11. Методические указания к изучению дисциплины

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ**

Л.Е.Рудельсон, М.М.Тверитнев

ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

ПОСОБИЕ
к изучению дисциплины

*для студентов III курса
специальности 230101
дневного обучения*

Москва – 2004

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей

Л.Е.Рудельсон, М.М.Тверитнев

ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

ПОСОБИЕ
к изучению дисциплины

для студентов III курса

специальности 230101
дневного обучения

Москва – 2004

ББК 6Ф.

Р

Рецензент к.т.н., доцент Л.А.Вайнейкис
Рудельсон Л.Е., Тверитнев М.М.

Р Пакеты прикладных программ: Пособие к изучению дисциплины. – М.: МГТУ ГА, 2004.-54 стр.

Пособие издается в соответствии с учебным планом для студентов специальности 220100 дневного обучения.

Рассмотрено и одобрено на заседаниях кафедры ВМКСС 09.04.04 и методического совета 13.05.04.

Редактор

ЛР №020580 от 05.09.01 г.		Подписано в печать
Печать офсетная	Формат 60x84/16	уч.-изд. л.
усл.печ.л.	Заказ №	Тираж экз.

Московский Государственный Технический Университет ГА
125993 Москва, Кронштадтский бульвар, д.20

Редакционно–издательский отдел
125493 Москва, ул. Пулковская, д.6а

1. ВВЕДЕНИЕ

Данная книга входит в комплект, подготовленный кафедрой ВМКСС для изучения дисциплины «Пакеты прикладных программ». Цель объединения – осветить темы, связанные с семейством популярных приложений – PowerPoint, Outlook, CorelDraw, Team Manager и т.д. Книги затрагивают вопросы работы с приложениями, а также теоретические аспекты. В предлагаемой брошюре речь идет о системе FrontPage и основах языка HTML разметки гипертекстов. Язык используется для публикаций и взаимодействия в сети Интернет (World Wide Web). Доступность данных (ресурсов) в этой сети обеспечивают три механизма.

- Единая *схема* наименования пути поиска ресурсов в Web, например *URI*.
- *Протоколы* доступа к именованным ресурсам через Web (*http*, *ftp* и т.д.).
- Гипертекст для простого перемещения по ресурсам, например HTML – язык, понятный всем компьютерам планеты, предоставляющий средства для:
  - публикации электронных документов с заголовками, текстом, таблицами, списками, фотографиями и т.д.;
  - загрузки удаленных ресурсов щелчком мыши по гипертекстовой ссылке;
  - разработки форм для взаимодействия с удаленными абонентами, для поиска информации, резервирования услуг, заказов товаров и т.д.;
  - включения электронных таблиц, видеоклипов, звуковых фрагментов и других приложений непосредственно в документы.

Основным качеством гипертекста, выделяющим его как особое средство познания, является возможность, не заказывая дополнительные материалы, получить всю необходимую для понимания изучаемого документа информацию. Для этого предусмотрен специальный механизм ссылок. В научной литературе всегда практиковался аналогичный прием. Развивая положения своей работы, авторы вводили в текст вместо цитат указания на труды предшественников, в которых были обоснованы известные утверждения, позволяющие получить новые оригинальные результаты. Электронные гипертексты автоматизировали эту читательскую технологию. Любой непонятный термин разъясняется щелчком мыши по ссылке в справочных системах и в документах HTML.

С этого и начался язык разметки. Первые версии ориентировались на дистанционное обучение и активный диалог коллег в ходе совместной работы, поддержанный средствами редактирования, представления графиков и таблиц, доступа к библиотекам. На этом этапе было достигнуто главное: узаконены соглашения, обеспечившие совместимость документов на различных браузерах (программах преобразования потока байтов в изображение и звук) и системных платформах. HTML понятен на любом компьютере, подключенном к сети.

Вовлечение в Web коммерческих пользователей, сделавшее ее действительно «всемирной паутиной», выдвинуло проблемы привлекательного пред-

ставления документов. Их решение породило ряд новых конструкций, элементов и атрибутов языка. В документы пришли звук, живопись, движение и т.д. Платой за новые возможности стала громоздкость, инженерные надстройки на не стыкующихся между собой усовершенствованиях. Следующий виток развития – отказ от обилия элементов и свойств, переход к разделению структуры документа и его оформления. Сейчас HTML находится именно на этом этапе своей истории. Всемирная сеть по-прежнему готова воспринимать каждую конструкцию, элемент, символ языка в комплекте с полным описанием своего представления с помощью атрибутов. В то же время такая практика объявлена нежелательной. Рекомендуется все, что касается стиля воспроизведения – особенностей шрифта, цвета, линий и т.д. – выносить в специальные наборы описаний (таблицы стилей), чтобы не отягощать конструкции документа указаниями относительно их вида или звучания.

Ряд определений и примеров в брошюре заимствованы из документа «Спецификация HTML 4.0». При работе над рукописью использованы материалы [1-7] обширной литературы по теме. Раздел 3 и §§ 4.41 – 4.44 подготовлены М.М.Тверитневым, остальное – Л.Е.Рудельсоном.

2. ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Спецификация HTML не всегда интерпретирует термины в их общеупотребительном значении. Вот несколько определений, вызывающих колебания на экзаменах. Начнем с известных из курса информатики.

Автор – согласно спецификации это не обязательно творец документа, а «человек или программа, пишущая или генерирующая документы в формате HTML». Существует следующее уточнение: «*средство разработки* – это отдельный случай автора, а именно программа, генерирующая код HTML».

Понятие «автор» определяет в HTML не столько класс объектов, как у юристов или даже в обыденной речи, сколько свойство произведений, значение которого – имя. Воспринимать существо этого термина предпочтительно именно в этом русле. Тогда не трудно будет, например, согласиться с тем, что лектор на кафедре не является автором студенческих конспектов, даже того, который вела стенографистка, ни разу не отступившая от текста.

Пользователь – это «человек, взаимодействующий с агентом пользователя для просмотра, прослушивания или другого применения документа HTML».

Агент пользователя – это «любое устройство, интерпретирующее документы HTML». Это визуальные браузеры, синтезаторы речи, устройства воспроизведения азбуки Брайля, поисковые машины и т.д. Вернемся к приведенному примеру и разберемся: кто или что у нас на лекции выступает в роли пользователя и агента пользователя.

Экземпляром документа, создаваемого на лекции, может служить любой конспект, конкретным автором которого (в терминах HTML) становится интеллект слушателя. Интеллект воспринимает и перерабатывает поступающую информацию, формирует ее в виде текста с рисунками. Агентом поль-

зователя является усвоенный слушателем навык фиксации (пером по бумаге) создаваемого его интеллектом документа. Пользователя на лекции нет. Он появляется в период сессии. Другой пример агента пользователя – техническое средство записи изображения и звука, в частности, видеокамера.

Конструкции языка. Понятие объединяет такие различные по назначению составные части языка, как элементы, атрибуты, ссылки на символы, комментарии, причем не только HTML, а любого наследника обобщенного стандартизованного языка SGML (*Standard Generalized Markup Language*), откуда они заимствованы. Назначению конструкции соответствует ее важность для создания и для представления документа. Элементы – это детали структуры, из которых набирается целое, создается документ. *Типы элементов* – это целостные структуры языка или указания на желательное поведение. Стандартные детали для создания документов. Формально объявление типа элемента включает три составные части: начальный тег, воспроизводимое содержание и конечный тег. Тег – это лишь часть элемента, заключенное в угловые скобки имя типа элемента. В скобках конечного тега перед именем элемента лидирует символ косой черты (слеш "/"). Поле начального тега используется, кроме объявления имени, для переопределения свойств конкретного экземпляра элемента. Теги позволяют программам обработки выделить в потоке байтов описание элемента и корректно воспроизвести его содержимое.

В HTML высок уровень избыточности, что позволяет в ряде случаев опускать части объявления элементов. Если фрагмент текста состоит из нескольких абзацев, то начало каждого из них однозначно указывает на окончание предыдущего, и наличие конечного тега становится необязательным. В таких случаях разрешается опускать в разметке конечные теги. Логическое развитие тенденции – разрешение пропускать и начальные, и конечные теги тех элементов, которые не могут не присутствовать в документе. Это элементы, открывающие и закрывающие страницу – <HTML>, или ее самостоятельные разделы: заголовок <HEAD> и тело <BODY>. По существу, их теги используются лишь для наглядности чтения человеком ранее размеченного документа, а программному обеспечению не нужны.

Ряд элементов HTML по своему замыслу не имеют содержимого и либо служат контейнерами (для упаковки или для изоляции от некорректных вмешательств) фрагментов документа, либо указывают браузеру желательное поведение в текущей точке разметки. Не имеет содержимого элемент BR перехода на следующую строку. Его роль – прерывание абзаца. Элемент FRAME тоже не создает никаких структур, это вложенная в экран рамка, как правило – не одна, для демонстрации автономных документов HTML. В таких элементах применение конечных тегов запрещено по определению: они пустые. Начало и конец едины. Все необходимое упаковано в определения атрибутов в начальном теге.

Атрибуты – это свойства элементов. Цвета, адреса, размеры. Каждый тип элемента поддерживает множество "своих" допустимых атрибутов, имеющих стандартные значения, заданные по умолчанию. Если, например,

нужно наделить заголовок каждого уровня индивидуальностью, то весь набор его уникальных атрибутов нужно повторять всякий раз, когда встречается очередной экземпляр. Четвертая версия HTML предлагает указывать желаемые свойства *таблицами стилей*, в которых этот набор фиксируется и объявляется классом один раз так же, как и заголовки одного уровня объявляются принадлежащими объявленному классу лишь однажды.

Наименования типов элементов и поддерживаемых ими атрибутов при обработке автоматически устанавливаются в верхний регистр, и вообще говоря, могут записываться авторами на любом регистре. Для наглядности восприятия разметки человеком, рекомендуется соблюдать общепринятую договоренность: имена элементов указываются символами верхнего, имена атрибутов – символами нижнего регистра. Переопределение атрибутов в начальном теге элемента осуществляется путем формирования списка пар "*имя атрибута – его значение*", разделенных пробелами.

