Содержание:

Введение

Программирование является деятельностью по созданию программ для вычислительных устройств путем специальных систем или языков программирования. Теоретические основы данной деятельности в полной степени соответствуют характерным чертам технических дисциплин. «Фундаментальные исследования в технических науках зачастую отождествляют с теоретическими исследованиями в технике, которые находятся между естественнонаучными, математическими теориями, с одной стороны, и инженерной практикой — с другой, и даже включают в себя элементы дедуктивно-аксиоматических теорий».

Появление научно-технических неклассических дисциплин отмечается как новая тенденция в развитии технических наук. Данные дисциплины появляются «в процессе обширного научного движения (прежде всего, системного), доработки и конкретизации общих методологических, к примеру, системных представлений и понятий, а также обобщения практики решения определенного класса научно-технических задач». Данные дисциплины не могут относиться ни к техническим, ни к естественным, ни к общественным наукам, они носят междисциплинарный, комплексный характер. Теоретические аспекты программирования вычислительных устройств можно причислить к таким дисциплинам.

Программирования языки представляют собой средство представления знания для компьютерных систем. Они предлагают концептуальные средства представления и возможности моделирования, которые приспособлены к решению конкретных задач. Многообразие конструкций таких языков, которое сложилось за достаточно краткий по историческим меркам период — около шестидесяти лет— является благотворной почвой для размышлений о вопросах методологического и логико-философского характера, которые связаны с формализованными языками. Дело в том, что одна и та же задача может решаться разными программными средствами. Основанием такого различия является не только модель вычислительного устройства, на котором будет выполняться алгоритм, но и язык, средствами которого данный алгоритм описан. «Эффективность машинных алгоритмов зависит во многом от используемых систем и языков программирования».

Сказанное выше позволяет сказать о важной роли, которую теория языков программирования играет в методологии компьютерных наук, что актуализирует в рамках философии техники задачу специализированного исследования, которое посвящено анализу становления данной теории.

В последнее время, говоря о программировании в Internet, зачастую имеют в виду создание публикаций с использованием языка разметки гипертекстовых документов HTML. Применение специализированных средств (HTML–редакторов) позволяет не только создавать отдельные динамически меняющиеся интерактивные HTML–документы, используя при этом данные мультимедиа, но и редактировать целые сайты.

Поэтому рассматриваемая тема актуальна.

Целью курсовой работы является изучение языка гипертекстовой разметки.

В соответствии с поставленной целью необходимо решить ряд задач, таких как:

- рассмотреть теоретические основы HTML;
- раскрыть особенности формирования синтаксиса, форматирования текста;
- охарактеризовать особенности формирования элементов HTML на странице.

Объектом исследования выступает язык гипертекстовой разметки. Предметом исследования – особенности его применения в программировании и системах обработки данных.

Теоретические вопросы гипертекстовой разметки

Основы HTML

HTML (HyperText Markup Language) - язык разметки гипертекста - предназначается для создания Web-страниц. Под гипертекстом в данной ситуации понимаютя текст, который связан с другими текстами указателями-ссылками.

HTML представляет собой простой набор кодов, которыми описывается структура документа. При помощи HTML можно выделить в тексте отдельные логические части (абзацы, заголовки, списки и т.д.), поместить на Web-страницу подготовленную картинку или фотографию, организовать ссылки на странице для

связи с другими документами.

HTML не задает точные и конкретные атрибуты форматирования документа. Конкретный тип документа определяет окончательно только программа-броузер на компьютере Интернет-пользователя.

HTML не является также языком программирования, но web-страницы могут включать в себя встроенные программы-скрипты на языках Visual Basic Script и Javascript и программы-апплеты на языке Java.

Даже, если в будущем не предполагается редакция «вручную» текста HTML (предполагается использование графических редакторов), знание языка HTML предоставляет возможность как лучше использовать данные средства, так и повысит шансы сделать HTML-документ более «читаемым» и доступным при просмотре браузерами различных фирм[1].

Главными компонентами HTML являются:

1. Тег (tag). Тег HTML представляет собой компонент, который командует Web-браузеру выполнить определенную задачу для вставки изображения или создания абзаца.
2. Аргумент (или атрибут). Атрибут HTML меняет тег. К примеру, можно выровнять изображение внутри тега или абзац.
3. Значение. Их присваивают аргументам и определяют вносимые изменения. К примеру, если для тега используют атрибут выравнивания, то можно указать значение данного аргумента. Значения могут являться текстовыми, типа right или left , а также числовыми, как, к примеру, высота и ширина изображения, где значения определяются размерами изображения в пикселях[2].

Теги представляют собой зарезервированные последовательности символов, которые начинаются с < (знака меньше) и заканчиваются > (знаком больше).

Открытие тега отличается от закрытия только наличием символа '/'.

Можно предположить, что есть гипотетический аргумент форматирования текста, который управляется кодом <X>, и необходимо применить его к словам «Это мой текст».

