Содержание:

Введение

В работе рассматривается языки гипертекстовой разметки такие ка HTML, XML, XHTML и HTML5.

Актуальность этих языков на сегодняшний день очень велика, и все они используются при разработках кроме XHTML, этот язык не удалось продвинуть в массы и сделать его жестким стандартом для разработчиков, но уделить ему внимания все равно надо, так как некоторые браузеры его еще поддерживают.

Как описывал автор Комолова Н. В своей книге «Самоучитель HTML (2-е издание)» что HTML сделал большой прорыв во всемирной паутине, благодаря этому язык стал стандартом для всех последующих языков гипертекстовой разметки, поэтому большое внимания в этой курсовой будет уделяться ему.

Сам язык появился в 1991 году, его разработал Британец Тимоти Джон Бернерс-Ли, полное название языка «HyperText Markup Language». Он был предназначенный для разметки и оформления документов World Wide Web.

В следствие развития интернета, HTML нуждался в расширении, в результате появился язык XML, имеющий простоту HTML, логику разметки SGML (Standard Generalized Markup Language- стандартный обобщённый язык разметки) и приспособленный к требованию интернета, но он не вытесняет HTML и SGML со своих мест, а только занимает те области где эти языки трудно использовать или невозможно. В книге «XML. Огромные возможности и легкость изучения» подробно расписано в каких областях используется язык XML.

В 6-м издание книги «HTML и XHTML» Чака Муссиано и Билла Кеннеди, подробно охарактеризован XHTM и идет сравнение его с HTML.

Сам XHTML был разработан в 2000 году, он должен был стать заменой языкам HTML и XML. Его задача была в том, чтобы убрать из вёрстки все теги и атрибуты форматирования, а также достигнуть большей строгости синтаксиса.

Самым последним появился HTML5 являющийся последней версией стандарта языка HTML, который несет в себе не столь большие изменения нежели его новые

подходы к проектированию благодаря поощрению лучших наборов практик за все время пока HTML развивался, все это и не только описаны в издании Брайна Хогана и этим он интересен для данной курсовой.

Цель работы состоит подробно разобрать эти языки: в чем их недостатки, разница между ними, как они устроены.

На практических примерах работы с HTML показать главную часть его структуры.

Глава 1 Понятие и виды языков разметки

1.1 Понятие и сущность языков разметки

Языки разметки являются строительными блоками, используемыми для создания веб-страниц или любых форм и размеров.

В мире существует много разных языков разметки. Для веб-дизайна и разработки есть три конкретных языка разметки, с которыми вероятно сталкивался каждый. Это HTML, XML, XHTML и HTML5.

Чтобы правильно определить этот термин, язык разметки - это язык, который аннотирует текст, чтобы компьютер мог манипулировать этим текстом. Большинство языков разметки читаются человеком, потому что аннотации написаны таким образом, чтобы отличать их от самого текста.

Любой текст, который появляется внутри одного из этих символов «<» а также «>», считается частью языка разметки, а не частью аннотированного текста и является тегом.

Каждый тег содержит символы «меньше» и «больше чем», чтобы обозначить его как часть разметки.

Когда вы форматируете текст для отображения на экране компьютера или другого устройства, вам необходимо различать сам текст и инструкции для текста.

«Разметка» - это инструкция для отображения или печати текста. Разметка не должна быть читаемой компьютером. Аннотации, сделанные в печатном виде или в книге, также считаются разметкой. Например, многие учащиеся в школе выделяют

определенные фразы в своих учебниках. Это указывает на то, что выделенный текст важнее окружающего текста.

Цвет выделения считается разметкой. Разметка становится языком, когда правила кодифицированы вокруг того, как писать и использовать эту разметку. У того же ученика может быть свой «язык разметки для заметок», если он кодифицирует правила, такие как «пурпурная подсветка - для определений, желтая подсветка - для деталей задачи, а карандашные заметки на полях - для дополнительной информации».

Большинство языков разметки определяются сторонним органом для использования многими разными людьми. Вот как работают языки разметки для Интернета. Они определяются консорциумом W3C или World Wide Web.

1.2 Виды языков разметки

HTML или HyperText Markup Language — язык разметки гипертекста является основным языком Интернета и наиболее распространенным языком, с которым работают веб-дизайнеры и разработчики.

Все веб-страницы написаны на языке HTML. HTML определяет способ отображения изображений, мультимедиа и текста в веб-браузерах. Этот язык включает элементы для соединения ваших документов (гипертекст) и создания интерактивных веб-документов (например, с помощью форм).

HTML - это определенный стандартный язык разметки. Он основан на SGML (стандартном обобщенном языке разметки). Это язык, который использует теги для определения структуры вашего текста. Элементы и теги определяются символами «<» и «>».

Хотя HTML на сегодняшний день является самым популярным языком разметки, используемым в Интернете, он не является единственным выбором для веб-разработки.

По мере развития HTML он становился все более и более сложным, а теги стиля и контента объединялись в один язык. В конце концов, W3C решил, что необходимо разделить стиль веб-страницы и контента. Тег, который определяет только содержимое, останется в HTML, в то время как теги, определяющие стиль, будут устаревшими в пользу CSS (каскадных таблиц стилей).

Самая новая пронумерованная версия HTML - это HTML5. Эта версия добавила больше возможностей в HTML и убрала некоторую строгость, наложенную XHTML.

Способ выпуска HTML был изменен с появлением HTML5. Сегодня новые функции и изменения добавляются без необходимости выпуска новой пронумерованной версии.

В HTML главную роль играют теги и атрибуты. Каждый элемент HTML должен быть заключен в **html** теги. Открывающий **<html>** тег должен появляться первым, а закрывающий **</html>** тег должен отображаться внизу документа. Любой другой элемент HTML должен появляться между этими двумя тегами как показано на рисунке №2.

```
<html>
<!-- здесь должен быть любой другой элемент  -->
</html>
```

2. Тег <html>

Элемент HTML - заголовка - это контейнер, который может содержать несколько элементов HTML, которые не являются видимыми частями страницы, отображаемым браузером.