Комбинации параметров. Ряд конструкций, типов элементов и атрибутов языка описываются схожими сочетаниями дескрипторов. Для упрощения выражений HTML такие часто употребляемые сочетания принято представлять комбинациями параметров. Когда агенты пользователя ссылаются на комбинацию по имени, она разворачивается в строку, которая может содержать имена других комбинаций параметров. Эти имена разворачиваются рекурсивно. Определение комбинации начинается с ключевого слова "<!ENTITY %", за которым следует имя комбинации, далее строка в кавычках, в которую разворачивается комбинация, и наконец, закрывающий символ ">". Экземпляры комбинаций параметров начинаются со знака "%", затем следует имя комбинации, которую замыкает необязательный символ ";". Пример комбинации элементов:

```
<!ENTITY  %  fontstyle    " TT | I | B | BIG | SMALL ">,
```

т.е. комбинация «стиль шрифта» представляется элементами «истинный (моноширинный) тип», «курсив», «полужирный», «крупный», «мелкий».

Комбинация атрибутов:

```
<!ENTITY % coreattrs  "--комбинация идентифика-
                       тор+класс+стиль+титул--
   id      ID              #IMPLIED   -- уникальный id в пределах док-та --
   class   CDATA           #IMPLIED   -- список классов (через пробелы) --
   style   %StyleSheet;    #IMPLIED   -- информация о стиле элемента   --
   title   %Text           #IMPLIED   -- текст всплывающей подсказки   --
">
```

Комбинация различных конструкций:

```
<!ENTITY % inline "#PCDATA|%fontstyle;|%frase;|%special;|%formctrl;">
```

Объявления элементов. Как и во всех компьютерных языках, авторам документов HTML не приходится объявлять типы элементов, которые описаны в формальных определениях типа документа (*Document Type Definition*

– *DTD*). Встречая в размеченном тексте очередной экземпляр элемента, агент пользователя обращается к DTD и воспроизводит указания автора в соответствии с найденными в первом теге инструкциями. Объявление начинается с ключевого слова “<!ELEMENT” и заканчивается символом “>”. Между ними указывают:

- Имя элемента.
- Символы обязательности тегов элемента. Два дефиса после имени элемента означают, что начальный и конечный теги являются обязательными. Дефис и следующая за ним буква “O” указывают, что конечный тег можно опустить. Две буквы “O” информируют, что можно опустить оба тега.
- Допустимое содержимое элемента, называемое *моделью содержимого*.

В следующем примере:

```
<!ELEMENT    UL  –  –    (LI) +>
```

- объявляется тип элемента UL – неупорядоченный список;
- два дефиса указывают, что начальный <UL> и конечный </UL> теги для этого типа элемента обязательны (иначе не ясно, где начать и где остановиться);
- модель содержимого для этого типа элемента (LI)+ устанавливает, что объявленный неупорядоченный список UL должен содержать, по меньшей мере (+), один элемент LI – строку списка.

В строке

```
<!ELEMENT    IMG  –  O    EMPTY>:
```

IMG – объявляется тип элемента – графическое изображение.

‘–‘ – дефис определяет обязательность начального тега, следующая за ним буква “O” в сочетании с моделью содержимого EMPTY запрещает использование конечного тега в экземплярах элемента IMG.

Ключевое слово EMPTY (пусто) означает, что экземпляры элемента не имеют содержимого.

Модель содержимого дает исчерпывающее описание, какие данные (или элементы) и в каком виде допустимы внутри элемента. Она фактически постулирует конструкцию элемента, а в широком смысле (композиция моделей объявленных элементов) – состав документа. Ее определения могут включать:

- имена допустимых и запрещенных элементов; например, тип элемента UL содержит экземпляры элемента LI, а элемент P не может включать другие элементы P;
- комбинации параметров (например, тип элемента LABEL может включать экземпляры комбинации параметров "%inline;" – только встроенные элементы);
- текст – обычные символы документа, не упакованные в структуры.

Пример составной модели содержимого:

```
<!ELEMENT  DL  –  –    (DT | DD)+>
```

Элемент DL – список определений – должен ограничиваться обоими тегами (чтобы знать, окончился список, или очередной элемент следует вложить внутрь него) и должен содержать хотя бы один или более элементов DT – терминов и DD – их определений, расположенных в любом порядке следования. При воспроизведении термины автоматически выделяются особым шрифтом, а их определения группируются в удобном для чтения виде. Элемент DL нередко используют для таких, например, целей, как оформление текстов драматургических сценариев (термин – имя персонажа, определение – очередная реплика).

Еще один характерный пример описания модели содержимого:

```
<!ELEMENT  A  – –   (%inline;)* – (A)>
```

Элемент A – якорь, устанавливающий либо место в тексте, щелчок мыши по которому приводит к автоматической загрузке нового ресурса, либо собственно новый ресурс, его начало. Модель содержимого элемента A – исходного (откуда) или целевого (куда) якоря гиперссылки – допускает включение любого встроенного фрагмента (%inline;)* текста. Символ "*" разрешает произвольное количество встроенных фрагментов, в том числе нулевое. Незаполненный якорь – понятие не очевидное. Из какого *реального* места в тексте исходит гиперссылка? В каком пустом целевом якоре находится ресурс, указанный ссылкой? Какова цель явного указания «–(A)», т.е. минус A?

Любой якорь исходного документа можно заполнить заново вызовом скрипта по произвольному внутреннему событию, например, по загрузке. Программный код скрипта разберется в ситуации – и в соответствии с ней сформирует текст, внедрит изображение, адресует ссылки. А явное запрещение вложенных якорей объясняется физическими причинами. Фактически гиперссылка есть вектор с началом и окончанием. И то, и другое – реальные адреса тех ресурсов, которые агент пользователя загружает в компьютер. Очевидно, что он не может загрузить абстрактную стрелку вектора, ему нужен вещественный поток байтов, поток физических нулей и единиц, следовательно, ссылки не могут быть вложенными. Таково ограничение HTML. Тип элемента A является частным случаем комбинации параметров "%inline;", но явно исключается сам из себя выражением –(A). Точно так же объявление типа элемента FORM (форма для ввода запроса) относит его к сочетанию параметров "%block;" уровня блока или элементов SCRIPT задания сценариев, и в то же время явно исключает из модели вложенные формы.

Объявления атрибутов. Список атрибутов, которые может иметь элемент, начинается ключевым словом "<!ATTLIST". За ним следует имя элемента, обладающего перечисляемыми свойствами. Основная часть объявления представлена списком, строки которого содержат определения атрибутов. Замыкает описание символ ">". Каждое определение атрибута задается следующей тройкой:

- имя атрибута;
- тип значения атрибута или явный набор величин;
- статус значения: является ли оно не явным (#IMPLIED), всегда обяза-

тельным (#REQUIRED) или фиксированным (#FIXED).

Примеры определений атрибутов представлены в таблице 1:

Таблица 1.

Имя	Тип	Статус	Пояснения
name	CDATA	#IMPLIED	-- уникальное имя элемента --
rowspan	NUMBER	1	-- число строк, охваченных ячейкой --
http-equiv	NAME	#IMPLIED	-- имя заголовка ответа http --
alt	%Text;	#REQUIRED	-- краткое альтернативное описание--
id	ID	#IMPLIED	-- уникальный идентификатор --
valign	middle | bottom | baseline | top	#IMPLIED	-- вертикальное выравнивание --

В этих примерах атрибут *name* – уникальное в пределах документа имя элемента – не является обязательным (#IMPLIED), тип его допустимых значений (CDATA) – текст, который может содержать ссылки на символы. Атрибут *rowspan* (сколько строк таблицы объединены общим заголовком) представляется числом, является обязательным, по умолчанию задается единицей. Для необязательного атрибута *http-equiv* (ответ) нужны значения типа NAME, для необязательного атрибута *id* – значения типа ID. Необязательный атрибут *valign* ограничен явными значениями из набора {*middle* | *bottom* | *baseline* | *top*}. Значения обязательного атрибута *alt* объекта устанавливаются текстом.

Уточним роли некоторых атрибутов в представлении элементов. Зачем указывать *rowspan*, если при заполнении ячейки таблицы ее границы раздвигаются по вертикали автоматически? Ровно на одну строку, когда набираемый текст достигает края. Что такое *border* – имя атрибута или его значение? В чем сходство и различие атрибутов *name* и *id*? С какой целью элементы поддерживают их одновременно? Каким образом выполняется вертикальное выравнивание, если *valign* = "*baseline*"?

По определению *rowspan* объединяет строки, для которых данная ячейка образует общий заголовок. Он используется не для увеличения пространства ячейки, как это произошло в нижней строке табл. 1, а для горизонтальной группировки раздела таблицы. Далее, значения атрибутов *name* и *id* представлены разными типами данных: *name* обеспечивает большее разнообразие имен, зато *id* может исполнять не только функции идентификатора элемента, но и коммутатора логических условий обработки текста или селектора таблиц стилей и т.д. Поясним на примере. Каждый гражданин России имеет фамилию (*name*) и номер паспорта (*id*). Их значения представлены разными типами данных, *name* обеспечивает большее разнообразие имен, зато *id* может исполнять не только функции идентификатора владельца, но и

коммутатора социальных услуг или селектора картотеки паспортного стола и т.д.

Термин *border* интерпретируется двояко: это атрибут элемента TABLE (таблица) и одновременно – значение атрибута *frame*, указывающее, что таблица при отображении должна окаймляться внешней рамкой. Атрибут *frameborder* (слитно) отвечает за наличие границы фрейма в многооконном документе. Наконец, *baseline* в вертикальном выравнивании означает, что во всех ячейках той строки таблицы, в которой находится ячейка с установленным в это значение атрибутом *valign*, текст должен располагаться так, чтобы первая строчка оказывалась на базовой линии, общей для всех ячеек строки. По умолчанию вся строка может выравниваться либо по первой строчке этой ячейки, либо по той ее строчке, в которой находится курсор. Случайно поместив фокус ввода в такую ячейку, мы можем незаметно для себя переформатировать всю строку таблицы.

Большинство элементов поддерживают пересекающееся множество атрибутов, и для их определения в DTD представлен ряд известных комбинаций. Самая распространенная записывается как сокращение “%attrs;”:

<!ENTITY % attrs “%coreattrs; %i18n; %events;”>

Сочетание комбинаций разворачивается при обработке рекурсивно: "%программа" ⇒ "%компьютерный код", "%перечень тем учебного курса", "%график работы конференции" и т.д. Выше рассматривался состав "%coreattrs;". Комбинация "%i18n;" объединяет атрибуты *lang* (код языка) и *dir* (“слева направо”, “справа налево”), определяющие язык и направление текста или содержимого таблицы. Комбинация "%events;" включает набор внутренних событий, состоящих в прерываниях компьютера от нажатия клавиш, движения и щелчков мыши, от смены фаз обработки. Значения атрибутов представляют собой скрытые от пользователя программные скрипты, не отображаемые в документе, но исполняемые при воспроизведении.

