti>.

Важен елемент от оптимизацията на страницата е избирането на подходящо заглавие. Специалистите в областта препоръчват заглавието на страницата (съдържанието на title тага) да не надвишава 70 символа. Това е така, защото Гугъл ограничават дължината на заглавията в резултатите си от търсене до 70 символа, а останалата част бива изрязвана при показване.Друга причина е, че колкото по-близо е разположена ключовата дума до началото на заглавието, толкова по-силен ефект има това върху позиционирането на страницата при търсене на дадената ключова дума. Изместването на думата по-навътре в заглавието постепенно намалява нейната тежест като фактор за позициониране в резултатите. Значението на позицията е представено графично на фигура 4.1.



***Фиг.4.1.*** *Значение на позицията на ключовата дума в заглавието*

Добра практика е името на сайта/марката да бъде поставено в края на заглавието, за да може ключовите думи да заемат челните места.

Друг важен елемент е създаването на качествено съдържание. Качественото съдържание, предразполага потребителите към коментиране и споделяне, което е важен фактор за доброто популяризиране на страницата. Основните елементи на качественото съдържание са неговата уникалност, да бъде лесно за разбиране и да привлича потребителите. Уникалността на съдържанието спомага за по-доброто позициониране в търсачките и приемане на съдържанието от потребителите. За да бъде дадено съдържание лесно за разбиране и въпреки това интересно, е важно предварително да се направи анализ на аудиторията, тъй като съдържанието, написано за експерти би било трудно за разбиране от посетители без широки познания в областта и обратно. Основен начин за задържане на вниманието на потребителя и повишаване на привлекателността на съдържанието е добавяне на изображения и видео към текста. Важен фактор тук е актуалността на съдържанието и редовното добавяне на ново съдържание към страниците. Добавянето на съдържание или обновяването на старо такова е знак към търсещите машини, за активността на сайта и влияе положително върху позиционирате му в резултатите при търсене. Не съществуват ясно изразени правила за дължината на съдържанието, но се препоръчва то да включва поне 250 думи. По-големия брой думи увеличава броя на комбинациите от ключови думи, на които страницата отговаря, но не трябва да се използват твърде много или излишни думи, тъй като това може да се отрази негативно на потребителското преживяване.

Задължителен елемент на оптимизацията за търсещи машини е осигуряване на „четимост“ на съдържанието от роботите, които обхождат страницата. Сайтове, използващи технологии като Flash, вътрешни рамки (IFrames) и AJAX, често са трудни за индексиране. Основният проблем е, че търсещите машини виждат само една страница, а останалите страници биват недостъпни за тях. Тъй като подобни сайтове стават все по-популярни, паралелно се разработват и решения на проблема. Пример за такова решение е съдържанието, доставяно на потребителя посредством AJAX да притежава реални URL адреси, където може да бъде прочетено и индексирано от обхождащите роботи. Това решението е приложено в социалната мрежа Пинтерест (Pinterest.com), където при отваряне на даден елемент от съдържанието, то се визуализира в модален прозорец, без да се презарежда страницата, но същевременно разполага със собствен URL адрес, на който може да бъде видяно и без помощта на JavaScript.

Гугъл препоръчват употребата на **rel=”alternate”** и **hreflang=”x”** атрибути при указването на линкове с превод на съответното съдържание. Примери за използването им e

<link rel=”alternate” href=”<http://example.com/bg>” hreflang=”bg” />

При предоставяне на преводи на различни документи на потребителя се препоръчва добавянето на rel и hreflang атрибутите като част от заглавните части на отговора на HTTP заявката.

Изображенията, използвани за подобряване качеството на съдържанието също трябва да бъдат подложени на оптимизация, която се разделя в две основни направления - подобряване на времето за зареждане и оптимизация за търсещите машини. По-кратко време за зареждане се постига чрез намаляване на размера на растера и качеството на изображението, както и при използване на подходящ формат. Препоръчва се при визуализиране на изображението като част от текста, да се използват така наречените тъмбнейлс(thumbnails) - умалени реплики на оригиналното изображение. Така се постига по-бързо зареждане на страницата и заедно с това може да се предостави възможност на потребителя да изтегли изображението в реалния му размер.

Като пример е разгледано изображение в JPEG формат с размери 1920x1200 пиксела и размер на файла 556,6 KB. Намалявайки размера на изображението пропорционално до 300х188 пиксела, се получава файл с големина 89 KB при 100 единици качество на изображението. Чрез намаляване на качеството на изображението от 100 на 60 единици размерът на файла се редуцира до 12,5KB с минимална видима загуба на качеството(таблица 4.1). По този начин постигаме намаляване на размера на файла повече от 44 пъти.

|  |  |
| --- | --- |
| 100 | 60 |
|  |  |

***Таблица 4.1*** *Сравнения на изображения с различно качество*

Оптимизацията за търсещи машини се състои в задаването на значещи имена на файловете и кратък описателен текст в alt атрибута на изображението. По този начин търсещите машини разбират какво е съдържанието на изображението. Например кода <img src=”/static/images/1.jpg” alt=”Image” />може да се оптимизира с добавяне на значещо заглавие, като след оптимизация получава следния код <img src=”/static/images/seo-optimization-process.jpg” alt=”SEO optimization process” />.

При споделяне на страница в социални мрежи като Фейсбук, Туитър и Гугъл Плюс, се препоръчва потребителят да вижда не само адреса на споделената страница, но и допълнителна информация за нея - изображение, заглавие, описание и т.н. Това се нарича преглед на страницата в социалните мрежи или „social snippets“ и спомага за повишаване на трафика от социaлни мрежи. Пример за такъв преглед е показан на фиг. 4.2.



***Фиг.4.2.*** *Пример за преглед на страница в социалната мрежа Facebook*

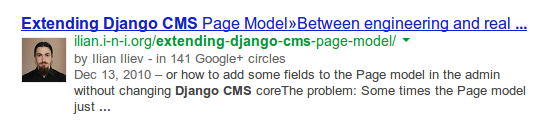
Изображението представя възможностите при споделяне на страница, използваща тагове за преглед в социалната мрежа Фейсбук. Тук отчетливо се виждат заглавието, адресът на страница и кратко описание, като се дава възможност на потребителя, който споделя връзката, да избере между няколко изображения или да сподели страницата без изображение.

Различните социални мрежи предлагат различни набори от тагове, с които се указват желаните от автора на страницата заглавие, описание и изображение, които да се визуализират при споделяне. Най-разпространен е Open Graph протокола на Фейсбук, който дефинира набор от мета тагове за указване на отделните елементи от прегледа на страницата.

* За дефиниране на предпочитаното за показване изображение: се използва   
  *<meta property="og:image" content="http://example.com/2012.jpg" />*
* Задаване на заглавието на страницата  
  *<meta property="og:title" content="Word Wide Web stats for 2012" />*
* Задаване на описанието на страницата  
  *<meta property="og:description" content="Description for the article." />*
* Дефиниране на типа на съдържанието. Ако такъв не е указван по подразбиране, се използва стойност website. Други възможни типове са: article, book, music, video and profile.  
  *<meta property="og:type" content="article" />*
* Задаване на URL адрес на страницата. По подразбиране се използва URL адресът, който потребителят е споделил, но може да се укаже и алтернативен такъв. Този таг най-често се използва с цел проследяване на трафика от социалните мрежи. Пример:  
  *<meta property="og:url" content="http://www.example.com/www-stats-2012?utm\_source=facebook&utm\_medium=social&utm\_campaign=www-stats-2012" />*

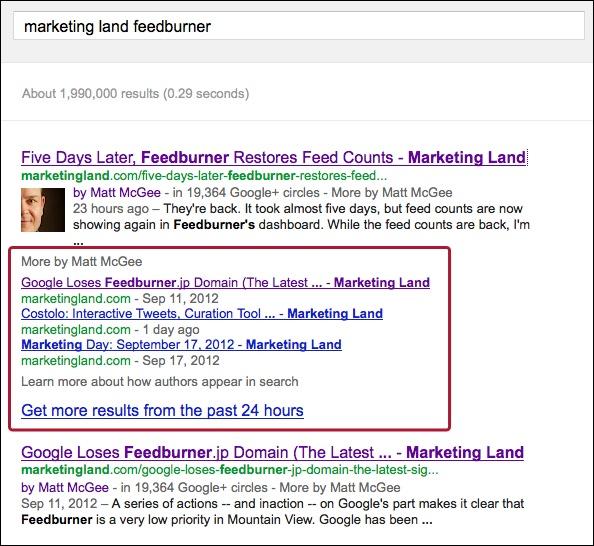
Тук utm\_source, utm\_medium и utm\_campaign са параметри за Google Analytics, които указват съответно източника, медията и уникалния идентификатор за съответната кампания

При представянето на страницата в резултатите от търсене (Search Enginre Resul Pages - SERP) - наред със заглавието и описанието (meta description тага) съществуват и други важни елементи, дефиниращи как дадена страница ще бъде визуализирана. Това е използването на информация за авторство и структурирани данни. Гугъл използват данните за авторство като допълнителен атрибут за определяне на авторитета на дадена страница. Освен това визуализирането на аватара като част от резултата от търсенето позволява на потребителите лесно да го разпознаят и увеличава броят посетители към съответната страница.



***Фиг.4.3.*** *Пример за визуализация на аватар като част от резултатите от търсене*

Друг важен факт е, че след избиране на даден резултат от търсенето, който предава информация за автора и последващо връщане към страницата с резултатите Гугъл може да предложи на посетителя списък със свързани статии от същият автор. Трябва да изминат поне две минути между избора на резултата и връщането към резултатите, за да се види въпросният ефект. Този времеви диапазон е необходим, за да се потвърди, че потребителят действително е заинтригуван от намерения материал и би имал желание да види и други резултати, свързани със същия автор. Пример за показване на подобни свързани статии е даден на фигура 4.4.