Эти элементы являются либо метаданными, которые описывают информацию о странице, либо помогают извлекать внешние ресурсы, такие как таблицы стилей CSS или файлы JavaScript.

<title> элемент является единственным элементом, который обязательный, который содержаться внутри **<head>** тега. Содержимое этого элемента отображается в виде заголовка страницы на вкладке браузера, а также используется поисковыми системами для определения заголовка страницы.

Все элементы HTML, которые могут использоваться внутри <head> элемента:

- <base>
- <link>
- <meta>
- <noscript>
- <script>
- <title> (обязательный)

Пример

```
<head>

<title>Дом</title>
<base/>
<link/>
<meta/>
<noscript/>
<script/>

</head>
```

показан на рисунке №3.

3. Пример элементов внутри <head>

Правила представления или оформления веб-страницы, написанные во внешнем файле CSS, должны быть связаны с веб-страницей с **link** элементом, который уведомляет браузер о том, какие таблицы стилей следует загрузить. Следующий синтаксис используется для **link**

```
<link href="../file_path/file_name.css" rel="stylesheet" title="Style Sheet Name">
```

внешней таблицы стилей в HTML документе как показывает рисунок №4.

4. Использования link для CSS

На рисунке №4 видно элемент **link** и его атрибуты **href**, **rel** и **title**.

Атрибуты содержат дополнительные фрагменты информации. Атрибуты принимают форму открывающего тега, а дополнительная информация размещается внутри.

Языки сценариев JavaScript является несомненным лидером рынка, обычно загружаются в **head** элемент страницы с помощью **script** тега. JavaScript может использоваться для выполнения самых разных задач, в том числе:

- Включение Google Analytics или других приложений для отслеживания посетителей.
- Условно добавив HTML5 Shiv для посетителей сайта, использующих старые браузеры.
- Загрузка библиотек JavaScript, таких как jQuery.

и многое, многое другое.

Основной синтаксис для script тега выглядит как показано на картинке №5.

```
<script src="../path/to/javascript_file.js">
//Приведенный выше атрибут src идентифицирует файл JavaScipt.
//Но, JavaScript также может быть добавлен между открывающим
// и закрывающим <script>
</script>
```

5. Тег script

Метаданные - это данные о данных. Это информация, которая описывает некоторую другую информацию подлым образом. Этот <meta> тег используется в документе HTML для предоставления метаданных высокого уровня о веб-странице: информации, которая описывает веб-страницу осмысленным образом, который может быть понят веб-сканерами и браузерами.

Вот несколько примеров метаданных которые мы можем предоставить веб-сканерам и браузерам в meta теге.

- Описание страницы и ключевые слова, которые описывают тему страницы.
- Информация об авторстве страницы.
- Инструкции для конкретных действий браузера.
- Подробные сведения о заголовке страницы, описании и авторе, которые будут использоваться при размещении страницы в социальных сетях или показе в поисковой выдаче.
- И многое, многое другое.

Атрибуты в <meta>:

- charset - Задает кодировку документа.
- content -Устанавливает значение атрибута, заданного с помощью
- name или http-equiv.
- http-equiv - Предназначен для конвертирования метатега в заголовок HTTP.
- name - Имя метатега, также косвенно устанавливает его предназначение.

Пример на картинке №6

```
<head>
   <title>Дом</title>
<!-- пример meta -->
   <meta charset="utf-8">
   <meta name="GENERATOR" content="Microsoft FrontPage 4.0">
   <meta name="ProgId" content="FrontPage.Editor.Document">
</head>
```

6. Пример meta

charset, Короткий для набора символов, это кодировка символов, используемый на веб странице. Почти во всех случаях пишется в UTF-8 кодировке.

Важно объявить этот **charset** первым в HTML документе, потому что браузеры перестанут искать кодировку символов после 512 байт и угадают, какую кодировку следует применять.

Это может создать некоторые угрозы безопасности, поэтому надо быть осторожней и объявите кодировку символов как один из первых элементов в HTML документе **head**.

С `<meta charset="utf-8">` интаксис для объявления UTF-8 в качестве кодировки символов на рисунке №7

7. Синтаксис определения кодировки

Весь контент, которы

```
<html>
<head>
   <title>Дом</title>
</head>
<body>
   <!-- это основной контейнер  -->
</body>
</html>
```

между открывающим и закрывающим **body** тегами. Тело - это основной контейнер контента, который составляет веб-страницу как показано на рисунке №8.

8. Основной контейнер

Разобрав немного что такое HTML можно переходить к следующему языку с названием XML который был введен консорциумом W3C в 1998 г. Его появление произошло из-за невозможности HTML описывать данные.

Расширяемый язык разметки - это язык, на котором основана другая версия HTML. Как и HTML, XML также основан на SGML. Он менее строг, чем SGML и более строг чем обычный HTML. XML обеспечивает расширяемость для создания различных языков.

XML - это язык для написания языков разметки. Например, если вы работаете над генеалогией, вы можете создать теги, используя XML для определения отца, матери, дочери и сына в вашем XML, например: <папа> <мама> <дочь> <сын>.

XML расшифровывается как расширяемый язык разметки. XML был разработан для хранения и транспортировки данных. XML был разработан, чтобы его могли читать как человек, так и машины(компьютеры).

XML играет важную роль во многих различных ИТ-системах. XML часто используется для распространения данных через Интернет.

XML и HTML были разработаны с разными целями:

- XML был разработан для переноса данных - с акцентом на то, что данные хранят.
- HTML был разработан для отображения данных - с акцентом на то, как данные выглядят.
- XML - теги не предопределены, как HTML - теги

XML широко используется в эпоху веб-разработки. Он также используется для упрощения хранения данных и обмена данными.

Основные особенности или преимущества XML приведены ниже.

Если необходимо отобразить динамические данные в HTML документе, каждый раз, когда данные изменяются, потребуется много работы для редактирования HTML — кода.

С XML данные могут храниться в отдельных файлах XML. Таким образом, можно сосредоточиться на использовании HTML и CSS для отображения и макета, и быть уверенным, что изменения в базовых данных не потребуют никаких изменений в HTML.