HTML-последовательность кодов и самого текста будет выглядеть следующим образом:

<X>Это мой текст</X>

Теги могут быть иерархически вложены друг в друга, но без пересечений, то есть является допустимым вложение вида <teg1><teg2></teg2> </teg1>, но не <teg1><teg2> </teg1></teg2>.

Действие вложенных тегов объединяется. К примеру, если внутрь тега, который создает жирное начертание шрифта, вложен тег курсива, то в итоге получится жирный курсив[3].

Особенности формирования синтаксиса, форматирования текста

1. Взаимное расположение элементов HTML, TITLE, HEAD, BODY должно быть стандартно на любой странице.

<HTML>

<HEAD>

<TITLE>.....</title>

</head>

<BODY>

.....

</body>

</html>

2. Следует всегда использовать конечные теги (не забывать </p>, </h1>, </table> и др.).

3. Не нарушать правила вложения тегов. Правильно это делать следующим образом: <H1>Заголовок крупный <H2> Заголовок меньше </h2> </h1>. Неверно: <H1>Заголовок крупный <H2> Заголовок поменьше </h1> </h2>

4. Любые полезные данные должны находиться между конечным и начальным тегами, которые указывают ее формат.

5. Все атрибуты располагают в начальном теге.

Кодирование цвета используют для раскрашивания горизонтальных линий, шрифтов, фона и других элементов. Цвета обозначают числовыми шестнадцатеричными кодами или английскими названиями.

Стандартные цвета приведены ниже.

Аквамарин	aqua	#00FFFF
Белый	white	#FFFFFF
Желтый	yellow	#FFFF00
Зеленый	green	#008000
Золотистый	gold	#FFD700
Индиго	indigo	#4B0080
Каштановый	maroon	#800000
Красный	red	#FF0000
Оливковый	oliv	#808000
Пурпурный	purple	#800080
Светло-зеленый	lime	#00FF00
Серебристый	silver	#C0C0C0

Серый	gray	#808080
Сизый	teal	#008080
Синий	blue	#0000FF
Ультрамарин	navy	#000080
Фиолетовый	violet	#EE80EE
Фуксиновый	fuchsia	#FF00FF
Черный	black	#000000

Градации красного представлены ниже.

Код	**Цвет**	**Код**	**Цвет**
#010000		#800000	
#100000		#900000	
#200000		#A00000	
#300000		#B00000	
#400000		#C00000	

#500000 #D00000

#600000 #E00000

#700000 #FF0000

Градации зеленого:

Код	Цвет	Код	Цвет
#000100		#008000	
#001000		#009000	
#002000		#00A000	
#003000		#00B000	
#004000		#00C000	
#005000		#00D000	
#006000		#00E000	
#007000		#00FF00	

Градации синего:

Код	Цвет	Код	Цвет
#000001		#000080	
#000010		#000090	
#000020		#0000A0	
#000030		#0000B0	
#000040		#0000C0	
#000050		#0000D0	
#000060		#0000E0	
#000070		#0000FF	

Градации оранжевого

Код	Цвет
#FFB000	1
#FFA800	2
#FFA000	3

#FF9800 4

#FF9000 5

#FF8800 6

#FF8000 7

#FF7800 8

#FF7000 9

#FF6800 10

#FF6000 11

#FF5800 12

Компьютерная радуга представлена ниже:

К

О

Ж

З

Г

С

Ф

Цвет шрифта может быть задан при помощи атрибута color в теге <FONT>, к примеру:

<FONT color=«FF5800»> Это цветной текст 1

<FONT color=«blue»> Это цветной текст 2

Чтобы задать цвет фона страницы используют атрибут color внутри тега <BODY>, к примеру:

<BODY color=« red»>

Существует три основные вида списков в HTML-документе:

- ◦ непронуменрованный;
  ◦ пронуменрованный;
  ◦ список описаний;

Можно создавать вложенные списки, используя при этом разные тэги списков или повторяя одни внутри других. Для этого нужно разместить одну пару тэгов (завершающий и стартовый) внутри другой. Будут ли элементы вложенного списка иметь те же маркеры, обозначающие элемент списка — зависит от браузера пользователя.

В пронумерованном списке браузер вставляет автоматически номера элементов по порядку. Это значит, что если удалить один или несколько элементов пронумерованного списка, то остальные номера будут пересчитаны автоматически.

Пронумерованный список начинается стартовым тэгом <OL> и заканчивается тэгом </OL>. Каждый элемент списка начинается с тэга <LI> и завершается тегом </LI>. К примеру:

<OL>

<LI>Программирование

<LI>Алгоритмизация

<LI>Проектирование

</OL>

Тэг <OL> может иметь следующие характеристики:

<OL TYPE=A|a|I|i|1 START=n> где: TYPE - вид счетчика:

- ◦ A — большие латинские буквы (A,B,C...)
  ◦ a — маленькие латинские буквы (a,b,c...)
  ◦ I — большие римские цифры (I,II,III...)
  ◦ i — маленькие римские цифры (i,ii,iii...)
  ◦ 1 — обычные цифры (1,2,3...)