***Фиг.4.4.*** *Пример за показване на свързани статии*

Структурираните данни от своя страна могат да се разделят на три основни типа:

* Микро данни (microdata)
* Микро формати (microformats)
* Инструмент за описание на ресурсите (Resource Description Framework - RDF)

Микро данните (microdata) са най-популярният от трите типа и залагат на използването на схемите, дефинирани в Schema.org. Този сайт е създаден чрез съвместната инициатива от Google, Yahoo! и Bing и съдържа колекция от HTML тагове, които авторите на страници могат да използват, за да улеснят търсещите машини при обработката на страниците и извличането на данните от тях. По този начин не само авторите могат да предадат допълнителна информация към търсачките, но и те от своя страна могат да използват информацията за предоставяне на по-качествени резултати от търсене. Микро данните използват три основни елемента, описващи обекта. Това са:

* Обхват (itemscope)
* Тип (itemtype)
* Атрибути (itemprop)

Обхватът дефинира, за кой обект от страницата се отнася информацията, заключена в използващия го таг.

Типът се използва в комбинация с обхвата с цел указване на типа на конкретния обект. На адрес <http://schema.org/docs/full.html> може да се види пълен списък с поддържаните типове, като някои от най-често използваните са:

* Личност (Person)
* Събитие (Event)
* Организация (Organization)
* Продукт (Product)
* Ревю (Review)

Всеки тип обекти притежава различни атрибути, които се дефинират чрез използване на itemprop. Общите за всички типове атрибути са: допълнителен тип (additionalType), описание (description), изображение (image), име (name) и адрес (URL). Пример за комбинирано използване на трите елементае показан на фигура 4.5

<div itemscope itemtype=”http://schema.org/Event”>  
<span itemprop=”name”>Concert Tonight</span>.at <span itemprop=”location”>Royal Albert Hall</span>starting at <time datetime="2013-06-08T19:30">June 8, 2013 at 19:30</time>.  
<div>

***Фиг.4.5*** *Пример за използване на микроданни*

При микро форматите се използва стандартният за HTML class атрибут заедно с указан в заглавната част на страницата профил. По този начин се указва на търсещите машини, как да интерпретират имената на класовете като индикатори за типа на съдържанието. Най-популярните микроформати са hCalendar - за информация, свързана със събития, hCard - информация за хора, компании и организации и hReview - при публикуване на ревюта за книги, представления, ресторанти и т.н. Примерно използване:

<head profile=”[*http://microformats.org/hcalendar*](http://microformats.org/hcalendar)”>

<div class=”vevent”>  
<span class=”summary”>AC/DC concert</span> at <span class=”location”>Vasil Levski Stadium  
</span>  
</div>

***Фиг.4.6*** *Пример за използване на микроформат*

Инструментът за описание на ресурсите(Resource Description Framework - RDF) използва така наречените речници, в които се дефинират двойки атрибут-стойност с цел описание на отделните елементи от съдържанието на документа.

<p vocab=””http://schema.org” typeof=”Person”>  
Hello, I am mr. <span property=”name”>Smith</span>!  
</p>

***Фиг.4.7*** *Пример за използване на RDF*

Изключително важна в последните години е и оптимизацията при локално търсене. Тъй като много компании предлагат своите продукти или услуги в определен географски регион, е важно правилното насочване на резултатите от търсене точно към потребителите, които могат да се възползват от тях. Могат да бъдат дадени множество примери за това - клиентите обикновено търсят водопроводчици, фризьори и ресторанти в града, в който живеят или се намират в момента, и не се интересуват от подобни услуги предлагани от компании в друг географски регион. Използвайки Google Keywords Tool можем да анализираме данните за търсенето на определени ключови думи в рамките на даден географски регион. Например при търсене на “locksmith” (ключар) в САЩ, резултатът показва, че за въпросната ключова дума месечно се извършват 820 хиляди локални търсения при 1,2 милиона глобални - ясен индикатор за важността на локалната оптимизация. Основните използвани похвати за този тип оптимизация включват:

* Редакция/добавяне на ключови думи
* Използване на структурирани данни за дефиниране на географската локация
* Създаване на локални страници
* Добавяне на обекта към социални мрежи

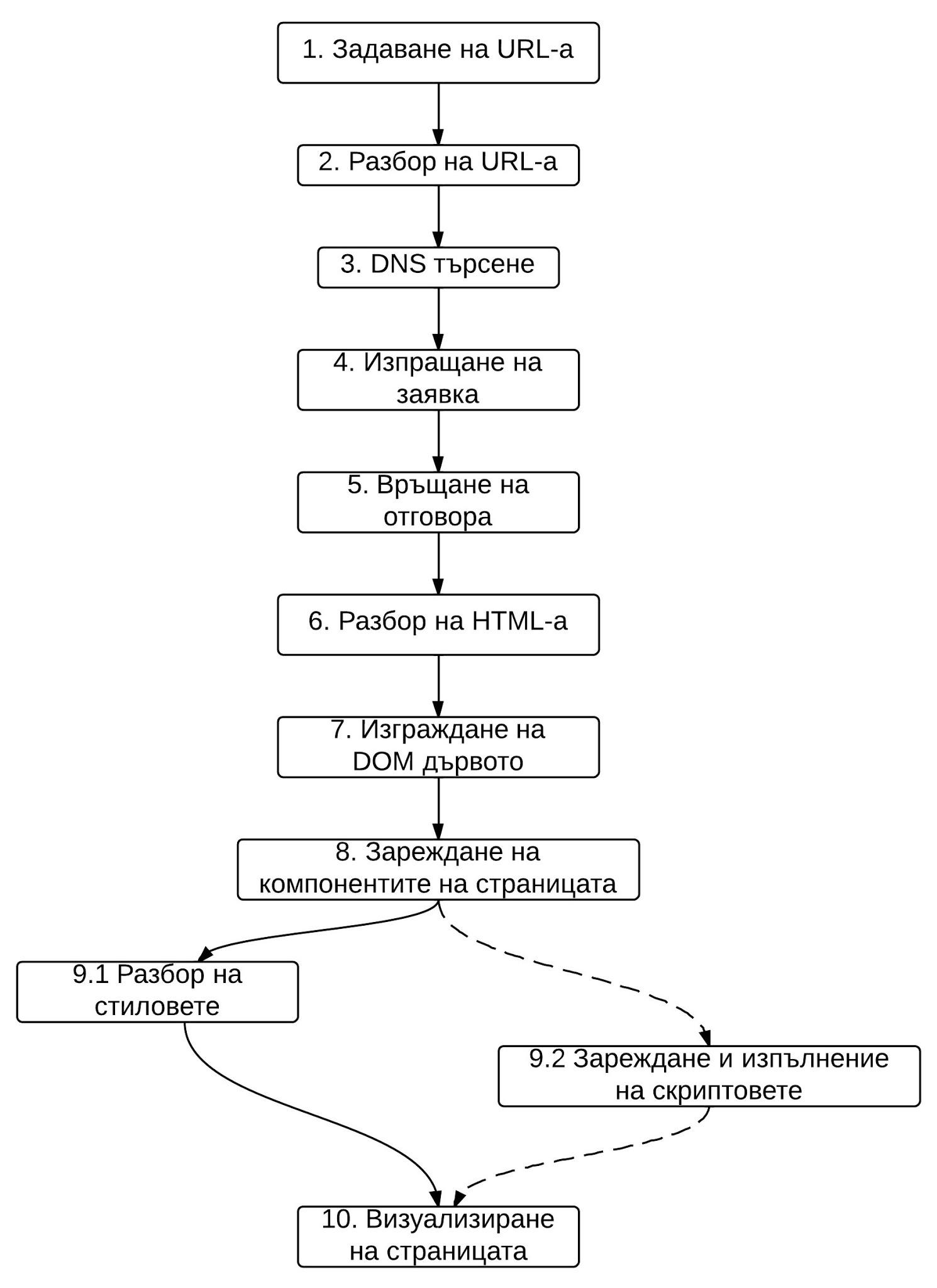
Добавянето на ключови думи, свързани с географското местоположение в заглавието, URL адреса и съдържанието на страницата, е сигурен индикатор за търсещите машини, че въпросното съдържание се отнася за конкретен географски регион. Примери за такива заглавия са “Ключар в София“, „Електротехник в Пловдив“ и др. Друг ефективен метод е използване на структурирани данни с цел предаване към търсещите машини географска информация. Чрез създаване на локална страница в Google+ или Googe Places for Business се подобрява както позиционирането в резултатите от търсене, така и се създава възможност на потребители да открият търсената информация не само чрез търсене в индексиращите машини, но и чрез навигация върху картата. Подобен ефект има и добавянето на обекта към социални мрежи с географска насоченост като FourSquare. За вериги от обекти се препоръчва за всеки обект да се създадат отделни страници, включващи географската информация в URL адреса си. Съответно всеки отделен обект трябва да се регистрира в Googe Places for Business и FourSquare.

1. **Производителност**

Скоростта на уеб страниците се е превърнала от екстра във фактор от първостепенно значение, защото ако приложението е бавно, то потребителите няма да го използват. Има представени данни от трите водещи търсещи машини Bing, Google и Yahoo, ясно показващи негативния ефект на продължителното зареждане върху трафика и потребителите.

В следствие на осезаемото влияние на скоростта на зареждане върху потребителите, разработчиците отдават все по-голямо внимание върху подобряването на производителността на страниците. Те целят намаляване на времето за зареждане на страницата, времето за реакция при действията на потребителя, както и увеличаване на максималния брой потребители, които могат да бъдат обслужени едновременно. Тъй като уеб страниците и приложенията се изграждат на базата на клиент-сървър архитектура, можем да дефинираме две основни зони за оптимизация – браузър и сървър.

