УДК 004.023

Н. С. Каргин, Н. И. Гурин
Белорусский государственный технологический университет

ИЗВЛЕЧЕНИЕ ОСНОВНОГО СОДЕРЖИМОГО ИЗ ВЕБ-СТРАНИЦ НА ОСНОВЕ АНАЛИЗА ВИЗУАЛЬНЫХ ХАРАКТЕРИСТИК ЭЛЕМЕНТОВ И ПРЕОБРАЗОВАНИЯ В JSON ФОРМАТ

В статье рассматриваются алгоритмы извлечения основного содержимого из веб-страниц и предлагается метод решения проблем, затрудняющих извлечение основного содержимого, на основе визуальных характеристик и внутреннего содержимого элементов страницы. В разработанном методе основное содержимое определяется одним корневым элементом и преобразуется в JSON формат, содержащий однозначные типы данных, описывающих абзацы, заголовки, изображения, видеозаписи, галереи и другие элементы страницы. Для отображения JSON формата не требуется браузер, что значительно расширяет его возможности применения в мобильной и встраиваемой технике ввиду большей эффективности. Применение в методе поиска корневого элемента позволяет улучшить качество и ускорить извлечение основного содержимого при обработке большого количества веб-страниц одного сайта и использовании персистентного хранилища для обработанных страниц.

Ключевые слова: веб, браузеры, HTML, CSS, JSON.

Для цитирования: Каргин Н. С., Гурин Н. И. Извлечение основного содержимого из веб-страниц на основе анализа визуальных характеристик элементов и преобразования в JSON формат // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2021. № 1 (242). С. 54–60.

N. S. Kargin, N. I. Gurin
Belarusian State Technological University

EXTRACTING THE MAIN CONTENT FROM WEB PAGES BY ANALYSING THE VISUAL CHARACTERISTICS OF THE ELEMENTS AND CONVERTING TO THE JSON FORMAT

The article discusses the algorithms of extraction of the main context from web pages. In addition, the article proposed the method of solving problems related to the difficulties of extraction of the main content. This method is based on the visual characteristics and internal content of web page elements. In the developed method the main content is defined by a single root element; this root element is converted to a JSON format containing unambiguous data types describing paragraphs, titles, images, videos, galleries, and other web page elements. The web browser is not required to display the JSON format; and this fact significantly expands its application capabilities in mobile and embedded technologies due to greater efficiency. Using the root element in the search method allows you to improve the quality of the extraction of the main content. Besides this, it speeds up the extraction during processing a large number of web pages on a single site and using permanent storage for the processed pages.

Key words: web, browsers, HTML, CSS, JSON.

For citation: Gurin N. I., Sahon E. S. Extracting the main content from web pages by analysing the visual characteristics of the elements and converting to the JSON format. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics*, *2021*, no. 1 (242), pp. 54–60 (In Russian).

Введение. Современные веб-стандарты и развитие инструментов для веб-разработки значительно упростили публикацию веб-документов, позволяя при этом достигать одинакового визуального результата десятками различных способов. Однако такое разнообразие стандартов и различных подходов к разработке веб-страниц усложняет автоматизированный анализ основного содержимого. При этом задача извлечения основного содержимого является базовой:

1) для новостных агрегаторов;
2) поисковых систем;
3) приложений для людей с ограниченными возможностями (озвучивание основного содержимого, изменение визуального представления при проблемах со зрением);
4) приложений, связанных с обработкой естественного языка (классификация текста по темам, создание краткого содержания и др.).

Средний размер веб-страниц стремительно растет уже десять лет. По данным HTTPArchive, на август 2020 г. медианное значение размера веб-страницы со всем его содержимым превышает 2 мегабайта [1]. Такой объем данных

снижает доступность веб-сайтов для пользователей с ограниченным каналом связи. Для решения данной проблемы появляются такие технологии, как Google AMP, Yandex Турбо, позволяющие пользователям получать облегченные версии страниц, содержащих только контент [2, 3]. В этой связи развитие и применение подобных технологий для извлечения основного содержимого веб-страниц является весьма актуальной задачей.