С помощью нескольких строк кода JavaScript можно прочитать внешний файл XML и обновить содержимое данных веб-страницы.

В реальном мире компьютерные системы и базы данных содержат данные в несовместимых форматах.

Данные XML хранятся в текстовом формате. Это обеспечивает программно-аппаратный независимый способ хранения данных.

Это значительно упрощает создание данных, которые могут использоваться различными приложениями.

Одной из самых трудоемких задач для разработчиков является обмен данными между несовместимыми системами через Интернет.

Обмен данными в виде XML значительно уменьшает эту сложность, поскольку данные могут быть прочитаны различными несовместимыми приложениями.

Обновление до новых систем (аппаратных или программных платформ) всегда занимает много времени. Большие объемы данных должны быть преобразованы, а несовместимые данные часто теряются.

Данные XML хранятся в текстовом формате. Это упрощает расширение или обновление до новых операционных систем, новых приложений или новых браузеров без потери данных.

Различные приложения могут получать доступ к данным не только на HTML страницах, но и из источников данных XML.

С помощью XML данные могут быть доступны для всех видов «читающих машин» (карманных компьютеров, голосовых машин, новостных лент и т. д.) и сделать их более доступными для слепых людей или людей с другими ограниченными способностями.

XML - документы создают иерархическую структуру, похожую на дерево, поэтому она называется XML деревом, которое начинается с «корня» и ветвится до «листьев».

XML документы используют само описание и простой синтаксис как показано на рисунке №20.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<purse>
  <coins>15</coins>
  <paper>6</paper>
  <cards>3</cards>
</purse>
```

20. Документ XML

Первая строка - это объявление XML. Он определяет версию XML (1.0) и используемую кодировку (ISO-8859-1 - Западно европейский набор символов).

Следующая строка, которая содержит **<purse>** описывает корневой элемент документа (например, говоря: «этот документ является кошельком»).

Следующие 3 строки описывают 3 дочерних элемента корня (**coins**, **paper**, **cards**).

И, наконец, последняя строка определяет конец корневого элемента это **</purse>**.

XML-документы должны содержать **корневой элемент.** Этот элемент является «родителем» всех других элементов.

Элементы в документе XML формируют дерево документа. Дерево начинается с корня и ветвится до самого нижнего уровня дерева.

Все элементы могут иметь подэлементы (дочерние элементы).

Пример на рисунке №21

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
  <child>
    <subchild> ..... </subchild>
  </child>
</root>
```

21. Дерево элементов XML

Термины «родитель», «ребенок» и «родной брат» используются для описания отношений между элементами.

Родительские элементы имеют детей. Детей на одном уровне называют братьями и сестрами.

Все элементы могут иметь текстовое содержимое и атрибуты (как в HTML).

Еще один пример с использованием XML, но уже из фреймворка автоматизации сборки проектов **Maven**.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>ru.testSpring</groupId>
    <artifactId>SpringTest</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.4.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
```

22. Файл pom.xml из Maven

Данный xml файл хранит в себе название, версии, id, пакетов и плагинов, а Maven их уже подтягивает. В данным случае собран **Java** проект **Spring boot**.

В XML очень удобно хранить данные, а потом вытягивать их, он очень гибок и прост, поэтому его часто используют в различных разработках.

В 2000 г. появился новый язык гипертекстовой разметки, который был назван **XHTML**. Он должен был стать новым стандартом и заменить все при ведущие стандарты.

XHTML - это HTML 4.0, переопределенный для соответствия стандарту XML.

XHTML был заменен в современном веб-дизайне на HTML5 с изменениями, которые произошли с тех пор. Вряд ли можно найти какие-либо более сайты, использующие XHTML, но если открыть более старый сайт, то можно столкнуться с XHTML.

Между HTML и XHTML не так много серьезных отличий, но можно отметить некоторые особенности, которые очень выделяются.

XHTML написан в нижнем регистре. В то время как HTML - теги могут быть написаны в верхнем регистре или в нижнем регистре, чтобы быть правильными, теги XHTML должны быть все строчные (многие веб-профессионалы пишут HTML строчными буквами, хотя это не является техническим требованием).

Все элементы XHTML должны иметь конечный тег. Элементы с одним тегом, нуждающиеся в закрывающей косой черте (/) в конце тега как показано на рисунке №22.

```
<hr />
<img />
```

22. Элементы с одним тегом

Все атрибуты должны быть указаны в XHTML. Некоторые люди удаляют кавычки вокруг атрибутов для экономии места, но они необходимы для правильного XHTML.

XHTML требует, чтобы теги были правильно вложены. Если открыть элемент, выделенный жирным шрифтом (<b>), а затем элемент курсива (<i>), то надо закрыть элемент курсива (</ i>) перед тем, как закрыть полужирный (</ b>) (оба этих элемента устарели, поскольку они являются визуальными элементами, HTML теперь использует и вместо этих двух).

Атрибуты HTML должны иметь имя и значение. Атрибуты, которые являются автономными в HTML, также должны быть объявлены со значениями, например, атрибут HR будет записан как noshade = "noshade".

XHTML имеет два основных диалекта, которые отличаются друг от друга с точки зрения того, позволяют ли они использовать некоторые элементы представления или нет.

Как следует из названия, диалект XHTML Strict не позволяет использовать такие элементы, как <font> и . Это также не позволяет использовать атрибуты элемента, такие как align и bgcolor.

Детали представления, такие как размер шрифта / тип и выравнивание, должны обрабатываться с использованием CSS.

Строгий диалект также требует, чтобы весь текст и изображения были встроены либо в элемент <p>, либо в элемент (используемый для определения разделов или разделов документа)

XHTML Transitional диалект не запрещает использование элементов представления и атрибутов, подобных тем, которые описаны в предыдущем абзаце.

XHTML Transitional был предназначен для разработчиков, которые хотят преобразовать свои старые страницы в более новую версию, но, вероятно, не потрудятся сделать это, если им придется исключить все настройки презентации.

XHTML Strict был предназначен для разработчиков, создающих новые страницы.

Разработчики XHTML определяют, какой набор правил следует их странице, добавив строку DOCTYPE вверху страницы.

Например, вот операторы DOCTYPE для XHTML Strict и XHTML Transitional на рисунке №23