Непросто уяснить себе, что события в компьютере квалифицируются в HTML как атрибуты элементов документа. Вы подвинули мышь, и это действие интерпретируется как свойство, например, абзаца или таблицы. И значением этого свойства становится программный код, вырабатывающий результат. Всплывает подсказка, меняется цвет гиперссылки, символ курсора и т.д.

Некоторые атрибуты играют роль булевых переменных. Их наличие в начальном теге подразумевает, что объявленное ими свойство присуще тому экземпляру элемента, в котором они упомянуты. Логические атрибуты могут принимать только одно значение: собственное имя атрибута. Разметка <OPTION *selected*> указывает агенту пользователя, что форма для запроса, отображаемая вероятному клиенту, должна уже в момент появления на экране содержать выбранный пункт меню. Объявление <OBJECT *declare*> говорит о том, что внедряемый в документ объект должен быть сформирован при загрузке, но его инициализация будет произведена позже, в зависимости от поведения пользователя, причем не исключена возможность воспроизведения в тексте многих копий с него, наполненных разным изобразительным и

смысловым содержанием.

Ссылки в документах HTML. Прозрачность понятия приводит на экзаменах к тому, что студент не в силах дать ему конкретное определение. Все знают, что главным отличием гипертекстов является простота переходов по ресурсам, что одним щелчком кнопкой мыши мы вызываем на отображение нужный нам фрагмент документа, что ссылкой в любом литературном труде называется указание на источник информации. Однако, судя по опыту тестирования, к этому общеизвестному определению удается подойти не сразу, независимо от уровня подготовки. Механизм ссылок настолько очевиден, что при разметке не возникает необходимости задерживаться на разъяснении самому себе деталей технической реализации. Попробуйте ответить, что такое ссылка в HTML. Элемент ли это языка, или атрибут? Что представляет собой это «главное отличие гипертекстов и залог успеха сети Интернет»?

Известно: ссылки связывают один ресурс Web с другим. Каждая из них имеет два конца (якоря) и направление. Ссылка начинается в источнике (исходном якоре), который не может не быть элементом документа, и указывает на целевой якорь, который может быть любым ресурсом – изображением, видеоклипом, документом, элементом в документе и т.д. По умолчанию со ссылкой связана загрузка другого ресурса, т.е. *поведение* или *действие* системы, достигаемое с помощью сознательных операций пользователя – щелчка мыши, ввода с клавиатуры, голосовой команды и т.д. В технической системе можно найти аналогию возникающей здесь путанице. Есть источники данных и центр обработки информации, соединенные каналами. Что такое канал: связь между элементами, или элемент (подсистема передачи данных)? Если на схеме изображена стрелка, то ответ не заставит себя ждать – это связь, скажете вы. Но если соединение представлено каскадом модемов, усилителей, линий задержки и других согласующих устройств, то вы постараетесь ответить более расплывчато: канал есть элемент системы, организующий связь между устройствами.

Сказанного достаточно, чтобы отсечь неверные ответы на вопрос, что такое ссылка. Нет, это ни элемент, ни атрибут. Как первое, так и второе реализуются всякий раз при воспроизведении документа – хотя бы своим альтернативным содержимым. А для реализации ссылки недостаточно факта загрузки. Нужен дополнительный импульс, вмешательство пользователя документа, активная обратная связь, управляющее воздействие. Ссылка – это потенциальное средство интерактивного взаимодействия, «спусковой крючок», ожидающий приказа. Сама по себе она не сработает. Это конструкция языка, обеспечивающая возможность простого перемещения по файлам. Реализовать эту возможность может не всякий элемент. Для этого он должен обладать особым свойством, он должен уметь указывать направление, путь к ресурсу. Он должен поддерживать хотя бы один атрибут адресации. Уметь описывать целевой якорь. Таких атрибутов несколько – *href, src, data* – и практически у всех один тип значения: %*uri*. В частности, его подмножества, ограниченные типами значений *id* и *cdata*, если целевой ресурс расположен внутри воспроизведенного документа. Каждый знает, какие типы элементов

способны образовать ссылки: – это, например, A, AREA, FORM, LINK, MAP.

Интернационализация. Речь идет о принятии стандарта ISO/IEC:10646 в качестве набора символов для документов HTML, узаконившего алфавиты всех существующих языков. Это наиболее содержательный стандарт, в котором решены вопросы представления национальных символов, направления письма, пунктуации и другие. Определены конструкции (ссылки на символы), позволяющие на любом компьютере ввести в документ любую букву или иероглиф любого языка с помощью стандартной латинской клавиатуры. Указание языка содержимого позволяет агенту пользователя при воспроизведении загрузить необходимые шрифты (или фонемы синтезаторов речи) и отобразить документ в том виде, в котором его задумали. К разработке четвертой версии языка разметки были привлечены лингвисты, и теперь документы можно писать на любом языке и свободно распространять их по всему миру. Реализована поддержка различных языков в одном документе. Это обеспечивает эффективное индексирование документов с помощью ключевых слов для поисковых машин, а также позволяет повысить качество полиграфии, преобразования текста в речь, корректные переносы слов на следующую строку и т.д.

Доступность. Термин не фигурирует в формальных конструкциях, и при изучении языка на него не обращают внимания, хотя реклама всех продвижений HTML представляет это свойство как магистраль развития Web. Сообщество пользователей сети растет, и возможности его членов существенно различаются. Особенно это касается людей с недостатками зрения и других органов чувств. Здесь имеются в виду соображения доступности сети именно для пользователей с физическими недостатками. Доступность – это расширение диапазона целевых устройств воспроизведения документов: синтезаторов речи, тактильных приставок азбуки Брайля. Сюда же относятся и усовершенствования традиционных устройств воспроизведения за счет следующих дополнений:

- рекомендовано использовать таблицы стилей вместо переопределения атрибутов, позволяющие быстро готовить изящные и красочные документы, не тратя времени на подбор или формирование собственного стиля;
- улучшены формы для запросов, добавлены "горячие" клавиши, семантическая группировка элементов и меток, включены активные метки;
- расширена сфера применения альтернативного текста, выводящего словесное описание тех частей документа, которые не могут воспроизводиться на устройствах с ограниченными возможностями (элементы OBJECT, MAP, IMG);
- добавлена поддержка всплывающих подсказок во всех элементах, введены сокращения и другие усовершенствования таблиц, дана возможность отображать таблицу по мере чтения, не дожидаясь окончания загрузки.

К категории доступности относят также удобство адаптации к будущим технологиям, которое появилось в HTML как результат разделения структуры и оформления документа благодаря таблицам стилей.

Набор символов документа и кодировка символов. Первое определяется

как множество абстрактных символов, которые могут входить в состав документа HTML. Второе постулирует форму представления этих символов двоичными нулями и единицами в файлах HTML-страниц, т.е. в потоке байтов, передаваемом по сети. Можно пояснить проще: возьмем *набор символов* русского алфавита и применим к нему *кодировку* азбукой Морзе. Компьютеру достаточно явного указания, какими битами кодировать тот или другой символ, в крайнем случае, указания правил выбора такой кодировки. Простейший способ проинформировать браузер о ней – использовать параметр “charset” в поле заголовка “Content-Type” протокола HTTP. Например, следующий заголовок HTTP объявляет, что пересылаемую страницу следует интерпретировать одним из диалектов японской кодировки EUC-JP:

Content-Type: text/html; charset = EUC-JP

Создатели языка дополнили эту рекомендацию практическим советом: некоторые серверы *блокируют* отправку параметра “charset”. Зато они не препятствуют передаче таблицы кодировки через заголовок документа с помощью установки тройки атрибутов элемента META:

<META http-equiv=”Content-Type” content=”text/html; charset = EUC-JP”>

Низший приоритет при выборе имеет установка “charset” в элементе, загружающем внешний ресурс. Если нет и его, то агенту пользователя остается «вспомнить» метод проб и ошибок и попытаться применить к загружаемому ресурсу все хранящиеся в его памяти таблицы по очереди. При отсутствии подходящей кодировки в окне диалога выпадает список всех известных браузеру кодировок, и пользователю предлагается выбрать из его строк нужную.

Набор символов каждого документа включает последовательность символов собственного репертуара. Каждый символ идентифицируется по его коду. Например, в наборе символов ASCII коды 65, 66, 67 означают символы "A", "B", "C" соответственно. Набора символов ASCII недостаточно для глобальной информационной сети, и поэтому совершен переход к универсальному набору символов, определенному в упомянутом выше ISO/IEC:10646. Серверы отправляют документы пользователям в виде потока байтов, браузеры интерпретируют их как последовательные символы. Способы преобразования могут меняться от простого соответствия один к одному до сложных схем и алгоритмов переключения. Серверы и прокси могут изменять кодировку на лету для выполнения запросов пользователей. Эта процедура получила название транскодирования.

В любом случае не исключена возможность, что агент пользователя не сможет отобразить все символы, употребленные в документе, например – из-за отсутствия соответствующего шрифта или когда символ имеет значение, которое не может выразиться средствами внутренней кодировки агента пользователя. Как правило, в таких случаях применяются механизмы оповещения пользователя об отсутствующих ресурсах.

Ссылки на символы – это не зависящий от кодировки механизм ввода

любых символов. Эти конструкции HTML могут принимать две формы: числовую и мнемоническую (комбинации ссылок). Любому символу любого языка стандартом ISO10646 (Unicode) поставлен в соответствие уникальный код. Нужно лишь знать его изображение, десятичное или шестнадцатеричное. Создатели языка разметки вообще рекомендуют применять шестнадцатеричную, а не десятичную форму представления, так как именно эта форма используется в стандартах наборов символов. Не все символы описаны на сегодняшний день десятичными ссылками. Синтаксис ссылки прост: лидирующий символ амперсанта “&”, далее решетка “#” и десятичный номер нужного символа Unicode. Если используется шестнадцатеричный номер, то перед его указанием, сразу после решетки, следует поставить метку ‘x’ любого регистра. Дополнительно предлагаются так называемые комбинации ссылок на символы. Вместо кодов они содержат мнемонические подсказки. Например, букву шведского алфавита å, т.е. “a” с колечком (a + ring), так и записывают “&aring”. Задавая “&Aring”, мы увидим в документе Å. В таблицу 2 сведены несколько примеров:

Таблица 2.