START=n - число, с которого начинают отсчет. К примеру:

<OL TYPE=I START=15>

<LI> Программирование

<LI> Алгоритмизация

<LI> Проектирование

</OL>

Для непронумерованных списков браузер использует обычно маркеры для пометки элемента списка. Вид маркера, как правило, настраивает пользователь браузера.

Список начинается стартовым тэгом <UL> и заканчивается тэгом </UL>. Каждый элемент списка начинается с тэга <LI>. К примеру:

<UL>

<LI>Программирование

<LI>Алгоритмизация

<LI>Проектирование

</UL>

Тэг <UL> может иметь следующий параметр: TYPE=disc|circle|square>. Тип тэга <UL> определяет внешний вид маркера — по умолчанию (disc), круглый (circle) или квадратный (square). К примеру:

<UL TYPE=square>

<LI>Программирование

<LI>Алгоритмизация

<LI>Проектирование

</UL>

Приведем пример вложенных списков:

<HTML>

<HEAD>

<TITLE> Список работников </TITLE>

</HEAD>

<BODY>

<H2> Список работников организации </H2>

<H3> Составлено : 30 июля 2006 года </H3>

<p>Этот список содержит фамилии, отчества и имена всех работников фирмы. <P>

<p>Список может использоваться только в служебных целях. <P>

<OL>

<LI> Дирекция

<UL>

<LI> Иванов И.И.

<LI> Петров К.В.

</UL>

</LI>

<LI> Отдел маркетинга

<UL>

<LI> Варшавская Е.Л.

<LI> Самсонов Д.М.

</UL>

</LI>

</OL>

</BODY>

</HTML>

Тэг <UL> может иметь следующие параметры: TYPE=disc|circle|squade> или <OL TYPE=A|a|I|i|1 VALUE=n>, в зависимости от того, в списке какого вида находится этот элемент.

Следовательно, атрибут TYPE определяет тип маркера или счетчика, а VALUE=n — значение для элемента пронумерованного списка (его номер). Все дальнейшие номера элементов списка будут отсчитываться от данного номера, а каждый элемент списка задаётся при помощи тега <LI>. К примеру:

<OL TYPE=I START=15>

<LI> Программирование

<LI TYPE=i VALUE=25> Алгоритмизация

<LI> Проектирование

</OL>

15. Программирование

2. Алгоритмизация
3. Проектирование

Список определений начинают с тэга <DL> и завершают тэгом </DL>. Этот список служит для создание списков типа «термин»-»описание». Каждый термин начинается тэгом <DT> , а описание — тэгом <DD>. К примеру:

<DL>

<DT> <B> Отдел маркетинга </B>

<DD> Этот отдел занимается продвижением услуг и продуктов

<DT> <B> Финансовый отдел </B>

<DD> Этот отдел занимается всеми финансовыми операциями

<DT> <B> Отдел кадров </B>

<DD> Этот отдел занимается набором и учетом новых работников организации, распределением отпусков, отслеживанием больничных листов и т.д.

</DL>

Особенности формирования элементов HTML на странице

Рисунки на web-странице

<IMG> является элементом для создания ссылки на графический файл (image). Он не содержит конечный тег — вся нужная информация задается с помощью аргументов. Данный элемент является универсальным: с его помощью могут использоваться изображения в гиперссылках, вставляться картинки в таблицы, размещаться рисунки на Web-странице, решаться задачи дизайна и т.д.

Требуемым атрибутом является src — указатель на файл графики:

src=«Ссылка на файл». К примеру: <IMG src=«cat5-web.jpg> — обычный рисунок, <IMG src=«1c5.gif»> — анимированный рисунок.

Обязательным и очень полезным атрибутом является атрибут alt. Он позволяет выводить текст в тех местах, в которых должны быть расположены рисунки. Страница может загружаться долго, и пока графические файлы на подходе, пользователь должен видеть, какие изображения он может получить.

К примеру: <IMG src=«cat5-web.jpg alt=«Фотография маленького котенка» >. <IMG src=«1c5.gif» alt=«Большая черная кошка, гуляющая сама по себе»>

Ширину и высоту области, в которой представлен рисунок, задают с помощью атрибутов height — высота и width— ширина.

К примеру: <IMG src=«bos2.gif» width=«76» height=«121»>

<img src=«bos2.gif» width=«176»>. Обратите внимание, что во втором случае изменен размер рисунка (мы изменили ширину, высота будет автоматически изменена). При этом происходит потеря качества изображения, следовательно, желательно задавать данные аргументы в согласно реальным размерам рисунка.

Атрибут border задает размер рамки вокруг объекта, к примеру, border=«4» ширина рамки задается в пикселях и в рассматриваемом примере равняется 4.

<IMG src=«cat5-web.jpg» alt=«Фото котёнка в рамке» width=«176» style=«border: 4px solid #9A2EEB;»>. Размер рамки и её цвет зададим, к примеру, при помощи стилей.

Полностью тег IMG должен выглядеть так:

<IMG src=«bos2.gif» width=«76» height=«121» alt=«Портрет маленького джентльмена»>

Если нужно использовать рисунок в качестве обоев страницы, то в теге <BODY> используется аргумент background с указанием адреса рисунка обоев.