```
<! DOCTYPE html PUBLIC "- // W3C // DTD XHTML 1.0 Strict // EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1- strict .dtd">

ИЛИ ЖЕ

<! DOCTYPE html PUBLIC "- // W3C // DTD XHTML 1.0 Transitional // EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1- transitional .dtd">
```

23. Операторы DOCTYPE

Хотя изначально XHTML задумывался как «следующий шаг в эволюции Интернета», он никогда не завоевывал прочных позиций в сообществе веб-разработчиков.

Основным фактором, препятствующим его принятию, было то, что для того, чтобы в полной мере воспользоваться преимуществами документа на основе XML, его нужно было показывать браузерам как «application / xhtml + xml», а не «text / html».

Большинство основных браузеров, таких как Firefox, Chrome и Safari, были созданы для работы с типом контента application / xhtml + xml. Заметным исключением был Internet Explorer, который до версии 9 не поддерживал контент «application / xhtml + xml», пользователей спрашивали, хотят ли они сохранить файл, когда встречаются документы такого типа.

В дополнение к проблеме с типом контента, технические достижения подорвали аргумент, что небольшие портативные устройства не могут загружать веб-страницы с приемлемой скоростью без использования синтаксического анализатора на основе XML.

Современные браузеры для смартфонов используют те же HTML-парсеры, что и их настольные аналоги.

Таким образом, в последние годы представление о мире, в котором браузеры анализируют все веб-страницы, поскольку XML и разработчики страниц должны создавать правильно оформленные документы, появляется все реже и реже.

W3C остановил разработку новой версии XHTML и переключил свое внимание на новую версию HTML (HTML5).

Это заставило некоторых заявить, что XHTML мертв, а HTML5 требует, чтобы браузеры продолжали исправлять плохо написанный HTML.

Это дает возможность авторам неаккуратных страниц продолжать свои небрежные привычки. При этом браузеры будут продолжать принимать страницы, созданные с помощью кодирования в стиле XHTML, поэтому разработчики, которые видят ценность в обслуживание их страниц в качестве XML могут продолжаться.

HTML5 является последней спецификацией языка HTML и представляет собой серьезный разрыв с предыдущими практиками разметки.

Цель глубоких изменений в языке состояла в том, чтобы стандартизировать многие новые способы использования его разработчиками, а также поощрять единый набор лучших практик в отношении веб-разработки.

Большинство отдельных изменений являются результатом более широких целей в дизайне языка.

Эти цели в первую очередь включают в себя:

- Поощрение семантической (значимой) разметки
- Отделение дизайна от контента
- Содействие доступности и отзывчивости дизайна
- Уменьшение совпадения между HTML, CSS и JavaScript
- Поддержка мультимедийных приложений, устраняя необходимость в плагинах, таких как Flash или Java

Довольно несильные изменения произошли в HTML5 от HTML прошлой версии. Также HTML5 предоставил новые элементы семантической разметки, которые улучшили структуру сайта, добавив смысловое значение хранящиеся в их теле.

Семантическая разметка означает разметку, которая имеет значение, а не разметку, которая просто выглядит определенным образом.

Например, <h1> тег подразумевает, что содержимым элемента является заголовок или заголовок всего документа.

Это семантическое значение было бы потеряно, если бы мы просто сделали текст жирным и большим без использования соответствующего тега.

В HTML всегда была доступна небольшая семантическая разметка: теги заголовка, атрибута и метаданные документа. Но этого было недостаточно.

В предыдущих версиях языка общие структурные элементы, такие как заголовки страниц, навигационные меню и разделы основного содержимого, были обозначены одним и тем же элементом HTML тег.

В HTML5 есть множество новых семантических элементов, предназначенных для указания базовой структуры страницы:

- <header>
- <nav>
- <main>
- <article>
- <aside>
- <section>

- <footer>

Также были введены новые текстовые (встроенные) элементы, такие как <address>и <time>.

Они помогают поисковым системам и другим службам легко находить информацию на странице для отображения в других контекстах.

В то же время, существующие встроенные элементы, которые производят различные эффекты, такие как **полужирный**, *курсив* и подчеркивание, были уточнены или переопределены, чтобы подразумевать конкретное семантическое значение.

Наряду со строго поощряющей семантической (значимой) разметкой, спецификация HTML5 категорически не поощряет бессмысленную разметку, предназначенную только для того, чтобы сообщать браузеру, как отображать вещи.

Это включает в себя такие вещи, как:

- объявление шрифтов и цветов текста
- настройка выравнивания или выравнивания текста
- размещение границ на столах
- определяя, как текст обтекает изображения

Большинство функций HTML, которые допускают подобные вещи, полностью устарели. Те немногие, которые все еще официально поддерживаются, приходят с предупреждениями о том, что они обычно не рекомендуются.

Есть в первую очередь две причины предпочитать это разделение:

- Сайт легче поддерживать и перепроектировать, если объявления всех стилей выведены в CSS файл
- Пользователи просматривают веб-контент в самые разные разрешения экрана - настольные компьютеры, ноутбуки, планшеты, мобильные телефоны, устройства для чтения RSS и многие другие.

Стили и дизайнерские решения, которые имеют смысл в одной среде, не всегда имеют смысл в другой. Поэтому гораздо лучше предоставить семантическую информацию и позволить контенту адаптироваться к экрану пользователя.

На основе рассмотренного материала по языкам гипертекстовой разметки: HTML,XML, XHTML и HTML5 можно заключить следующие.

Основным у этих языков можно подчеркнуть это тег и одинаковую реализация, с помощью тегов они хранят или отображают данные

HTML и HTML5 используются для работы с отображением данных, но HTML5 может работать с новыми элементами и разметкой, имеет мультимедийную поддержку.

XML описывает данные, хранит их и передает. Его теги не придерживаются стандартных наборов как это делает остальные языки гипертекстовой разметки. XHTML не сильно отличается от HTML, но есть ряд особенность таких как, в HTML можно допустить небрежную ошибку и браузер все равно нормально отобразит страницу, но в XHML этого сделать не получится так как он основан на XML который является строгим языком разметки, браузер просто выдаст синтаксическую ошибку.

Глава 2 Работы с HTML

HTML является стандартизированным языком гипертекстовой разметки, большинство веб-страниц содержат это описание разметки.

Так как HTML является стандартом для всех последующих языков гипертекстовой разметки, большинство тегов поддерживается, поэтому будем использовать его в примерах.

2.1 Работа с текстом, ссылки и картинки

Работа с текстом в HTML является основной его составляющей, которую должен знать каждый.

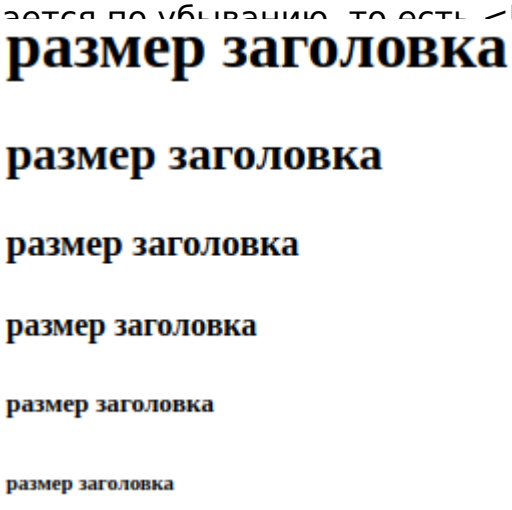
Есть много различных тегов для оформления текста, например, заголовки в тексте пишутся следующим образом:

- <h1>.
- <h2>.
- <h3>.
- <h4>.

- <h5>.
- <h6>.

Размер заголовка уменьшается по убыванию, то есть <h1> самый большой, а <h6>

