**2.Что такое HTML?**

Термин HTML (Hyper Text Markup Language) означает "язык маркировки гипертекстов". Это понятие более широкое, включает в себя Интернет и локальные сети, редакторы, браузеры, разнообразные программные продукты, компакт-диски, обучающие курсы, дизайн и многое другое. HTML – своеобразная противоположность сложным языкам программирования, известным только специалистам.

**2.1HTML как явление нашей жизни**

HTML давно перестал быть просто языком программирования. Человек, изучавший этот язык, обретает возможность делать сложные вещи простыми способами и, главное, быстро, что в компьютерном мире не так уж и мало. Гипертекст подходит для включения элементов мультимедиа в традиционные документы. Практически именно благодаря развитию гипертекста, большинство пользователей получило возможность создавать собственные мультимедийные продукты и распространять их на компакт-дисках. Такие информационные системы, выполненные в виде набора HTML-страниц, не требует разработки специальных программных средств, так как все необходимые инструменты для работы с данными (WEB-браузеры) стали частью стандартного программного обеспечения большинства персональных компьютеров. От пользователя требуется выполнить только ту работу, которая относится к тематике разрабатываемого продукта: подготовить тексты, нарисовать рисунки, создать HTML-страницы и продумать связь между ними.

HTML является основой моды в Интернете.

HTML, как основа создания WEB-страниц, имеет прямое отношение и к новому направлению изобразительного искусства – WEB-дизайн. Художнику в Интернете недостаточно просто нарисовать красивые картинки, оригинальный логотип, создать новый фирменный стиль. Он должен еще поместить все это в Сети, продумать связь между WEB-страницами, чтобы все двигалось, откликалась на действие пользователя, поражало воображение, вызывало желание создать что-нибудь свое, оригинальное в этой области.

**2.2Версии HTML**

Первая версия HTML была разработана в 1989 году Тимом Бернес-Ли для популярного в прошлом браузера Mosaic. Но в то время ни для языка, ни для браузера не нашлось достойного применения. В 1993 году появился HTML+ , и эта версия также осталась практически незамеченной. Начало широкого использования гипертекста дала версия 2.0  которая, появилась в июне 1994 года. Это был год роста популярности WWW по всему миру. Элементы, включенные в версию 2 , в большинстве своем используются по сей день.

В версии 3.0 HTML, которая появилась год спустя, была реализована возможность прорисовки математических символов (знаков интервала, бесконечности, дроби, скобок и т.д.) при помощи элементов языка. Под эту версию был разработан браузер Arena. Но этот проект оказался тупиковым и не получил дальнейшего распространения.

В 1996 году появился HTML 3.2. Это было новаторское решение, в спецификацию языка были введены фреймы, которые стали теперь весьма популярны у разработчиков WEB-страниц. Даже сейчас на основе этой спецификации можно реализовать интересные дизайнерские решения. Практически все современные браузеры поддерживают версию 3.2, поэтому авторы WEB-страниц уверенны в работоспособности всех элементов.

Наряду с официальными спецификациями языка, которые разрабатывались организацией W3C (W3 Консорциум), компании-производители браузеров создавали собственные элементы (расширения). Впоследствии, некоторые из этих элементов, после получения всеобщего признания включились в спецификацию следующих версий языка. Но новаторское решение – фреймы, не были включены в спецификацию 3.2. Но браузеры поддерживали фреймы и многие книги, посвященные HTML, содержали описание фреймов без упоминания о том, что это нестандартные элементы. В последствии, фреймы стали стандартом де-факто. В версии 4 они уже были включены на полном основании.

И наоборот, элементы APPLET и SCRIPT, необходимые для расширения HTML другими программными кодами  версии 3.2, не сыграли той роли, которую были призваны сыграть. Это объясняется тем, что браузеры различных версий по-разному  интерпретировали программы на разных языках JAVA, JAVASCKRIPT, Visual Basic (VBScript). В результате не удалось получить достаточно надежный работающий код, и данные языки использовались любителями HTML в основном для экспериментов.