К примеру: <BODY background=«wood.jpg»>

Гиперссылки

<A> — один из наиболее важных элементов языка, который обеспечивает создание гиперссылок. Зачастую используют следующий шаблон:

Произвольный текст <A href=«адрес ссылки»>Текст для щелчка

К примеру, следующим образом выглядит гиперссылка в тексте: Если вы хотите вернуться к материалам урока «Гиперссылки», то вам сюда, при этом следует предварительно поставить имя закладки перед тем местом в документе, на которое происходит переход. Закладка в документе задаётся так: . В данном случае это имя zakl2.

Следующим образом задают шаблон тогда, когда видимая часть гиперссылки представляет собой рисунок:

<A href=«Адрес ссылки»> <IMG src=«Ссылка на рисунок»>

К примеру: Чтобы вернуться к уроку «Гиперссылки» нажмите на кнопку <A href=«index.html#zakl2»><IMG src=«knopka.gif» width=«30» height=«20»>

Внутри тега <BODY> используют аргумент link — задает цвет ссылок на всей Web-странице, vlink — задает цвет посещенных ссылок, alink — задает цвет активных ссылок (цвет появляется при нажатии мыши).

Например: <BODY link=«0000FF» vlink=«CC0066» alink=«FF0000»>

По умолчанию используют ссылку на файлы текущей папки (той, где располагается файл Web-страницы). В данном случае указывают имя файла, к примеру: page2.htm, photo35.gif и т.д.

Зачастую используют относительные ссылки на папки, что позволяет легко изменять местоположение комплекса страниц на диске. Если в текущей папке существует другая, в которой размещаются нужные файлы, ссылка строится по следующему шаблону:

href=«./Папка/файл.тип». Здесь на структуру вложенных папок указывает точка перед наклонной чертой. Если нужно указать папку, которая находится на том же уровне вложенности, что и текущая, добавляется еще одна точка: href=«../Папка/файл.тип».

Если ссылка указывает на какой-то Web-ресурс, то она выглядит так: href=«http://www.netscape.com».

Когда гиперссылку используют для указания адреса электронной почты, ее выбор обеспечивает не переход к новому документу, а запуск почтовой программы на компьютере для отправки сообщения указанному адресату. Обычно такая ссылка

размещается в конце страницы для обеспечения связи с автором страницы или Web-мастером, к примеру:

<A href=«mailto:goncharov@online.ru>Алексей Гончаров

Таблицы

Таблицы представляют собой очень удобное средство форматирования данных на Web-станице. Они позволяют решать дизайнерские задачи, а именно: выравнивать части страницы друг относительно друга, размещать текст и рисунки рядом, управлять цветовым оформлением и т.д.

При создании таблиц используют принцип вложения, а именно: внутри основного элемента таблицы <TABLE> создают ряд элементов, которые определяют строки <TR>, а внутри данных элементов размещают элменты для описания каждой ячейки в с строке <TD>.

<TABLE> </table> — внешний элемент таблицы.

<TR> </tr> — элемент, который задает строку таблицы. Конечный тег может не использоваться, поскольку строка завершается там, где начинается следующая строка.

<TD> </td> — элемент, который задает ячейку таблицы. Конечный тег можно также не использовать.

Ширина таблицы можно задаваться точно в пикселах или в процентном отношении к ширине страницы в окне браузера.

К примеру, если следует создать таблицу определенного размера, то указывают следующее:

<TABLE width=«500»>

<TR>

<TD> Ширина этой таблицы 500 пикселей и она состоит из одной строки и одного столбца.</td>

</tr>

</table>

Если нужно получить таблицу на всю ширину экрана, не заботясь при этом, какое разрешение монитора (800x600, 1024 x 768, 1280 x 1024) у того, кто будет просматривать нашу Web-страницу, то следует задать ширину страницы как 100%.

<TABLE width=«100%»>

<TR>

<TD> Ширина этой таблицы 100%.</td>

<TD> и она состоит из одной строки и двух столбцов </td>

</tr>

</table>

Для всей таблицы может задаваться цвет фона: bgcolor=«Цвет» или bgcolor=«#RRGGBB», к примеру:

<TABLE width=«100%» bgcolor=«#00CC99»>

<TR>

<TD> Ширина этой таблицы 50%.</td>

</tr>

<TR>

<TD> и она состоит из двух строк и одного столбца </td>

</tr>

</table>

Может быть задан цвет ячеек таблицы отдельно.

<table width=«600» border=«1» cellspacing=«0» cellpadding=«5» align=«center»>

<tr>

<td bgcolor=«#00FFFF»></td>

<td bgcolor=«#CCFF00»></td>

<td bgcolor=«#FF6633»></td>

</tr>

<tr>

<td bgcolor=«#0000FF»></td>

<td bgcolor=«#33FF66»></td>

<td bgcolor=«#FF00FF»></td>

</tr>

<tr>

<td bgcolor=«#CCCCCC»></td>

<td bgcolor=«#9933FF»></td>

<td bgcolor=«#FFFFCC»></td>

</tr>

</table>

Шириной боковой грани управляет аргумент border. Может быть задана ширина боковой грани таблицы в пикселах.

