УДК 004.45
Поморова О.В. д.т.н., проф., Фурман Ю.І. магістр
Хмельницький національний університет

ДОСЛІДЖЕННЯ ШВИДКОДІЇ АВТОМАТИЧНОЇ ПОБУДОВИ DOM-СТРУКТУР ДЛЯ WEB РЕСУРСІВ

Поморова О.В., Фурман Ю.І. Дослідження швидкодії автоматичної побудови DOM-структур для WEB ресурсів. В статті представлено результати дослідження особливостей автоматичної побудови DOM-структур web-браузерами та швидкодії браузерів. Розглядаються методи створення та результати побудови DOM-структур різними web-браузерами. Дослідження швидкодії проводиться шляхом визначення швидкості побудови DOM для різних типів web-сторінок. Аналізуються основні проблеми, що виникають при створенні DOM-структур.
Ключові слова: Структура web-документа, DOM-структура, об'єктна модель документа, оцінка швидкодії.

Поморова О.В., Фурман Ю.И. Исследование быстродействия автоматического построения DOM-структур для WEB ресурсов. В статье представлены результаты исследования особенностей автоматического построения DOM-структур web-браузерами и быстродействия браузеров. Рассматриваются методы создания и результаты построения DOM-структур разными web-браузерами. Исследование быстродействия проводится путем определения скорости построения DOM для различных типов web-страниц. Анализируются основные проблемы, возникающие при создании DOM-структур.
Ключевые слова: Структура web-документа, DOM-структура, объектная модель документа, оценка быстродействия.

Pomorova O.V., Furman Y.I. Research of speed for automatic construction of DOM-structures for WEB resources. The article presents the results of studies of the DOM-structures automatic construction by web-browsers and browser's performance. The methods of creating and the results of constructing DOM-structures of various web-browsers are considered. Research carried out by determining the speed of the DOM constructing for different types of web-pages. The main problems encountered when creating a DOM-structures are analyzed.
Keywords: Structure of web-document, DOM-structure, document object model, evaluation of performance.

Вступ. На сьогодні web-серфінг є невід'ємною частиною життя користувачів глобальних мереж. Доступ до сервісів електронної пошти, соціальних мереж та до інших web-сайтів користувачі отримують за допомогою спеціалізованих програм – web-браузерів. Основними їх функціями є: побудова DOM-структур, завантаження скриптів, стилів, рендеринг web-сторінок, і т.і. Користувачі надають перевагу тим браузерам, які швидко видають повний перелік web-ресурсів. Задачею браузера у цьому випадку є побудова структури сайту та автоматичне виявлення шаблонів розмітки, тому для розробників пошукових машин актуальним є пошук найбільш оптимальних алгоритмів, котрі швидко і ефективно будують DOM-структури сайтів [1].

DOM (Document object model) – об'єктна модель документа, котра є представленням документа у вигляді ієрархії структурних частин –вузлів. DOM розпізнає вміст web-сторінок і структурує їх в зручній для використання іншими засобами формі [2]. Також DOM надає можливість доступу до динамічної модифікації структури, змісту та оформлення документу. Вірно побудована DOM-структура надає пошуковій машині вичерпну інформацію про вміст web-ресурсу та суттєво зменшує час пошуку. DOM-структури різних ресурсів значно відрізняються між собою, тому актуальною задачею є приведення їх до одного шаблону та пошук однакових рис. Вирішення цієї задачі дозволить більш ефективно проводити аналіз web–ресурсів та зменшити час їх опрацювання.

Постановка задачі. Метою дослідження є виявлення особливостей автоматичної побудови DOM-структур різними web-браузерами та оцінка їх швидкодії.

Для досягнення цієї мети потрібно:

- дослідити відомі методи побудови DOM-структур;
- провести експериментальні дослідження із визначення швидкості побудови DOM-структур різними web-браузерами.

Методи побудови DOM-структур. DOM-структури дозволяють представляти web–ресурси у зручній для їх опрацювання формі, також вони надають можливість доступу до динамічної модифікації структури, змісту та оформлення web–документів. Кожен вузол DOM-структури розглядається як об'єкт із властивостями, методами і подіями, які підкоряються відношенню наслідування і об'єднуються в масиви чи колекції.