знак	10-тичная ссылка	16-ричная ссылка	комбинация	пояснения
å	&#229	&#xE5 (&#Xe5)	&aring	шведская å
&	&#38	&#x26	&amp	амперсант
<	&#60	&#x3C	&lt	строго меньше
水	&#27700	&#x6C34	не введена	иероглиф ‘вода’
Α	&#913	&#x391	&Alpha	заглавная альфа
α	&#945	&#x3B1	&alpha	строчная альфа
£	&#163	&#xA3	&pound	знак фунта
©	&#169	&#xA9	&copy	авторское право
♥	&#9829	&#x2665	&hearts	черные червы
⇐	&#8656	&#x21D0	&lArr	двойная стрелка
"	&#8222	&#x201e	&bdquo	двойная кавычка
•	&#8226	&#x2022	&bull	маркер списка
°	&#176	&#xB0	&deg	знак градуса
″	&#8243	&#x2033	&Prime	секунды, дюймы

Ряд символов для однозначной интерпретации рекомендуют указывать не явно, а ссылками на них. Упомянем их, поскольку они часто используются для специальных целей:

- "<" – представляет собой знак <;
- ">" – представляет собой знак >;
- "&" – представляет собой знак &;
- """ – представляет собой знак ".

При необходимости употребления в тексте символа "<" следует использовать ссылку "<" во избежание путаницы с началом тега. Аналогично, следует использовать "<" вместо ">", чтобы избежать проблем со старыми браузерами, ошибочно принимающими их за окончание тега. Рекомен-

дуется использовать "&" вместо "&", чтобы не перепутать символ с началом ссылки на него. Символ двойной кавычки также желательно указывать ссылкой, потому что он может использоваться в качестве разделителя значений атрибутов.

Типы данных языка разметки. Коротко обсудим основные типы данных, которые могут составлять содержимое элементов или значения атрибутов.

Информация о регистре. Определения атрибутов могут устанавливаться с учетом регистра, и в DTD этот факт постулируется следующими кодами.

CS – значения учитывают регистр, т.е. "a" и "A" интерпретируются браузерами различным образом.

CI – значения не учитывают регистр, т.е. "a" и "A" интерпретируются одинаково.

CN – значение не зависит от регистра, например, потому что это цифра.

CA – информация о регистре заключена в определении элемента или атрибута.

CT – учет регистра установлен в определении типа содержимого.

Если значение атрибута представлено списком, то код учета регистра применяют к каждому его элементу.

Универсальный идентификатор ресурсов (URI). Адрес документа, или ресурса, который нам необходим. Каждый ресурс в Web располагается по собственному адресу, состоящему из трех частей:

- схема наименования механизма, используемого для доступа к ресурсу;
- имя компьютера, на котором располагается ресурс;
- имя собственно ресурса, заданное в виде пути.

Рассмотрим URI спецификации HTML 4.0 на сервере W3C:

http://www.w3.org/TR/PR-html4/index.html

Этот документ можно получить по протоколу HTTP, он располагается на компьютере www.w3.org, путь к этому документу – TR/PR-html4/index.html. В документах HTML можно встретить и другие схемы, например, mailto для электронной почты и ftp для протокола FTP.

Некоторые URI указывают на местоположение фрагмента документа внутри ресурса. Этот тип URI заканчивается символом "#", за которым следует указатель, или идентификатор фрагмента. Например, следующий URI указывает на раздел документа:

http://somesite.com/html/top.html#section_2

Подмножество *относительных* URI не содержит информации о схеме наименования. Путь в нем указывает ресурс на машине, на которой находится текущий документ. Относительные URI могут содержать компоненты относительного пути (например, ".." означает «один уровень выше в иерархии») и идентификаторы фрагментов. Для разрешения относительных URI используют базовые URI, указанные в воспроизводимом документе.

Цвета. Значения цвета указываются шестнадцатеричным числом, кото-

рому предшествует знак диеза "#" или названием, которое учитывает регистр. Отметим некоторые оттенки – Black = "#000000" – черный, White = "#FFFFFF" – белый. А также Maroon = "#800000" – коричневый и Red = "#FF0000" – красный. Аккуратное использование цветовой палитры позволяет существенно повысить информативность документа и удобства чтения. Однако следует учитывать ряд сопутствующих обстоятельств:

- использование элементов и атрибутов HTML для указания цветов объявлено нежелательным, для этой цели рекомендованы таблицы стилей;
- избегайте сочетания цветов, вызывающих у пользователей затруднения при восприятии;
- если цветное изображение использовано в качестве фона, не забывайте подбирать к нему соответствующий цвет шрифта;
- не рассчитывайте на одинаковое воспроизведение созданной цветовой гаммы на компьютерах разных платформ.

Длины. Определяют линейные размеры всех представленных на экране элементов. Выражаются целыми числами пикселей, либо долей известного отрезка в процентах, либо в относительных единицах (частях известного отрезка). Таким образом, значение "50" означает 50 пикселей, а значение "50%" означает половину доступного (на экране, на бумаге) пространства. Относительная длина имеет форму "i*", где "i" – целое число. При распределении пространства между конкурирующими элементами, агенты пользователя сначала отводят место для длин, определенных в пикселях и процентах, а затем делят оставшееся место между относительными длинами. Каждая относительная длина получает часть доступного пространства, пропорциональную целому числу i. Указание "*" равноценно записи "1*".

Дата и время. Представляется в разных форматах. Самый популярный – год, месяц, день, час, минуты, секунды. И указатель часового пояса (сдвиг относительно Всемирного скоординированного времени). Вот пример:

ГГГГ-ММ-ДДТчч:мм:ссУЧП,

где: ГГГГ – год, указанный четырьмя цифрами,

ММ – месяц, заданный двумя цифрами, с января по сентябрь – с лидирующим нулем;

ДД – день месяца, 01 – 31;

Т – разделитель (начало строки текущего времени);

чч – две цифры часов 00 – 23, «до и после полудня» не допускается;

мм – две цифры минут 00 – 59;

сс – две цифры секунд 00 – 59;

УЧП – указатель часового пояса (со знаком и величиной сдвига времени относительно нулевого меридиана).

Данные сценария. Программные коды для создания динамических страниц и обеспечения правильной реакции компьютера на происходящие с документом события. Могут быть содержимым элемента SCRIPT и значением атрибутов внутренних событий. Браузер не должен оценивать данные сценариев в разметке HTML, а должен передавать их на исполнение ядру

скриптов. Скрипты, выполняемые при загрузке, могут динамически изменять содержимое документа. Технологически это осуществляется в три шага. Сначала оценивается корректность сценария, далее с его помощью генерируется текст, полученные данные также оцениваются, и в случае успешного завершения результат становится новым фрагментом документа. Скрипты, созданные для реакции системы на внутренние события, выполняются, когда эти события происходят именно с теми элементами, для которых определены соответствующие атрибуты. Они могут осуществлять контроль вводимой в поля формы информации, отображать на экране подсказки, изменять внешний вид элементов.

3. СИСТЕМА FRONTPAGE

Microsoft FrontPage как универсальная система управления узлами Web эксплуатируется давно. Поддерживаемые пакетом функции перешли из разряда экзотики в категорию повседневной работы. Современные офисы, в том числе – кабинеты Государственной Службы Гражданской Авиации, не мыслятся без собственного узла с информацией о департаменте или компании, доступного через Интернет, как и без публикации корпоративных данных на внутреннем узле. Это не просто модно, – это, прежде всего, удобно и полезно. Технологии Интернета обеспечивают надежный доступ к информации, и система управления FrontPage является популярным инструментом их использования. Как и другие приложения пакета Microsoft Office, обсуждаемое тоже обеспечено средствами гибкой настройки. Можно изменить параметры всей системы, файлов и проектов, открываемых в нем, чтобы получить желаемый результат.

При работе с Microsoft FrontPage достаточно оперировать двумя понятиями, которые мы напомним. Под *веб-страницей* понимается собственно документ в формате HTML. Она не обязательно должна содержать ссылки на другие страницы (например, http://www.khokhlov.ru/last.html). Под *веб-сайтом* понимается совокупность веб-страниц, связанных между собой гиперссылками и объединенных общей темой и целью, например: предоставить информацию об МГТУ ГА.

Система FrontPage позиционируется разработчиком как инструмент для создания веб-сайтов. Помимо редактора HTML-страниц, она включает средства управления веб-сайтами. На рис. 3.1 представлено окно системы после запуска. Все средства для манипуляции веб-сайтами вызываются с помощью левой панели инструментов. Средства редактирования веб-страниц доступны