<TABLE width=«100%» bgcolor=«#00CC99» border=«3» >

<TR>

<TD> </td>

<TD> Ширина данной таблицы 300 пикселей</td>

<TD> </td>

</tr>

<TR>

<TD> и она состоит из двух строк и трех столбцов</td>

<TD> </td>

<TD></td>

</tr>

</table>

Можно выполнить грани таблицы невидимыми, для этого ширину бордюра таблицы следует задать равной 0:

<TABLE width=«100%» bgcolor=«#00CC99» border=«0» >

<TR>

<TD> </td>

<TD> Ширина данной таблицы 300 пикселов</td>

<TD> </td>

</tr>

<TR>

<TD> и она состоит из двух строк и трех столбцов</td>

<TD> </td>

<TD></td>

</tr>

</table>

Есть набор аргументов, которые нужны для выравнивания данных в ячейках таблиц. Аргумент align позволяет выравнивать сведения в ячейках по горизонтали. Он принимает следующие значения:

left — выравнивание влево;

right — выравнивание вправо;

center — центрирование.

Аргумент valign позволяет выравнивать текст по вертикали. Значения могут быть следующими:

top — выравнивание по верхнему краю ячейки

center — выравнивание по центру

baseline — выравнивание по первой строке.

<table width=«100%» border=«1» cellspacing=«0» cellpadding=«5» align=«center»>

<tr> <td width=«257»>Выравнивание по горизонтали</td>

<td width=«233» align=«center»> По центру </td>

<td width=«217» align=«left»>По левому краю </td>

<td width=«246» align=«right»> По правому краю </td>

</tr>

<tr>

<td width=«257» height=«112»>Выравнивание по вертикали</td>

<td width=«233» height=«112» valign=«top»>По верхнему краю</td>

<td width=«217» height=«112» valign=«middle»>По центру</td>

<td width=«246» height=«112» valign=«baseline»>По нижнему краю</td>

</tr>

</table>

Разметка web-страницы с использованием таблицы

Разметка Web-странцы должна производиться с использованием таблицы. Возможны разные варианты разметки. Рассмотрим некоторые из них.

1 вариант

Разметка страницы производится при помощи таблицы шириной на весь экран, вне зависимости от того, каким является разрешение экрана (width=100%). В рассматриваемой ситуации удобно создать таблицу, которая состоит из двух столбцов и двух строк. Верхняя строка будет отводиться под заголовок страницы, левый столбец будет отводиться под меню Web-сайта.

2 вариант

Разметка страницы производится при помощи использования таблицы шириной 760 пикселей, которая выровнена по центру экрана. В рассматриваемом случае удобно создать таблицу, которая состоит из трех строк и одного столбца. Верхняя строка будет отводиться под заголовок страницы, вторая строка будет отводиться под меню Web-сайта, а третья строка отводится под содержание сайта.

Если нужно разместить внутри текста страницы какие-то изображения, фотографии и пр., то в данном случае также можно использовать таблицы. В приведенном ниже примере во вторую ячейку второй строки вставлена таблица, которая состоит из трех столбцов и двух строк. В первую и в третью ячейки первой строки вставлены иллюстрации, а во вторую ячейку первой строки — название страницы. Во все ячейки второй строки введен текст страницы.

Фреймы

Используя фрэймы, которые позволяют разбивать Web-страницы на большие скроллируемые подокна, можно в значительной степени улучшить функциональность и внешний вид информационных систем и Web-приложений. Каждое подокно, или фрэйм, может иметь следующие характеристики:

1. Каждый фрэйм имеет свой URL, что позволяет загружать его вне зависимости от других фрэймов.
2. Каждый фрэйм имеет собственное имя (параметр name), которое позволяет переходить к нему из другого фрэйма.
3. Размер фрэйма может меняться пользователем прямо на экране с помощью мыши (если это не запрещено указанием специализированного показателя).

Данные параметры фрэймов позволяют создавать продвинутые интерфейсные решения, к примеру:

1. Размещение стратегических даных, которую автор считает нужным постоянно показывать пользователю, в одном статическом фрэйме. Этим может являться copyright, графический логотип фирмы, набор управляющих кнопок.
2. Помещение в статическом фрэйме оглавления всех или части Web-документов, которые содержатся на Web-сервере, что позволяет быстро находить пользователю интересующие его данные.
3. Создавать окна итогов запросов, когда в одном фрэйме находится сам запрос, а в другом итоги запроса.
4. Создавать формы типа «мастер-деталь» для Web-приложений, которые обслуживают базы данных.

Фреймы

Формат документа, который использует фрэймы, внешне напоминает формат обычного документа, только вместо тэга BODY используют контейнер FRAMESET, который содержит описание внутренних HTML-документов, т. е. саму информацию, которая размещается во фрэймах.

<HTML>

<HEAD>...</HEAD>

<FRAMESET>...</FRAMESET>

</HTML>

Тем не менее, фрэйм-документ представляет собой специфичный вид HTML-документа, потому что он не содержит элемент BODY и какую-то информационную нагрузку соответственно. Он описывает только фрэймы, которые будут содержать информационные данные (помимо случая двойного документа).