```
<!DOCTYPE html>
<html>
<head>
<title>main</title>
</head>
<body>
<h1>размер загаловка</h1>
<h2>размер загаловка</h2>
<h3>размер загаловка</h3>
<h4>размер загаловка</h4>
<h5>размер загаловка</h5>
<h6>размер загаловка</h6>
</body>
</html>
```



9. Слева как выглядит в блокноте, а справа в браузере

В данном примере **<h1>** и **<h2>** должны использоваться для наиболее важных заголовков, а остальные теги следует использовать для подзаголовков и менее важного текста.

Боты поисковых систем используют этот порядок при расшифровке, какая информация является наиболее важной на странице.

Добавить текст на HTML страницу просто, используя элемент <**p>** который создает новый абзац. Мы помещаем весь наш обычный текст внутри элемента **<p>**.

При написании текста в HTML, есть ряд других элементов, которые мы можем использовать для управления текстом или для его отображения определенным образом.

Другие ключевые элементы:

- <b> - делает текст жирным, им можно выделить важную информацию.
- - делает абсолютно тоже самое что и <b>.
- <i> - делает текст курсивным.
- - делает тоже самое что и <i>.
- <mark> - выделяет текст цветом.

- <small> - делает текст меньше (сжимает его).
- <strike> - зачеркивает текст.
- <u> - подчеркивает текст.
- <ins> - тоже подчеркивает текст, как и <u>.
- <sub> - уменьшает текст и делает его снизу.
- <sup> - уменьшает и делает текст с верху (возведение в степень).

На рисунке № 10 показан пример со всеми выше описанными тегами, должен отметить что некоторые из них повторяют действия других тегов.

На примере работы с текстом, на верхнем рисунке отображено окно блокнота, все элементы закрыты в теге <body>, на нижнем рисунке показано как читает браузер все что написано в блокноте.

```
<body>
<p><b><mark>b</mark> выделяет важное</b> чем просто текст, то есть делает его жирным</p>
<p><strong><mark>strong</mark> делает тоже что и b,</strong> делает его жирным</p>
<p><i><mark>i</mark> делает текст курсивным</i></p>
<em><mark>em</mark> тоже делает курсивным</em>
<p><mark>mark</mark> подкрашивает фон текста</mark></p>
<p><small><mark>small</mark> делает текст меньше</small> чем основной</p>
<p><strike><mark>strike</mark> зачеркивает текст</strike></p>
<p><mark>u</mark> <u>подчеркивает наш текст</u></p>
<p><ins><mark>ins</mark> тоже подчеркивает текст</ins></p>
<p><sub><mark>sub</mark></sub> уменьшает текст и делает его с низу </p>
<p><sup>sup</sup> уменьшает и делает текст с верху(возведение в степень)</p>
</body>
```

b выделяет важное чем просто текст, то есть делает его жирным

strong делает тоже что и b, делает его жирным

i делает текст курсивным

em тоже делает курсивным

mark подкрашивает фон текста

small делает текст меньше чем основной

~~strike зачеркивает текст~~

u подчеркивает наш текст

ins тоже подчеркивает текст

sub уменьшает текст и делает его с низу

sup уменьшает и делает текст с верху(возведение в степень)

10. Пример работа с текстом, отображение в блокноте и в окне браузера

Как видно на рисунке №10 все эти теги должны быть открыты и закрыты вокруг рассматриваемого текста.

В итоге мы можем оформить текст как нам угодно используя при этом правильные теги, но при слишком большом объеме текста все может оказаться очень запутанным, поэтому лучше использовать таблицу стилей.

Текст не только можно оформить красиво, но можно еще сделать его ссылкой. Интернет страница состоит не только из текста, но и из множества ссылок в поисковиках и сайтах.

Почти все, на что мы нажимаете во время просмотра веб-страниц — это ссылка которая ведет на другую страницу посещаемого вами сайта или на внешний сайт.

Элемент ссылки обозначается как <a>. Этот элемент имеет атрибуты чтобы управлять ссылкой, пример href на рисунке №11.

