Содержание:

Введение

Не смотря на то что тема задания звучит как Языки гипертекстовой разметки, основной такой язык существует всего один. Называется он HTML.

Это аббревиатура, которая так и расшифровывается: **HyperText Markup Language** – язык разметки гипертекста. Так же существует язык XHTML (Extensible Hypertext Markup Language), повторяющий по своей сути HTML. Данный язык стал естественной ступенью развития технологий в связи с распространением сети Интернет. Нужно было придумать что-то, что поможет отображать информацию пользователям в более удобном и красивом виде.

В данной курсовой работе мы рассмотрим историю появления языка HTML, принципы его спецификации, основные приёмы и инструменты работы с ним, а также ознакомимся с целями, которые можно решить с помощью этого языка.

Глава 1. Что такое HTML

HTML – это язык гипертекстовой разметки.

Язык разметки (текста) в компьютерной терминологии — набор специализированных символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении. Принадлежит классу компьютерных языков.

Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст (как последовательность слов и знаков препинания), но и дополнительную информацию о различных его участках — например, указание на заголовки, выделения, списки, шрифты и т. д. В более сложных случаях язык разметки позволяет вставлять в документ интерактивные элементы и содержание других документов.

На сегодняшний день основная цель языка HTML – это создание ВЕБ страниц для визуализации и структурирования передачи информации в сети Интернет.

В чистом виде HTML нигде не применяется. Его комбинируют с другими технологиями, позволяющими добиться наиболее качественного результата, такими как таблицы каскадных стилей (css) и язык программирования JavaScript. Сейчас с помощью HTML в сочетании с JavaScript и CSS (каскадные стили) можно создавать по-настоящему многофункциональные динамические страницы, включающие в себя мультимедиа элементы, анимации и даже полноценные трёхмерные объекты. Но так было не всегда. Изначально HTML позволял создавать только статические страницы с форматированным текстом. Об это подробнее в следующей главе.

Глава 2. История появления языка HTML

Первая версия языка появилась в 1986 году. Она была разработана одним учёным из CERN, Тимати Джоном Бернерс-Ли. Цель, которую он преследовал – форматирование научной документации. На первых парах не было ни параметров шрифтов, ни цветовых схем, только разметка. Таким образом, изначально HTML позволял выделять в тексте заголовки, абзацы, списки и им подобные структурные элементы. Результат обработки или «воспроизведения» HTML не должен был зависеть от технических особенностей аппаратных средств его визуализации, поскольку не содержал в себе параметров этой визуализации. Со временем такая особенность языка разметки гипертекста была частично утрачена.

Сам по себе язык гипертекстовой разметки появился не на пустом месте, он брал за основу стандарт обобщённого языка разметки (SGML), который в некотором роде можно считать прообразом расширяемого языка разметки данных XML (eXtensible Markup Language). Стандарт XML в наше время приобрел огромную популярность благодаря большому количеству своих расширений, используемых в компьютерных технологиях.

SGML (англ. *Standard Generalized Markup Language* — стандартный обобщённый язык разметки) — метаязык, на котором можно определять язык разметки для документов. Но и SGML – это ещё не начало истории. Он, в свою очередь является наследником разработанного в 1969 году в IBM языка GML (Generalized Markup Language), который не стоит путать с Geography Markup Language, разрабатываемым Open GIS Consortium. На развитие же языка GML оказал влияние ещё один представитель семейства языков компьютерной разметки. Первым языком с четким и ясным различием между структурой и видом

документа был Scribe, был создан и описан докторской диссертацией Брайана Рейда (Brian Reid) в 1980 году. Scribe был революционным в количестве способов обработки, не в последнюю очередь из-за введенной идеи стилей, отделенных от собственно текста и грамматики и управляющих использованием описательных элементов. Scribe оказал влияние на разработку языка GML

Изначально SGML был разработан для совместного использования машинно-читаемых документов в больших правительственных и аэрокосмических проектах. Он широко использовался в печатной и издательской сфере, но его сложность затруднила его широкое распространение для повседневного использования.

Основные части документа SGML:

1. SGML-декларация — определяет, какие символы и ограничители могут появляться в приложении;
2. Document Type Definition — определяет синтаксис конструкций разметки. DTD может включать дополнительные определения, такие, как символьные ссылки-мнемоники;
3. Спецификация семантики, относится к разметке — также даёт ограничения синтаксиса, которые не могут быть выражены внутри DTD;
4. Содержимое SGML-документа — по крайней мере, должен быть корневой элемент.

Язык SGML предоставляет множество вариантов синтаксической разметки для использования различными приложениями. Изменяя SGML-декларацию, можно даже отказаться от использования угловых скобок, хотя этот синтаксис считается стандартным, так называемым *concrete reference syntax*.

Пример синтаксиса SGML:

<quote type="example"**>**

typically something like **<italics>**this**</italics>**

</quote>

SGML стандартизован ISO: «ISO 8879:1986 Information processing—Text and office systems—Standard Generalized Markup Language (SGML)».

Языки HTML и XML произошли от SGML. HTML до версии 4 включительно был приложением SGML, а XML — это подмножество SGML, разработанное для

упрощения процесса машинного разбора документа. Другими приложениями SGML являются SGML Docbook (документирование) и «Z Format» (типография и документирование).

Началом отсчёта входа языка HTML в повседневную жизнь пользователей принято считать 1991 год, в тоже время была придумана и сама аббревиатура.

И так, мы имеем аббревиатуры SGML, HTML, XML и XHTML, и необходимо понять, кто из них кто. На самом деле все просто: SGML это не что иное, как набор правил, на основе которых можно строить любые языки разметки.

До 1994 года HTML по-прежнему использовался только для структурной разметки данных, хотя в его составе уже появились теги для выделения текста жирным или курсивом. В том же 1994 году создается организация W3C (World Wide Web Consortium) – Консорциум всемирной паутины, которую возглавляет, что вполне логично, тот самый Тим Бернерс-Ли, и в 1995 году в свет выходит рекомендация HTML 2.0. Создатели HTML уже тогда понимали, что со временем их детище из языка статичной разметки текста эволюционирует в основной инструмент создания динамических интернет ресурсов. Главным дополнением HTML 2.0 стало появление в составе языка форм с наборами пользовательских элементов управления, которые должны были использоваться для ввода пользователем параметров HTTP запросов.

После выхода второй версии сразу же началась работа над следующим поколением HTML. В 1997 году выходит рекомендация HTML 3.2, которая дополнила язык разметки таблицами, фреймами, изображениями и некоторыми другими важными тегами. Но самым главным достижением 3-й версии является то, что ее авторы вновь вернулись к проблеме визуализации разметки в браузере, вспомнили про то, что HTML должен размечать лишь структуру документа и не должен содержать непосредственно в себе параметры графических стилей отображения элементов в браузере. Результатом их работ над HTML 3.2 стало появление самостоятельного языка CSS (Cascading Style Sheets) – каскадных таблиц стилей, код которого можно теперь подключать к коду разметки HTML и тем самым настраивать внешний вид страницы.

К выходу 4-й версии HTML в 1997 году сотрудники W3C избавили свое детище от тех ненужных элементов, которые с появлением CSS стали устаревшими и компрометировали идею отделения разметки структуры от параметризации представления. Но из-за таких мелочей никто не стал бы городить новую версию.

Основное достижение рекомендаций HTML 4.0 – появление объектной модели страницы (Document Object Model, DOM), элементами которой теперь можно было манипулировать посредствам скриптовых языков программирования, исполняемых браузерами. Самым популярным таким языком программирования является JavaScript. HTML плюс DOM плюс JavaScript равно Dynamic HTML или просто DHTML, который ознаменовал прорыв в веб-дизайне. Теперь элементы загруженной интернет страницы могли изменять свой внешний вид в ответ на действия пользователя, а также добавлять новые и удалять имеющиеся элементы. В 24.12.1999 году выходит последняя редакция 4-й версии языка разметки гипертекста – HTML 4.01.