DOM – це програмний інтерфейс, що визначає:

- інтерфейси і об'єкти, що використовуються для представлення і маніпуляції web-документами;
- семантику всіх інтерфейсів і об'єктів, включаючи їх атрибути і реакцію на події;
- взаємозв'язки між цими інтерфейсами і об'єктами.

Варто зазначити, що web-браузери не зобов'язані використовувати DOM для виконання html-документу, але для реалізації динамічної зміни вигляду сторінки вона є обов'язковою, адже саме DOM є інструментом, який дозволяє скриптовим мовам програмування бачити вміст html-сторінок і стан web-браузера [1,3].

Є чотири рівні DOM, які стандартизував Консорціум W3C, кожен з них є доповненням іншого:

- рівень 0 – використовувався у web-браузерах Netscape і Internet Explorer. Містить деякі примітивні методи і властивості. Частково увійшов до стандарту HTML 4;
- рівень 1 – описує програмні інтерфейси для роботи з XML і HTML;
- рівень 2 – розширює опис інтерфейсів для XML, HTML, добавляє підтримку XHTML, описує події W3C, описує роботу з CSS. На сучасному етапі є основним рівнем;
- рівень 3 – розширює існуючі специфікації доповненнями, що призначені для спрощення представлення DOM і покращують взаємодію з сучасними web-технологіями.

Коректна побудова DOM є важливою у контексті швидкості і надійності доступу до web-контенту. Не дивлячись на її стандартизацію, різні web-браузери мають свої, специфічні особливості побудови DOM, які застосовуються для покращення функціонування, зручності використання та відображення web–документів. На сьогодні використовують два методи створення подібних структур:

- із використанням розпізнавання тегів;
- із використанням розпізнавання тегів і їх візуального представлення.

Метод, що використовує лише теги є досить простим. Більшість html тегів працюють в парі. Кожна пара складається з відкриваючого і закриваючого тегів (позначаються < > і < /> відповідно). В межах кожної тег–пари може знаходитись різна кількість інших тег-пар, але всі вони відповідають визначеній html-документом структурі. Html-сторінка будується за певними правилами, тому створити на основі її коду DOM-структуру у вигляді дерева досить просто. В дереві кожна пара тегів представлена у вигляді вузла, а вкладені в них теги – нащадками даного вузла.

Перед побудовою дерева з html-кодом потрібно провести певні дії:

- «очищення» html-коду. Деякі теги не потребують закриваючих тегів (наприклад <li>,<hr>,<p>), хоча вони і мають їх. Для використання даного методу побудови DOM-структури кількість закриваючих і відкриваючих тегів повинна бути збалансована. Погано форматовані теги теж повинні бути виправлені. Тут виникає проблема з вкладеністю, адже знайти помилку (відсутність тегу) у багаторівневій структурі досить складно. Для цього часто застосовують стороннє програмне забезпечення.
- Побудова дерева. DOM-структура у вигляді дерева будується наступним чином: в першу чергу відбувається витяг токенів (значимих частин, що відповідають певній структурі), починаючи з початку документу. Для цього створюється порожній стек *S*, котрий містить усю інформацію про предків елемента DOM-структури, що обробляється. Після зчитування стартового тегу <*e*>, для даного елемента створюється вузол d_e . Будь-які атрибути чи значення, пов'язані з даною парою тегів аналізуються і зберігаються, створюючи таким чином вузол у DOM-структурі. Якщо стек *S* не пустий, це означає що вузол d_e вже створений. Він стає крайнім правим дочірнім вузлом створеного раніше батьківського вузла. Створена пара (d_e, *e*) заноситься в стек. Якщо *e* – заключний текст, для нього створюється окремий вузол в DOM-структурі і прив'язується як «текстовий» дочірній до вузла d_e. Аналіз продовжується поки не буде зчитаний закриваючий тег </*e*>. Далі здійснюється перевірка на відповідність відкриваючого тегу, що знаходиться на вершині стека і закриваючого тегу.
- У випадку неспівпадання імен подальший аналіз документу припиняється, і він вважається некоректно сформованим. В іншому випадку вершина стеку очищається і аналіз продовжується. Після обробки останнього символу документу, якщо стек *S* – порожній, то DOM-

структура сформовано коректно, інакше вхідний документ потребує додаткового «очищення» і виправлення помилок [4].