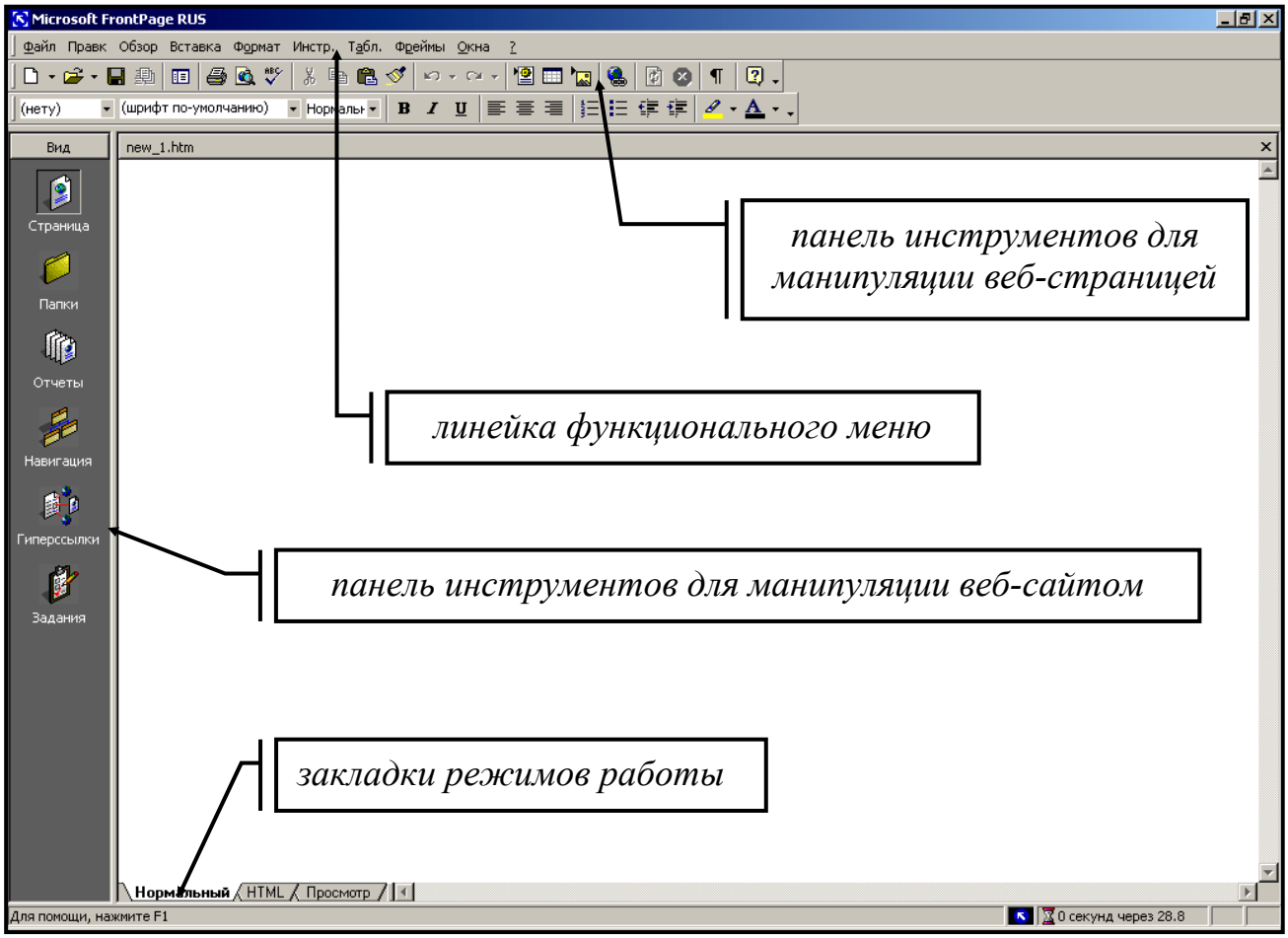

Рис. 3.1. Microsoft FrontPage 2000. Общий вид

через меню и верхнюю панель инструментов. Схема работы проста: задать структуру узла, упаковать в эту структуру свои страницы и создать связи между ними.

В состав FrontPage входят несколько заготовок сайтов с различной структурой страниц. При входе в систему автоматически создается веб-сайт, содержащий одну веб-страницу. Если вы желаете воспользоваться такой заготовкой – необходимо обратиться в меню «Файл»/«Новый»/«Сайт». Появится окно выбора заготовки сайта (см. рис. 3.2). Не нужно описывать все возможные заготовки. Для выполнения лабораторной работы достаточно использовать:

- One Page Web – веб-сайт с одной, заглавной, веб-страницей;
- Empty Web – тот же веб-сайт с одной безымянной веб-страницей;
- Personal Web – создается прототип персонального веб-сайта с несколькими веб-страницами (фотоальбомом, списком интересов и т.д.).

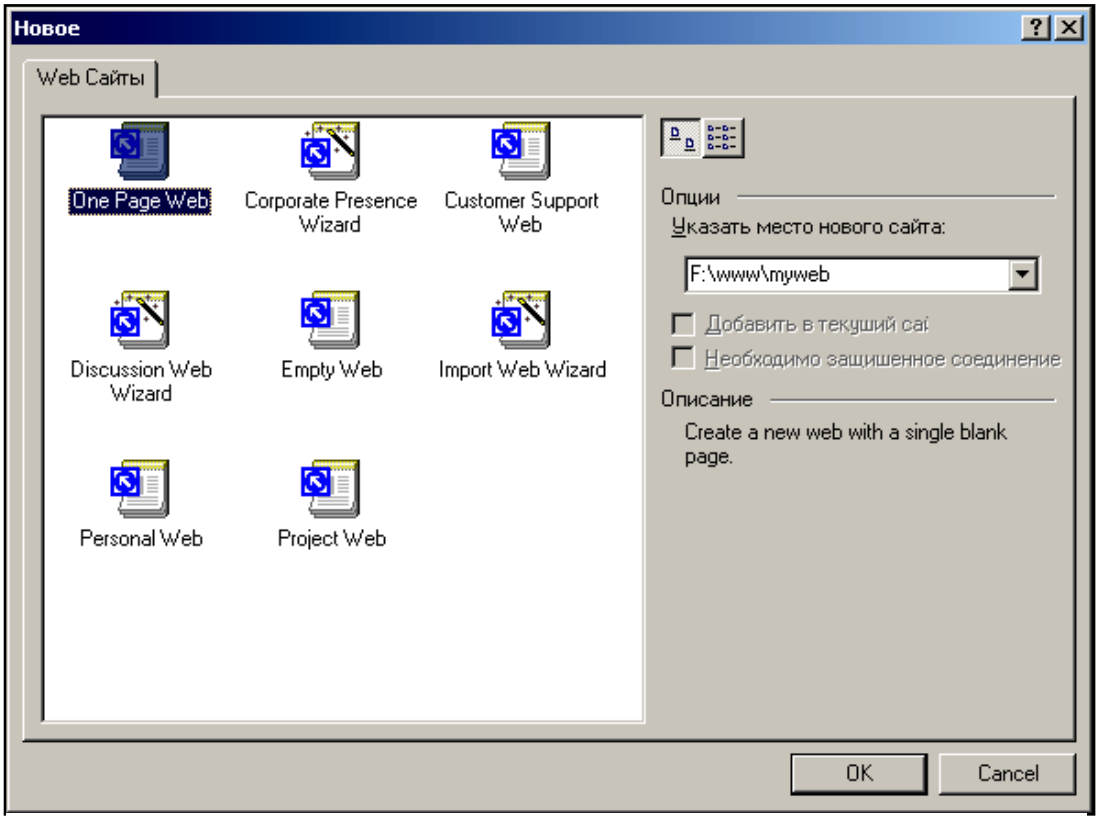

Рис. 3.2. Выбор заготовки сайта

Заготовки Empty Web или One Page Web помогают создать любой сайт «с нуля», не навязывая стиль или структуру сайта. Для управления структурой сайта существует левая панель инструментов. По нажатию кнопки «Страница» (она сверху) появляется редактор HTML-страниц. Кнопка «Папки» предъявит список папок, входящих в состав веб-сайта. Папками можно манипулировать: создавать, удалять, перемещать. Все эти функции доступны через меню «Файл» и контекстное меню, вызываемое щелчком правой кнопки «мыши» на имени папки. С помощью подменю «Обратить в сайт» папку пакуют в отдельный, не зависимый от ныне редактируемого, сайт. Нажатие кнопки «Отчеты» вызовет список отчетов о страницах, входящих в веб-сайт (см. рис. 3.3). Двойной щелчок левой кнопки «мыши» по элементу списка выводит на экран более подробную информацию по выбранному отчету.

Имя	Подсчет	Размер	Описание
Все файлы	0	0KB	Все файлы на текущем сайте
Картинки	0	0KB	Файлы картинок в текущем сайте (GIF, JPG, BMP, и т.д.)
Несвязанные файлы	0	0KB	Файлы в текущем сайте, которые не могут быть достигнуты стартуя с вашей Домашней страницы
Связанные файлы	0	0KB	Файлы в текущем сайте которые можно достугнуть стартуя от вашей Домашней страницы
Медленные страницы	0	0KB	Страницы в текущем сайте которые грузятся дольше чем 30 секунд при скорости 28.8
Старые файлы	0	0KB	Файлы в текущем сайте не измененные за последние 72 дней
Последние добавленные страницы	0	0KB	Файл в текущем сайте добавленные за последние 30 дней
Гиперссылки	0		Все гиперссылки в текущем сайте
Непроверенные гиперссылки	0		Гиперссылки указывающие на непроверенные конечные файлы
Сломанные гиперссылки	0		Гиперссылки указывающие на недоступные файлы
Внешние гиперссылки	0		Гиперссылки указывающие на файлы вне текущего сайта
Внутренние гиперссылки	0		Гиперссылки указывающие на другие файлы внутри текущего сайта
Ошибки компонента	0		Файлы в текущем сайте с компонентами выдают ошибку
Невыполненные задания	0		Задания в текущем сайте которые не отмечены как выполненные
Неиспользованные темы	0		Темы в текущем сайте которые не применены ни к камому файлу

Рис. 3.3. Отчет о создаваемом веб-сайте

На рис. 3.3, в разделе «Описание», отображены такие фразы, как:

- «за последние 30 дней»;
- «дольше, чем 30 секунд»;
- «за последние 72 дней».

Эти параметры настраиваются через меню «Инструменты» / «Опции» / «Просмотр отчетов». Нажатие кнопки «Навигация» вызывает схему вашего веб-сайта со всеми его страницами и гиперссылками на них. По нажатию кнопки «Гиперссылки» появятся: список страниц сайта и указание всех ссылок на выбранную из этого списка страницу со всех страниц сайта, а также все ссылки с этой страницы на все другие страницы сайта.

Перейдем к главному в системе Front-Page, к редактору веб-страниц. Собственно редактор состоит из трех закладок: «Нормальный», «HTML» и «Просмотр». По первой закладке вы попадаете в окно редактора, в котором, как в редакторе Word, можно создавать текст документа (рис. 3.4). Рисунок состоит из трех частей. Верхняя представляет окно «Нормальный», средняя – окно «Просмотр», нижняя – окно редактора. Это – три режима работы со страницей.

Режим «Нормальный» предназначен для редактирования страницы практически точно в таком же виде, как она будет выглядеть в процессе просмотра Web-страницы пользователем сети Интернет. Другое название режима – «видишь то, что получишь», т.е. страница отображается практически точно в таком же виде, как она будет выглядеть в программе просмотра, такой как Internet Explorer. В этом режиме на экране представлены заголовок страницы, панель навигации в виде кнопок под заголовком, панель навигации в виде гиперссылок в левой части страницы – и текст в правой части. Параметры панели навигации в виде кнопок можно изменять. Эти изменения распространятся на все страницы сайта, поскольку созданные автоматически панели находятся в общих для всех страниц областях, расположенных по краям страницы. Можно удалить лишнюю панель навигации, можно добавить новую панель.

По второй закладке вы попадаете в окно редактора HTML. Здесь отображен код, который вы набирали клавишами в первом окне. Существует и обратная связь – все то, что вы написали во втором окне, найдет свое отражение в первом. Режим HTML позволяет редактировать страницу в текстовом представлении. Используйте этот режим, когда хорошо овладеете языком разметки гипертекстов.