С цел по-добро разбиране на работата и ефекта на отделните методи за оптимизация на скоростта на зареждане на страниците е добре да се опише пълният процес на зареждане на дадена страница - от въвеждането на адреса на страницата или натискане на дадена връзка до момента, в който всички ресурси от страницата са заредени в браузъра и изпълнението на всички скриптове е приключило. Процеса на зареждане на страницата е показан на фиг.5.1

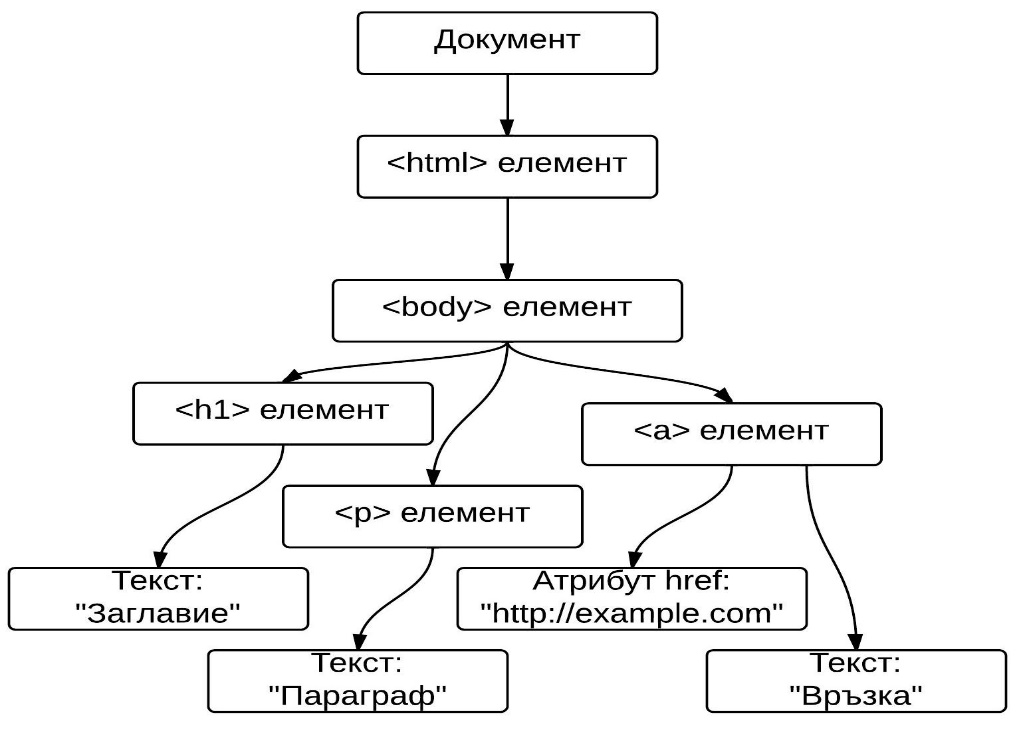


***Фиг. 5.1*** *Процес на зареждане на уеб страница*

1. Потребителят задава URL на страницата в адресната лента на браузъра или натиска върху връзка, водеща към съответната страница.
2. Браузърът прави разбор на URL-а и определя типа на протокола, най-често HTTP, но се използват също така HTTPS, FTP и др. Също така се определят домейна, порта, пътя към съответната страница или ресурс, както и GET параметрите ако има такива.
3. Прави се DNS търсене (DNS Lookup), което цели да се определи IP адреса, отговарящ за съответния домейн.
4. Създава се мрежова връзка между браузъра и Интернет сървъра, отговарящ на съответното IP, по която се изпраща заявка за съответната страница.
5. Сървърът генерира отговор, като отговорът може да съдържа както статичен файл, разположен на сървъра, така и динамично съдържание, генерирано на база заложени в кода на страницата или приложението правила. Отговорът се изпраща обратно към клиента.
6. Браузърът прочита съдържанието на отговора и в случай, че това е HTML документ прави разбор. Разчитат се отделните тагове, като в случай на неправилен синтаксис грешките се обработват и се генерира максимално близка до реалната структура.
7. Създава се DOM (Document Object Model) дърво, където всеки таг или атрибут е представен като отделен елемент в дървото. Например на база следния примерен HTML код, показан на фиг.5.2, ще бъде генерирано DOM дървото от фиг. 5.3.

<html>  
 <body>  
 <h1>Заглавие</h1>  
 <p>Съдържание</p>  
 <a href=”http://example.com”>Връзка</a>  
 </body>  
 </html>

**Фиг.5.2.** *Примерен код на HTMLстраница*



**Фиг.5.3.** *Дървовидна структура на DocumentObjectModel-а*

1. След изграждането на DOM дървото се започва постепенно зареждане на отделните компоненти на страницата - стилове, скриптове, изображения и т.н..
2. Разбор и зареждане на стилове и скриптове:
   1. След зареждането на даден стилов ресурс, се прави разбор на съдържанието му и стиловете, съдържащи се в ресурса започват да се прилагат последователно върху отделните елементи в DOM дървото. Тъй като стиловете не променят DOM дървото, а само дефинират атрибути, свързани с изрисуването му паралелно на тяхното зареждане могат да се зареждат и други ресурси.
   2. Тъй като скриптовете могат да доведат до промени в структурата на DOM дървото , по време на тяхното зареждане, разбор и изпълнение всички други процеси биват спрени. Тук има едно изключение. Някои браузъри като Mozilla (Firefox) и WebKit (Chrome, Safari) използват отделни нишки, за да продължат да зареждат външни ресурси по време на обработката на скриптове. По този начин се подобрява цялостната скорост на зареждане на външните ресурси и съответно страницата, но скриптовете, модифициращи DOM дървото продължават да се изпълняват единствено в основната нишка.
3. След като DOM дървото е генерирано и основните ресурси са заредени и обработени, се преминава към изчертаването и окончателното визуализиране на страницата.
4. **Оптимизация в браузъра**

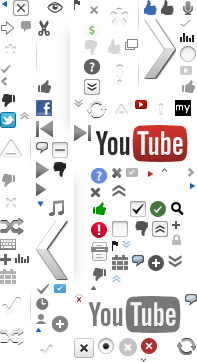
Процесът на оптимизация на производителността на страниците в уеб браузъра включва различни методи, свързани с подобряване скоростта на зареждане на страницата, като комбинира техники за намаляване на размера на страницата и използваните от нея ресурси, кеширане с цел намаляване на трафика и др. Този процес се нарича също така оптимизация при клиента.

Процесът на оптимизация при клиента рефлектира върху стъпки 3, 6, 7, 8 и 9 от диаграмата за зареждане на страницата. Съществуват 14 основни правила за подобряване скоростта на зарежда на страниците:

1. Намаляване на броя на HTTP заявките
2. Използване на мрежа за доставяне на съдържание - Content Delivery Network(CDN)
3. Добавяне на указания за валидност в заглавната част на HTTP отговора
4. Използване на GZip компресия за ресурсите
5. Разполагане на стиловете в началото на страницата
6. Разполагане на скриптовете в края на страницата
7. Избягване на скриптови изрази в стиловите ресурси
8. Използване само на външни скриптове и стилове
9. Намаляване на броя на хостовете, използвани при зареждане на ресурси
10. Минимизиране на скриптовете
11. Избягване на използването на пренасочвания
12. Премахване на дублиращи се скриптове
13. Конфигуриране на ETag
14. Кеширане при AJAX заявките

**Намаляване на броя на HTTP заявките**

Чрез намаляването на броя на HTTP заявките се редуцира времето за създаване и затваряне на нови връзки със сървъра. Основната техника тук е комбиниране на всички стилове и скриптове в два файла –по един за всеки тип медия. Като допълнителен подход се използва комбинирането на изображения, свързани с дизайна в така наречените “CSS Sprites”. Това е единично изображение, представляващо комбинация от множество по-малки, като в отделните части на страницата се визуализира само съответната част от него. Примерно CSS Sprite изображение е показано на фиг. 5.4.



**Фиг.5.4.** *Пример за CSS Sprite*

**Използване на мрежа за доставяне на съдържание - Content Delivery Network (CDN)**

Следващият метод е използването на мрежи за доставяне на съдържание CDN (Content Delivery Network). Те представляват обединение от уеб сървъри, разпределени на различни географски позиции с цел по-ефективното доставяне на съдържание до потребителите. На фиг. 5.5 е показано географското разположение на сървърите на Google.