Цей метод застосовується до більшості web-сторінок. Однак, деякі погано форматовані теги важко виявити, що призводить до некоректної побудови DOM-структури і створює труднощі при подальшому витягу інформації.

Метод, що використовує теги і їх візуальне представлення використовує візуальне представлення тегів (положення на екрані), для більш коректного розпізнавання зв'язків між ними. Через те, що браузери при рендеренгу web-сторінки більш толерантно відносяться до помилок, цей метод допомагає надійніше і точніше сформувати DOM-структуру. DOM-структура завжди буде коректно будуватись, якщо web-браузер правильно створюватиме візуальне представлення web-сторінки. У web-браузері кожен html-елемент відображається у вигляді прямокутника. Тобто, DOM-структура може формуватись на основі вкладених прямокутників у візуальному представлені web-сторінки. Для цього потрібно виконати наступні дії [2]:

- визначити крайні точки прямокутника кожного html-елемента, що будується браузером;
- дотримуючись послідовності відкриваючих тегів, провести перевірку вкладеності, тобто перевірити чи входить прямокутник, що досліджується, у площу розташування іншого прямокутника.

Даний метод є більш надійним і застосовується більшістю web-браузерів. Однак виникають ситуації коли і за допомогою нього неможливо побудувати DOM- структуру, у такому випадку проводиться більш ретельна перевірка web-документа на коректність і виправляються помилки.

Особливості побудови DOM-структур сучасними web-браузерами. Розглянемо особливості створення DOM-структур найбільш популярними на сьогодні web-браузерами, такими як: Internet Explorer 8, Google Chrome, Firefox 36 і Opera 12. Дані web-браузери використовують метод побудови DOM-структур з використанням візуального представлення і розпізнавання тегів, але кожен з них має свої особливості, які досить суттєво впливають на швидкість створення дерева і ефективність його подальшого використання. Основною такою особливістю є парсинг – метод розпізнавання пробілів, переходів на новий рядок, і в цілому тексту.

Розглянемо метод парсингу для кожного браузера окремо. Для цього побудуємо і використаємо Шаблон 1.