```
<body>
<a href="https://www.youtube.com">Youtube</a>
</body>
```

Youtube

11. <a> ссылка с атрибутом href и отображение в браузере

Первая часть атрибута указывает на страницу, которая откроется после того, как нажмете на ссылку.

Между тем, вторая часть атрибута содержит текст, который будет отображаться посетителю.

Если мы создаем свой собственный веб-сайт, мы, скорее всего, разместим все свои страницы на профессиональном веб-хостинге. В этом случае внутренние ссылки на вашем веб-сайте будут **** Перейти по ссылки****.

Тег ссылки также можно используя с тегами оформления текста, таким образом ссылка может быть красиво оформлена. Пример на рисунке №12.

```
<body>
<a href="google.com"><h1>GOOGLE</h1></a>
<a href="google.com"><small>GOOGLE</small></a>
<a href="google.com"><mark>GOOGLE</mark></a>
</body>
```

GOOGLE

GOOGLE GOOGLE

12. Тег <a> с тегами оформления текста

В HTML оформление текста и создание ссылок это еще не самое главное, в современном цифровом мире изображения это все.

<IMG> тег имеет все, что нужно для просмотра изображений на сайте. Как и элемент ссылки **<a>**, **<img>** также содержит атрибуты.

Атрибут содержит информацию для компьютера относительно источника, высоты, ширины и альтернативного текста изображения.

Можно определить границы и другие стили вокруг изображения, используя атрибут класса.

Типы файлов, обычно используемые для файлов изображений в интернете: .**jpg**, . **png** и (все меньше и меньше) .**gif**.

Дополнительный текст важен для обеспечения правильного ранжирования вашего сайта на поисковые системы, а также для слабовидящих посетителей вашего сайта.

<img> тег обычно записывается как показано на картинке №13.

13. Тег img

Также, как и с тестом, картинку можно сделать ссылкой

окружив тег <img> тегом <a> и прописав атрибут **href**. В итоге при наведении курсора мышки на картинку в браузере, курсор меняет свой значок на и картинка становить активной.

2.2 Списки и таблицы в HTML

В веб-дизайне есть 3 различных типа списков, которые можно добавить на сайт.

Упорядоченный список

Первый **<ol>**: это упорядоченный список содержимого.

Например:

1. Песок
2. Щебень
3. Цемент.

Внутри тега <ol> перечисляем каждый элемент списка внутри тегов ** </ li>**.

Пример показан на рисунке №14.

```
<body>
<ol>
 <li>Песок</li>
 <li>Щебень</li>
 <li>Цемент</li>
</ol>
</body>
```

1. Песок
2. Щебень
3. Цемент

14. Список

Неупорядоченный список

Второй тип списка, который можем включить - это неупорядоченный список <ul>. Это больше известный как список маркеров и не содержит цифр.

```
<body>
<ul>
 <li>Песок</li>
 <li>Щебень</li>
 <li>Цемент</li>
</ul>
</body>
```

- Песок
- Щебень
- Цемент

Пример на картинки №15.

15. Список

Список определений

Список определений пишется <dl> тегом.

Пример кода используемый <**dl**> тег выглядит следующим образом на картинке № 16.

```
<body>
<dl>
 <dt>Ресурсы</dt>
 <dd>Щебень</dd>
 <dd>Цемент</dd>
</dl>
</body>
```

Ресурсы
Щебень
Цемент

16. Список <dl>

Текст с окруженным тегом **<dd>** будет отображаться с отступом, а **<dt>** будет как к примеру заголовок или глава.

Такие список можно использовать в разных ситуациях, например, в меню сайта или содержание какого-нибудь текста.

Не редко на веб-сайтах можно увидеть таблицы. При рисовании таблицы нужно открыть элемент с открывающим тегом <table>. Внутри этого тега мы структурируем таблицу, используя строки таблицы **<tr>** и ячейки **<td>**.

```
<table>
<tr>
 <td>Строка 1 - колонка 1</td>
 <td>Строка 1 - колонка 2</td>
 <td>Строка 1 - колонка 3 </td>
 </tr>
<tr>
 <td>Строка 2 - колонка 1</td>
 <td>Строка 2 - колонка 2</td>
 <td>Строка 2 - колонка 3</td>
</tr>
</table>
```

№17.

Столб 1 - колонка 1 Столб 1 - колонка 2 Столб 1 - колонка 3
Столб 2 - колонка 1 Столб 2 - колонка 2 Столб 2 - колонка 3

17. Таблица HTML в блокноте, браузере

Получается создать 2-рядную таблицу с 3 ячейками в каждом ряду которые заполняются различным текстом.

Сами таблицы очень часто встречаются на веб-сайтах с различными товарами: еда, одежда и т.д.

Также в таблице можно сделать рамку чтоб видеть разделения ячеек в ней, для этого нужно указать атрибут **border** в теге <table> **"<table border="1">"** и в итоге получится рамки как на примере №18

Таблица размеров обуви

Россия	Великобритания	Европа	Длина ступни, см
34,5	3,5	36	23
35,5	4	36⅔	23–23,5

18. таблица с рамками

Благодаря разделительной рамки таблица на рисунке №18 становится более читаемой чем на предыдущем примере. В сегодняшних реалиях данный атрибут не используется, его заменяют внешние стили.

2.3 Формы в HTML

Веб-формы используются практически всеми веб-сайтами для самых разных целей. Пользователи форумов и социальных сетей используют формы для добавления контента и взаимодействия с другими пользователями.

Веб-сайты, которые можно настраивать для создания персонализированного интерфейса, например, настраиваемые каналы новостей, используют формы, позволяющие пользователям контролировать содержимое, отображаемое на странице.

И почти каждый веб-сайт использует формы, позволяющие посетителям сайта связаться с организацией или лицом, управляющим веб-сайтом.

Каждая веб-форма должна быть завернута в **<form>** теги. В большинстве случаев все поля формы будут вложены между этими тегами.

Существует несколько атрибутов, которые необязательно могут использоваться с **form** элементом, в том числе: **accept-charset** этот необязательный атрибут используется для идентификации кодировок символов, приемлемых для сервера, и ввода формы обработки кода.

Если указано более одной кодировки, между каждой кодировкой должен быть один пробел. Если оставить это поле пустым или не указывать, кодировка по умолчанию будет соответствовать той же кодировке, что и документ, содержащий форму.

action: этот атрибут используется для указания URL - адреса, куда должны отправляться данные формы (например: *http://example.ru/ form_file.php*). Это поле было обязательным до HTML5, но теперь является необязательным.

autocomplete: этот атрибут говорит браузеру предлагать посетителю ответы в форме на основе сохраненных записей. Значением по умолчанию является *autocomplete*="*on*".

Если нужно выключить авто заполнение, то надо также отключить его в каждом поле, которое может разрешить авто заполнение.

enctype: используется для указания способа кодирования данных формы. Используется только в том случае, если указан атрибут *method*.

Есть три возможных значения:

- **application/x-www-form-urlencoded**: значение по умолчанию, которое заменяет все пробелы на «+» и преобразует все специальные символы в значения ASCII HEX.
- **multipart/form-data**: ничего не закодировано. Входные данные загружаются на сервер в точности так, как они вводятся в форму.

text/plain: пробелы преобразуются в символы «+», но другие символы не кодируются.

input элемент является очень гибким элементом, который внешний вид и поведение могут резко измениться на основе **type** атрибута, приложенный к элементу.

Наиболее распространенные **type** значения включают в себя: **text** значение по умолчанию для **type=""** атрибута. Определяет одну строку текста шириной 20

символов.

Ширина может быть скорректирована с помощью атрибута размера.

password: тип пароля в основном совпадает с текстовым полем, за исключением того, что символы, введенные в поле пароля, маскируются.

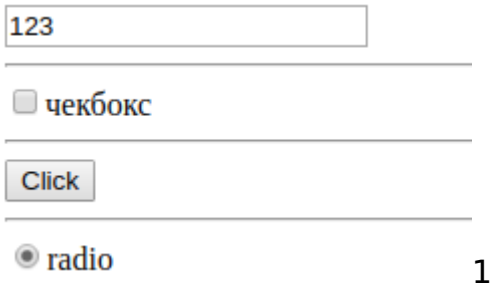
radio: радиокнопки могут использоваться для предоставления нескольких опций, из которых может быть выбран только один.

checkbox: флажки аналогичны переключателям, но одновременно могут быть активны несколько вариантов. Это означает, что несколько значений могут быть отправлены с набором флажков, в то время как набор переключателей будет принимать только одно значение.

submit: значение типа отправки создает кнопку отправки формы. При нажатии форма выполнит действие, указанное в *action* атрибуте, связанном с *form* элементом.

Многие формы используют только один или два input типа, а самые простые формы создаются с использованием только перечисленных выше типов. Пример показан на картинке №19.