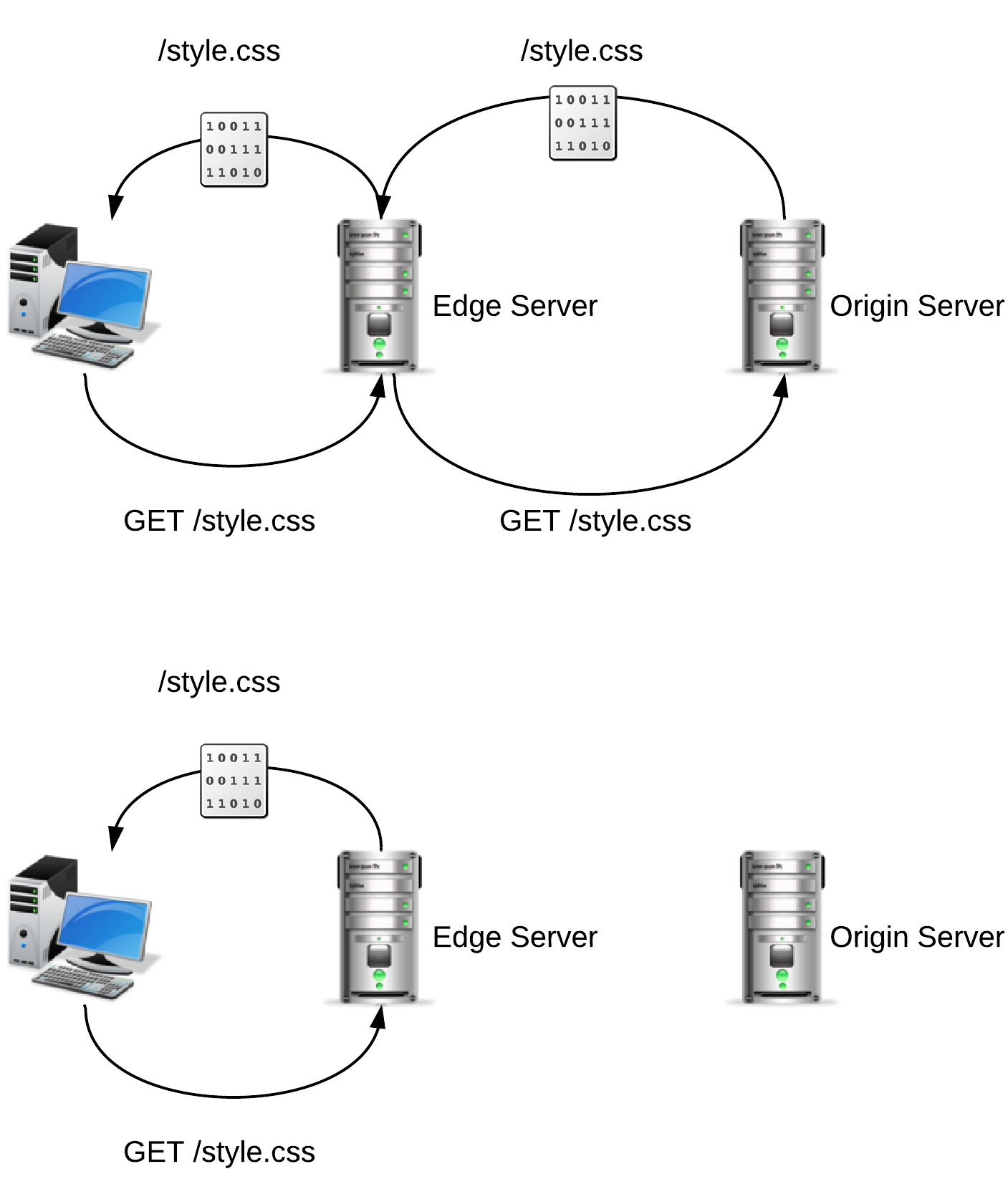


***Фиг.5.5*** *Географско разположение на сървърните центрове на Google*

В този тип мрежи сървърите са разделени в две основни групи – периферни (edge) и източници (origin) . Когато потребителят направи заявка за определен ресурс, разположен в мрежата на CDN-a, първата част от тази заявка е DNS запитването с цел определяне на IP-адреса на сървъра, отговарящ за съответния домейн. За разлика от случаите, когато за домейна отговаря едно единствено IP, което се връща автоматично, тук DNS сървъра изчислява адреса на коя от машините в мрежата да отговори на потребителя. Параметърът с основна тежест при това изчисление е географската позиция на потребителя, поради презумпцията, че колкото по-близо е разположен сървърът до клиента толкова по-бързо съдържанието ще бъде доставено. Заедно с географката локация, за по-добро определяне на сървъра, от който съдържанието може да бъде доставено най-бързо, се използват и други параметри, като натовареността на сървърите в мрежата, скоростта на връзката на клиента и др. В резултат от това изчисление се връща адреса на един от периферните сървъри. След това заявката на клиента се насочва към съответния сървър.

Ако той не разполага с поисканото съдържание се обръща към сървърите източници, от които изтегля съответния ресурс (Фиг.5.6). След това ресурсът се връща към потребителя, а периферният сървър го запазва заедно с параметър наречен “време за живот” (TTL - Time To Live), указващ колко време той може да бъде съхраняван на сървъра преди да бъде изтрит или маркиран като изтекъл.

Ако поисканото съдържание вече съществува на периферния сървър и не е изтекло, то се връща директно към потребителя без да се прави заявка към източника. По този начин след първото поискване на ресурса той е запазен на периферния сървър и може да се използва повторно, както от потребителя, който пръв го е поискал, така и от други потребители насочени към същия сървър.



***Фиг.5.6*** *Схема на кеширане при използване на CDN*

**Добавяне на указания за валидност в заглавната част на HTTP отговора**

Добавянето на указания за валидност в заглавната част на HTTP отговора се извършва чрез директивата Expires. По този начин сървърът указва на браузъра, че може да използва дадения ресурс до определен момент във времето. Съдържанието на Expires включва дата и час, след които валидността на ресурса може да се счита за изтекла. Използват се абсолютни дата и час, като използваният формат трябва да отговаря на този, дефиниран в RFC 1123. Пример:

*Expires: Thu, 01 Dec 1994 16:00:00 GMT*

Тук е важно да се отбележи, че ако в заглавните части е включена директивата Cache-Control в комбинация с max-age то тя ще презапише стойността на Expires, както и, че използването му не означава автоматично, че ресурсът няма да бъде изтрит или променен преди, след или в указания момент. Cache-Control хедърът се появява във версия 1.1 на HTTP протокола с цел да се превъзмогнат ограниченията поставени от Expires. Въпросната max-age директива показва за какъв период от време даденият ресурс може да бъде кеширан, като стойността му се задава в секунди. Ако изминалото време от първото зареждане на ресурса до негово следващо поискване е по-малко от времето, специфицирано в max-age, ресурсът може да се зареди локално от кеша на браузъра, без да се прави HTTP заявка. По този начин се намалява не само трафикът, но и броя заявки, а от там и скоростта на зареждане на страниците.

**Използване на GZip компресия за ресурсите**

Чрез използване на компресия на данните, базирана на алгоритъма DEFLATE, се намалява размера на пренесеното съдържание, което от своя страна води до редуциране на времето за пренос и зареждане на страницата. Важен за отбелязване факт тук е, че не всички елементи от страницата трябва да се подлагат на компресия. Компресиращият алгоритъм има добър ефект върху самите страници, стиловете, скриптовете и подобно текстово съдържание, но няма ефект върху PDF и DOC документи или изображения, тъй като те използват специфична за конкретния формат компресия. С въвеждането на версия 1.1 на HTTP протокола се разширяват възможностите на хедъра Accept-Encoding. По този начин браузърите получават възможност да изпращат данни, които да указват на сървъра дали клиентът поддържа методи за компресиране на данните, както и предпочитания метод. Примери за такова използване на Accept-Encoding са:

* Accept-Encoding: compress, gzip
* Accept-Encoding:
* Accept-Encoding: \*
* Accept-Encoding: compress;q=0.5, gzip;q=1.0
* Accept-Encoding: gzip;q=1.0, identity; q=0.5, \*;q=0

Поддържаните от протокола стойности за типа на кодирането са gzip, compress, deflate и identity. При gzip се използва командата за компресиране gzip (GNU zip), действието на която е базирано на алгоритъма за кодиране на Лемпел-Зив (Lempel-Ziv) – LZ77 с 32 битова контролна сума (CRC), използвана при проверката за грешки. При подаване на стойност compress се използва стандартната за UNIX системите команда compress, като формата на кодирането е адаптация на алгоритъма на Лемпел-Зив-Уелч (Lempel-Ziv-Welch) – LZW. Deflate комбинира алгоритъма на Лемпел-Зив и кодирането на Хъфман, аidentity използва оригиналното кодиране на съдържанието, т.е. съдържанието не се трансформира. Заедно с типа на кодирането може да се подаде и втори атрибут qvalue(q), който указва предпочитанията на браузъра за съответния тип кодиране. Ако стойността му е нула, това означава, че типът кодиране е неприложим. При подаване на повече от един тип кодиране и използване на qvalue, кодирането с максимална стойност е с най-висок приоритет. Използваните за компресия алгоритми не водят до загуба на качество или с други думи разкодираното (разкомпресирано) съдържание отговаря напълно на оригинала.

**Разполагане на стиловете в началото на страницата**

Поставянето на стиловете в началото на файла позволява зареждането им да започне веднага след първоначалната обработка на HTMLстраницата и да продължи паралелно с това на останалите елементи. По-този начин основните елементи на страницата и текстовете получават своите стилове преди всички изображения и скриптове да бъдат заредени. По този начин се създава усещане у потребителя, за прогресивно зареждане на страницата, което води до подобряване на потребителското преживяване. Обратно на това при разполагане на стиловете в края на страницата, се увеличава периода, през който потребителят изчаква тяхното зареждане и съответно вижда версия на сайта, в която липсват данните за подреждането и изгледа на елементите.

**Разполагането на скриптовете в края на страницата**

Противно на правилото за стиловете, скриптовите ресурси трябва да бъдат разположени в края на страницата, като така се цели да се избегне блокирането на процеса на зареждане и обработка на останалите ресурси. Причините за това блокиране са описани в точка 9.2 от процеса на зареждане на страницата, но в обобщение може да се каже, че по време на зареждането на външни скриптове се блокира зареждането и обработката на всички елементи от страницата, намиращи се след скрипта. Допълнителен факт тук е, че някои браузъри не поддържат паралелното сваляне на ресурси по време на зареждането на скриптове. Поставянето на скриптовете в края на страницата позволява пълното зареждане на страницата и стиловете преди да се достигне до обработката на скриптовете и съответно блокирането на останалите процеси.

**Избягване на скриптови изрази в стиловите ресурси**

Този метод за оптимизация е широко разпространен, но в последствие губи своето практическо приложение. Причината за това е, че през 2008-ма година, Майкрософт обявяват своето намерение да преустанови използването на този тип изрази във версия 8 на Internet Explorer. По този начин се цели по-добро спазване на стандартите за езика, подобряване на производителността и повишаване на сигурността на браузъра. Поради това можем да разглеждаме метода като неактуален за текущото състояние на уеб индустрията.

**Използване само на външни скриптове и стилове**

При този метод за оптимизация се избягва директната употребата на стилови и скриптови команди като част от кода на HTML страницата, а вместо това те трябва да бъдат изнесени като външни за уеб страницата ресурси. По този начин не само се премахва негативният ефект от блокиране на зареждането на страницата по време на обработката на скриптове, но и се намаля размерът на основната страницата. Второто основно предимство на използването на външни стиловете и скриптовете е, че по този начин се позволява тяхното кеширане и по-бързото им изтегляне при презареждане на страницата или навигация в рамките на сайта.