Шаблон 1. Тестова Html-сторінка

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
        <html>
   <head>
              <title>Документ</title>
     <script>window.onload = function() { alert(document.body.childNodes.length) }</script>
           </head>
           <body>
              <div id="id ">Дані</div>
              <ul>
                 <li style="background-color:blue">Рядок 1</li>
                 <li class="string2">Рядок 2</li>
              </ul>
              <div id="testr">Тест;</div>
           </body>
        </html>
```

Після відкриття Шаблону 1 в різних браузерах були отримані результати, відображені на рисунку 1.

Отже, результати побудови DOM-структур в браузерах відрізняються, хоча і не досить суттєво. Основною відмінністю є саме розпізнавання переходу на новий рядок. Internet Explorer, будуючи DOM-структуру, забирає непотрібні знаки переходу на новий рядок, в той час як Google Chrome і Firefox навпаки, чітко акцентують на них увагу і представляють їх у вигляді дочірніх вузлів. Браузер Opera використовує специфічний алгоритм, за яким він може додати пустий елемент для своїх внутрішньо програмних цілей.

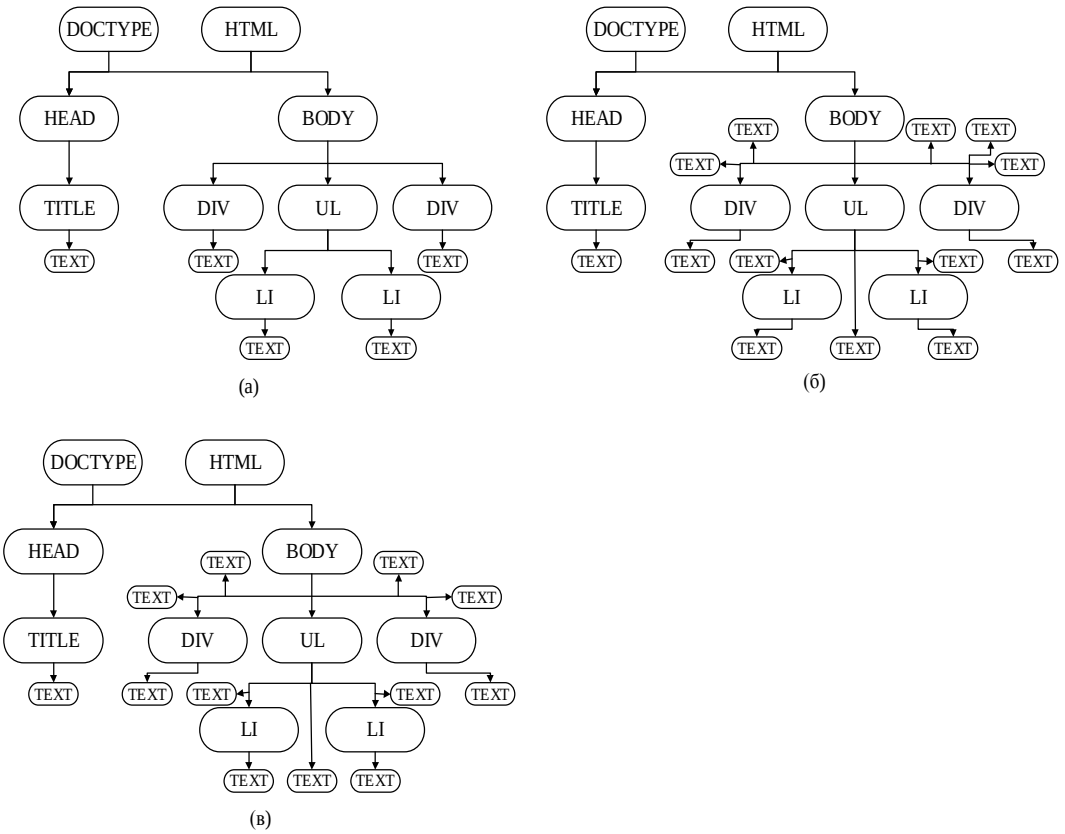

Рис. 1 DOM-структури побудовані за Шаблоном 1:
(а) – в IE, (б) – в Opera, (в) – в Google Chrome і Firefox

Дослідження швидкодії автоматичної побудови DOM-структур для web-ресурсів. Швидкість побудови DOM-структури є досить важливим показником для web-браузерів, адже чим швидше побудується DOM, тим швидше запрацюють скрипти і відобразяться всі елементи web-сторінки. Оцінити швидкість автоматичної побудови DOM-структур досить важко, адже не дивлячись на їх стандартизацію, вони є унікальними для кожного web-браузера. Дослідимо швидкість побудови DOM-структури в залежності від вмісту html-документа і його об'єму.

Тестування проведемо наступним чином: використаємо два html-документи: перший файл представляє собою лінійну структуру, тобто в тегові <body> створимо теги один за одним, у другому файлі тег представимо в деревовидній структурі з нарощуванням вкладеності. Нарощування проводитимемо поки кількість елементів не досягне певних точок: 50, 100, 500, 1000 і 5000 елементів відповідно. При досягнені цих величин проведемо побудову DOM-структури і визначимо час, за який вона буде побудована.