На данный момент актуальной считается пятая версия HTML.

HTML5 внёс очень много изменений, взяв на себя такие функция, как проигрывание мультимедиа и отрисовка изображений на виртуальном полотне, оттеснив популярную до недавнего времени технологию Flash.

Глава 3. Эволюция языка HTML

Версия 1.0 содержала в основном спецификации по созданию гиперссылок, отступов и заголовков.

На смену ей пришла версия 2.0, утверждённая в 1994 г. Международной комиссией по стандартам в Intenet (IETF - Internet Engineering Task Force) в качестве стандарта. В этой версии появились возможности работы со встроенными изображениями и интерактивными формами.

Дэйв Раггет (Dave Raggett) из организации W3C предложил расширенный набор спецификаций HTML, который стал известен как HTML+ (HTML 3.0).

Тут появляется поддержка создания таблиц и URL в виде адресов электронной почты, так что гиперссылка может служить для автоматической отправки сообщений. Третья версия также поддерживает создание списков с произвольной степенью вложенности, а элементы списков могут включать горизонтальные разделители.

Были добавлены новые тэги для запуска программ поиска информации в Web. Ещё одно нововведение - возможность определять текст или графику как объекты. Это позволяет перетаскивать мышью текст или рисунки из браузера в другие

приложения.

Но стандарт HTML 3.0 так и не был принят, т.к. разработан он был без участия Netscape и Microsoft.

Тогда W3C в мае 1996 года разработал и в январе 1997 года принял стандарт HTML 3.2 со стандартизованным синтаксисом, в который уже были включены многие дополнительные дескрипторы, введённые фирмами Netscape и Microsoft.

Хотя HTML 3.2 и является относительно новым стандартом, некоторые его ограничения уже стали раздражать разработчиков, стремящихся к расширению возможностей (многим из них нравились новые нестандартные команды, даже если при их использовании приходится преодолевать различия между браузерами).

Требования разработчиков заставили фирмы - изготовители браузеров принять новый стандарт досрочно. Рабочий вариант стандарта HTML 4.0 был принят только 8 июня 1997 года, но фирмы-изготовители уже стараются приспособиться к новым спецификациям.

За разработку стандартов HTML в основном отвечает Консорциум WWW.

Консорциум World Wide Web (Consortium WWW) - это некоммерческая организация, основанная в октябре 1994 г. с целью разработки и реализации стандартов HTML и WWW. В неё входят более 165 коммерческих и академических организаций, в том числе такие гиганты, как Netscape и Microsoft. Этот консорциум возглавляет создатель Web - Тим Бернерс Ли. Проблема W3C заключается в том, что стороны никак не могут прийти к единому мнению о том, какими должны быть стандарты. Netscape и Microsoft, как и другие фирмы, активно борются за принятие своих вариантов стандарта.

Одной из наиболее актуальных проблем, связанных с публикациями в Internet, является создание стандартного формата документов, который бы позволил пользователям просматривать файлы на любом компьютере с любой ОС. Не все хотят изучать информацию в онлайновом режиме, ведь многие платят именно за время подключения. Поэтому имеет смысл размещать большие документы, такие как детальные отчёты или документация, в отдельных файлах, которые могут быть пересланы на локальный компьютер и затем прочитаны в режиме “off-line”. До последнего времени большинство документов хранилось в формате PostScript, который, как и UNIX, был фактическим стандартом Internet. Однако, с появлением других форматов переносимых документов такое положение дел начинает

меняться.

Справедливости ради нужно отметить, что каждая компания стремиться создать свой собственный стандарт на переносимые документы. Например, Adobe Acrobat - это набор средств для создания и просмотра документов в формате PDF (Portable Document Format - формат переносимых документов). В этом формате может быть представлен практически любой документ.

Четвёртая версия HTML продержалась очень долго, HTML5 начал свой путь только в 2004 году, а на мировую арену вышел лишь 28 октября 2014 г.

Если HTML4 устраивал всех больше десяти лет, зачем нужно было обновляться в 2014? Самое значимое различие между старыми версиями HTML и HTML5 заключается в интеграции видео и аудио в спецификации языка. Кроме того, в HTML5 вошли следующие обновления:

- были удалены устаревшие элементы, такие как center, font и strike;
- улучшение правил парсинга сделало его более гибким и совместимым;
- появились новые элементы video, time, nav, section, progress, meter, aside и canvas;
- новые атрибуты для инпутов, в том числе email, URL, dates и times ;
- новые атрибуты, в том числе charset, async и ping;
- новые API с офлайн кэшированием и поддержкой drag-and-drop и т.д;
- поддержка векторной графики без сторонних программ типа Silverlight или Flash;
- поддержка MathML улучшила отображение математических обозначений;
- благодаря JS Web worker API, JS теперь может работать в фоновом режиме;
- глобальные атрибуты типа tabindex, repeat и id теперь можно применять ко всем элементам.

Обновление стало невероятно глобальным, первое время HTML5 считали чуть ли не самостоятельным языком, однако после введения повсеместной поддержки данного стандарта сама формулировка HTML5 стала не корректной.

Язык развивается и по сей день и в будущем было принято отойти от практики выдачи номерных версий html. Все изменения будут вносится в существующую спецификацию, а язык будет носить название просто HTML.

Такие глобальные обновления не могли пройти сами собой, не задев сопряжённые технологии. Вместе с HTML5 вышел в мир и новый стандарт каскадных таблиц

стилей CSS3.

Так как с момента появления каскадных стилей было принято разделять разметку и оформление, введение такого большого числа новшеств, тегов и атрибутов в язык разметки не могло не задеть и язык оформления стилей. Новые технологии заставляли идти в ногу со временем и давать возможность добиваться современного результата веб страниц без применения сторонних технологий.

Также глобальное изменение претерпела не только структура самого документа, список используемых тегов и поддерживаемых технологий, но методика работы с ним и принципы построения документа.

Изначально все документы на HTML были линейным текстом с небольшим визуальным форматированием, затем были добавлены изображения и эти документы приобрели более презентабельный вид. Но с течением времени росли и потребности пользователей, с момента ввода в спецификацию HTML поддержки таблиц, веб документы стали строится по более сложной схеме. Появилась так называемая табличная вёрстка.

При этой вёрстке ведь документ представлялся в виде большой таблицы из строк и столбцов, некоторые из них были объединены в большие ячейки, таким образом получалась сложная структура, позволяющая добавить на страницу дополнительные информативные и функциональные элементы, такие как меню навигации, информационные поля, визуальные украшения и фирменные знаки.

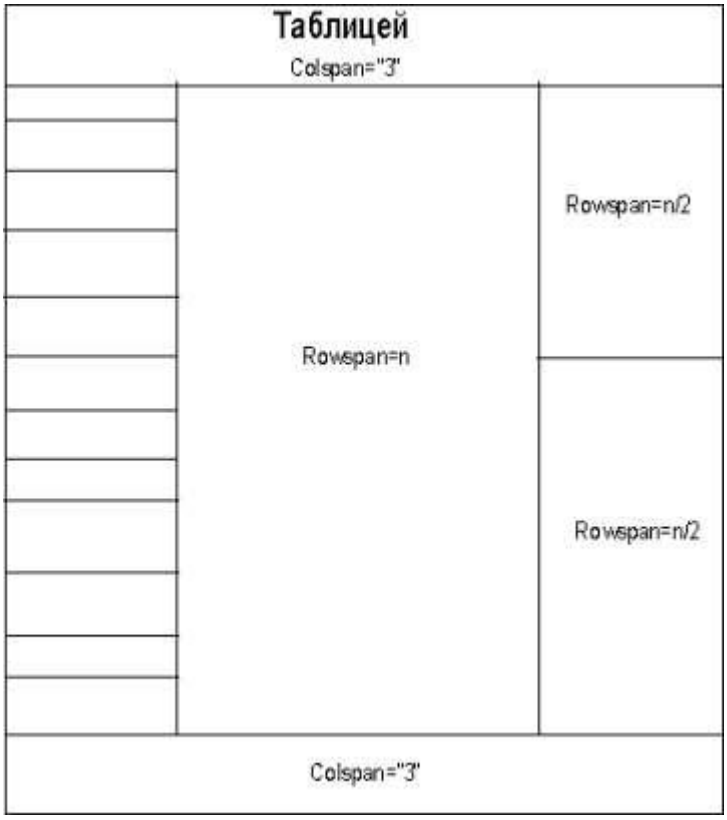


Рис 1. Пример разделения таблицы с применением табличной вёрстки

Данный метод существовал очень долго, но в конце концов изжил себя, так как для описания таблиц требуется описать много дополнительных тегов и атрибутов, код страницы становился перегружен информацией.