Процесс извлечения основного содержимого затрудняется тем, что веб-документы содержат в себе множество неинформативной или нетекстовой информации: CSS стили, JavaScript код, навигационные и декоративные элементы, генерируемый пользователями контент (например, комментарии).

Для решения данной проблемы в настоящее время применяются два основных подхода: на основе правил и семантики веб-документов.

Подход на основе правил показывает хороший результат, но при этом является крайне трудоемким и неустойчивым к изменению верстки веб-страниц, что значительно усложняет его использование.

Подходы на основе семантики веб-документов ориентируются на название элементов, классов, идентификаторов, размещении в веб-документе, однако современные стандарты HTML и JavaScript уже сейчас позволяют разработчикам создавать собственные элементы разметки с собственными названиями. При бурном развитии веб-технологий и браузеров данный подход становится неэффективным, по крайней мере на страницах, на которых применяются собственные элементы.

В работе предлагаются собственные алгоритмы извлечения основного содержимого из веб-страниц и методы решения проблем, затрудняющих извлечение основного содержимого на основе визуальных характеристик и внутреннего содержимого элементов страницы.

Основная часть. Базовая структура HTML документа включает в себя:

– DOCTYPE, декларация о типе документа;

– HTML, корневой элемент страницы;

– HEAD, элемент, содержащий метаинформацию о странице, подключаемые файлы скриптов, стилей и т. д.;

– BODY, элемент, выступающий контейнером для всего видимого содержимого.

HTML элементы определяются открывающимся и закрывающимся тегом. Некоторые элементы, например BR, не содержат закрывающегося тега и являются «пустыми».

HTML разметка определяет содержание и структуру веб-контента; другие технологии обычно используются для описания внешнего вида (CSS) или поведения веб-страницы (JS).

Сложности при извлечении основного содержимого:

1) множество неинформативной или нетекстовой информации (CSS, JS);

2) скрытые элементы, обеспечивающие правильную работу веб-страницы;

3) источником данных не всегда является HTML разметка;

4) применение несемантической верстки веб-разработчиками;

5) неоднозначность элементов;

6) вариативность подходов к верстке.

Веб-разработчики часто скрывают какие-то данные, связанные с работой веб-страницы, в скрытые элементы TEXTAREA, INPUT или прячут блоки за видимой областью экрана. Такие элементы могут располагаться в любой части страницы или, например, в элементе статьи (рекламный блок). Исключение таких элементов – дополнительная сложность при извлечении основного содержимого.

Распространение SPA (Single Page Application) – одностраничных приложений – и инструментов для их разработки вносит еще одну сложность с правильной интерпретацией содержимого страницы [4]. В таких веб-приложениях источник данных (например, содержимое статьи) может располагаться в JavaScript коде в формате JSON или подгружаться через XMLHttpRequest, Fetch API. Для автоматической обработки и извлечения информации из таких документов недостаточно только обработчика HTML, необходим полноценный браузер. Использование браузера требует отдельных ресурсов: процессорное время, оперативная и видеопамять. Это может быть критичным в приложениях, в которых извлечение основного содержимого происходит не на стороне клиента, а на стороне сервера.

Проблема кросс-платформенности веб-приложений касается и приложений, связанных с извлечением основного содержимого. Неоднозначность интерпретации некоторых элементов, JavaScript или CSS может происходить и на уровне различных браузеров.

HTML позволяет разработчикам верстать страницы абсолютно разными способами и по разным методологиям, но получать одинаковый визуальный результат в браузере. Стандарты дают возможность разработчикам размещать текстовые узлы рядом с блочными, оборачивать их и т. д., но для правильной интерпретации таких сценариев требуется веб-браузер, чтобы учитывать визуальную составляющую веб-страницы.

Семантические HTML элементы описывают свое смысловое значение для браузера и разработчика. Например, к семантическим элементам

можно отнести: article, header, main, summary; к несемантическим: div, span. По названию семантического элемента можно понять его содержимое. Для того чтобы проанализировать возможность использования семантической верстки в качестве «индикатора» для поиска основного содержимого статьи, были проанализированы топ-15 новостных сайтов в мире по состоянию на август 2020 г. [5]. В табл. 1 приведен результат анализа верстки сайтов.