Отримані результати представлені на рисунку 2. На осі x вказано кількість елементів DOM-структури, на осі y – час в мілісекундах.

Отримані результати суттєво відрізняються. І хоча часові затримки здаються досить малими, але зі збільшенням кількості елементів у html-документі час побудови DOM-дерева збільшується і після 1000 елементів падіння швидкості відбувається за лінійною залежністю.

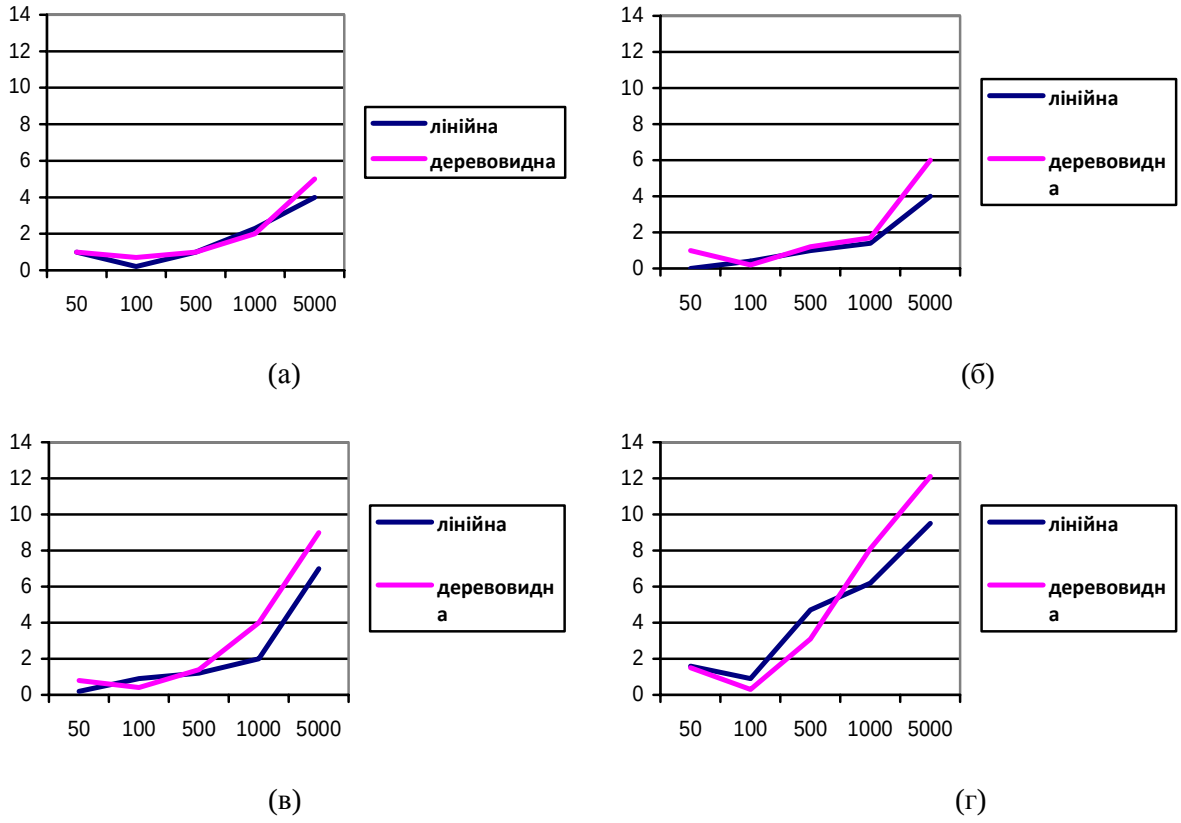

Рис. 2 Швидкість створення DOM-структур:
(а) – в Opera 12, (б) – в Firefox 36, (в) – в Google Chrome, (г) – в IE 8

Висновок. Сучасні web-браузери використовують методи, які дійсно швидко створюють DOM-структури для web-сторінок довільної складності. Результати дослідження особливостей побудови DOM-структур різними web-браузерами та швидкодії браузерів показали, що найбільш нормалізована DOM-структура була створена браузером Internet Explorer 8, але за швидкістю виконання цього процесу цей web-браузер помітно програє своїм конкурентам. Google Chrome і Firefox орієнтовані на прискорення побудови DOM-структури і збільшення її в об'ємі, що допомагає цим web-браузерам досягти більш швидкого завантаження web-сторінок.

Однак, при великій кількості елементів в html-документі, усі web-браузери починають витрачати час (спочатку лише декілька мілісекунд) на створення дерева, і цей час зростає при збільшенні кількості елементів web-сторінки.

Отже, при великій кількості елементів web-сторінок, створення DOM-структури займає десятки секунд, що є досить критичним чинником на сьогодні.

Автори вважають перспективними підходами до підвищення швидкодії web-браузерів при відображенні web-ресурсів застосування більш ефективних алгоритмів для побудови DOM-структур із використанням розпаралелювання, кращого парсингу, більшої толерантності до помилок і вимогою зменшення кількості рівнів лінійної та деревовидної структур.

1. Document Object Model (DOM) [Електронний ресурс] : / Philippe Le Hégaret, Ray Whitmer. – 2005. [Електронний ресурс]. – Режим доступу: http://www.w3.org/DOM
2. C-H. Chang and S. Lui. IEPAD. Information Extraction Based on Pattern Discovery.In Proc. of the Tenth Intl. World Wide Web Conf. (WWW'01), 2001.– 720 c.
3. S. Chakrabarti. Mining the Web. Discovering Knowledge from Hypertext Data. – Morgan Kaufmann, 2003.–345c.
4. W. Chen. New Algorithm for Ordered Tree-to-Tree Correction Problem. [J. Algo-rithms], 2001.–280c.