Официальная спецификация HTML 4 (Dynamic HTML) появилась в 1997 году. В это время уже было очевидно, что дальнейшее развитие гипертекста будет осуществляться за счет скрипт - программирования. Это оказалось немногим более эффективным, чем вводить в язык все новые элементы. Появившиеся в то время браузеры (Netscape Navigator 4, Microsoft Internet Explorer 4 и др.) уже достаточно надежно интерпретировали программный код (был установлен определенный уровень стандартизации). Однако проблемы у разработчиков еще остались. В качестве примера можно отметить, что многие скрипты начинаются с определения версии браузера, чтобы потом использовать тот или иной фрагмент кода. Очевидно, что на программиста ложится обязанность тестирования страниц на всех популярных в настоящие время браузерах.

В результате, использование всех возможностей Dynamic HTML  стало уделом программистов достаточно крупных организаций, где есть условия для разработки сложных программ и всестороннего их тестирования. Создателям личных WEB-страниц подчас приходится искать компромисс между надежностью и новаторством, чтобы получить достаточно грамотный HTML-код.

**2.3Структура проекта.**

<!doctype html>

<html lang="ru">

<head>

<title>Job</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="css/style.css">

<link href='https://fonts.googleapis.com/css?family=Raleway' rel='stylesheet' type='text/css'>

<link rel='stylesheet prefetch' href='http://maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css'>

<style>

.Otcenter{

text-align: center;

display: none;

}

</style>

</head>

<body>

</div>

<div class="container">

<div class="card"></div>

<div class="card">

<h1 class="title">Добавление данных</h1>

<form action="" method="post">

<div class="input-container">

<input type="text" name="ID" required>

<label for="Username">Введите ID спортсмена</label>

<div class="bar"></div>

</div>

<div class="input-container">

<input type="text" name="Name" required>

<label for="Username">Введите имя спортсмена</label>

<div class="bar"></div>

</div>

<div class="input-container">

<input type="text" name="Fname" required>

<label for="Username">Введите фамилию спортсмена</label>

<div class="bar"></div>

</div>

<div class="input-container">

<input type="text" name="Result" required>

<label for="Username">Введите результат спортсмена</label>

<div class="bar"></div>

</div>

<div class="button-container">

<button><span>Добавить</span></button>

</div>

</form>

</div>

Если рассмотреть исходныq текст то можно легко увидеть схожесть их структуры. Это объясняется тем, что документы создаются по определенным правилам. В основу синтаксиса языка  HTML лег стандарт ISO 8879:1986 «Information processing. Text and office systems. Standard Generalized Markup Language (SGML)». Правда, существует большое различие между стандартом официальном и стандартом фактическим. HTML постоянно  развивается, дополняется новыми элементами.

Рассмотрим основные элементы HTML.

**<html></html>**

Означает документ на языке HTML. Я уже упоминал о том, что одним из принципов языка является многоуровневое вложение элементов. Данный элемент является самым внешним, так как между его начальным и коечным тегами должна находится вся WEB – страница. В принципе этот элемент. Он имеет атрибуты version, long и dir, которыми в данном случае редко кто пользуется, и допускает вложение элементов HEAD, BODY и других, определяющую общую структуру WEB-страницы. Конечным тегом </html> оканчиваются все документы в формате HTML.

**<head></head>**

Область заголовка Web – страницы, её первая часть. HEAD  служит только для формирования общей структуры документа. Этот элемент может иметь атрибуты IANG, DIR, должен включать элемент TITLE и допускает вложение элементов BASE, META, LINK, OBJECT, SCARIPT, STULE.

**<title></title>**

Элемент для размещения заголовка Web – страницы. Строка текста , расположенного внутри этого элемента, отображается не в документе, а в заголовке окна броузера. Эта строка часто используется при организации поиска в WWW. Эта строка должна достаточно точно отражать назначение документа.

**<style></style>**

Элемент для описания стиля некоторых элементов Web – страницы. В файле Strukt. Htm назначены шрифты для элементов h2, code. Для каждого элемента существует стилевое оформление по умолчанию, поэтому употребление элемента  style не обязательно, но желательно. Этот элемент введён сравнительно недавно, без него немыслима работа в приложениях World, Excel.