Таблица 1

Результат анализа верстки 15 самых популярных новостных сайтов в мире

Сайт	Семантические элементы
news.yahoo.com	Используются
huffpost.com	То же
cnn.com	Элемент article включает лишнее содержимое
nytimes.com	То же
foxnews.com	– // –
nbcnews.com	Используются
dailymail.co.uk	Отсутствуют
washingtonpost.com	Используются
theguardian.com	То же
wsj.com	Элемент article включает лишнее содержимое
abcnews.go.com	То же
bbc.com	– // –
usatoday.com	– // –
latimes.com	Используются

В результате анализа самых популярных новостных сайтов выявлено, что на одном сайте семантические элементы отсутствуют полностью, на семи сайтах семантические элементы применяются, но проанализировать веб-страницы только на основе данного признака и извлечь основное содержимое (текст новости) было бы невозможно, так как семантические элементы включали в себя лишнее содержимое (комментарии, рекламу, рекомендации и т. д.).

Таким образом, более чем 50% из выборки не используют семантическую верстку или используют ее таким образом, что смысловое значение элемента не передает его содержимое.

Для решения задачи извлечения основного содержимого разработан метод, который ориентируется на визуальные характеристики элементов, текстовые узлы документа и стремится определить только один корневой элемент с основным содержимым веб-страницы. Основные шаги метода:

1) сбор метаинформации веб-страницы;

2) формирование множества всех текстовых узлов веб-страницы;

3) определение корневого элемента с основным содержимым;

4) преобразование корневого элемента в JSON формат.

Все шаги выполняются в браузере с помощью подключения JS скриптов в тело страницы. Скрипт извлечения основного содержимого может быть выполнен как на сервере, например в среде NodeJS с применением библиотеки Puppeteer, так и в браузере пользователя с помощью расширений [7].

Сбор метаинформации веб-страницы. На данном шаге формируется JS объект, содержащий метаинформацию веб-страницы: заголовок, описание, изображение, время публикации, время редактирования, название хоста и сайта, ссылку на документ, иконку сайта. Для извлечения данной информации используются теги из элемента HEAD по стандартам OpenGraph [8] или их альтернативы. В табл. 2 приведены названия полей объекта с метаинформацией и объекты, из которых происходит извлечение.

Таблица 2

Объект метаинформации о странице

Поле объекта	Место извлечения
title	Метатеги og:title, twitter:title, title
description	Метатеги description, og:description
publishedTime	Метатег article:published_time
modifiedTime	Метатег article:modified_time
image	Метатег og:image, twitter:image
host	Переменная window.location.host
siteName	Метатеги og:site_name, apple-mobile-web-app-title, application-name
href	Переменная window.location.href
favicon	Элементы типа LINK apple-touch-icon, icon

Данная информация может использоваться для отображения краткой информации о странице, группировке по сайтам, времени публикации и т. д.

Формирование множества текстовых узлов. Скрипт извлечения делает обход всех элементов документа. Если текстовый узел содержит более одного слова, то он попадает в множество.

Элемент и его дочерние текстовые узлы не обходятся и не попадают в множество при наступлении одного из следующих условий:

1) название элемента style, link, script, noscript, input, textarea, svg, button и некоторые другие «технические» элементы;

2) в названии класса элемента или его идентификатора есть вхождения со словами comment, related, tags, date, share, author, promo и некоторые др.;

3) реальный визуальный размер элемента меньше 8 пикселей в ширину или высоту и свойство display не равен inline или inline-block;

4) элемент расположен за границами видимой области браузера;

5) элемент визуально скрыт (display равен none или hidden, opacity равно 0).