Спустя некоторое время на смену табличной вёрстке пришла, так называемая, блочная.

Это произошло с введением в спецификацию тега DIV

Блочная вёрстка давала больше возможностей при меньшей нагруженности кода страницы.

Пользователи не видели разницы, так как страница визуально могла выглядеть точно так же, но создание и поддержка страниц с блочной вёрсткой не только были проще и быстрее, но и давали больше возможностей и большую производительность.

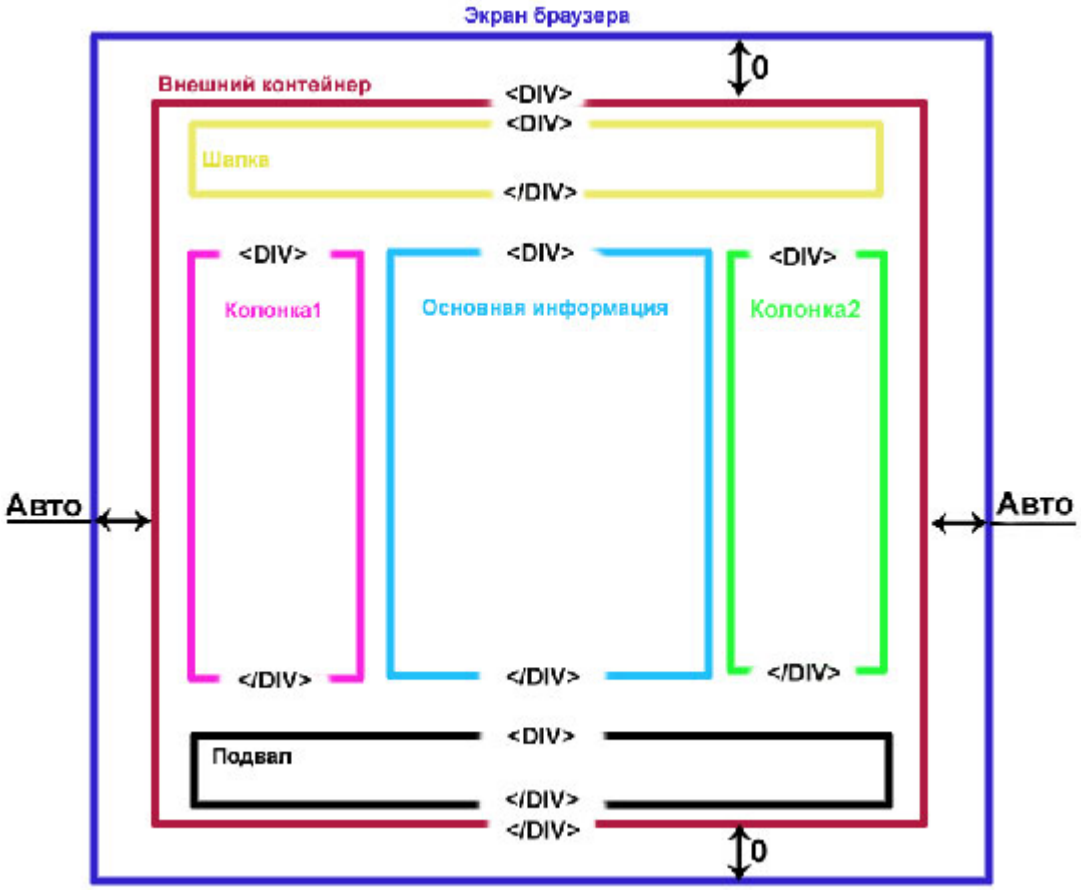


Рис 2. Пример структуры страницы с применением блочной вёрстки.

Глава 4. История развития CSS

CSS(Cascading Style Sheets) с английского переводится как каскадные таблицы стилей, - это язык, который используется как средство оформления web-страниц, а именно для работы с шрифтами, цветами, полями, таблицам, картинками, расположением элементов.

Основной целью, которая послужила для разработки языка CSS, являлось разделение разметки содержимого (которое написанного на HTML или другом языке программирования) и представления документа (написанного на CSS). Результат этого разделения помогает увеличить доступность документа, предоставить большую гибкость, а также уменьшить сложность и повторяемость в структурном содержимом, создает удобное описание дизайна и стилистики web-страницы и ее содержимого. Еще CSS дает возможность предоставить один документ в различных стилях или методах вывода (например, таких как экранное представление, чтение голосом, печать).

Первое упоминание о CSS было в 1994 году, когда Хоком Виум Ли предложил использовать CSS (Каскадные таблицы стилей) для стилистического оформления web-страниц. У него не сразу получилось продвинуть свою технологию, - только через пару лет ему удалось привлечь внимание к CSS. Итак, 17 декабря 1996 года опубликована первая спецификация (CSS1) и она была рекомендована к использованию Консорциумом Всемирной паутины (W3C).

После небольшого успеха положение дел у технологии CSS пошли куда лучше и 12 мая 1998 года (через 2 года) была принята рекомендация W3C для CSS2. Следующим этапом была CSS 2.1 - версия W3C от 8 сентября 2009 года, она была построена на базе CSS2, и была работой над исправлением существующих ошибок.

Сегодня актуальна версия CSS3, которая максимально расширена по сравнению с предыдущими версиями. CSS3 включают эффекты теней, а также скругленные углы у блоков. В CSS3 появилась возможность устанавливать изображения в качестве фона и использовать их как границы. Намного проще и удобнее стала работа с анимацией – теперь для ее создания достаточно возможностей самой CSS3, стало не нужно работать с языком Java Script.

Были внесены такие новшества как:

- Фон и границы: CSS3 позволяет разработчикам по-новому стилизировать фон и границы HTML-элементов. Ниже представлены CSS-свойства, которые позволяют это сделать:
- border-radius – позволяет закруглять границы (border) HTML-элементов
- border-break – заканчивает границу элемента, которая была рассположена в точке разрыва (конец страницы, колонки, строки)
- border-image – устанавливает изображение как границу (вместо использования стилизированных границ)
- box-shadow – добавляет тень элементу
- Многоколоночное расположение элементов: Данная возможность CSS3 позволяет отображать информацию в нескольких колонках, с помощью меньшего количества HTML-кода. Содержание в многоколоночном элементе распределяется по принципу: от колонки к колонке. Column box находится в родительском div-контейнере, содержащему контент. Свойства column-count и column-width определяют количество колонок и их ширину. Все колонки равны по ширине, высоте и имеют определенное расстояние между собой.
- Расположение элементов: Главная задача данного модуля CSS - разделение контента и представления. CSS3 упрощает позиционирование элементов

двумя новыми методами: шаблонное позиционирование (Template Based Positioning) и отображение в виде закладок (Tabbed Displays). Шаблонное позиционирование позволяет расположить элементы в ячейках условной сетки. Отображения с помощью закладок располагает элементы в виде закладок, показывая только выбраный элементу.