<**META>**

Элемент содержит служебную информацию, которая не отражается при просмотре Web- страницы. Внутри него нет текста в обычном понимании, поэтому нет и конечного тега. Каждый элемент Meta содержит два основных атрибута, первый из которых определяет тип данных, а второй – содержание. Вот несколько примеров meta – данных:

Дата, обозначающая срок годности документа:

Name – “Expires”  content = “Дата”

Адрес электронной почты:

Name = “Reply – to” content = “ Имя @ Адрес “

Имя автора Web-страницы:

Name = “Author” content = “ Имя автора”

Набор ключевых слов для поиска:

Name = “Keywords” content = “слово 1, слово 2, слово 3…”

Краткое описание содержания Web- страницы:

Name = “Description” content = “Содержание страницы”

Описание типа и характеристик Web – страницы:

Name = “Content – Type” content = “Описание страницы”

Указание приложения, в котором была создана Web – страница:

Name = “Generator” content = “Название HTML – редактора”

Атрибут Name используется приложением – клиентом для получения дополнительной информации о Web – страницах и их упорядочения. Его часто заменяют атрибутом http – equiv. Он используется сервером для создания дополнительных полей при выполнении запроса.

Элемент META может содержать URL. Шаблон атрибута таков:

URL = “http: // адрес”

**<body></body>**

Этот элемент заключает в себе гипертекст, который определяет собственно Web – страницу. Конечный тег этого элемента надо искать в конце HTML-файла. Внутри элемента BODY можно использовать все элементы, предназначенные для дизайна Web-страницы. Внутри начального тега элемента BODY можно расположить ряд атрибутов, обеспечивающих установки для всей страницы целиком.

Атрибут, определяющий фон страницы, является одним из самых важных для дизайна.

background = “Путь к файлу”

Более простое оформление фона сводится к заданию его цвета:

bgcolor = “#RRGGBB”

Цвета фона задается тремя двухразрядными шестнадцатеричными числами, которые определяют интенсивность красного, зеленого и синего. Оба приведенных выше атрибута не являются альтернативными и часто используются совместно: если по каким-либо причинам не может быть найден рисунок фона, используется цвет.

Поскольку фон станицы может изменятся, необходимо иметь возможность подбирать соответственный цвет текста. Для этого имеются следующей атрибут:

text = “#RRGGBB”

Для задания цвета текста гиперссылок используются следующие атрибуты:

link = “#RRGGBB”

Точно так же можно задать цвет для просмотренных гиперссылок:

vlink = “#RRGGBB”

Можно также указать изменение цвета для последней выбранный пользователем гиперссылки:

Alink = “#RRGGBB”

# 2.4 Правила синтаксиса

При использовании каждого элемента важно знать, какие элементы могут располагаться внутри него*,*и внутри какихэлементов может находиться он сам. Так, взаимное расположение элементов HTML, HEAD, TITLE, BODY должно быть стандартным на любой странице, в тех случаях, когда не используются фреймы. Если страница представляет собой документ планировки фреймов, то вместо элемента BODY используется элемент FRAMESET.

Существуют группы элементов, которые используются совместно. К ним относятся элементы для создания таблиц, списков, фреймов. В этом случае порядок вложения элементов определяется логикой создания того или иного объекта на странице. Таблицы и фреймы часто используются для того, чтобы разместить детали страницы (рисунки, текст и др.) в определённом порядке. Например, располагая рисунок внутри ячейки таблицы, можно добиться определённого его положения.

Большое количество элементов, которые используются для форматирования текста, допускают самые разнообразные варианты вложения. И сами они обязательно должны располагаться внутри определённых элементов.

**2.5 Что такое PHP?**

PHP – это скрипт-язык (scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере. Проще всего это показать на примере:

<?php

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', '');

define('DB\_NAME', 'n3');

if (!mysql\_connect(DB\_HOST, DB\_USER, DB\_PASS)) {

exit('Cannot connect to server');

}

if (!mysql\_select\_db(DB\_NAME)) {

exit('Cannot select database');

}

mysql\_query('SET NAMES utf8');

function search ($query)

{

$query = trim($query);

$query = mysql\_real\_escape\_string($query);

$query = htmlspecialchars($query);

if (!empty($query))

{

if (strlen($query) < 1) {

$text = '<p>Слишком короткий поисковый запрос.</p>';

} else if (strlen($query) > 50) {

$text = '<p>Слишком длинный поисковый запрос.</p>';

} else {

$q = "SELECT \* FROM info WHERE ID LIKE '%$query%'

OR Name LIKE '%$query%' OR Fname LIKE '%$query%' OR Result LIKE '%$query%'";

$result = mysql\_query($q);

Основное отличие от CGI-скриптов, написанных на других языках, типа Perl или C – это то, что в CGI-программах вы сами пишете выводимый HTML-код, а, используя PHP – вы встраиваете свою программу в готовую HTML-страницу, используя открывающий и закрывающий теги (в примере <?php и ?>).

