Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Анализ алгоритмов построения, обновления и отображения гипертекстового документа при помощи виртуальной объектной модели»

Студент группы **ИУ7-56Б** _____ **П. А. Калашков**
(Подпись, дата) (И.О. Фамилия)

Руководитель _____ **Д. Е. Бекасов**
(Подпись, дата) (И.О. Фамилия)

***2022** г.*

СПИСОК ИСПОЛНИТЕЛЕЙ

Расчетно-пояснительная записка 25 с., 5 рис., 0 табл., X ист., X прил.

КЛЮЧЕВЫЕ СЛОВА

РЕФЕРАТ

Расчетно-пояснительная записка 25 с., 5 рис., 0 табл., X ист., X прил.

КЛЮЧЕВЫЕ СЛОВА

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящем отчете о НИР использованы ссылки на следующие стандарты:

1) DOM Living standart [1]
2) HTML Living standart [4]

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

HTML — HyperText Markup Language — язык гипертекстовой разметки

DOM — Document Object Model — объектная модель документа

ВВЕДЕНИЕ

Работа с гипертекстовыми документами является неотъемлемой частью жизни каждого человека, пользующегося Всемирной сетью. Часто возникает потребность в просмотре различных гипертекстовых документов, а также в выполнении операций, приводящих к их изменению. Возникает вопрос: каким образом стоит отображать документ, и производить операции его обновления и построения?

Для взаимодействия с гипертекстовыми документами, входящими в сеть Интернет, существуют программы-браузеры. Преимущественнная часть браузеров использует стандарт [1], обеспечивающий использование объектной модели документа [2].

Целью данной работы является анализ алгоритмов построения, обновления и отображения гипертекстового документа при помощи виртуальной объектной модели. Для достижения поставленной цели необходимо выполнить следующие задачи:

1) изучить принципы работы объектной модели документа;
2) изучить принципы работы виртуальной объектной модели документа;
3) сравнить и проанализировать трудоёмкости алгоритмов с использованием объектной модели документа и виртуальной объектной модели документа на основе теоретических расчётов.

1 Аналитическая часть

1.1 Гипертекстовые документы

Гипертекстовым документом является документ, состоящий из текстовых страниц, имеющих перекрёстные ссылки. В данной работе под гипертекстовыми документами будут подразумеваться документы, написанные при помощи языка гипертекстовой разметки (англ. *HyperText Markup Language* — HTML) [3], а именно документы, соблюдающие стандарт HTML5 [4].

Данный выбор обусловлен тем, что использование HTML5 получило широкое распространение в Всемирной сети благодаря рекомендации [5] к использованию от Консорциума Всемирной паутины (англ. *World Wide Web Consortium* — W3C). Вследствие данной рекомендации HTML документы поддерживают большинство самых распространённых браузеров в России, такие, как Google Chrome, Яндекс.Браузер и Safari.

1.2 Объектная модель документа

Объектная модель документа (англ. *Document Object Model* — DOM) [2] — программный интерфейс для HTML, XML и CSV документов. Он обеспечивает структурированное представление документа (дерева), и определяет способ, по которому структура может быть доступна для программы, для изменения структуры документа, его стиля и содержания. Представление DOM состоит из структурированной группы узлов и объектов, которые имеют свойства и методы.

Стадарт W3C DOM [1] формирует основы DOM, реализованные в большинстве современных браузеров. Многие браузеры предлагают расширения за пределами данного стандарта, поэтому необходимо проверять работоспособность тех или иных возможностей DOM для каждого конкретного браузера.

Для описания структуры DOM потребуются следующие термины: корневой, родительские и дочерние элементы. Корневой элемент находится в основании всей структуры и не имеет родительского элемента. Дочерние элементы

не просто находятся внутри родительских, но и наследуют различные свойства от них. Рассмотрим, как выглядит DOM-представление следующего HTML документа.

Листинг 1: Пример простого HTML документа

```
<!DOCTYPE HTML>
<html>
<head>
  <link/>
  <meta/>
  <title/>
</head>
<body>
  <header/>
  <section>
    <div/>
    <p/>
    <a/>
  </section>
  <footer/>
</body>
</html>
```

Корневым элементом здесь является *html*, он не имеет родительского элемента и имеет два дочерних — *head* и *body*. По отношению друг к другу элементы *head* и *body* являются сиблингами (братьями и сестрами). В каждый из них можно вложить еще много дочерних элементов. Например, в head обычно находятся *link*, *meta*, *script* или *title*.