- Позиционирование по сетке. Цель этого модуля CSS - упрощение позиционирования элементов: мы создаем сетку и выравниваем элементы внутри ячеек. Сетка может иметь один из следующих типов:
- Явная сетка – сетка с явно заданными столбцами и строками.
- Естественная сетка – создается автоматически, например в многоколоночных элементах (края с обеих сторон колонки создают границы сетки).
- Сетка по умолчанию - элементы, которые не подходят под вышеперечисленные определения, рассматриваются как имеющие сетку с одной единственной ячейкой.
- Новые селекторы CSS3. В CSS3 были добавлены новые виды селекторов. Некоторые из них приведены ниже:
- E:only-of-type - выбирает элемент, который является ближайшим sibling элементом того же типа
- E:not(s) - возвращает любой элемент, который не соответствует простому селектору s
- E ~ F - возвращает элемент F, который следует после элемента E
- E:nth-child(n) - возвращает элемент, который является n-ым по счету дочерным элементом исходного
- E:nth-last-child(n) – выполняет то же, что и предыдущий селектор, но счет идет от последнего элемента
- E:nth-of-type(n) - возвращает элемент, который является n-ым по счету sibling элементом того же типа
- Расширенная поддержка медиа-устройств. Цель нововведения - расширить поддержку различных типов медиа путем добавления css-свойств, вычисляемых с учетом текущего типа медиа. Некоторые из них могут быть использованы для задания высоты, ширины и цвета элемента в зависимости от медиа типа. Это позволит настраивать внешний вид страницы для различных устройств без переключения режимов отображения страницы. Благодаря этому стала считаться устаревшей методика создания отдельно мобильного сайта, отдельно компьютерного. Пользователи и мобильных и полноценных компьютерных устройств заходят на один и тот же сайт, но свойства CSS определяют какой именно вариант вёрстки страницы необходимо показывать

пользователю. Также с помощью медиа запросов можно определять отдельный внешний вид для вывода, например, на печать.

Глава 5. Основные принципы работы HTML

Основной элемент HTML, как и любого другого языка разметки – это теги. Тег – это некое ключевое слово, заключённое в острые скобки <>. Теги бывают открывающими и закрывающими. Закрывающий тег отличается от открывающего наличием слеша перед ключевым словом, например: .

Любой документ на языке HTML представляет собой набор элементов. Именно тегами и отмечается начало и конец этих элементов. Элементы могут быть *пустыми*, то есть не содержащими никакого текста и других данных (например, тег перевода строки
). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь *атрибуты*, определяющие какие-либо их свойства (например, размер шрифта для тега <font>). Атрибуты указываются в открывающем теге. Вот примеры фрагментов HTML-документа:

- Текст между двумя тегами — открывающим и закрывающим.
- Здесь элемент содержит атрибут href, то есть гиперссылку.
- А вот пример пустого элемента:

Регистр, в котором набрано имя элемента и имена атрибутов, в HTML значения не имеет (в отличие от XHTML). Элементы могут быть вложенными. Например, следующий код:

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="utf-8" />

<**title**>HTML Document</**title**>

</**head**>

<**body**>

<**p**>

<**b**>

Этот текст будет полужирным, <**i**>а этот — ещё и курсивным</**i**>.

</**b**>

</**p**>

</**body**>

</**html**>

даст такой результат:

Этот текст будет полужирным, *а этот — ещё и курсивным*.

Кроме элементов, в HTML-документах есть и *сущности* (англ. *entities)* — специальные символы. Сущности начинаются с символа амперсанда и имеют вид &*имя*; или &#*NNNN*;, где *NNNN* — код символа в Юникоде в десятичной системе счисления.

Например, © — знак авторского права(©). Как правило, сущности используются для представления символов, отсутствующих в кодировке документа, или же для представления «специальных» символов: & — амперсанда (&), < — символа «меньше» (<) и > — символа «больше» (>), которые некорректно записывать «обычным» образом, из-за их особого значения в HTML.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML <!DOCTYPE…>, которая обычно выглядит примерно так:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

Если эта строка не указана, то добиться корректного отображения документа в браузере становится труднее.

Далее обозначается начало и конец документа
тегами <html> и </html> соответственно. Внутри этих тегов должны находиться теги заголовка (<head></head>) и тела (<body></body>) документа.

Сами по себе документы HTML – это просто текстовые файлы. Их отрисовкой должны заниматься специализированные программы, называемые браузерами. Браузеры отображают документ в его форматированном виде. Обычно они предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются Google Chrome, Mozilla Firefox, Opera, Internet Explorer и Safari.

Из-за наличия на рынке большого количества браузеров возникает ситуация, при которой одна и та же страница может отображаться по разному на различных устройствах и даже на одном устройстве в окне разных браузеров. Это добавляет неудобства верстальщикам – специалистам по созданию HTML страниц. Поэтому и разрабатываются стандарты и дополнительные инструменты, в основном на основе каскадных стилей (css), позволяющие писать универсальные интерфейсы, отображаемые одинаково на всех устройствах и браузерах. Такая вёрстка называется кроссбраузерной.

Стоит так же рассмотреть вопрос каскадных стилей, так как они являются неотъемлемой частью работы с html документов. С момента создания CSS из HTML было вырезано практически всё, что конфликтовало с принципом разделения разметки и оформления.

Синтаксис каскадных стилей подразумевает последовательное описание свойств всей страницы, групп элементов или отдельных элементов на странице. Свойства описываются по принципу «название: значение». При перечислении нескольких свойств в конце строки обязательно ставить знак «;»

Стили можно применять несколькими способами

1. как атрибут тегов:

нижний колонтитул

```
<style type="text/css">

td {

font-family : "trebuchet ms", Verdana, Arial, Sans-Serif;

font-size : 14px;

}

</style>
```

1. Можно описывать непосредственно внутри html документа, заключая правила оформления страницы между тегами <style></style>
2. Либо подключить внешний файл с расширением .css.

```
<link rel="stylesheet" href="main.css">
```

```
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
.small_box {
font-size: 8pt;
}
body, html {
width: 100%;
height: 100%;
}
```

Не обязательно выбирать один из вариантов, можно комбинировать их в любом соотношении.

Стиль с элементом можно связать одним из трёх способов:

1. По имени тега: теги пишутся как есть без дополнительных символов.
2. По имени класса тега: у каждого тега можно задать атрибут class, по значению этого атрибута и можно связывать стиль с элементами. Это делается с помощью указания имени класса и установки знака «.» перед именем. Например, у тега text имя класса равно SomeClass, значит в файле стилей к нему можно будет обратиться так: .SomeClass {}
3. По имени уникального идентификатора конкретного элемента. Идентификатор указывается в атрибуте id любого тега на странице, а образение к нему происходит с подставлением перед идентификатором символа «#». К элементу text стоит обращаться так: #SomeClass {}

Глава 6. Отличия HTML и XHTML

Если в основе HTML лежит стандарт SGML, то в основе XHTML заложен XML.