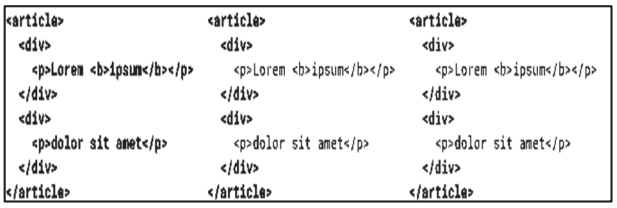
Определение корневого элемента с основным содержимым. Данный этап алгоритма может выполняться несколько раз в случае, если корневой элемент не обнаружен (в реализации алгоритма 3 раза максимум). В список кандидатов на роль корневого элемента попадают все элементы из множества, сформированного на предыдущем этапе (при первом выполнении – это все элементы из множества текстовых узлов), плюс их родительские элементы.

Для каждого кандидата высчитывается количество слов, которое он содержит. Список кандидатов сортируется по убыванию по данному признаку. Затем из списка кандидатов удаляются все элементы, которые меньше порога – процентное значение от наибольшего элемента (в реализации алгоритма используется 80%).

Если в списке остался только один кандидат, то он и будет являться корневым для основного содержимого веб-страницы. Если в списке более одного элемента, то происходит проверка на родственность кандидатов. Родственными считаются элементы, которые являются родителями или потомками друг друга. Если элементы не являются родственными между собой, то считается, что корневой элемент не обнаружен и происходит повторное выполнение данного шага при условии, что не превышен лимит поиска. Если элементы являются родственными, то корневым считается элемент с наибольшим количеством слов.

Схематичное изображение работы данного шага и его итераций представлено на рисунке. Повторный проход необходим для того, чтобы избежать ситуации, при которой абзацы текста имеют двойную или тройную вложенность в другие HTML элементы.

Данную проблему можно было бы решить, применяя алгоритм упрощения DOM дерева: если у элемента только один дочерний, то содержимое данного элемента заменяется на содержимое его дочернего элемента. Однако применение такого алгоритма приводило бы к тому, что элементы теряли свои визуальные характеристики, так как CSS селекторы являлись неактуальными для обновленного DOM дерева.



Пример итераций шага поиска корневого элемента

Преобразование корневого элемента в JSON формат. Целью данного преобразования является упрощение сложных и неоднозначных HTML элементов в простые и однозначные JSON объекты, которые могут быть обработаны другими программами без применения XML-парсера или браузера.

Основные шаги преобразования:

1) очистка корневого элемента по тем же правилам, по которым исключались элементы при формировании множества текстовых узлов;

2) формирование массива абзацев текста;

3) преобразование дочерних элементов в альтернативные JSON объекты.

Формирование массива абзацев текста происходит следующим образом: содержимый текст (свойство innerText у HTMLElement) корневого элемента очищается от повторяющихся и лишних пробелов и переносов текста, разбиение на массив происходит по символу переноса текста. Каждый абзац сопоставляется с узлом страницы, в котором он начинается и заканчивается. В процессе сопоставления для абзаца формируется массив атрибутов, применяемых к тексту. Атрибут представляет собой объект, содержащий тип форматирования (жирный, курсив, подчеркивание, ссылка), числовые позиции from и to, определяющие границы форматирования. Ссылки дополнительно содержат поле href. Все атрибуты текста, за исключением ссылки, определяются на основе визуальных характеристик (CSS свойства font-weight, font-style).

После формирования списка абзацев с их атрибутами происходит обход всех дочерних элементов корневого узла. На данном шаге HTML элементы соотносятся с JSON альтернативами, представленными в табл. 3.

Важно отметить, что этапы формирования множества текстовых узлов и поиска корневого элемента могут быть пропущены при условии, что в процессе преобразования других страниц с этого же веб-сайта происходит сохранение XPath или CSS селектора к корневому элементу с основным содержимым.