По третьей закладке попадаем в окно просмотра, которая покажет, как ваша веб-страница будет выглядеть в браузере. Режим «Просмотр» позволяет не только просмотреть страницу в том виде, в котором ее отобразит браузер, но и проверить функции этой страницы, такие как: переходы по ссылкам, корректное отображение рисунков, использование объектов и форм и т.д.

Переход из режима в режим осуществляется, как и в других пакетах, щелчком мыши по закладке нужного режима. Начнем с первой закладки.

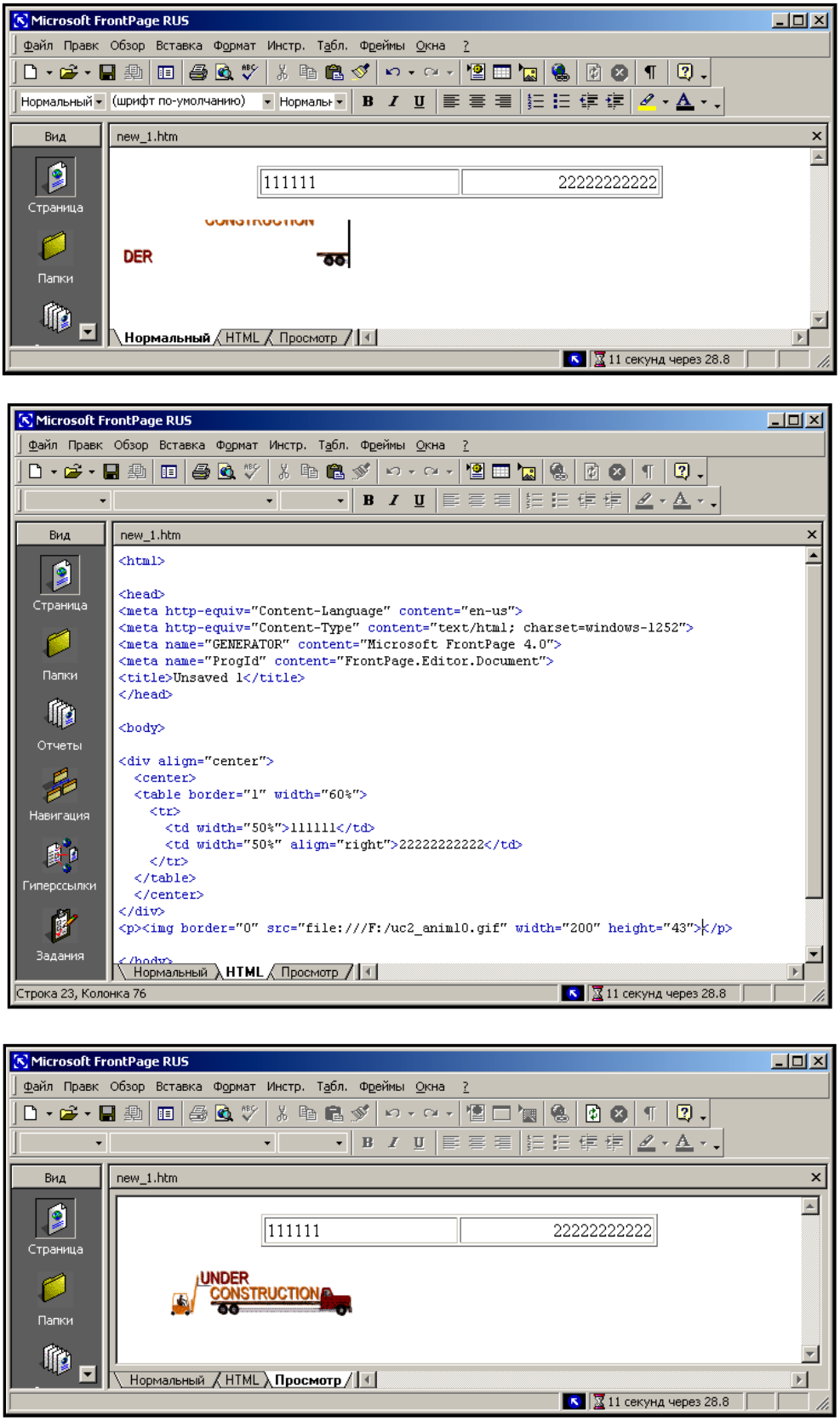

Рис. 3.4. Закладки окна редактора веб-страниц

Щелчком правой кнопки мыши на каждом редактируемом объекте можно получить всплывающее контекстное меню, как это показано на примере элемента «таблица» (см. рис. 3.5). При этом всегда можно отредактировать свойства страницы: язык, кодировку символов, заголовок, фон, шрифт, поля и т.д.

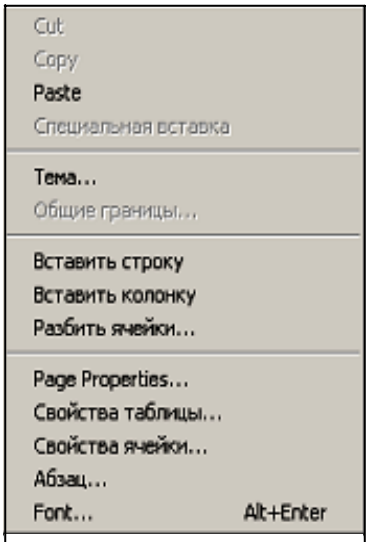

Рис 3.5. Контекстное меню

Первая закладка редактора веб-страниц по содержанию фактически совпадает с хорошо известным окном редактора Microsoft Word. Обычно работа с ней не вызывает затруднений, и в данном изложении касаться ее мы не будем.

Исходное меню в FrontPage – это «Обзор». Помимо дублирования левой панели инструментов, оно содержит пункт «Показать теги», щелчок левой кнопкой мыши по которому включает или выключает режим показа тегов при редактировании документа на первой закладке редактора (см. рис. 3.6).

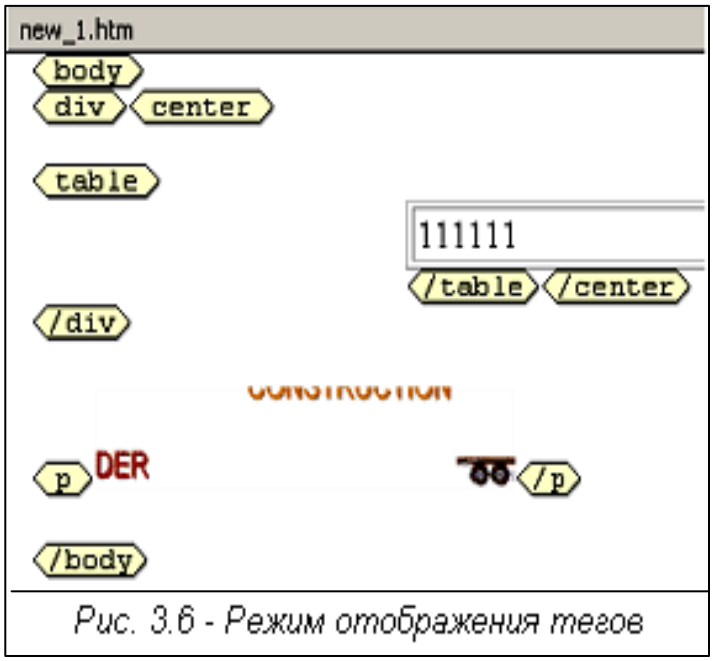

Рис. 3.6 - Режим отображения тегов

Меню «Вставка» используется, чтобы внедрять в документ различные объекты: изображения, компоненты, формы, документы в формате других изделий Microsoft Office и т.д. Будьте очень внимательны при добавлении этих вставок в формы. Для того чтобы они работали, на веб-сервере должны быть дополнительно установлены FrontPage Web Server Extensions. Взаимный обмен данными между клиентским веб-сервером и компьютером основан на методе, который реализован с помощью веб-ботов. Они так и называются: «обменные» компоненты FrontPage для обеспечения передачи данных. Эта технология поддерживается только веб-серверами, на которых установлен FrontPage Web Server Extensions. Веб-бот может вызвать функцию записи информации, введенной в файл, в форму, функцию поиска информации по сайту, функцию вывода даты последнего обновления веб-страницы и т.д. Введены эти компоненты для того, чтобы начинающим проектировщикам не составило труда создать, например, страничку поиска информации по своему веб-сайту, или гостевую книгу. В тексте HTML-документа веб-бот записывается так:

```
<!--webbot bot="SaveResults" U-
File="fpweb:///_private/form_results.txt"S-
Format="TEXT/CSV" S-Label-Fields="TRUE" -->
```

В меню «Формат» особого внимания заслуживают два пункта: «Динамик HTML эффекты» и «Тема». «Динамик HTML эффекты» вызывает окно добавления реакций на различные события, происходящие с произвольным элементом веб-страницы. В результате добавления эффекта в теле документа появляется заготовка функции на языке JavaScript, которую вы заполняете в соответствии со своим замыслом.

Пункт «Тема» предназначен для оформления веб-страницы аналогично одному из шаблонов (тем), предлагаемых создателями FrontPage.

4. АНАЛИЗ ТИПОВЫХ ТЕСТОВ ПО КУРСУ

В данном разделе обсуждаются экзаменационные и контрольные тесты прежних лет и варианты ответов на них.

4.1. *Назначение системы FrontPage:*

Варианты ответов:

1. Универсальный инструмент форматирования титульных листов, фронтисписов и оглавлений документов.
2. Универсальный инструмент для обмена и управления узлами в Web.
3. Универсальная настольная издательская система.
4. Универсальный текстовый процессор.

Система FrontPage позиционируется изготовителем как инструмент для создания веб-сайтов. Помимо редактора HTML-страниц, она включает средства управления веб-сайтами и обмена между ними. Первый вариант говорит, что ее возможности ограничены форматированием весьма ограниченного спектра составных частей документов. Ответ неверный. Правильный вариант второй.

Утверждение третьего варианта ошибочно. Язык разметки гипертекстов нередко называют средством публикации в Интернете. Однако под термином «издательская система», введенным в обиход еще до появления всемирной паутины, понимаются обычно пакеты прикладных программ, предназначенных для подготовки макетов типографской продукции, для верстки страниц и блоков печатных изданий.

Редактирование текстов является одной из функций системы FrontPage, но не основной. Текстовые процессоры, например Word, способны образовывать ссылки, допускают формирование гипертекстов, однако эти возможности не являются для них главными. При этом они не могут воспроизводить страницы HTML в том виде, в котором их представит браузер. У текстовых редакторов и у средств обмена в сети Web разные назначения. Четвертый вариант неверен.