Функциональность у XHTML и HTML одинаковая, но отличается принцип построения документа и его обработки. Поскольку XHTML является расширением XML, то все требования к правильно сформированному (well-formed) XML документу сохраняются. Вот те самые дополнительные требования к разметке документа, если он должен соответствовать стандарту XHTML:

1. Каждый тег XHTML должен быть закрыт. Если HTML позволял конструкции типа
 или <hr>, то в XHTML они должны выглядеть только так:
<hr/>. Менее тривиальным является следующий вариант разметки, который устраивает HTML, но не является корректным с точки зрения XML: boldbold_and_italicitalic. Правильным XHTML аналогом будет являться следующая разметка boldbold_and_italicitalic. На основе приведенного примера, первое ограничение я бы дополнил формулировкой: XHTML не допускает частичного пересечения области действий тегов разметки. Если это обстоятельство и создает какие-то дополнительные сложности верстальщикам, то эти сложности с лихвой компенсируются контролем над ошибками со стороны сервисов XML. Причины всех этих ограничений проявятся дальше.
2. XHTML не поддерживает сокращенной формы записи атрибутов. Это означает, что в XHTML нет сокращенной формы записи булевых атрибутов, а само значение атрибутов всегда должно быть в кавычках. Если в HTML следующий код считался корректным <input type=textbox readonly value='anytext'/>, то в XHTML приведенная конструкция должна выглядеть следующим образом: <input type='textbox' readonly='readonly' value='anytext'/>.
3. Специальные символы в XHTML должны быть представлены в виде кодов. Это означает, например, что символы < и >, если они не являются частью разметки, должны в тексте обозначаться, как < и > соответственно. Если такой вариант не устраивает, например, если требуется в разметку добавить программный код (Java-script, VBScript), то для этих целей следует использовать раздел CDATA, содержимым которого может быть любая символьная информация, в том числе специальные символы разметки. Вот пример: <script type="text/javascript"><![CDATA[Код скрипта]]> </script>

4. Все символы, используемые в именах тегов и атрибутов должны быть строчными. Вот это ограничение уже не является наследием XML, поскольку XML настаивает только на том, чтобы и открывающий и закрывающий теги были записаны одинаковым набором символов, в который могут входить как строчные, так и заглавные символы. Это ограничение, скорее, результат стремления избежать путаницы и оптимизировать скорость обработки документа. Кодировка символов в XHTML, как и в XML по умолчанию UTF-8.
5. Корневой элемент в XHTML должен быть один. Другими словами это означает, что тег HTML должен присутствовать в XHTML всегда! Стандарт HTML не настаивал на присутствии тегов <HTML> и <BODY> – разметку можно было начинать с любого тега и корневого элемента могло не быть вообще.

Есть ещё несколько отличий, чуть менее очевидных, но точно так же исходящих из того, что XHTML является расширением XML, а HTML нет.

Обработчик HTML является, по сути, интерпретатором. Он обрабатывает документ последовательно, и именно это обстоятельство позволяет ему исправлять ошибки разметки. Многим известно, что документ HTML в памяти браузера представлен в виде объектной модели DOM. Именно DOM является основой DHTML (Dynamic HTML) – симбиоза HTML и JavaScript, который способен “оживлять” статичную разметку, обрабатывая события пользователя.

Код javascript и DOM способны одни фрагменты документа “на лету” заменять другими или существенно изменять стиль их отображения. Преобразования эти осуществляются на стороне клиента, благодаря чему перезагрузки страницы не требуется. Все выглядит красиво и динамично. Именно благодаря DOM стали возможны все выпадающие меню и списки на страницах браузера. Для корректного формирования объектной модели интерпретатору HTML жизненно необходимо исправлять ошибки верстальщиков, добавляя закрывающие теги, исправляя частичные пересечения области действия тегов разметки (пример выше) и т.п. Обработчик XML, в свою очередь, больше походит на компилятор: он обрабатывается сразу весь документ.

Если документ не является well-formed (например, нарушает стандарты xml), то обработчик (парсер) сообщает об ошибке и отменяет формирование DOM целиком.

Логично, что в случае XHTML происходит все то же самое. Для XML и его расширений действует правило: “Либо все, либо ничего”, в то время как парсер HTML не сообщает об ошибках, старается их исправлять и практически всегда

производит, как минимум, частичную обработку HTML страницы. Может показаться, что такое поведение интерпретатора HTML более удобно, нежели парсера XHTML, однако есть нюансы.

Исправление HTML происходит на стороне клиента, т.е. интернет браузером. Различных браузеров достаточно много, не говоря уже о большом количестве версий каждого из них. Это обстоятельство не гарантирует вам, что во всех случаях ошибки будут исправлены одинаково и результат этого исправления вас устроит. Также можно утверждать, что на исправление ошибок тратится дополнительное время. Не стоит в этом полагаться на браузер.

На данный момент не все интернет браузеры поддерживают XHTML. Для того чтобы поэкспериментировать с этим новым стандартом локально, достаточно файлу с гипертекстом дать расширение .xhtml и открыть его в поддерживающем XHTML интернет браузере, например, в Opera. Если разметка не будет соответствовать well-formed XML, то браузер выведет сообщение об ошибке. Если у тега не будет определено пространство имен xmlns=http://www.w3.org/1999/xhtml, то вы увидите просто xml, а не обработанную страницу.

При разработке сайтов рекомендуется придерживаться правил XHTML, и проверять страницу на соответствие этому стандарту локально. А какой обработчик будет использован браузером пользователя не так важно, поскольку если страница соответствует синтаксису XHTML, то синтаксису HTML она соответствует точно. Принудительно использовать парсер XHTML можно заставить браузер в любой момент, настроив должным образом заголовки интернет страниц на сервере.

Глава 7. Инструменты для работы с HTML

Так как документы html являются текстовыми, для работы с ними подойдёт абсолютно любой текстовый редактор. Но ввиду специфику содержимого лучше выбирать редакторы, которые будут обеспечивать подсветку синтаксиса языка разметки. Это значительно упрощает работу в разы.