**Намаляване на броя на хостовете използвани при зареждане на ресурси**

Този метод спомага за подобряване скоростта на зареждане. Както е обяснено в схемата за процеса на зареждане на страниците, всяка заявка започва с така нареченото DNS търсене, което отнема средно между 20 и 120 милисекунди. Броят на тези търсения зависи пряко от броя на уникалните хостове в кода на страницата, а не от броя на заявките. Причината за това е, че както браузърите, така и операционните системи се грижат за кеширане на резултатите от тези търсения. Важно уточнение тук е, че кеширането в браузъра се различава от това, извършвано в DNS сървъра на интернет доставчика. При второто обикновено периодът за кеширане е по-продължителен и макар то да намалява времето за търсене, не се постига моменталният ефект както при кеширане в браузъра или операционната система. Това се дължи на факта, че времето необходимо за изпълнение на DNS заявката до друга система, е значително по-голямо от това при обръщение към кеша на браузъра или операционната система. Освен намаляването на броя на хостовете се прилага и метода на повторно използване на съществуващи връзки към сървъра, наричани още перманентни връзки (Persistent Connections, Keep-Alive Connections). Във версия 1.0 на HTTP протокола се използва заглавната част “Connection: keep-alive”, като поставянето ѝ в заявката към сървъра указва, че браузърът поддържа повторното използване на връзки, а наличието ѝ в отговора от сървъра - че той ще запази връзката отворена с цел нейното повторно използване. Във версия 1.1 на протокола всички връзки се считат за перманентни освен в случай, че е указано друго. Повторното използване на една връзка за пренос на множество ресурси води до намаляване на общия брой обръщения към сървърите, на които са разположение съответните ресурси, както и избягване на допълнителното време, използвано за създаването и затварянето на всяка отделна връзка. Заедно с това се намалява и натоварването на сървъра, тъй като общият брой на едновременно отворените връзки намалява. Недостатък при използването на keep-alive е, че при изтегляне на един или малък брой ресурси, връзката може да остане отворена дори след изтеглянето на последния. По този начин лесно може да бъде достигнат максималният брой връзки за сървъра, въпреки че някои от тях не са активни, а такива, които чакат да изтече времето до тяхното затваряне. За да се избегне това, времето за запазване на връзката се конфигурира на сървъра на база съществуващия трафик, броя на едновременните заявки, потребители и средния брой заявки в рамките на една потребителска сесия.

**Минимизиране на скриптовете**

Този метод включва два процеса за обработка на скриптовете,поставени под общо име. Това са процесите на минимизиране и на объркване (obfuscation) на кода. При минимизирането се премахват “излишни” символи, с което се цели намаляване на размера на скрипта. Обикновено това са коментари, интервали, табулация и знаци за нов ред.Объркването, от своя страна, не само премахва излишните символи, но и обърква кода замествайки имена на променливи и функции, като по този начин той става по-труден за четене. Макар този процес да е възникнал като опит да се предотврати процеса на „обратно инженерство“ (reverse engineering) на кода,той има положителен ефект поради намаляването на размера на крайния файл. И двата процеса влияят единствено на размера на скрипта, но не и на неговото действие или производителност.

**Избягване на използването на пренасочвания**

С цел подобряване скоростта на зареждане на страниците е препоръчително да се избягва използването на пренасочвания. Във версия 1.1 на HTTP протокола, са дефинирани 8 различни кода, с номера от 300 до 307, означаващи типа на пренасочването. Изключение тук прави единствено код номер 304, известен като “304 Not Modified”, който не представлява истинско пренасочване, а указва на браузъра, че ресурсът се счита за непроменен спрямо зададените в заявката параметри. От останалите кодове най-честа употреба имат тези с номера 301 и 302, указващи съответно постоянно и временно пренасочване. Независимо от типа на пренасочването (с изключение на 304) наличието му води до създаване на нова заявка. Така за всеки ресурс се генерират две отделни заявки, една за оригиналния URL адрес, чийто отговор съдържа кода на пренасочването и една за адреса, към който води пренасочването и чрез която се изтегля съответният ресурс. Освен времето за генериране на двата отговора от сървъра, в случай, че адресът, към който води пренасочването е в рамките на друг домейн, времето за зареждане на ресурса се увеличава допълнително и заради процеса на DNS търсенето. И в двата случая пренасочването води до създаване на нова връзка към сървъра, която добавя допълнително време към процеса на зареждане. По този начин използването на пренасочвания, независимо от техния тип, води до допълнително забавяне при зареждането на страници или отделни ресурси.

**Премахване на дублиращи се скриптове**

Премахването на дублиращи се скриптове пряко рефлектира върху размера и скоростта на зареждане на страниците. През 2007-ма година изследване на десетте най-големи сайтове от САЩ установява, че в два от тях има дублиращи се скриптове. Според изследването, най-честата причина за дублиране на скриптове е големият им брой в страницата, както и големият брой на разработчиците в компания. За да се осъвремени изследването могат да се анализират някои от най-посещаваните сайтове и да се намери броят на зарежданите скриптове за всеки от тях(табл.5.1):

|  |  |
| --- | --- |
| Сайт | Брой скриптове |
| Facebook | 23 |
| Amazon | 18 |
| Google | 3 |
| Youtube | 2 |
| Wikipedia | 12 |
| StackOverflow | 12 |
| Twitter | 5 |
| LinkedIn | 17 |

***Таблица 5.1.*** *Брой използвани скриптове в сайтове*

Не се наблюдават дублиращи се скриптове, но впечатление прави големият брой ресурси (над 10), зареждани в значителна част от сайтовете. Големият брой скриптове повишава възможността един скрипт да бъде зареден повече от веднъж, което от своя страна може да има негативен ефект върху работата на сайта в три направления:

* Повишен брой заявки и трафик
* Увеличаване на времето за обработка и зареждане
* Възникване на грешки и/или промяна в работата на скрипта

Повишеният брой заявки и трафик се проявяват при референциране на една и съща версия на даден скрипт при използване на различни URL-адреси за зареждане на скрипта. Това води до създаването на нова заявка към сървъра за всеки отделен адрес, реализиране на допълнителен трафик и увеличаване на времето за зареждане.

Дори да се използва кеширане при всяко следващо референциране към даден скрипт той ще бъде обработен и изпълнен отново. Това ще доведе не само до увеличаване на времето за изпълнение на скрипта, но и до евентуално блокиране на изпълнението и обработката на други ресурси.

В отделни случаи повторното изпълнение на даден скрипт може да доведе до възникване на грешки във функционалността или неочаквани резултати. Пример за това е библиотеката която конвертира специфичните за Туитър ключови низове, наричани още хеп-тагове, във връзки към други страници. Тъй като при конвертиране въпросния низ се затваря в таг за връзка, повторното му изпълнение би довело до създаването на нов таг за връзка в вече съществуващия такъв.

За щастие в съвременния уеб съществуват различни инструменти, позволяващи работа със скриптове и намаляващи шанса за тяхното дублиране. Пример за такива инструменти са RequireJS, HeadJS и LoadJS.

**Конфигуриране на ETag**

Конфигурирането на заглавната част ETag, подобно на Expires, спомага за по-доброто определяне на това дали даден ресурс е бил променен в рамките на периода между две заявки. ETag-а се добавя в заглавната част на отговора, когато съдържанието на ресурса бъде предоставено на браузъра. Когато браузърът отново поиска същия ресурс, той може да провери дали съдържанието му е било променено чрез добавянето на If-None-Match в заглавната част на заявката. В случай, че ресурсът не е бил променен сървърът би трябвало да върне отговор със статус “304 Not Modified” и празно съдържание. В противен случай очакваният отговор трябва да е със статус “200 OK” и да включва съдържанието на ресурса. Различават се два вида валидация на ETag-ове - силна и слаба. Разликата във формата на изписване е, че слабата добавя низа “W/” пред стойността на тага, например - W/"123456789". Силната валидация се използва за проверка дали съдържанието на ресурса е побитово идентично на това на оригиналния такъв. Слабата показва, че двата ресурса са семантично идентични и може да бъде използвано кеширано копие на съответния ресурс. Според сайта WebPageTest.org “В повече случаи ETag-овете не трябва да бъдат използвани, освен в случай на изрична нужда от тях” (“ETag headers should generally not be used unless you have an explicit reason to need them”). Причината се крие във факта, че по подразбиране форматът по-който се генерира ETag стойността е в пряка зависимост от конкретния сървър и това може да бъде проблем при системи, използващи множество сървъри. Затова за мултисървърни системи трябва или ETag частта да бъде конфигурирана, така че да не включва информация за конкретния сървър, или изцяло да бъде премахната от отговора на заявките. Според данни на W3Techs за Май, 2013 г. 15.5% от всички сайтове използват ETag-ове, като силната валидация се използва в 98,5% от случаите, а слабата в останалите 1.5%. Сред популярните сайтове, които използват ETag са социалните мрежи Туитър и Пинтерест и платформата Блогър.

**Кеширане при AJAX заявките**

Освен за ресурсите като стилове, скриптове и изображение кеширането може да се приложи и при използването на асинхронни заявки (AJAX). Абревиатурата AJAX означава „Asynchronous JavaScript and XML“, в превод „Асинхронен JavaScript и XML“ и представлява обединение от техники, позволяващо на браузъра да създава асинхронни заявки към сървъра. По този начин могат да бъдат създавани нови заявки дори и след окончателното зареждане на страницата без това да води до презареждане или преминаванекъм друга страница. В своя статия HTTPWatch обясняват, че AJAX заявките следва да се разглеждат като нормални HTTP заявки тоест, за тях важат всички правила, записани в HTTP протокола и следва да бъдат кеширани по същият начин. Според дефиницията на HTTP протокола методите POST, PUT, PATCH и DELETE следва да бъдат използвани в случаи, когато се цели промяна на данните на сървъра или съществуващата сесия и не бива да бъдат кеширани. Кеширането следва да се използва само в случай, че използваната заявка служи единствено за извличане ресурси и няма ефект върху сървъра. Този тип методи се наричат сигурни („safemethods“) и включват GET, OPTIONS и HEAD. Тъй като последните два рядко се използват при разработката на уеб страници и приложения, се разглежда кеширането само в случая, когато AJAX заявките използват GET метода. Тъй като зареждането на стилове и/или скриптове при AJAX заявки се използва по-рядко и основната употреба е за извличане на данни от сървъра, основните посоки, в които трябва да се работи са задаването на коректни Expires и Cache-Control заглавни части, базирани на бизнес логиката на приложението, както и използването на GZIP компресия.