Представим общий синтаксис фрэймов:

<FRAMESET COLS=«value» | ROWS=«value»>

<FRAME SRC=«url1»>

<FRAME ...>

...

</FRAMESET>

Общий контэйнер FRAMESET описывает все фрэймы, на которые разделен экран. Можно разделить экран на несколько горизонтальных или несколько вертикальных фрэймов. Тег FRAME описывает каждый фрэйм в отдельности. Рассмотрим каждый компонент более детально[4].

Тэг <FRAMESET> имеет завершающий тэг </FRAMESET>. Все, что может находиться между этими тэгами, это тэг <FRAME>, вложенные тэги <FRAMESET> и </FRAMESET>, а также контейнер из тэгов <NOFRAME> и </NOFRAME>, который позволяет строить двойные документы для браузеров, которые поддерживают фрэймы и не поддерживают фрэймы.

Данный тэг имеет два взаимоисключающих параметра: COLS и ROWS.

ROWS=«список-определений-горизонтальных-подокон». Этот аргумент содержит описания некоторого числа подокон, которые разделены при помощи запятых. Каждое описание представляет собой числовое значение размера подокна в пикселах, процентах от всего размера окна или связанное масштабное значение. Число подокон определяется числом значений в списке. Общая сумма высот подокон должна представлять собой высоту всего окна (в любых измеряемых размерах). Отсутствие атрибута ROWS определяет один фрэйм, величиной во все окно браузера[5].

Синтаксис используемых видов описания величин подокон:

value — простое числовое значение, которое определяет фиксированную высоту подокна в пикселах. Это не лучший способ описания высоты подокна, потому что разные браузеры имеют разный размер рабочего поля, не говоря уже о разных экранных разрешениях у пользователя. Если все же используется этот способ описания размера, то рекомендуется сочетать его с каким-то другим, чтобы в итоге было получено 100%-ное заполнение окна браузера пользователя.

value% — значение величины подокна в процентах от 1 до 100. Если общая сумма процентов описываемых подокон превышает 100, то размеры всех фрэймов пропорционально снижаются до суммы 100%. Если, соответственно, сумма меньше 100, то размеры пропорционально повышаются.

value* —значение value в рассматриваемом описании является необязательным. Символ «*» указывает на то, что все оставшееся место будет принадлежать этому фрэйму. Если указаны два или более фрэйма с описанием «*» (к примеру «*,*»), то оставшееся пространство разделяется между этими фрэймами поровну. Если перед

звездочкой стоит цифра, то она указывает пропорцию для этого фрэйма (во сколько раз одно будет больше аналогично описанного чистой звездочкой). К примеру, описание «3*,*,*», говорит, что будут созданы три фрэйма с размерами 3/5 свободного пространства для первого фрэйма и по 1/5 для двух других.

COLS=«список-определений-горизонтальных-подокон». То же самое, что и ROWS, но разделяет окно не по горизонтали, а по вертикали.

Совместное использование этих параметров может привести к непредсказуемым итогам. К примеру, строка: <FRAMESET ROWS=«50%,50%» COLS «50%,50%»> может привести к ошибочной ситуации.

Рассмотрим примеры.

<FRAMESET COLS=«50,*,50»> — описывает три фрэйма, два по 50 точек слева и справа, и один внутри данных полосок.

<FRAMESET ROWS=«20%,3*,*»> — описывает три фрэйма, первый из которых занимает 20% площади сверху экрана, второй 3/4 оставшегося от первого фрэйма места (т.е. 60% всей площади окна), а последний 1/4 (т.е. 20% всей площади окна).

<FRAMESET ROWS=«*,60%,*»> — аналогично предыдущему примеру.

Тэги <FRAMESET> могут быть вложенными, т.е., к примеру:

<FRAMESET ROWS=«50%,50%»>

<FRAMESET COLS=«*,*»

</FRAMESET>

</FRAMESET>

FRAME

<FRAME SRC=«url» [NAME=«frame_name»] [MARGINWIDTH=«nw»] [MARGINHEIGHT=«nh»]

[SCROLLING=yes|no|auto] [NORESIZE]>. Этот тэг определяет фрэйм внутри контейнера FRAMESET.

SRC=«url» описывает URL документа, который будет отображаться внутри рассматриваемого фрэйма. Если он отсутствует, то будет отображаться пустой

фрэйм.

NAME=«frame_name». Этот аргумент описывает имя фрэйма. Имя фрэйма может использоваться для определения действия с этим фрэймом из другого HTML-документа или фрэйма (как правило, из соседнего фрэйма этого же документа). Имя обязательно должно начинаться с символа. Содержимое поименованных фрэймов может быть задействовано из других документов с помощью специального атрибута TARGET, который описан ниже.

MARGINWIDTH=«value». Данный аргумент может использоваться тогда, когда автор документа хочет указать сбоку величину разделительных полос между фрэймами. Значение value указывается в пикселах и не может быть менее единицы. По умолчанию это значение зависит от реализации поддержки фрэймов используемым клиентом браузером.