Отличие PHP от JavaScript, состоит в том, что PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется.

Любители Internet Information Server найдут, что PHP очень похож на Active Server Pages (ASP), а энтузиасты Java скажут, что PHP похож на Java Server Pages (JSP). Все три языка позволяют размещать код, выполняемый на Web-сервере, внутри HTML страниц

**2.6 Возможности PHP**

В нескольких словах – на PHP можно сделать все, что можно сделать с помощью CGI–программ. Например, обрабатывать данные из форм, генерировать динамические страницы, получать и посылать куки (cookies).

Кроме этого в PHP включена поддержка многих баз данных (databases), что делает написание Web-приложений с использованием БД до невозможности простым.

Вот неполный перечень поддерживаемых БД:

|  |  |  |
| --- | --- | --- |
| Adabas D | InterBase | Solid |
| dBase | mSQL | Sybase |
| Empress | MySQL | Velocis |
| FilePro | Oracle | Unix dbm |
| Informix | PostgreSQL |  |

Вдобавок ко всему PHP понимает протоколы IMAP, SNMP, NNTP, POP3 и даже HTTP, а также имеет возможность работать с сокетами (sockets) и общаться по другим протоколам.

**2.7 Краткая история PHP**

Началом PHP можно считать осень 1994 года, когда Расмус Лердорф (Rasmus Lerdorf) решил расширить возможности своей Home-page (Домашней страницы) и написать небольшой движок для выполнения простейших задач. Такой движок был готов к началу 1995 года и назывался Personal Home Page Tools. Умел он не очень много – понимал простейший язык и всего несколько макросов.

К середине 1995 года появилась вторая версия, которая называлась PHP/FI Version 2. Приставка FI – присоединилась из другого пакета Расмуса, который умел обрабатывать формы (Form Interpritator). PHP/FI компилировался внутрь Apache и использовал стандартный API Apache. PHP скрипты оказались быстрее аналогичных CGI – скриптов, так как серверу не было необходимости порождать новый процесс. Язык PHP по возможностям приблизился к Perl, самому популярному языку для написания CGI-программ. Была добавлена поддержка множества известных баз данных (например, MySQL и Oracle). Интерфейс к GD – библиотеке, позволял генерировать картинки на лету. С этого момента началось широкое распространение PHP/FI.

В конце 1997 Зээв Сураски (Zeev Suraski) и Энди Гутманс (Andi Gutmans) решили переписать внутренний движок, с целью исправить ошибки интерпретатора и повысить скорость выполнения скриптов. Через полгода, 6 июня 1998 года вышла новая версия, которая была названа PHP 3. К лету 1999 года PHP 3 был включен в несколько коммерческих продуктов. По данным NetCraft на ноябрь 1999 PHP использовался в более чем 1 млн. доменах.

**2.7 Почему нужно выбирать PHP**

Разработчикам Web-приложений нет необходимости говорить, что web-страницы - это не только текст и картинки. Достойный внимания сайт должен поддерживать некоторый уровень интерактивности с пользователем : поиск информации, продажа продуктов, конференции и т.п. Традиционно все это реализовалось CGI-скриптами, написанными на Perl. Но CGI- скрипты очень плохо масштабируемы. Каждый новый вызов CGI, требует от ядра порождения нового процесса, а это занимает процессорное время и тратит оперативную память. PHP предлагает другой вариант – он работает как часть Web-сервера, и этим самым похож на ASP от Microsoft.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти.

Программы, написанные на PHP, достаточно легкочитаемы. Написанный PHP – код легко зрительно прочитать и понять, в отличие от Perl-программ.

Недостатки PHP

· PHP является интерпретируемым языком, и, вследствие этого, не может сравниться по скорости с компилируемым С. Однако при написании небольших программ, что, в общем-то, присуще проектам на PHP, когда весь проект состоит из многих небольших страниц с кодом, вступают в силу накладные расходы на загрузку в память и вызов CGI-программы, написанной на С.