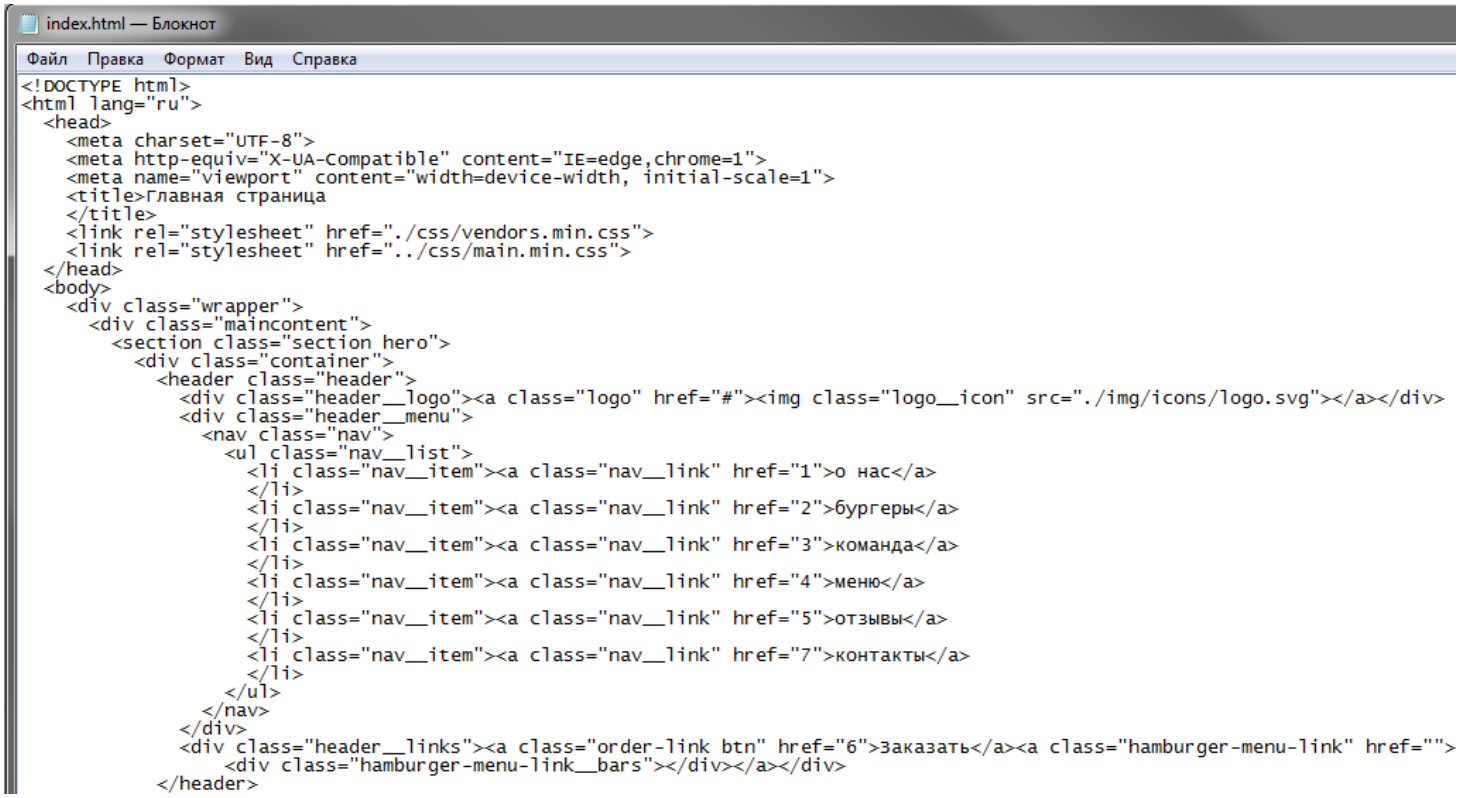
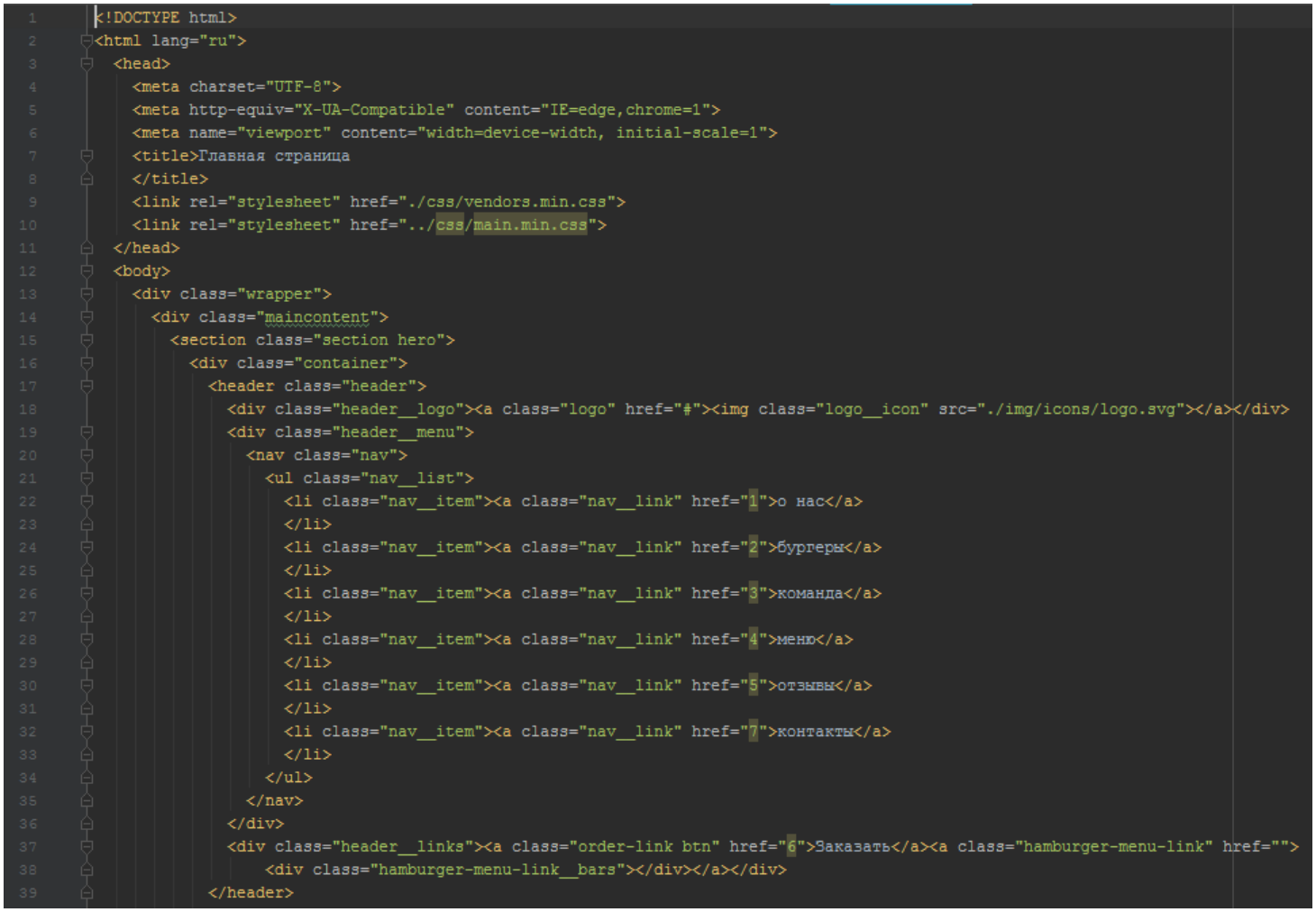


Рис 3. Пример html документа в редакторе без подсветки синтаксиса.



```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Главная страница
    </title>
    <link rel="stylesheet" href="./css/vendors.min.css">
    <link rel="stylesheet" href="../css/main.min.css">
  </head>
  <body>
    <div class="wrapper">
      <div class="maincontent">
        <section class="section hero">
          <div class="container">
            <header class="header">
              <div class="header__logo"><a class="logo" href="#"><img class="logo__icon" src="./img/icons/logo.svg"></a></div>
              <div class="header__menu">
                <nav class="nav">
                  <ul class="nav__list">
                    <li class="nav__item"><a class="nav__link" href="1">о нас</a>
                    </li>
                    <li class="nav__item"><a class="nav__link" href="2">бургеры</a>
                    </li>
                    <li class="nav__item"><a class="nav__link" href="3">команда</a>
                    </li>
                    <li class="nav__item"><a class="nav__link" href="4">меню</a>
                    </li>
                    <li class="nav__item"><a class="nav__link" href="5">отзывы</a>
                    </li>
                    <li class="nav__item"><a class="nav__link" href="7">контакты</a>
                    </li>
                  </ul>
                </nav>
              </div>
              <div class="header__links"><a class="order-link btn" href="6">Заказать</a><a class="hamburger-menu-link" href="">
                  <div class="hamburger-menu-link__bars"></div></a></div>
            </header>
```

Рис 4. Пример html документа с подсветкой синтаксиса.

Среди таких редакторов можно выделить несколько наиболее популярных и удобных. От самых простых: notepad++ и sublime text, до серьёзных и коммерчески выгодных: WebStorm, PhpStorm, Visual Studio, Atom.io.

Так же для работы с html со всеми вышеупомянутыми редакторами применяется множество расширений и дополнительных инструментов, ускоряющих работу. Такие инструменты умеют автоматически заканчивать имена тегов, компилировать законченный документ из сокращённого синтаксиса, автоматически обновлять окно браузера для отображения внесённых в документ изменений «на лету» и многое другое.

Существует множество онлайн (работающих в окне браузера) редакторов, облегчающих работу фронтэнд разработчиков. Фронтэнд (англ. Frontend) разработчики – это специалисты, занимающиеся созданием клиентской части веб приложений, то есть как раз и работают с html, css и JavaScript.

Но не все работают с html файлами только в текстовом формате. Существуют редакторы, позволяющие менять именно визуальную составляющую документа, она так и называются: Визуальные редакторы.

Визуальные или же WYSIWYG редакторы позволяют редактировать именно внешнюю составляющую будущей веб страницы, а сам код разметки, таблиц каскадных стилей и javascript скриптов формируются программой в фоновом режиме. То есть при работе с ними разработчик апеллирует не текстом разметки, а графическими элементами, из которых состоит страница.