MARGINHEIGHT=«value». То же самое, что и MARGINWIDTH, но для нижних и верхних величин разделительных полос.

SCROLLING=«yes | no | auto». Данный аргумент позволяет задавать наличие полос прокрутки у фрэйма. Параметр yes указывает, что полосы прокрутки будут присутствовать у фрэйма в любом случае, параметр no напротив, что полос прокрутки не будет. Auto определяет наличие полос прокрутки только при их необходимости (значение по умолчанию).

NORESIZE. Этот аргумент позволяет создавать фрэймы без возможности изменения размеров. По умолчанию размер фрэйма может быть изменен с помощью мыши также просто, как и размер окна Windows. NORESIZE отменяет эту возможность. Если у одного фрэйма установлен атрибут NORESIZE, то у соседних фрэймов тоже не может изменяться размер со стороны данного.

NOFRAMES этот аргумент используют тогда, когда нужно создать документ, который может быть просмотрен как браузерами, которые поддерживают фрэймы, так и браузерами, которые их не поддерживают. Этот тэг помещают внутри контейнера FRAMESET, а все, что находится внутри тэгов <NOFRAMES> и </NOFRAMES> игнорируется браузерами, которые поддерживают фрэймы.

Рассмотрим реализацию фрэймов для такого разбиения окна:

+--------------------------+

| | |

| | |

| Link1 | Link2 |

| | |

| | |

+-------------------------+

| | | |

| | | |

| Link3 | Link4 | Link5 |

| | | |

| | | |

+-------------------------+

<FRAMESET ROWS=«*,*» <NOFRAMES> <H1>Ваша версия WEB-браузера не поддерживает фрэймы!</H1> </NOFRAMES>

<FRAMESET COLS=«65%,35%»>

<FRAME SRC=«link1.html»>

<FRAME SRC=«link2.html»>

</FRAMESET>

<FRAMESET COLS=«*,40%,*»>

<FRAME SRC=«link3.html»>

<FRAME SRC=«link4.html»>

<FRAME SRC=«link5.html»>

</FRAMESET>

</FRAMESET>

С появлением фрэймов сразу появляется вопрос: «Как сделать так, чтобы нажимая на ссылку в одном фрэйме, инициировать появление информации в другом?»

Ответом на этот вопрос является планирование взаимодействия фрэймов (далее — планирование). Каждый фрэйм может иметь собственное имя, которое определяет параметр NAME при описании этого фрэйма. Также есть специальный аргумент — TARGET, который позволяет определять, к какому фрэйму относится операция. Формат данного атрибута следующий:

TARGET=«windows_name»

Этот аргумент может встречаться внутри разных тэгов: TARGET в тэге <A> — это самое прямое использование TARGET. Обычно при активизации пользователем ссылки соответствующий документ появляется в том же окне (или фрэйме), что и исходный, в котором была сделана на него ссылка. Добавление атрибута TARGET позволяет произвести вывод документа в другой фрэйм. К примеру: <A HREF=«mydoc.html» TARGET=«Frame1»> Переход в фрэйм № 1 </A>

Размещение TARGET в тэге BASE позволит не указывать при описании каждой ссылки фрэйм — приемник документов, которые вызываются по ссылкам. Это удобно, если в одном фрэйме находится меню, а в другой — выводится информация. К примеру:

Документ № 1.

<FRAMESET ROWS=«20,*»>

<FRAME SRC=«doc2.htm» NAME=«Frame1»>

<FRAME SRC=«doc3.htm» NAME=«Frame2»>

</FRAMESET>

Документ № 2 (doc2.htm).

<HTML>

<HEAD>

<BASE TARGET=«Frame2»>

</HEAD>

<BODY>

<A HREF=«url1»> Первая часть</A>

<A HREF=«url2»> Вторая часть</A>

</BODY>

</HTML>

Также можно включать параметр TARGET в описание ссылки при создании карты изображения в теге <AREA>. К примеру:

<AREA SHAPE=«circle» COORDS=«100,100,50» HREF=«http://www.softexpress.com» TARGET=«Frame1»>

Это же относится и к определению формы. В этом случае, после обработки переданных параметров формы результирующий документ появится в указанном фрэйме.

<FORM ACTION=«url» TARGET=«window_name»>. Внимание! Имя окна (фрэйма) в параметре TARGET должно начинаться с латинской буквы или цифры. Также следует помнить, что есть зарезервированные имена для разрешения специальных ситуаций.

Зарезервированные имена фрэймов нужны для разрешения специализированых ситуаций. Все они начинаются со знака подчёркивания. Любые другие имена фрэймов, которые начинаются с подчеркивания, будут браузером игнорироваться.

TARGET=«_blank» — это значение определяет, что документ, который был получен по ссылке, будет отображен в новом окне браузера.

TARGET=«_self» — это значение определяет, что документ, который был получен по ссылке, будет отображен в том же фрэйме, в котором находится ссылка. Это имя удобно для переопределения окна назначения, которое было указано ранее в тэге BASE.