4.2. *Элементами и атрибутами языка HTML называются:*

Варианты ответов:

1. Конструкции (структуры) языка SGML, формализованные в определениях типа документа HTML, а также желательное поведение при воспроизведении документа.
2. Теги простейших объектов HTML, со связями и отношениями между ними, закрепленными заданными свойствами и их значениями.
3. Совокупность средств обеспечения взаимодействия в Web.
4. Типы данных языка HTML.

По определению, типы элементов – это целостные структуры языка или указания на желательное поведение. Стандартные детали для создания документов. Атрибуты – это свойства элементов. Цвета, адреса, размеры. Каждый тип элемента поддерживает ограниченное множество допустимых атрибутов. Об этом говорит верный первый вариант.

Ошибка второго варианта состоит в том, что теги являются, на самом деле, составными частями формального представления элементов HTML, необходимыми для их распознавания программами. Ответ утверждает обратное, что элементы вместе с атрибутами принадлежат множеству тегов.

Третий вариант говорит о, что элементы и атрибуты в своей совокупности покрывают все множество средств обмена в сети. Это неправильно. Для взаимодействия нужны механизмы, процедуры, схема адресации, протоколы и язык разметки, а не его ограниченное подмножество, указанное в данном ответе.

Согласно четвертому варианту, элементами и атрибутами называются типы данных языка. Типы данных могут служить ограничением значений атрибутов, но никак не элементами или атрибутами. Ответ неправильный.

4.3. *Технология активных серверных страниц предназначена с целью:*

Варианты ответов:

1. Для активизации протокола http при взаимодействии в Web.
2. Для динамического создания и форматирования HTML-страниц.
3. Для поддержания совместимости версий HTML на стороне сервера.
4. Для поддержания совместимости различных версий HTML как на стороне сервера, так и на стороне клиента.

Напомним о такой услуге интернета как поиск информации. Нам нужно выяснить цену и место продажи товара, или расписание поездов, или другие справочные данные. Свой вопрос мы формулируем в словесном виде: товар, вокзал, стоимость. Сервер принимает от нас указание, какие компоненты источников Web нам необходимы, преобразует полученные ключевые слова в SQL-запрос к базе данных и форматирует документ, содержащий не весь необъятный ресурс, а только то, что было заказано. Выдает страницу, которая создается автоматически с помощью технологии активных серверных страниц.

Первый вариант неверный. Любой протокол лишь указывает схему об-

мена, это свод правил, которые не переходят из пассивного в активное состояние. Протокол указывается наименованием. Правильный ответ второй.

Третий и четвертый варианты ошибочны. Если нам достался старый браузер, то никакая активизация «на лету» не научит его поддерживать позднейшие продвижения, нужно сознательно скрывать от него нововведения. Для этого в языке существует ряд приемов, среди которых – альтернативные тексты, заключение скриптов в теги, воспринимаемые браузерами как комментарии и т.д.

4.4. *Что такое World Wide Web (Web)?*

Варианты ответов:

1. Новая информационная технология для запросов (заказов), рассылки и получения данных.
2. Гипертекстовая система интерактивного взаимодействия пользователей в реальном масштабе времени.
3. Всемирная централизованная информационно-поисковая система доступа к компьютерным ресурсам.
4. Сеть информационных ресурсов, использующая три механизма: (схема именования ресурсов, протоколы доступа и гипертекст для движения по ним).

Согласно спецификации, World Wide Web – это сеть информационных ресурсов. Для того чтобы сделать эти ресурсы доступными самой широкой аудитории, используются три механизма: схема наименования пути поиска, протоколы доступа к ресурсам, гипертекст для простого перемещения по ресурсам.

Сеть компьютеров не является технологией, как подсказывает неправильный первый вариант. Она может поддерживать технологию, т.е. последовательность операций, но являться ею не может. Второй вариант тоже неверный. Web, безусловно, является системой, и с ее помощью можно «интерактивно» общаться с друзьями и коллегами. Однако это лишь одна из функций сети, а не ее определение или общее назначение. Полная независимость и децентрализация узлов сети позволяет без колебаний отмести третий вариант. Верный ответ, заимствованный из спецификации HTML-4.0, четвертый. Предшествовавшие версии нуждались в довольно громоздких инструкциях, обеспечивающих взаимодействие узлов с помощью согласующих шлюзов, однако теперь их функции выполняются без участия авторов документов.

4.5. *Универсальный идентификатор ресурсов (Universal Resource Identifier), или URI, это*:

Варианты ответов:

1. Указатель ресурсов производительности и памяти компьютера.
2. Указатель тематики ресурсов, содержащихся на стороне сервера.
3. Указатель адреса ресурса, содержащий наименования: протокола доступа, компьютера и документа.
4. Пиктограмма поиска в Web на панели инструментов браузера.

Каждый ресурс в Web – документ, изображение, видеоклип, программа – имеет адрес, который можно закодировать с помощью универсального идентификатора ресурсов *(Universal Resource Identifier – URI*). Адрес состоит из следующих трех частей:

- схема наименования механизма, используемого для доступа к ресурсу;
- имя компьютера, на котором располагается ресурс;
- имя собственно ресурса, заданное в виде пути.

Ресурсы в Интернете информационные, требуют для своей идентификации указания адреса, определителя местоположения, а не номиналов быстродействия и емкости, и поэтому первый вариант неверен.

Второй вариант содержит одно лишнее слово: «тематика». URI ничего не говорит о содержимом документа, это путь к ресурсу. Ответ ошибочный.

Неправилен и четвертый вариант. Пиктограмма, несомненно, указывает на принципиальные возможности поиска ресурсов, но никак их не идентифицирует. Вот если щелкнуть по ней мышью, тогда перед нами распахнется окно запроса. В его поле ввода необходимо вписать тот самый идентификатор, о котором идет речь. Верный ответ третий.

4.6. *Что такое HTML*?

Варианты ответов:

1. Обязательный тег документа, созданного на языке HTML.
2. Международный стандарт для реализации транзакций «клиент-сервер».
3. Протокол передачи данных в сети Интернет.
4. Язык публикации электронных документов, загрузки информации, взаимодействия с удаленными службами, включения приложений в документы.

Вариант первый: в разметке документов, как правило, фигурирует тег <HTML>. Однако все помнят, что тег этот не обязателен. Уже с первой строки (объявление типа документа) программное обеспечение распознает, какому пакету передать на обработку поступающий поток байтов. Этот тег полезен человеку, читающему распечатку текста HTML, а браузер в нем не нуждается.

Второй вариант содержит другую ошибку. На самом деле, HTML – это язык разметки, который использует стандарты, но сам он стандартом не является. Перед ним поставлены другие задачи.

Третий вариант. Протоколом передачи гипертекстов является не HTML, a http – HyperText Transfer Protocol. И ряд других, упомянутых ранее, о которых не следует забывать: ftp, mailto и т.д. Ответ неверный.

Правильный ответ четвертый. HTML – это язык публикации, язык загрузки информации удаленных источников, язык внедрения объектов и других приложений в электронные документы.

4.7. *Интернационализация в HTML 4.0 означает*:

Варианты ответов:

1. Развертывание Web на территориях государств мирового сообщества.

2. Поддержка различных языков в одном документе.

3. Компьютерный перевод текстов с языка исходного документа на язык, задаваемый элементом LANG.

4. Компьютерный перевод текстов с языка исходного документа на язык, задаваемый атрибутом lang.

Стандарт ISO/IEC:10646 узаконил в качестве набора символов для документов HTML алфавиты всех существующих языков. В нем решены вопросы представления национальных символов, направления письма, пунктуации и другие. Определены конструкции (ссылки на символы), позволяющие на любом компьютере ввести в документ любую букву или иероглиф любого языка с помощью стандартной клавиатуры. Реализована поддержка различных языков в одном документе. Это обеспечивает эффективное индексирование документов с помощью ключевых слов для поисковых машин, а также позволяет повысить качество полиграфии, преобразования текста в речь, корректные переносы слов на следующую строку и т.д. Первый ответ неправильный.

Второй вариант верный, хотя и неполный. Понятие интернационализации в HTML включает в себя, кроме сказанного в нем, поддержание разноязычных версий одного документа. Другой упущенный нюанс: синтезаторы речи, встречая в документе фразы на различных языках, воспроизводят их с необходимым прононсом, усиливая эффект присутствия.

Третий вариант рассчитан на недостаток знания. Элемента LANG в HTML пока не существует, равно как и компьютерный перевод не входит в компетенцию языка разметки гипертекста. Отсюда неправомерность и четвертого варианта. Атрибут lang подсказывает агенту пользователя, какой набор символов нужно загрузить для воспроизведения документа, но не более того. Переводом занимаются совсем другие пакеты прикладных программ.

4.8. *Доступность в HTML 4.0 означает:*

Варианты ответов:

1. Наличие механизмов таблиц стилей, скриптов, кадров, объектов.

2. Новые конструкции таблиц с большим количеством возможностей и новые свойства форм.

3. Расширение возможностей для пользователей с физическими недостатками, улучшенный интерфейс, адаптация к новым технологиям.

4. Возможность составлять документы на любом языке и передавать их по всему миру.

Как и предыдущий термин, доступность тоже не относится к числу фундаментальных понятий языка. Можно ежедневно упражняться в общении через Интернет, даже не подозревая об этом свойстве. Однако продвижение всякого товарного продукта на рынке не обходится без рекламы его достоинств, и не принимать в расчет показатели доступности нельзя. Здесь имеют-

ся в виду соображения доступности сети именно для пользователей с физическими недостатками. Доступность – это расширение диапазона целевых устройств воспроизведения документов: синтезаторов речи, тактильных приставок азбуки Брайля.

Первый вариант пропагандирует гибкость представления документа в четвертой версии HTML. Таблицы стилей освобождают нас от скрупулезного оформления каждого элемента. Достаточно один раз установить свой собственный «почерк», и он навсегда станет спутником производимых нами документов. Является ли это синонимом доступности? Если не вспомнить определение термина, то в какой-то мере с этим можно согласиться. Вариант рассчитан на ситуацию домыслов и предположений, а не четкого знания.