Самыми популярными WYSIWYG редакторами на рынке являются: Adobe Dreamweaver, Adobe Muse, KompoZer, Amaya и многие другие. Они очень удобны с той точки зрения, что разработчик сразу же видит результат. Нет необходимости писать сотни строк кода, подгонять стили, постоянно обновлять окно браузера. Всё происходит «на лету». Так же они ускоряют процесс создания прототипа будущей веб страницы либо позволяют создать веб документы, не обладая знаниями в области компьютерной вёрстки и программирования.

Визуальные редакторы делятся на несколько типов: предназначенные в первую очередь для вёрстки, такие как например Dreamweaver.

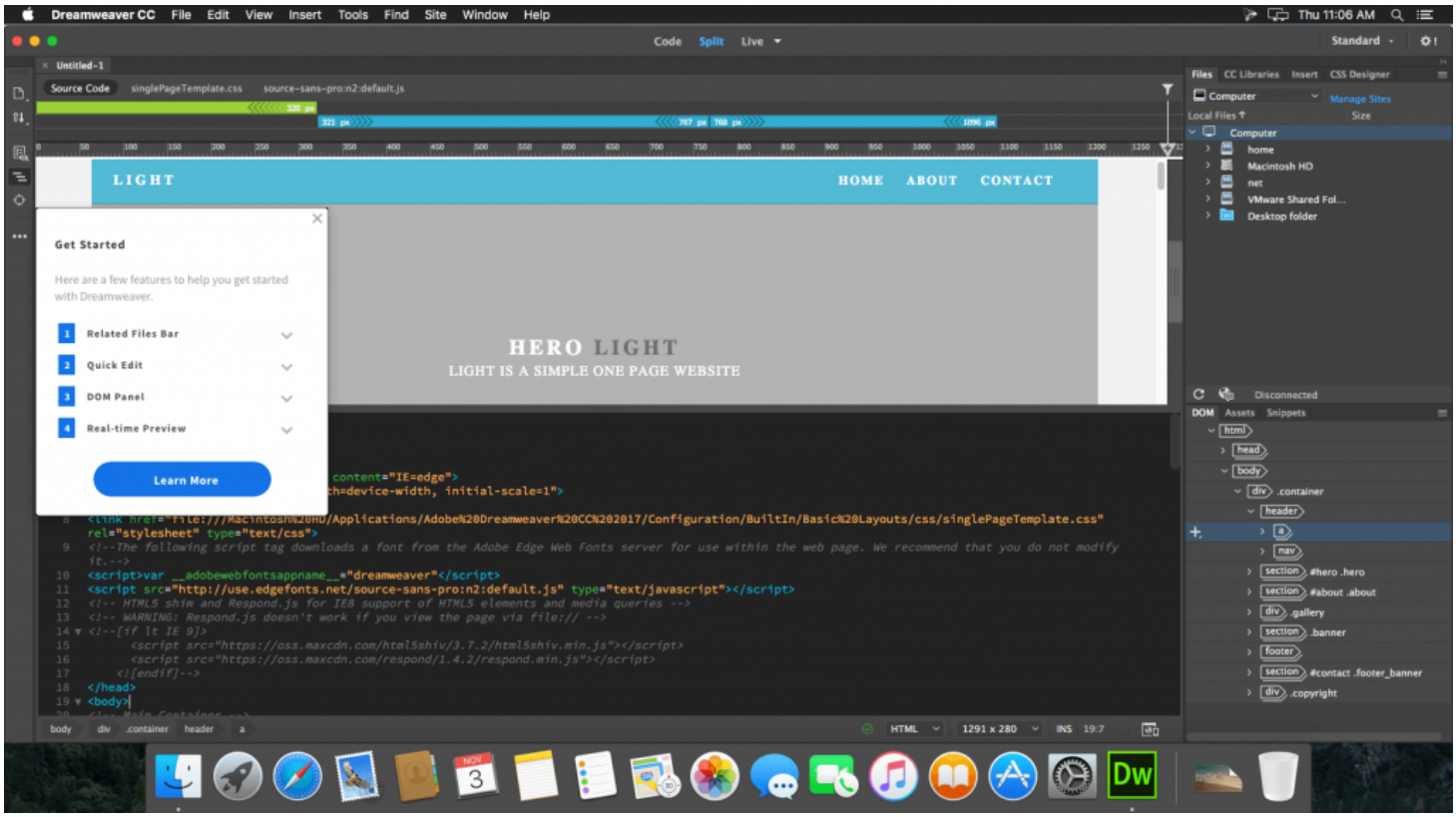


Рис 5. Пример окна визуального редактора Adobe Dreamweaver

Или же заточенные на создание прототипов и дизайна, например Adobe Muse.

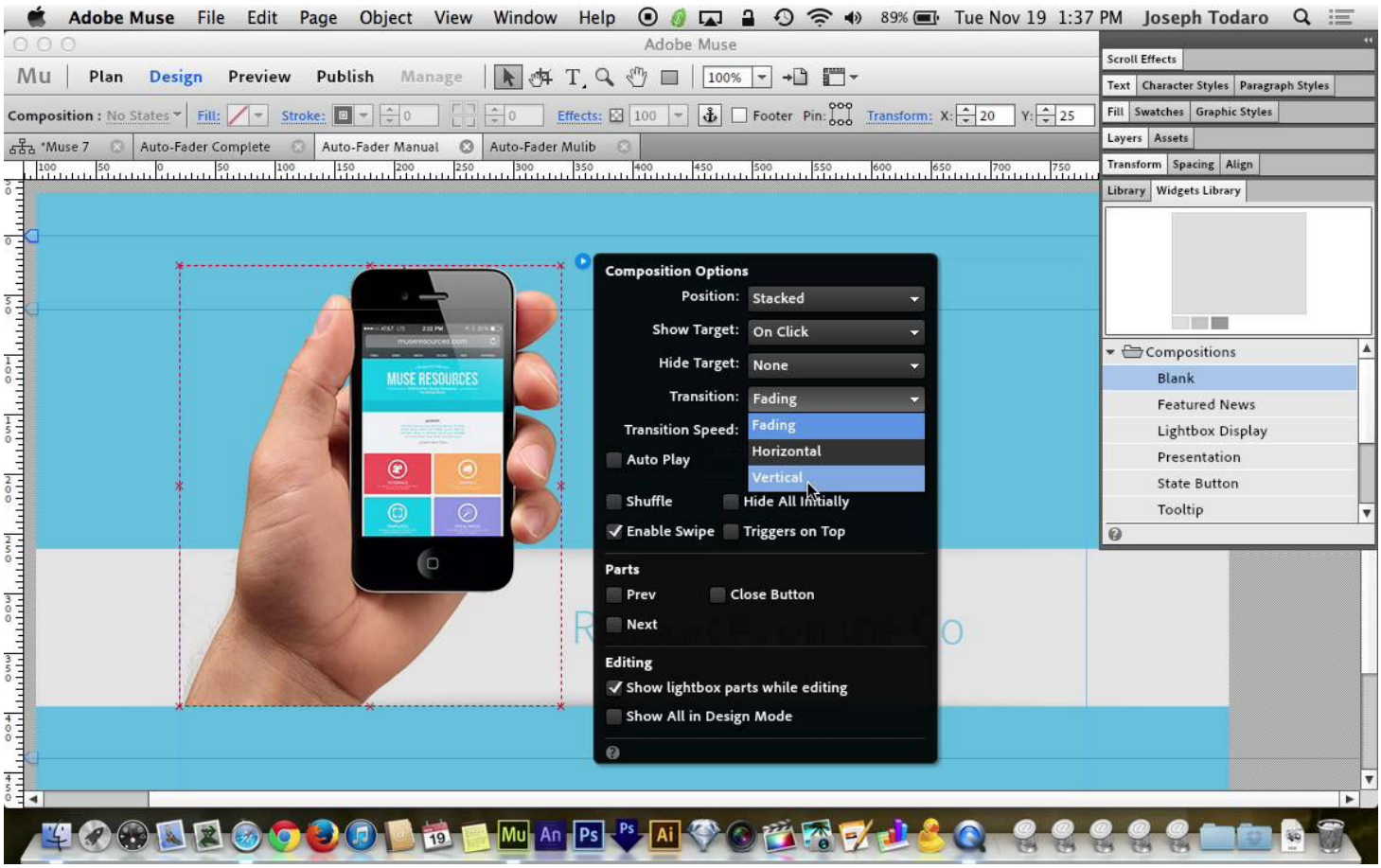


Рис 6 Окно визуального редактора Adobe Muse.