```
<form>
<input type="text" value="123"><hr>
<input type="checkbox">чекбокс<hr>
<input type="submit" value="Click"><hr>
<input type="radio">radio
</form>
```



19. Элементы ввода input

Несмотря на то, что данный **type** атрибут является наиболее используемым и наиболее полезным **input** атрибутом, есть несколько других атрибутов, которые необходимо знать для создания полезных форм.

name: *name* назначенный *input* элемент будет отправлен вместе со значением, введенным в соответствующее поле. Другими словами, если значение «Петя» было введено в *input* элемент с этим кодом, *<input type="text" name="first_name">* отправленное значение будет «*first_name = Петя*».

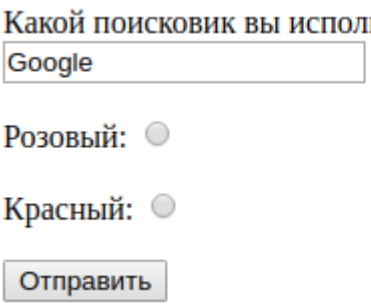
value: значение элемента ввода выполняет различную функцию в зависимости от типа ввода, к которому он применяется. При указание этого атрибута в *submit*, *reset* или *button* типах, значение используется в качестве текста на кнопке.

Применительно к текстовым полям, он предоставляет значение по умолчанию, связанное с полем.

При связывании с флажком или переключателем значение предоставляет значение, которое будет связано с этим полем, если оно выбрано

```
<form>
<p>Какой поисковик вы использовали?:<br> <input type="text" name="Referrer"
value="Google"></p>
<p>Розовый: <input type="radio" name="color" value="pink"></p>
<p>Красный: <input type="radio" name="color" value="red"></p>
<input type="submit" value="Отправить">
</form>
```

Пример ниже



20. Входные атрибуты value

readonly: когда *readonly* применяется как атрибут *input* элемента, значение в поле не может быть изменено.

JavaScript можно использовать для удаления **readonly** атрибута после выполнения какого-либо другого действия, например, нажатия кнопки или выбора флажка, например, **readonly** может быть применен к типу **submit** ввода и удален, когда установлен флажок подтверждающий, что пользователь принял условия обслуживания сайта.

disabled: используем этот атрибут, чтобы отключить все поля в форме.

size: используем *size* атрибут с *input* типами текста, чтобы указать видимую ширину поля без ограничения количества символов, которые могут быть введены в поле.

maxlength: ограничить эти поля определенным количеством символов.

checked: предварительно выбирает флажок или переключатель при загрузке формы, этот атрибут применяется к input элементу.

Эти атрибуты имеют широкую поддержку и обычно используются с формами, с которыми мы сталкиваемся каждый день.

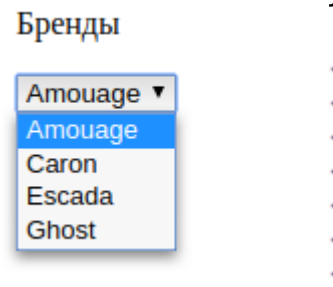
2.4 Выпадающие меню, текстовые области

Входные данные не единственные элементы, которые можно использовать для создания полей формы.

Другие типы элементов могут быть связаны с формами для создания раскрывающихся списков или параметров, текстовых областей произвольной формы и гибких кнопок.

Чтобы создать раскрывающийся список предварительно заполненных параметров, из которых посетитель веб-сайта может выбрать параметр, используется **select** элемент для создания поля и вложите несколько **option** элементов, чтобы создать различные параметры, которые должны отображаться в раскрывающемся меню.

Например, чтобы создать выпадающее меню каталога парфюмерии для вымышленного магазина косметики, можно использовать следующий код,