TARGET=«_parent» — это значение определяет, что документ, который был получен по ссылке, будет отображен в родительском окне, независимо от параметров FRAMESET. Если родительского окна нет, то это имя аналогично «_self».

TARGET=«_top» — это значение определяет, что документ, который был получен по ссылке, будет отображен на всей поверхности окна, независимо от наличия фрэймов. Использование этого параметра удобно в случае вложенных фрэймов.

Заключение

Таким образом, можно сделать следующие выводы.

Гипертекстовый документ — текстовый файл, который имеет специализированные метки, которые называются тегами, впоследствии они опознаются браузером и используются им для отображения содержимого файла па экране компьютера.

При помощи этих меток можно выделять заголовки документа, менять цвет, начертание и размер букв, вставлять графические таблицы и изображения. Но главным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок — специализированных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

Список использованных источников

1. Абхалимова, Р. С. Информационные технологии XXI века // Экономика и социум. - 2014 г. - № 2-5 (11). - С. 234-236.
2. Азаров, Б.Ф. Теория систем управления: Учебное пособие / Б.Ф. Азаров, И.В. Карелина и др. - СПб.: Лань, 2013. - 424 с.
3. Астахова И.Ф., Толстобров А.П., Мельников В.М. SQL в примерах и задачах. – Мн.: Новое знание, 2011. – С.4.
4. Боуман Дж.С., Эмерсон С.Л., Дарновски М. Практическое руководство по SQL. – Вильямс, 2011. – С.56-90.
5. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз информационных данных. Полный курс. - Вильямс, 2010. – С.125.
6. Дейт К. SQL и реляционная теория. Как грамотно писать код на SQL. – Символ-Плюс, 2010. – С.123.
7. Дейт К., Дарвен Х. Основы будущих систем баз информационных данных. Третий манифест. – Янус-К, 2011. – С.102.

8. Дунаев В.В. Базы информационных данных. Язык SQL. – СПб. : БХВ-Петербург, 2010. – С.88.
9. Дж. Кастаньетто, Х.Рават, С.Шуман, К.Сколло, Д.Велиаф «Профессиональное PHP программирование». – Пер. с англ. – СПб: Символ-Плюс, 2010. – С.76.
10. Жилин Д.М. Теория систем: опыт построения курса. – КомКнига, 2011. – С.123.
11. Иванова Г.С. – «Основы программирования» Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2010. – С.156.
12. Коггзолл Джон. PHP 5. Полное руководство – М.: Вильямс, 2012 – С.336.
13. Кренке Д. Теория и практика построения баз информационных данных. – Питер, 2010. – С.206.
14. Мирошниченко Г. Реляционные базы информационных данных. Практические приемы оптимальных решений. – СПб. : БХВ-Петербург, 2011. – С.199.
15. Новиков Б., Домбровская Г. Настройка приложений баз информационных данных. – BHV, 2011. – С.22.
16. Советов Б.Я., Цехановский В.В., Чертовской В.Д. Базы информационных данных. Теория и практика.– Высшая школа, 2010. – С.49.
17. Скотт В. Эмблер, Прамодкумар Дж. Садаладж Рефакторинг баз информационных данных. Эволюционное проектирование. – Вильямс, 2010. – С.36.
18. Тоу Д. Настройка SQL. Для профессионалов. – Питер, 2011. – С.103.
19. Фейт С. TCP/IP. Архитектура, протоколы и реализация (включая IP версии 6 и IP Security) – Питер, 2011. С.196.
20. MySQL. Библиотека профессионала – Киев: Диалектика, 2012 – С.170-179.
21. PHP/MySQL для начинающих – Кудиц-образ, 2010 – С.44-108.
22. Теория и практика построения баз информационных данных: Д. Крёнке. – Питер, 2011. – С.223-250.
23. Microsoft Access 2007. Шаг за шагом: Практическое пособие / Пер. с англ. – М.: ЭКОМ, 2011. – С.63.
24. Багриновский К.А. Хрусталев Е.Ю. Новые информационные технологии. – М.: ЭКО, 2011. – С.122.
25. Информатика и информационно-коммуникационные технологии. Базовый курс: И.Г. Семакин, С.В. Русаков, Л.В. Шестакова. - М: БИНОМ, Лаборатория знаний, 2010. – С. 169.

1. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз информационных данных. Полный курс. - Вильямс, 2010. – С.125. ↑

2. Астахова И.Ф., Толстобров А.П., Мельников В.М. SQL в примерах и задачах. – Мн.: Новое знание, 2011. – С.4. ↑

3. Абхалимова, Р. С. Информационные технологии XXI века // Экономика и социум. - 2014 г. - № 2-5 (11). - С. 234-236. ↑

4. Боуман Дж.С., Эмерсон С.Л., Дарновски М. Практическое руководство по SQL. – Вильямс, 2011. – С.56-90. ↑

5. Азаров, Б.Ф. Теория систем управления: Учебное пособие / Б.Ф. Азаров, И.В. Карелина и др. - СПб.: Лань, 2013. - 424 с. ↑