Данный HTML документ будет иметь следующее DOM-представление:

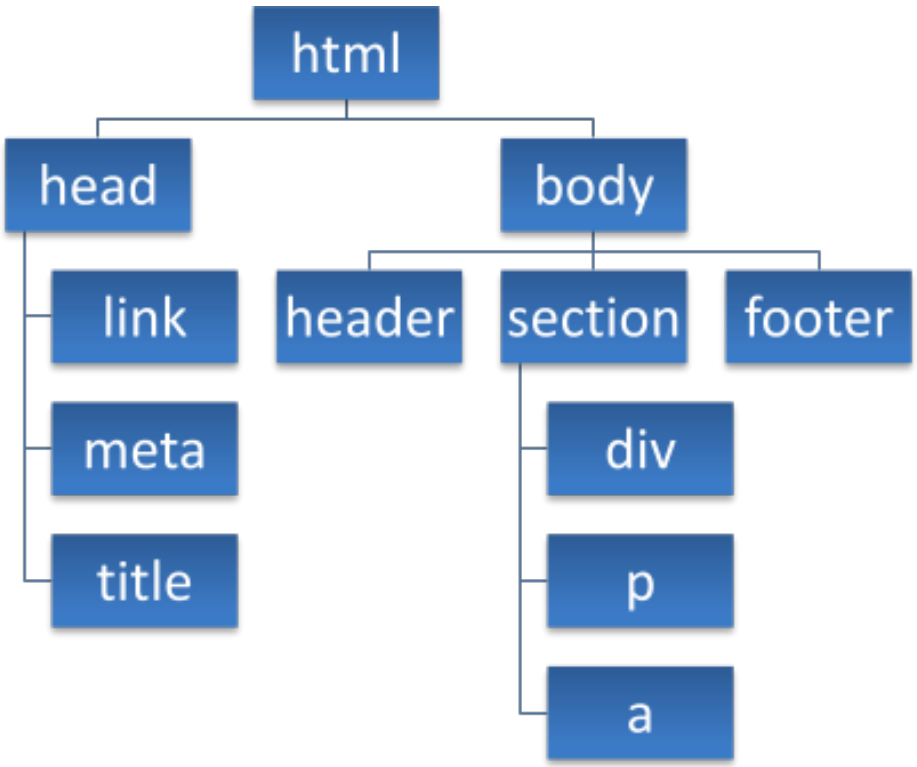

Рисунок 1 – Пример DOM-представления для простого HTML документа

Все эти теги не являются уникальными, и в одном документе может быть по несколько экземпляров каждого из них.

В *body* могут находиться разнообразные элементы. Например, в родительском *body* — дочерний элемент *header*, в элементе *header* — дочерний элемент section, в родительском section — дочерний *div*, в *div* — элемент *h3*, и, наконец, в *h3* — элемент *span*. В этом случае *span* не имеет дочерних элементов, но их можно добавить в любой момент. Это можно описать так:

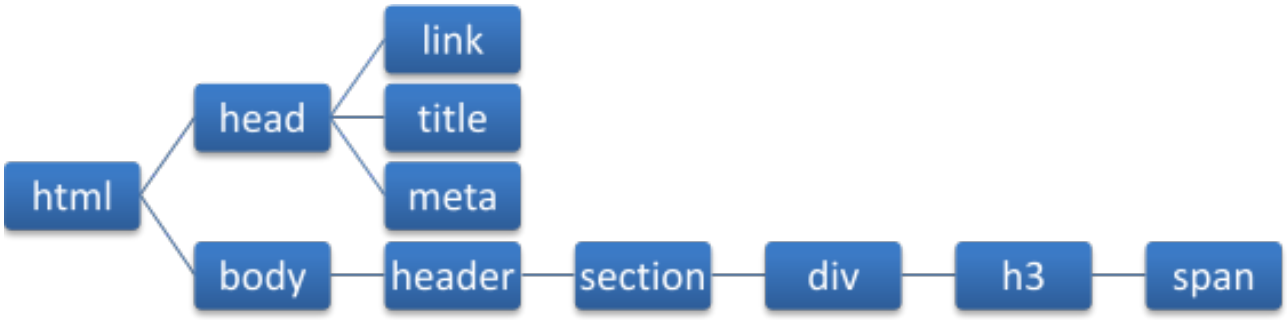

Рисунок 2 – Пример DOM-представления для чуть более сложного HTML документа

А если бы система была бы более разветвлённая и с большим количеством вложений — так:

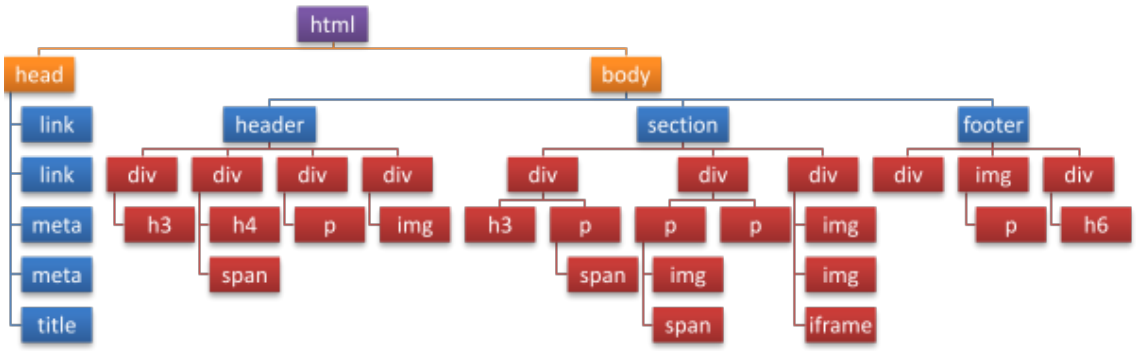

Рисунок 3 – Пример DOM-представления для сложного HTML документа

На схеме изображено довольно большое DOM-дерево, и его сложно воспринимать из-за его размера. Для удобства часто используется система многоуровневых списков. Например, предыдущее дерево можно преобразовать в такой список:

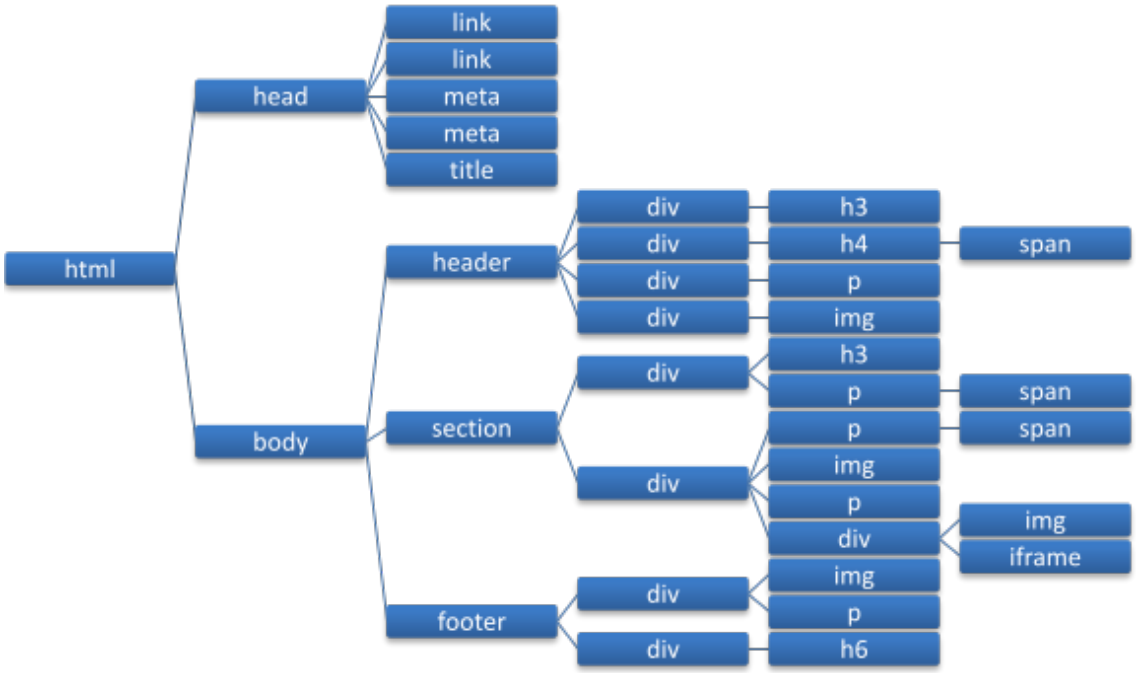

Рисунок 4 – Пример DOM-представления для сложного HTML документа в виде списка

Элементы могут наследовать не все, но многие свойства своих родителей — например, цвет, шрифт, видимость, размеры и т.д.

Таким образом, чтобы задать стиль шрифта на всей странице, потребуется не прописывать цвет для каждого элемента, а задать его только для body. А чтобы изменить наследуемое свойство у дочернего элемента, нужно прописать только ему новые свойства. Наследование удобно для создания единообразной страницы.

1.2.1 Алгоритм построения DOM

Рассмотрим алгоритм построения DOM-дерева по имеющемуся HTML документу.

Для того, чтобы построить дерево объектной модели, требуется обработать документ и произвести операцию вставки в получающееся дерево n раз, где n - количество элементов.

1.2.2 Алгоритм отображения DOM

Полученное DOM-дерево необходимо отобразить, чтобы пользователь получил возможность увидеть результаты отображения у себя на экране.

Чтобы это сделать, придётся сделать полный обход DOM-дерева и отобразить каждый из имеющихся элементов.