В случай на зареждане на скриптове или стилове или пренасянето им като част от съдържанието на отговора, те трябва да бъдат минимизирани с цел намаляване размера на отговора и съответно неговото по-бързо изтегляне от сървъра.

1. **Оптимизация на производителността на сървъра**

Разглеждат се различни методи за подобряване на производителността на кода на страницата или приложението, разположен на уеб сървъра, като оптимизация на заложените в кода алгоритми и използване на специфичните за езика методи за постигане на по-добра производителност. Разглежда се процеса на повторно използване на връзките на уеб сървъра. Тук промените рефлектират върху скоростта, с която сървърът връща отговор при направена заявка или стъпка 5 от диаграмата за зареждане.

Използването на клъстърни конфигурации и облачни системи (cloud-computing) с цел по-лесно скалиране е частен случай на процеса на оптимизация, характерен единствено за сайтове и приложения с голямо натоварване, затова в случая ще бъдат споменати само основните методи. Най-често използваният е употребата на системи за разпределение на натоварването (“loadbalancer”), които служат за разпределение на заявките между два и повече сървъра. Обикновено това се прави, чрез “round-robin” алгоритъм. Друг метод е използването на опашки за по-тежки операции и изчисления. По този начин се избягва едновременното стартиране на множество силно ресурсоемки процеси, а чрез системи за известяване потребителят бива уведомен, когато заявката му е изпълнена.

При стандартните по размер сайтове и приложение основен метод за оптимизация остава кеширането. Докато при клиента то се използва с цел намаляване на трафика и броя HTTP заявки, то използването му от страна на сървъра цели намаляване на натоварването и подобряване на скоростта за отговор на заявките. То може да бъде разделено в две основни групи:

* Кеширане на цели страници/ресурси
* Кеширане на блокове данни от приложението

Най-често прилаганият метод за кеширане на цели страници и ресурси е използването на кеширащ прокси сървър. Той играе ролята на посредник в комуникацията между клиента и приложението, което предоставят поисканите от клиента данни.Предимството е, че при определени условия ресурсите могат да бъдат съхранени в прокси сървъра, така че при следващо поискване да бъдат сервирани директно без да се налага да бъдат наново генерирани. За реализиране на такъв тип кеширане могат да бъдат ползвани софтуери като Varnish, Squid, NginX и др. В зависимост от избраната архитектура прокси сървърът може да споделя един и същи хардуер с уеб приложението или да е разположен на отделна компютърна система. Обикновено се използва в случаи, когато едно и също динамично съдържание може да бъде заредено при всички потребители. Например публични страници, несъдържащи специфична за потребителя информация. По този начин кодът на страницата се генерира само при първоначалното поискване и при всяко следващо такова се предоставя кешираното копие. В случай, че валидността на копието е изтекла, то се генерира наново. Като условия за продължителността на валидност на копието се използват същите полета в заглавната част на отговора, както и при използването на кеша на браузъра. Също така някои от прокси сървър решенията позволяват да се конфигурират условията за опресняване на кеша, така че при евентуално отпадане на сървъра с приложението, прокси сървърът да продължи да предоставя информацията, съхранена в кеша му до възстановяване на работата на приложението. Това решение е много подходящо за сайтове, чиято информацията се изменя рядко, но се изисква постоянна наличност на сайта. Пример за такъв тип сайтове са юридически портали, сайтове на държавни институции, корпоративни портали и др.

Кеширането на цели страници от приложението се използва, когато достъпът до даден ресурс се регулира на база зададени в приложението правила. Тук отново групи от потребители получават едно и също съдържание, без то да се генерира наново за всяка отделна заявка. Приложението се грижи да установи дали потребителят отговаря на условията за достъп до него.

При приложения, където голяма част от данните зависят от конкретния потребител се използва кеширането на блокове от данни в кода на приложението. Скоростта на генериране на отговора се увеличава, тъй като се избягват сложни и бавни обработки на данни. Друга честа употреба на този метод е в случаите, когато приложението прави заявки към външни системи. Резултатът от заявката може да бъде кеширан, така че да се избегне забавянето от повторното обръщение. Обикновено за целта се използват външни приложения, съхраняващи данните в двойки от ключ и стойност. Пример за такива приложения са Memcached и Redis. Всеки ключ е уникален, а стойностите могат да бъдат типове данни от използвания програмен език или обекти. Принципът на използване е показан чрез примерен код (фиг.5.7):

cache\_key = 'unique\_cache\_key' // уникален ключ  
cache\_time = 1800 // периода на валидност на данните  
result = cache.get(cache\_key)  
if not result:  
result = get\_result()  
cache.set(cache\_key, result, cache\_time)  
return result

***Фиг.5.7*** *Примерен код за употреба на кеширане*

Процесът е следният - прави се проверка за наличието на данни за съответния ключ в кеша. Ако такива съществуват, те се връщат като резултат и програмата продължава изпълнението си. При липса на данни за подаденият ключ се изпълнява код, който да генерира необходимите данни. Те се записват в кеша със съответния ключ и период на валидност, след което програма продължава изпълнението си.

Например приложение, при което времето за генериране на отговора е 200 ms, а за трансферирането му до потребителя 50 ms, общо време за отговор 250 ms. Ако максималният брой връзки, които уеб сървърът може да обслужва едновременно е 1000, то лимитът им би бил достигнат при създаване на нова връзка на всеки 0,25 ms или 4000 връзки на секунда. Ако чрез използване на кеширане времето за генериране на отговора бъде намалено 4 пъти, то общото време за отговор става 100 ms, а броят на връзките, които могат да бъдат реализирани всяка секунда нараства до 10000 т.е. 2,5 пъти.

Използването на кеширане дава отражение и върху работата на базите данни, използвани от приложението. Кеширането на цели страници или блокове данни води не само до намаляване на броя заявки към базата и на нейното натоварване, но и до подобряване на конкурентоспособността чрез намаляване на времето, в което таблиците биват заключени.

1. **Потребителско Преживяване**

Потребителското преживяване (user-experience) отразява ефективността, удобството и удовлетворението на потребителя при използване на даден продукт или услуга. Според ISO 9241-210 потребителското преживяване е персоналното възприятие и реакция в резултат от употребата на продукт, система или услуга. Терминът „потребителско преживяване“ (user experience) бива въведен от Донал Норман в средата на 90-те години на 20-ти век. За неговото определяне се изследват реакциите и емоциите на потребителя, както и производителността му при изпълнение на определени задачи. При софтуерните продукти, уеб страниците и приложенията в частност изследването на потребителското преживяване също така включва анализ на взаимодействието човек-компютър и човек-продукт, както и скоростта за разучаване на продукта.

Представено по друг начин потребителското преживяване отразява усещането, създадено у потребителя по време на престоя му на сайта. Цели се развиване на чувство за сигурност и удобна работа. Основните фактори тук са дизайнът и информационната архитектура. Често присъствието или липсата на отделни елементи в дизайна влияе подсъзнателно на потребителите и създава определени усещания в тях. Използват се различни цветови гами в зависимост от тематика и насочеността на сайтовете. Потребителите често имат предварителни очаквания за това какво ще намерят на даден сайт на база опит с други подобни сайтове. Важна функция на дизайна е да насочва потребителя към определени елементи от сайта - били те съдържание или линкове и бутони. Информационната архитектура от своя страна се грижи да улесни ориентацията на потребителите, така че бързо и лесно да могат да намерят търсената от тях информация.

Основните критерии за оценка и измерване на потребителското преживяване са:

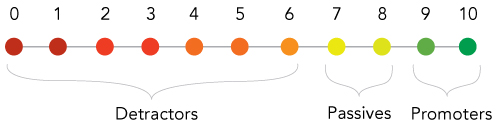
* леснота за опознаване на продукта
* леснота за ползване - при познаване на продука
* удовлетвореност на потребителя
* желание за повторна/последваща употреба
* количество грешки по време на работа с продукта

МeasuringUsability публикуват изследване, съдържащо данни за най-важните метрики на потребителското преживяване:

* средна успеваемост за изпълнение на задачите
* стойност на нетните промоутъри за софтуер и уебсайтове
* средна стойност на скалата за използваемост на системите
* процент на успешно извършените задачи

Процента на успешно изпълнените задачи е една от най-лесните за изследване метрики на ползваемостта и потребителското преживяване. За целта се създава списък от задачи и ясни критерии кога дадена задача се смята за изпълнена. Изследвайки над 1200 различни задачи, МeasuringUsability откриват, че средният процент на успеваемост при софтуерни и уеб приложения е 78%.

Стойност на нетните промоутъри (net promouter score) сред софтуерните потребители е метрика, отчитаща вероятността продуктът или услугата да бъде препоръчан от клиент. Използва се скала от 0 до 10, където 0 означава “не бих препоръчал”, а 10 “бих препоръчал”. След това отговорите на потребителите се разделят в 3 групи (фиг.5.8):



***Фиг.5.8*** *Скала на вероятността за препоръчване на продукта*

* промоутъри (promoters) - стойност 9 и 10
* пасивни (passives) - стойности от 7 до 8
* негативни (detractors) - стойности от 0 до 6

За да се получи крайната стойност, броят на негативните потребители се изважда от броя на промоутърите и се определя неговият процент спрямо общия брой промоутъри (формула .5.9).