Таблица 3

Описание типов элементов в JSON формате

Тип элемента	Альтернативный HTML элемент	Описание
header	H1, H2, H3, H4, H5, H6	Заголовок. Дополнительно содержит поле l, указывающее на уровень заголовка (от 1 до 6). В поле text находится текстовое содержимое элемента
paragraph	Наиболее близкий P, но может быть любой блочный элемент	Абзац текста. В поле text находится текстовое содержимое элемента. Может содержать массив attributes
image	FIGURE с дочерними элементами IMG и FIGCAPTION	Изображение. Содержит в себе: 1) url – адрес изображения 2) size – объект с шириной и высотой изображения 3) thumbnail – превью изображения в строке base64 с максимальным размером 40 пикселей по большей стороне 4) caption – массив с объектами типа paragraph, подпись к изображению
list	UL, OL	Список. В поле children содержит массив объектов paragraph. В поле style указывается стиль списка (нумерованный, ненумерованный)
remoteVideo	IFRAME	Видео из сервиса YouTube или Vimeo. Содержит: 1) information с объектом, содержащим название сервиса и идентификатор видео 2) caption – массив с объектами типа paragraph, подписи к видео
video	VIDEO	Видео. Содержит: 1) src, указывающее на адрес видеофайла 2) булево поле isAnimation (true, если у элемента есть атрибут loop) 3) ratio – соотношение сторон видео 4) caption – массив с объектами типа paragraph, подписи к видео
audio	AUDIO	Аудио. Содержит поле src со ссылкой на файл и поле caption – массив объектов типа paragraph
gallery	Прямой альтернативы нет	Галерея изображений. Содержит поле caption с объектами типа paragraph и поле images с объектами типа image
delimiter	HR, но может быть любой элемент, представляющий собой горизонтальную линию	Горизонтальный разделитель

Большинство контент-ориентированных веб-сайтов используют шаблонизаторы для рендеринга HTML и у страниц меняется только часть с основным содержимым. Следовательно, при накоплении некоторого количества таких селекторов для новых страниц искать этот элемент уже нет необходимости: достаточно получить из базы данных наиболее часто встречающийся путь, что позволяет ускорить извлечение основного содержимого.

Разработанный метод был протестирован на 111 веб-страницах с 25 различных веб-сайтов, включая упомянутые ранее популярные веб-сайты из табл. 1. Тестирование включало в себя ручное сравнение извлеченного содержимого и оригинальных веб-страниц.

В результате тестирования было выявлено 6 веб-страниц, для которых не удалось корректно определить корневой элемент, что составляет 5,41% от всех страниц. Следует учитывать, что сканирование выполнялось для небольшого количества веб-страниц с одного веб-сайта и не применялась оптимизация с «запоминанием» корневого элемента для веб-сайта.

На 8 веб-страницах были выявлены недостатки, связанные с некорректным преобразованием рекламных блоков или «ленивых» изображений (изображения, которые начинают загружаться только при приближении видимой области окна браузера к ним).

Для реализации ленивых изображений применяются различные техники с помощью JavaScript, а HTML в исходной разметке выглядит исключительно как заглушка и данную проблему нельзя решить только с помощью анализа HTML. Для ее решения была реализована функция,

которая выполняет прокрутку страницы до конца корневого элемента с основным содержимым и ожидает загрузку изображений в нем.

Для оценки эффективности представления основного содержимого в JSON формате был произведен замер исходных HTML документов (без учета подключаемого JavaScript, CSS) и преобразованного основного содержимого:

– средний размер HTML документа составляет 291 килобайт;

– средний размер извлеченного содержимого в JSON формате 16 килобайт.

Таким образом, представление основного содержимого в JSON формате в 18 раз эффективнее. Минимальный размер JSON документа составил 3 килобайта, максимальный – 63; минимальный размер HTML документа 16 килобайт, максимальный размер – 1462 килобайта. При этом следует учитывать, что JSON документы уже содержат подробную информацию об изображениях и их уменьшенные копии.

Было проведено сравнительное тестирование разработанного метода и реализация алгоритма Readability в браузерах Mozilla Firefox и Apple Safari. Для данного тестирования было разработано две визуально одинаковые страницы. Одна из страниц использует стандарт собственных элементов для представления содержимого, а вторая применяет семантические HTML элементы. Разработанные веб-страницы содержат небольшой текст, навигационные элементы и комментарии.

Браузер Firefox не смог определить элемент с основным содержимым на странице, на которой присутствуют собственные элементы. На странице с семантическими HTML элементами проблемы не возникло.