1.2.3 Алгоритм обновления DOM

Обновление структуры DOM — распространённая и часто используемая операция, производимая, например, в случае, когда документ должен меняться в ответ на действия пользователя (любая активность на странице).

Рекоммендация W3C в таком случае призывает повторно отрисовать обновлённое дерево с нуля — то есть, каждый раз, когда необходимо обновить дерево, будет использоваться алгоритм отображения.

1.3 Виртуальная объектная модель документа

Основной проблемой DOM является то, что он никогда не был приспособлен для динамического интерфейса (UI). Он предоставляет удобный программный интерфейс, позволяющий взаимодействовать с ним из кода, но это не решает проблем с производительностью. Современные социальные сети, такие, как ВКонтакте, Twitter или Facebook будут использовать тысячи DOM узлов, взаимодействие с которыми может занимать ощутимое для пользователя время.

Одним из решений данной проблемы служит технология виртуальной объектной модели, которая, хоть и не является стандартом, позволяет по-прежнему взаимодействовать с DOM, но делать это как можно реже.

Виртуальная объектная модель документа (англ. *Virtual Document Object Model* — VDOM) [8] — концепт программирования, в которой идеальное («виртуальное») представление пользовательского интерфейса хранится в памяти и синхронизуется с «настоящим» DOM при помощи алгоритмов согласования.

Вместо того, чтобы работать с DOM напрямую, согласно концепции VDOM работа происходит с его легковесной копией, хранящийся непосредственно в памяти компьютера. За счёт этого операции вставки, сравнения, удаления и обхода узлов дерева происходит быстрее благодаря отсутствию необходимости отрисовывать изменения после каждой операции.

Когда в пользовательский интерфейс добавляются новые элементы, создаётся виртуальная модель DOM, представленная в виде дерева. Если состояние любого из элементов изменяется, создаётся новое виртуальное дерево DOM. Затем это дерево сравнивается с предыдущим виртуальным деревом.

Рассмотрим, как осуществляется представление DOM- и VDOM- дерева на примере следующего простого HTML доокумента:

Листинг 2: Простой HTML документ

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <link/>
</head>
<body>
  <ul class="list">
    <li class="list__item">List item</li>
  </ul>
</body>
</html>
```

Рисунок 5 – DOM-дерево для простого HTML документа

Листинг 3: VDOM-дерево для простого HTML документа

```
const vdom = {
  tagName: "html",
  children: [
    { tagName: "head" },
    {
      tagName: "body",
      children: [
        {
            tagName: "ul",
            attributes: { "class": "list" },
            children: [
                {
                    tagName: "li",
                    attributes: { "class": "list__item" },
                    textContent: "List item"
                } // end li
            ]
        } // end ul
      ]
    } // end body
  ]
} // end html
```

Данный пример служит лишь для демонстрации того, как может осуществляться представление VDOM. Реальные деревья содержат намного большее число узлов, а элементы DOM имеют гораздо больше полей (классы, идентификаторы, стили, обработчики событий, поля данных, типы и значения, вспомогательные поля и т.п.).

1.3.1 Алгоритм построения VDOM

Рассмотрим, как происходит построение VDOM-дерева по гипертекстовому документу.

Чтобы построить виртуальное дерево, требуется обработать документ и осуществить операцию вставки в виртуальное представление n раз, где n — количество элементов. После этого необходимо сделать полный обход виртуального дерева и для каждого элемента создать соответствующие ему узлы DOM-дерева.

2 Конструкторская часть

3 Технологическая часть

4 Исследовательская часть

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. DOM Living standart [Электронный ресурс]. — Режим доступа: https://dom.spec.whatwg.org/, свободный (дата обращения: 09.11.2022)

2. Document Object Model (DOM) [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model, свободный (дата обращения: 09.11.2022)

3. HTML [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/ru/docs/Web/HTML, свободный (дата обращения: 09.11.2022)

4. HTML Living standart [Электронный ресурс]. — Режим доступа: https://html.spec.whatwg.org/multipage/, свободный (дата обращения: 09.11.2022)

5. HTML5 is W3C recommendation [Электронный ресурс]. — Режим доступа: https://www.w3.org/blog/news/archives/4167, свободный (дата обращения: 09.11.2022)

6. Яндекс.Радар Браузер в России [Электронный ресурс]. — Режим доступа: https://radar.yandex.ru/browsers, свободный (дата обращения: 09.11.2022)

7. Сбалансированные деревья, М.В. Губко. — М.: Институт проблем управления РАН [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/sbalansirovannye-derevya/viewer, свободный (дата обращения: 09.11.2022)

8. Виртуальный DOM и детали его реализации в React [Электронный ресурс]. — Режим доступа: https://ru.reactjs.org/docs/

faq-internals.html#gatsby-focus-wrapper, сводный (дата обращения: 12.11.2022)

ПРИЛОЖЕНИЕ А