*, (5.9)*

Където P e броят на промоутърите, а N – броят на негативните потребители.

Например при изследвани 370 души сред които 150 промоутъра, 100 пасивни и 120 негативни, получаваме, че стойността на нетните промоутъри е 20%.

Изследвайки 1000 потребители на популярни софтуерни продукти, МeasuringUsability определят, че средната стойност на нетните промоутъри е 21%. Същият параметър за уебсайтове и уеб приложения има стойност 14%. Изводът тук е, че потребителите са по-лоялни към софтуерните продукти отколкото към уебсайтовете и шансът те да бъдат препоръчани е значително по-малък.

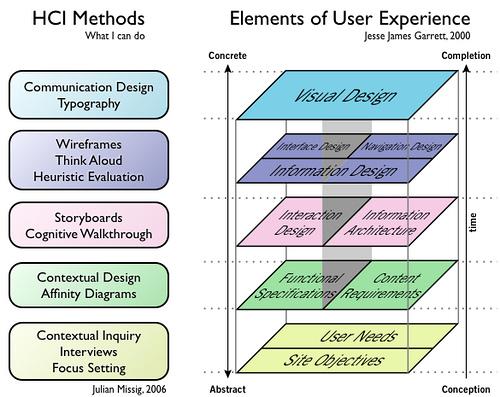
Средната стойност на скалата за използваемост на системите (SUS - system usability scale) представлява комбинация от 10 въпроса, като всеки един от тях притежава 5 отговора. Всеки отговор носи от 1 до 5 точки, като една точка означава “пълно несъгласие”,а пет “пълно съгласие”. Използваните въпроси са:

* Бих се радвал да използвам често тази система
* Намирам системата ненужно сложна
* Смята, че системата е лесна за употреба
* Смятам, че ще се нуждая от техническа помощ при работата ми със системата
* Намирам множеството функции в системата за добре интегрирани
* Смятам, че има твърде много непостоянство в системата
* Смятам, че повечето хора ще се научат да използват системата много бързо
* Намирам системата тромава при употреба
* Чувствам се сигурен при работата ми със системата
* Трябваше да науча много неща преди да мога да работя със системата

В посоченото изследване при разглеждане на 500 приложения се установява, че средната стойност на скалата на ползваемост е 68.

Високият процент на успешност при задачите (процентът на изпълнените спрямо всички зададени задачи) е в пряка връзка със стойността на SUS. При SUS над 80 се наблюдава 100% изпълнение на поставените задачи.

За да се оптимизира потребителското преживяване на дадена уебстраница, приложение или друг софтуерен продукт, е важно да се познават основните елементи, които го изграждат (фиг.5.10).



***Фиг.5.10*** *Елементи на потребителското преживяване*

Основните елементи на потребителското преживяване движейки се в посока от концепцията към крайния продукт са дефинирани в 5 слоя:

* Потребителски нужди и цели на сайта
* Функционална спецификация и изисквания към съдържанието
* Информационна архитектура и проектиране на взаимодействията –
* Информационен дизайн
* Визуален дизайн

Изхождайки от дадена идея или набор от такива, се задава общата насока на сайта или продукта. След това на база маркетингови и потребителски проучвания се дефинират нуждите от целите на приложението.

След като целите на сайта са определени се преминава към изработването на функционалната спецификация и изискванията към съдържанието. Дефинират се основните елементи на съдържанието, както и необходимите за сайта функционалности, така че да бъдат задоволени потребителските нужди.

След това се преминава към създаване на структурен дизайн на информационните елементи, така че да се постигне интуитивна работа на потребителите. Проектират се взаимодействията потребител-система като се цели максимално улеснение на потребителя за намиране на информация и изпълнение на задачи. Тук е и мястото на информационната архитектура. Създаване на структурата и организиране на съдържанието в сайта, така че да се улесни навигацията на потребителя между отделните информационни ресурси.

Създаването на информационния дизайн следва това на архитектурата. Информационният дизайн е начин на представяне на информацията, така че да се улесни нейното възприемане. В този етап се осъществява и дизайнът на клиентската част и навигацията. Работи се за постигане на улеснено и интуитивно придвижване на потребителя между отделните информационни ресурси на страницата

Последният елемент е визуалният дизайн. Тук се дефинира графичното представяне на разработените в предните слоеве модели - избор на подходяща палитра, шрифтове и графични елементи. Тези елементи са от изключителна важност за постигане на усещане за комфорт у потребителя.

Тъй като в смисъла на термина оптимизация се съдържа предпоставката за съществуваща система, която ще се променя, в този пример се приема, че първият и вторият слой вече са дефинирани и оптимизация се прилага в трети, четвърти и пети слой. В отделни случаи оптимизация се прави и в първите два слоя, като това е свързано с промяна на бизнес модела и/или предлаганите от сайта възможности.

Информационната архитектура дефинира разположението на отделните информационни елементи в цялостната система. Съществуват множество модели за информационни архитектури, като най-използваните сред тях са:

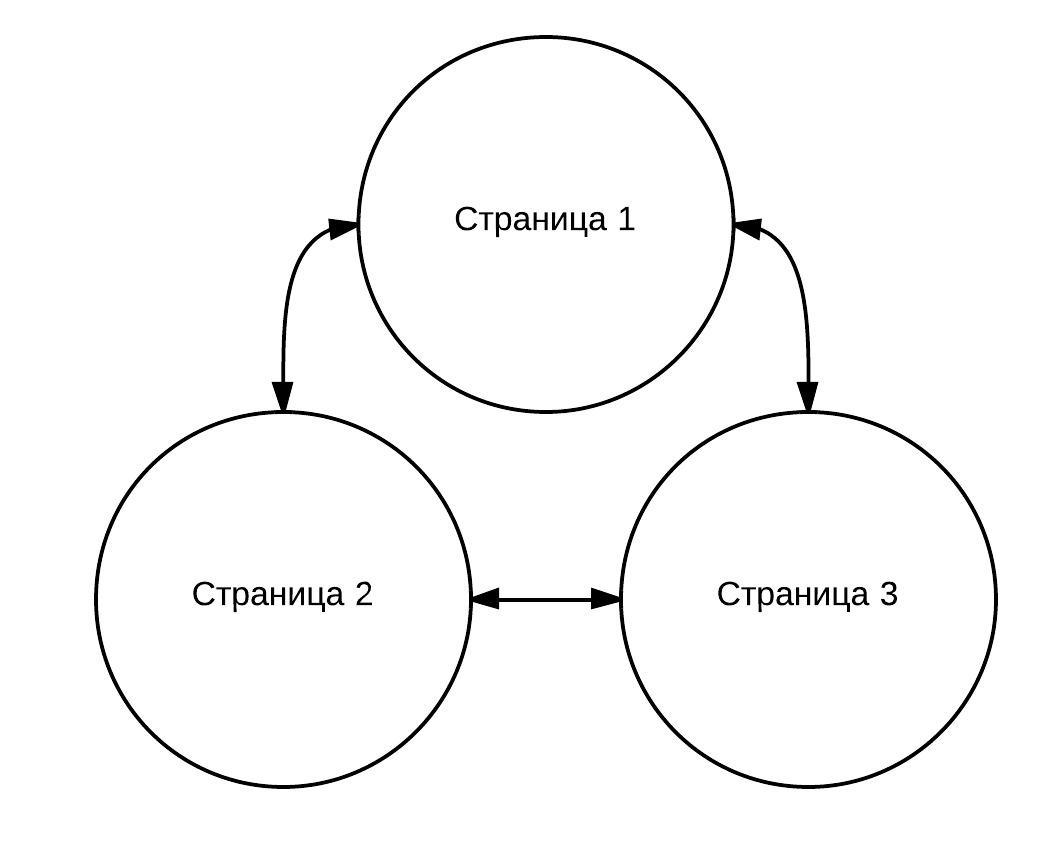
* единична страница
* плоска структура
* индекс страница
* йерархична структура
* смесена йерархия

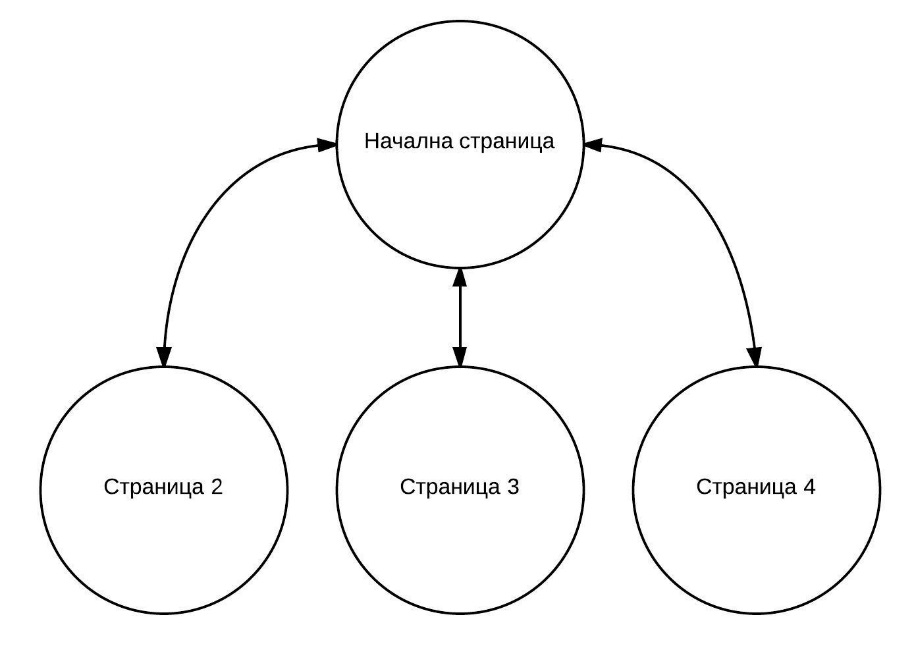
Моделът с единична страница(фиг.5.11) е подходящ за кратка реклама на продукт или лична/фирмена страница за контакт. В този модел фокусът е върху малка по обем информация, която да подтикне посетителя да осъществи връзка с фирмата или лицето, чиято е страницата. Често се използва и като начална страница за стартиращи фирми или в случай на преработка на сайта, когато не е подходящо използването на съществуващия до момента такъв в процеса на разработка на новата страница.