· Не такая большая база готовых модулей, как, например, СPAN у Perl. С этим ничего нельзя поделать – это дело времени. В PHP 4 разработчики предусмотрели специальный репозиторий PEAR, аналогичный CPAN, и я думаю, очень скоро будет написано достаточное количество модулей для его наполнения.

**2.8Кратко про CSS.**

Под термином CSS мы понимаем фразу "Каскадные Таблицы Стилей". Они внедряются в HTML-код и не требуют никаких специальных редакторов и компиляторов, ведь интерпретируются оные обычным браузером.

Каскадные таблицы стилей представляют собой описания различных HTML-элементов и созданы они для расширения свойств последних. Впервые стили были предложены WWW Consorcium'ом в рамках разработки спецификации HTML 3.0, однако реально эту шнягу стали поддерживать только в 1997 году. Насладиться CSS имели возможность лишь счастливые обладатели таких браузеров, как Netscape Navigator 4.0 и Internet Explorer 4.0.

На этом с историческими фактами позвольте завязать и приступить непосредственно к делу. CSS открывает нам новые, ранее неизвестные и недоступные грани. С их помощью мы получаем возможность более изысканно оформить свой Web-сайт, придать ему новые черты и симпатичное лицо.

Синтаксис CSS довольно прост в изучении, поэтому освоение каскадных стилей является делом достаточно легким, но весьма и весьма полезным. Для того, чтобы не оказаться голословным, я хочу привести небольшой пример. Возьмем типичную таблицу HTML и поставим атрибут border="1". После интерпретации данного кода браузером мы получим самый обыкновенный результат, к которому привык наш опытный глаз. Другое дело, что таблицы подобного рода не совместимы ни с одним мало-мальски хорошим дизайном. Ну куда годится эта псевдо-трехмерная линия, представляющая собой границу? Вот тут и приходят на помощь CSS, по средствам которых можно без лишнего гемора оформить любую таблицу на свой вкус, пример:

body {

background-color: rgba(186, 186, 186, 0.69);

color: #666666;

font-family: 'RobotoDraft', 'Roboto', sans-serif;

font-size: 14px;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

**2.8 Основы**

Как же описывать свойства некоторых элементов страницы? На самом деле, в этом нет ничего сложного, и самый примитивный случай присвоения стилей выглядит так:

имя элемента {свойство: значение;}

Здесь «имя элемента» - это название какого-либо HTML-тэга, чьи свойства мы желаем задать. Для простоты и наглядности я возьму элемент BODY и попытаюсь с помощью стилей сделать фон страницы зеленым. Конечно, такой шаг в Web-дизайне просто не допустим, но мы с вами только учимся.

Нижеследующий код заключен в шапке документа, то есть между тэгами <HEAD> и </HEAD>. Я не стал приводить полный листинг Web-страницы, чтобы вам легче было разобраться:

<STYLE TYPE=”text/css”>

BODY {background-color: green;}

</STYLE>

Попробуйте загрузить данный код в браузере, и вы увидите, что фон действительно стал ядовито-зеленым. В качестве цвета не обязательно указывать полное название последнего. Как и в обычном HTML вы можете прибегнуть к таблице RGB.

Из примера видно, что, хотя мы описывали тэг <BODY>, в стилях он прописан без ограничителей < и >. И действительно, если вы поставите данные знаки, то браузер будет пребывать в замешательстве, воспринимая <BODY>, как элемент тела, а не описываемый тэг.

При работе со стилями необходимо помнить, что не все пользователи Сети имеют современные браузеры. Такие динозавры не смогут правильно интерпретировать CSS-код, и обязательно сделают какую-нибудь пакость, испортив тем самым ваше творение. Но не стоит отчаиваться и пускать на клавиатуру сопли, ведь от такого поведения есть лекарство, имя которому «комментарии». Перепишем нашу каскадную таблицу следующим образом:

<STYLE TYPE=”text/css”>

<!—

BODY {background-color: green;}

-->

</STYLE>

При таком раскладе, старые браузеры пропустят все содержимое стилей, а новые выполнят их, так как они понимают, что в данном случае комментарии несут иной смысл.