```
<p>Бренды</p>
<select name="perfumery">
<option value="amouage">Amouage</option>
<option value="caron">Caron</option>
<option value="escada">Escada</option>
<option value="ghost">Ghost</option>
</select>
```

21. Выпадающее меню

Именно **value** будет отправлено вместе с формой, а текст между открывающим и закрывающим тегами это то, что представляется посетителю, заполняющему форму.

Например, если посетитель выбирает «Caron», фактическим значением, представленным в форме, будет caron который указан в **value**.

Все текстовые вводы, например, **<input type="text">** принимают только одну строку текста. Однако, если нужно хотите создать большую текстовую область для более длинного ввода текста, поле ввода в одну строку не будет работать.

textarea элемент является правильным выбором для создания большой области ввода текста, способную принимать ввод текста, который не будет отображаться наилучшим образом на одной линии.

Есть три части textarea:

1. textarea создаются путем вставки открытия и закрытия тегов. Любой текст, вложенный между тегами, будет загружен в текстовую область при загрузке формы и будет отправлен вместе с формой, если посетитель, отправляющий форму, не удалит текст из textarea.
2. Размер текстовой области определяется с помощью rows атрибут, чтобы определить количество строк текста, которые должны отображаться без изменения размера текстовой области.
3. Чтобы установить предопределенную ширину нужно использовать cols атрибут. Применяемое значение будет количеством символов, которые появятся в одной строке перед переносом во вторую строку.

Элементы текстовой области могут быть изменены. Тем не менее, rows и cols атрибуты будут определять размер самого textarea когда заходишь на страницу браузера, а также будет установлено минимальное пространство, область может быть изменена, чтобы соответствовать размеру, Пример на рисунке № 22.

```
<textarea rows="3" cols="60" placeholder="здесь введенный текст"> </textarea>
<br>
```

22. Текстовая область

Этот код создаст две текстовые области одинакового размера, высотой три строки и может принимать 60 символов в строке.

Они могут быть изменены до любого размера большего размера по умолчанию. Особое внимание стоит уделить, как текст заполнителя был добавлен к первому с placeholder атрибутом, во втором элементе он просто вложен между открывающим и закрывающим тегами.

Если посмотреть на рисунок №19 в низу, можно увидеть, как я уменьшил текстовую область нижнего **textarea**.

В то время как **textarea** размер может быть задан с помощью строк и столбцов, лучше использовать CSS для стиля и размера текстовых областей.

Как видно HTML очень удобно работать с содержанием и стилями, но популярность CSS заменила всю стилевую работу в HTML, и за ним осталось только содержание.

С помощью CSS можно создавать различные анимации и стилизовать любой тег HTML, покрасить в любой цвет текст и фон, весь стиль что когда-то писался в HTML теперь пишется во внешнем файле CSS, что во много раз упростило и улучшило работу веб-дизайнеру и разработчикам.

Заключение

В заключение можно отметить, что все языки гипертекстовой разметки имеют не большую разницу между друг другом, все они были спроектированы с самого первого языка HTML, но с некоторыми отличиями в реализации. Можно выделить отдельно язык XML так как он не отображает данные на странице, а только хранит и передает их. XHTML взял в себя XML что его сделало очень строгим, и он требует больших усилий чем HTML.

HTML5 отличается от HTML новыми тегами, которые облегчают нам разметку и поиск по коду, на пример тег <nav> (элемент меню) проще найти чем из кучи тегов в котором лежит это меню, или тот же <header>будет содержать шапку сайта. Также в HTML5 реализован большое количество возможностей, без сторонних технологий, которые облегчают разработку сайта.

В HTML большое количество тегов для работы с текстом, но они уже считаются устаревшими и не используется, отдавая предпочтение к внешним таблицам стилей таких как CSS.

XHTML можно убрать на полку и не трогать, его трудно использовать без помощи специальных программ по сравнению с HTML который можно написать на простом блокноте, поэтому вряд ли можно найти кого-нибудь кто его еще использует.

Список используемой литературы

1. Н. Комолова. HTML. Самоучитель: самоучитель / Е. Яковлева - 2-е издание .: Питер, 2011. - 284 с.

2. М. Чак HTML и XHTML. Подробное руководство: / К. Билл - 6-е издание .: Символ-Плюс, 2008. - 746 с.

3. Д. Джон Основы веб-программирования с использованием HTML, XHTML и CSS 2-е издание .: Эксмо, 2010. - 768 с.

4. М. Пилгрим Погружение в HTML5.: БХВ-Петербург, 2011. - 293 с.

5. К. Сухов HTML5 - путеводитель по технологии .: ДМК Пресс, 2013. - 352 с.

6. Кох Д. XML. Огромные возможности и легкость изучения: самоучитель / К. Дэвидсон пер. с англ. Чайкина А. И 2007. - 256 с.

7. Б. Хоганс HTML5 и CSS3. Веб-разработка по стандартам нового поколения.- СПБ.: Пи-тер, 2012.- 272 с.

8. Ф. Бен HTML5 и CSS3.Разработка сайтов для любых браузеров и устройств .: Питер, 2014.- 304 с.

9. К. Игорь HTML, XHTML и CSS на 100% .: Питер, 2010.- 440 с.

10. Ш. Стивен HTML, XHTML и CSS. Библия пользователя .: Вильямс, 2011.- 656 с.

11. Д. Кит HTML5 для веб дизайнеров .: Манн, Иванов и Фербер, 2012. - 112 с.

12. Е. В. Мальчук HTML и CSS. Самоучитель. – М.: Вильямс, 2008.- 416 с.

13. М. Хольцшлаг Использование HTML и XHTML. Специальное издание .: Вильямс, 2004. – 736 с.

14. XML. Справочник / Э. Р. Гарольд, У. С. Минс пер. с англ. Л. Фрейдин .: Символ-плюс, 2002.- 567 с.

15. Д Дакетт Разработка и дизайн веб-сайтов пер. с англ. М. Райтман .: Эксмо, 2013.- 480 с.