Но если бы всё было так хорошо, никто бы не пользовался редакторами кода. Здесь так же есть свои минусы: визуальные редакторы ВЕБ страниц генерируют код разметки и стилей не опираясь на производительность, лаконичность, человеческий опыт и дальнейшую читабельность документов.

Код, полученный на выходе из таких инструментов в большинстве своём очень громоздкий, содержит множество лишних элементов, перегружен свойствами и слабо читаем, что усложняет его дальнейшую поддержку.

К таким редакторам стоит прибегать в одном из нескольких случаев:

- Дальнейшая поддержка документа не планируется. Необходимо создать «одноразовый» документ.
- Необходимо в короткие сроки разработать прототип для его демонстрации и затем уже переходу к разработке полноценного веб документа.
- Нет возможности привлечь к разработке квалифицированных специалистов

- Недостаточно знаний для создания страницы с нуля, но достаточно для оптимизации автоматически сгенерированного года.

Так как на сегодняшний день понятие HTML неразрывно связано с созданием веб страниц, к программам, без которых этот процесс не обходится, стоит так же отнести и графические редакторы. К HTML они относятся косвенно, однако, без них не происходит создание практически ни одного сайта, а также они имеют вспомогательный функционал, способный упростить настройку некоторых элементов на странице, к примеру, копированием уже готового CSS кода.

К самым известным из них относятся следующие программы: Adobe Photoshop, Adobe Illustrator и Coral Draw.

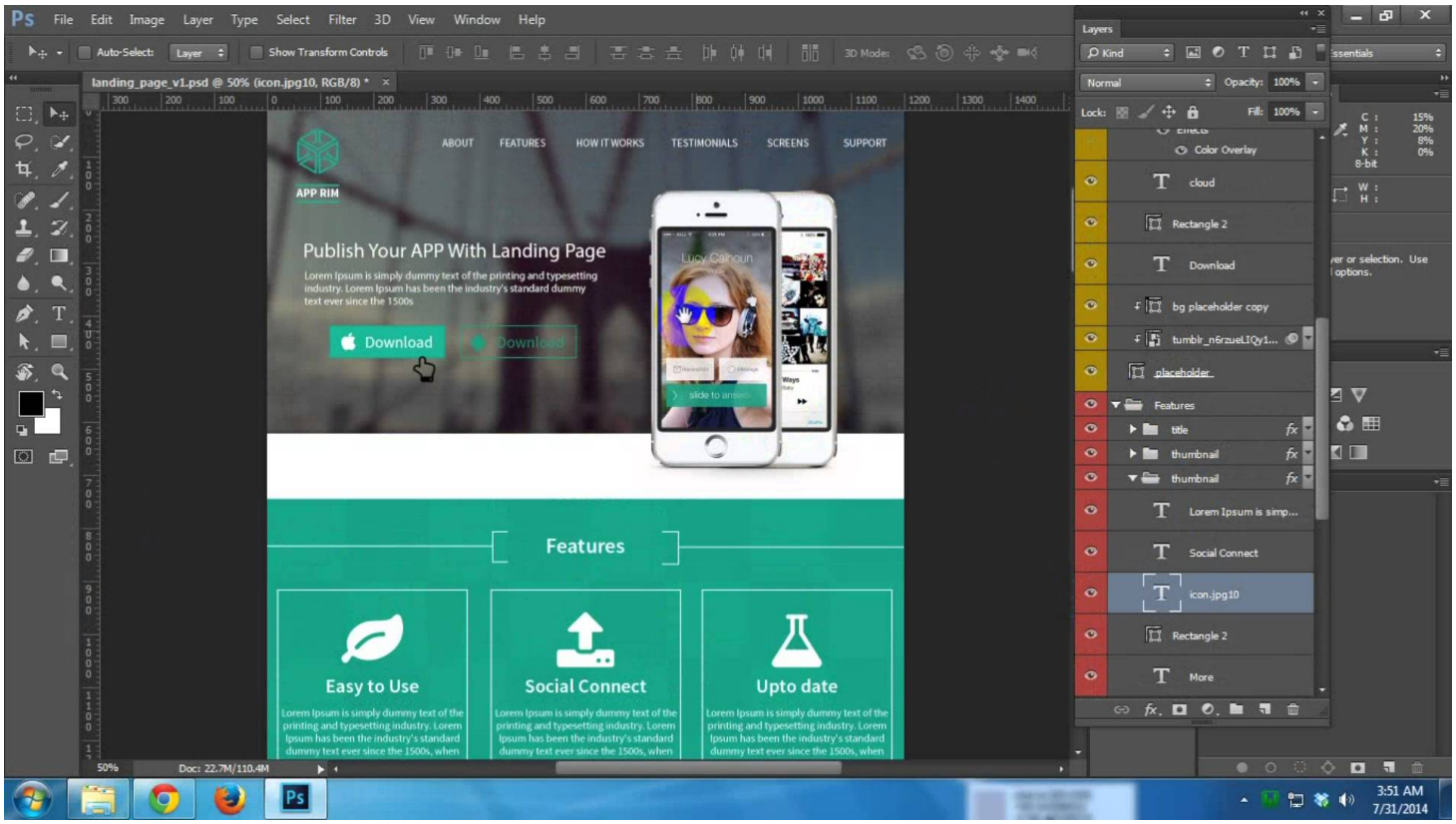


Рис 7. Окно программы Adobe Photoshop применяемой для дизайна веб страниц.

Заключение

HTML – одна из немногих технологий, которая, появившись ещё в середине 80-ых годов прошлого столетия, не просто дожила до наших дней, но и продолжает развиваться.

Являясь практически единственным в своей нише стандартом, HTML не имеет конкурентов. Это позволяет ему развиваться планомерно и без спешки.

Изобретённый некогда как язык форматирования научной документации, на сегодняшний день HTML завоевал 100% рынка веб страниц. Даже страницы, созданные с помощью других технологий, таких как Flash или WebGL, вынуждены использовать HTML в качестве обёртки для своих объектов.

На ближайшее время смерть HTML как стандарта языка разметки не предвидется.

Используемая литература

1. Пол Хейн / Схематика, Стандарты и Стиль [Текст]: учебное пособие, Пол Хейн. – 2010. – 250 с.
2. В. В. Дунаев, Е. Беляева / (X)HTML, скрипты и стили. Самое необходимое [Текст]: учебное пособие, В. В. Дунаев, Е. Беляева. – БХВ-Петербург, 2009. – 496 с.
3. Эрик Фримен, Элизабет Робсон / Изучаем HTML, XHTML и CSS [Текст]: учебное пособие, Эрик Фримен, Элизабет Робсон. – Питер, 2012. – 720 с.
4. Муссиано Ч., Кеннеди Б. / HTML и XHTML. Подробное руководство [Текст]: Учебное пособие / Муссиано Ч., Кеннеди Б. – Символ-Плюс, 2008. – 752 с.