Браузер Safari смог корректно определить элемент с основным содержимым для двух страниц, но не смог корректно отобразить его на странице, на которой присутствуют собственные элементы: все подряд идущие абзацы текста сливаются в один.

Разработанный метод корректно определил и преобразовал основное содержимое на двух страницах, так как не ориентируется на семантику HTML элементов.

Заключение. Технологии извлечения основного содержимого веб-страниц встречаются в самых разных приложениях и сервисах, которыми пользуются миллионы людей ежедневно: поисковые системы, новостные агрегаторы, приложения для чтения новостей, мессенджеры и многие др.

Разработанный метод извлечения основного содержимого на основе визуальных характеристик и внутреннего содержимого элементов не ориентируется на семантику элементов HTML документа, что делает алгоритм устойчивым к разнообразию подходов к верстке и применению на странице собственных элементов.

Выделение корневого элемента с основным содержимым веб-страницы позволяет улучшить качество работы алгоритма (неправильное определение основного содержимого, скорость работы) при обработке большого количества страниц из одного источника.

Конвертирование основного содержимого в JSON формат позволяет избавиться от сложности и неоднозначности HTML разметки, что значительно упрощает машинную обработку основного содержимого, а для отображения или озвучивания статей в таком формате не требуется браузер, что особенно актуально для мобильных устройств и встраиваемой техники с ограниченными ресурсами.

Список литературы

1. State of the Web [Электронный ресурс] // HTTPArchive, 2020. URL: https://httparchive.org/reports/state-of-the-web (дата обращения: 05.11.2020).

2. AMP on Google [Электронный ресурс] // Google Developers. URL: https://developers.google.com/amp (дата обращения: 05.11.2020).

3. Турбо-страницы для владельцев сайтов [Электронный ресурс] // Яндекс. URL: https://yandex.ru/adv/turbo (дата обращения: 05.11.2020).

4. SPA (Single-page application) [Электронный ресурс] // MDN. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (дата обращения: 05.11.2020).

5. Top 15 Most Popular News Websites [Электронный ресурс] // eBiz, 2020. Август. URL: http://www.ebizmba.com/articles/news-websites (дата обращения: 05.11.2020).

6. Custom Elements [Электронный ресурс] // W3C, 2018. 3 мая. URL: https://www.w3.org/TR/custom-elements/ (дата обращения: 05.11.2020).

7. Puppeteer v 5.4.1 [Электронный ресурс] // Puppeteer. URL: https://pptr.dev (дата обращения: 05.11.2020).

8. The Open Graph protocol [Электронный ресурс] // Facebook, 2010. URL: https://ogp.me (дата обращения: 05.11.2020).

References

1. State of the Web. Available at: https://httparchive.org/reports/state-of-the-web (accessed 05.11.2020).

2. AMP on Google. Available at: https://developers.google.com/amp (accessed 05.11.2020).

3. Turbo-stranitsy dlya vladel'tsev saytov [Turbo pages for website owners]. Available at: https://yandex.ru/adv/turbo (accessed 05.11.2020).

4. SPA (Single-page application). Available at: https://developer.mozilla.org/en-US/docs/Glossary/SPA (accessed 05.11.2020).

5. Top 15 Most Popular News Websites. August. 2020. Available at: http://www.ebizmba.com/articles/news-websites (accessed 05.11.2020).

6. Custom Elements. 3 May. 2018. Available at: https://www.w3.org/TR/custom-elements/ (accessed: 05.11.2020).

7. Puppeteer v 5.4.1. Available at: https://pptr.dev (accessed 05.11.2020).

8. The Open Graph protocol. 2010. Available at: https://ogp.me (accessed 05.11.2020).

Информация об авторах

Каргин Николай Сергеевич – магистрант кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: hello@karh.in

Гурин Николай Иванович − кандидат физико-математических наук, доцент кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: ngourine@mail.ru

Information about the authors

Kargin Nikolay Sergeevich – Master's degree student, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: hello@karh.in

Gurin Nikolay Ivanovich − PhD (Physics and Mathematics), Assistant Professor, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: ngourine@mail.ru

Поступила после доработки 12.01.2021