***Фиг.5.11*** *Модел на единична страница*

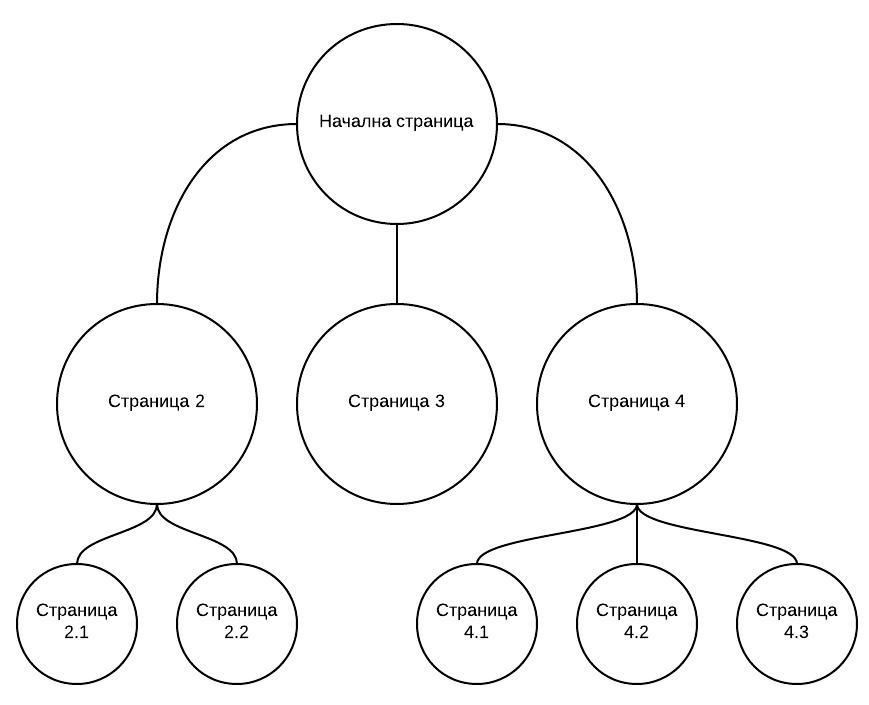
При плоската структура (фиг.5.12) всички страници са разположени на едно ниво и са с еднаква важност. Пример за това са сайтове брошури или такива с малък брой страници.

**Фиг.5.12** **Плоска структура на уеб-сайт**

Моделът с индекс страница (фиг.5.13) използва една страница като връзка към множество под-страници.

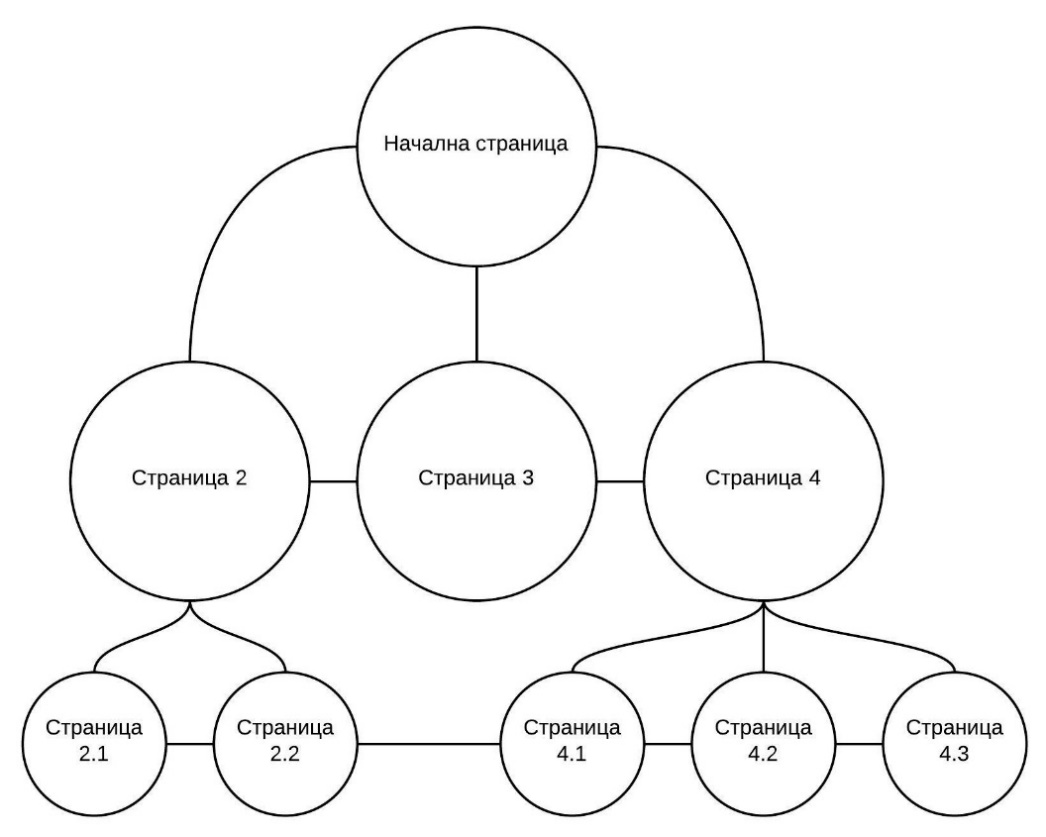
***Фиг.5.13*** *Модел с индекс страница*

При йерархичната структура (фиг.5.14) всяка страница разполага с една или повече под-страници, образувайки дървовидна структура. Тук отделните клонове в дървото са свързани единствено със своя родител, но не и с другите страници.



***Фиг.5.14*** *Модел на йерархична структура*

В случая на смесена йерархия (фиг.5.15) страниците са подредени, подобно на йерархична структура, но се позволява директна връзка между отделните страници на дадено ниво.

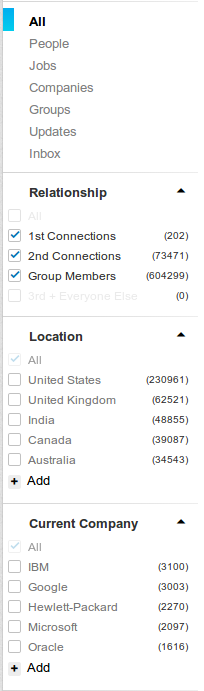
***  
Фиг.5.15****Модел на смесена йерархична структура*

При дизайна на навигацията е силно разпространено правилото за трите клика. То предполага, че всяка информация в сайта трябва да е достъпна за потребителите чрез не повече от три натискания. Правилото се базира на предположението, че потребителите биха напуснали сайта в случай, че не са открили търсената от тях информация след третия клик.

Множество изследвания опровергават това правило. Пример за такова изследване е проведеното от Джейкъб Нилсен и Хоа Лорангер, които откриват, че възможността на потребителите да намерят определен продукт в сайт за електронна търговия е нараснало с 600% след заместването на 3 степенната навигация с 4 степенна. Резултатите от това изследване намират широка употреба в този тип сайтове и стимулират въвеждането на “фасетно търсене” като навигационен елемент.

Фасетното търсене е навигационна техника, при която потребителя използва комбинация от множество различни филтри с цел достигане до продукт или информация, най-пълно отговарящи на неговите критерии. Пример за такова търсене е прегледа на библиотечен каталог и филтриране на съществуващите издание на база комбинация от жанр, година на издаване, читателски рейтинг и ключова дума. Използването на този метод за търсене стартира през 2000 г. и през 2012 става основен елемент при онлайн магазините и сайтове, представящи множество продукти или информационни ресурси.

На фигура 5.16 е показан пример за използване на фасетно търсене в професионалната социална мрежа LinkedIn:

***  
Фиг.5.16*** *Пример за фасетно търсене*

Друг важен аспект за подобряване на потребителското преживяване в съвременните страници е интерактивността. Тя представлява възможност на потребителя лесно и удобно да комуникира със сайта или системата. Развитието на технологиите доведе до възможност за почти мигновена реакция на приложеният в отговор на действията на потребителите. Важен за отчитане факт е все по-разширяващата се употреба на AJAX. Тази техника позволява постоянна комуникация между потребителя и приложението без презареждане на страницата, като процеса на обмен на информацията остава скрит за потребителя. Това, което той вижда е как отделни елементи от страницата се променят в резултат на неговите действия. Добър пример за интерактивност е търсенето в реално време. Възможността потребителят да вижда възможни резултати още докато пише своята заявка дава удобство и води до по-бърза работа със системата.

Последен фактор, влияещ на потребителското преживяване е достъпността. Под достъпност се разбира възможността на потребители с увреждания да могат пълноценно да използват уеб страниците. С подобряването на достъпността се подобрява ползваемостта и сайтовете стават по-лесни за разчитане от търсещите машини. Подобряването на достъпността може лесно да се постигне чрез употребата на текстове към линковете и изображенията, както и чрез правилно подреждане на съдържанието в HTML страницата, така че то да отговаря на визуалното му представяне.

Анализа на факторите, дефиниращи успеха на дадена страница е основна предпоставка за правилното определяне на необходимите типове оптимизация. От своя страна, детайлното познаване на отделните методи за оптимизация и спецификите на тяхното приложение позволява ясното определяне на тяхната сложност за изпълнение и ефективност, както при самостоятелно приложение, така и в комбинация с други методи. Многостранният ефект на някои от методите е допълнително доказателство за взаимовръзката между отделните типове методи и необходимостта от комбинираното им използване.