Второй ответ исповедует ту же концепцию. Действительно, формы и таблицы помогают нам выбрать из обилия информации те фрагменты, которые нам необходимы, отсеивая все постороннее. Можно вспомнить поисковые машины и тоже назвать их облегчением доступа, но доступностью назвать нельзя, потому что спецификация определяет ее так, как это сформулировано в правильном третьем варианте. Ошибочен и четвертый ответ, тоже рассчитанный на ситуацию домыслов. Возможность составлять документы на любом языке и передавать их по всему миру является не доступностью, а целью развертывания Web.

4.9.*Новые отличительные свойства таблиц в HTML 4.0.*

Варианты ответов:

1. Новая модель таблиц дает возможность структурированного представления массивов однородной информации.
2. Новая модель таблиц дает возможность группировать столбцы и отображать данные по мере приема (не ожидать всю таблицу для ее отображения).
3. Новая модель таблиц дает возможность генерировать название, заголовки и подзаголовки документа и его фрагментов.
4. Новая модель таблиц позволяет форматировать документ.

Модель таблиц действительно позволяет «форматировать, генерировать и структурировать» данные. Правда, не только новая. Конструкции строк, столбцов и ячеек словно созданы для их использования как инструмента упорядоченной подачи материала с рисунками, рубрикацией и другими визуальными эффектами. Многие авторы даже списки и оглавления упаковывают в таблицы. А заодно и абзацы, формулы, чертежи, изменения шрифта. Результатом становится задержка загрузки. Страдает и разбивка таблиц по страницам документа. Как бы тщательно вы ни подгоняли строки под кромки листа, ваш читатель простым изменением масштаба окна может нарушить дизайн оригинала.

Все ответы логически непротиворечивы. Модель «дает возможность». Единственный правильный ответ – второй. Только в новой модели таблиц предусмотрена структурная группировка столбцов и введены элементы горизонтального разделения THEAD, TBODY, TFOOT, помогающие (если гра-

мотно ими пользоваться) браузеру моментально отобразить начало таблицы, не дожидаясь, пока ее содержимое загрузится полностью. Другие варианты говорят о качествах новой модели, унаследованных от предшественников. Следовательно, их нельзя назвать «отличительными», как это определено вопросом.

4.10. *Составные документы в HTML 4.0. формируются с помощью*:

Варианты ответов:

1. Стандартного механизма фреймов и набора фреймов (типы элементов FRAME и FRAMESET).
2. Стандартного механизма фреймов, якорей и ссылок (элементы FRAME, A и LINK).
3. Стандартного механизма скриптов (элемент SCRIPT и его наследники).
4. Стандартного механизма для внедрения иерархии объектов и приложений в документы HTML (элемент OBJECT и его наследники).

По определению составной документ отличается тем, что включает в себя ресурсы, расположенные по разным адресам. Текст может находиться на одном сервере, формулы на другом, фотографии на третьем. Каков же «стандартный механизм формирования» составного документа?

Первый вариант говорит о многооконном представлении. В общем случае в каждое окно загружаются автономные ресурсы. В частном случае они могут оказаться связанными, но для фреймовых конфигураций это несущественно. Важно, что во фреймы можно загружать разнородные источники. Элемент IFRAME из их множества вообще не образует самостоятельного окна, а внедряется в тело документа без фреймов. Как и элемент IMG. Дело в том, что «стандартный механизм» фреймов специально разработан для отображения информации в нескольких окнах. Его задача – одновременная демонстрация разных документов, или их фрагментов, но не постраничное воспроизведение составного документа в одном окне. Следовательно, первый ответ неверен.

Второй вариант. Скомпилировать составной документ без якорей и ссылок невозможно, но их «стандартный механизм» создан совершенно с другой целью: для простого перемещения по ресурсам. Для «выгрузки» изучаемого документа и загрузки на освободившееся место нового, указанного ссылкой. Она указывает путь доступа к ресурсам, а не формирует единый документ из разных источников. С ее помощью мы переходим от одного документа (или фрагмента) к другому, а не преобразуем их в составные части единого целого.

Аналогично с третьим вариантом. Скрипты зачастую бывают полезны при формировании составных документов, но цель их реализации – это *интеллектуальные формы, сценарии, анимированные документы*. Динамическое формирование дополнений, повышающих удобства восприятия человеком текста или другой формы представления информации.

Верный ответ четвертый. Элемент OBJECT и механизм его внедрения

изначально задуманы как средство компиляции составного документа. Относительно «наследников» отметим неоднозначность определения. Здесь имеются в виду как элементы IMG и APPLET, частные функции которых «подхватывает» более общий элемент OBJECT, так и возможность параметрического описания с помощью элемента PARAM свойств конкретного экземпляра объекта.

4.11. *Таблицы стилей в HTML 4.0. представляют собой*:

Варианты ответов:

1. Механизм управления структурой документа – блоками, конструкциями, встроенными элементами, абзацами, фразами.
2. Механизм управления представлением документов – шрифтами, выравниванием, цветами и т.д.
3. Механизм управления структурой таблиц – заголовками, группированием строк и столбцов, шириной и высотой ячеек.
4. Комбинацию элементов (с их атрибутами) построения таблиц.

Тип элемента TABLE (таблица) и конструкция HTML «таблица стилей» (StyleSheets) сходны лишь по созвучию в русском переводе. Если вы представляете себе разницу между структурой документа и его оформлением, то выбор ответа не составит затруднений. Правдоподобие первого варианта поверхностно. Оно основано на общеизвестном свойстве: любая таблица структурирует данные. Однако таблицы стилей не только не управляют структурой, но даже не являются таблицами. Это – перечень установленных значений атрибутов. И управляют они не структурой, а представлением элементов.

Верный второй вариант утверждает следующее. Мы фиксируем в таблице стилей приемлемые для наших целей значения атрибутов. Объявляем их принадлежность целому классу элементов. И вместо того, чтобы описывать в начальных тегах интересующих нас экземпляров элементов необходимые свойства, указываем лишь их принадлежность введенному классу.

Третий вариант смешивает детали управления структурой и оформления. Согласно ему, таблицы стилей управляют заголовками, группированием, размером ячеек. Если интерпретировать вариант буквально, то получим следующее. С помощью таблиц стилей можно управлять свойствами заголовков, объединением строк и столбцов, высотой и шириной ячеек. Для исключения такого толкования в текст сознательно введено дополнение: «управление структурой таблиц». Другими словами, управление *элементами* CAPTION, THEAD и TFOOT, элементами COL и COLGROUP, но никак не *атрибутами* span, или align, или других. Вариант неверный. Таблицы стилей устанавливают значения атрибутов, а не структуру элементов.

Четвертый вариант называет таблицы стилей комбинациями элементов построения таблиц. Его нелепость очевидна: таблица в HTML задается элементом TABLE, а не комбинацией параметров. Модель содержимого элемента TABLE включает в себя ряд других элементов, но формально они не представляют собой комбинации. Таблицы стилей отличаются от комбинаций па-

раметров свободой использования. Они сосредоточивают в себе не жесткое объявление параметров в определении типа документа DTD, а описание свойств частного класса экземпляров элементов, указываемое авторами документов.

4.12. *Назначение скриптов в HTML*:

Варианты ответов:

1. С помощью скриптов авторы могут создавать динамичные страницы.
2. С помощью скриптов авторы могут производить дистанционное обучение языкам программирования.
3. Скрипты необходимы пользователям для организации доступа к системам управления базами данных.
4. Скрипты организуют устойчивые переходы по гиперссылкам.

Правильный первый вариант объявляет, что скрипты предназначены для создания динамичных Web-страниц. Динамические страницы отличаются от статических тем, что содержат программируемые элементы. Можно заставить элементы менять цвет и размеры при попадании на них курсора мыши, или вообще скрывать или делать их видимыми в зависимости от действий пользователя. Можно выполнять поиск в библиотеках. Мы посылаем на сервер запрос, сформулированный нами словесно в полях специальной формы. Например, прислать хранящиеся в базе данных статьи, опубликованные автором X в журнале Y в период с N-го года по настоящее время. Программа обработки формы предоставляет скрипту, формирующему SQL-запрос, указанные параметры, и в ответ генерирует страницу с таблицей, содержащей необходимые данные.

Второй вариант: дистанционное обучение языкам программирования. Если бы процесс обучения сводился к поиску нужной информации, то можно было бы согласиться с этим утверждением. Добавив еще одно допущение: компиляторы языков высокого уровня являются экзаменаторами, выставляющими оценки (сообщения компилятора). И что эти средства включены в скрипты, дистанционно управляющие учебным процессом. – Очевидно, что эта функция не входит в компетенцию языка разметки гипертекстов. Ответ неверный.

Третий вариант – доступ к данным – рассчитан на невнимательность. Да, с помощью скриптов мы организуем SQL-запросы и «на лету» формируем для клиента выжимку из БД. Но в варианте речь идет о доступе к системе управления базой данных, а не к ее начинке. Причем утверждается, что доступ организует не автор документа, а пользователь. Вариант неверный.

Четвертый вариант. Конечно, скрипты добавляют интеллектуальности в переходы по гиперссылкам форм или объектов. Но стабильнее переходы от этого не становятся. Понятие «переход» вообще плохо вяжется с устойчивостью. Скорее, это символ изменчивости. Ответ неправильный.

4.13.*Укажите ответ, который НЕ ЯВЛЯЕТСЯ характеристикой приложения SGML*:

Варианты ответов:

1. *SGML*-объявление (указывает, какие символы и разделители могут отображаться в данном приложении).

2. *SGML*-определение типа документов (DTD – определяет синтаксис конструкций разметки).

3. *SGML*-спецификация (описывает семантику разметки).

4. Система определения языков *SGML*-разметки (содержит информацию о структуре, представлении и семантике в документе).

Как известно, приложением SGML является любой язык разметки, определенный в этом стандарте. Вариант первый указывает, какие символы и разделители могут отображаться в конкретном приложении. Является ли такое объявление характеристикой языка? Да, безусловно, является. К тому же выводу приходим во втором и третьем вариантах. Да, определение типа документа и спецификация языка являются его характеристиками. А вот четвертый вариант выходит за рамки приложения, он говорит о системе определения любого возможного, а не конкретного языка разметки.

Здесь прямая аналогия с любыми языками: с алгоритмическими, с математикой, с разговорными. Может ли служить характеристикой языка романа «Война и мир» наука лингвистика? Или литературоведение? Четвертый вариант оперирует схожими категориями. Если колебания остались, пойдем с другой стороны. Является ли характеристикой языка, которым написан роман, имя этого языка? Или, как в первом ответе, алфавит и знаки препинания? Или, как во втором, синтаксис и семантика? Если искать нужно неверный ответ, тогда правильный вариант четвертый. Правильный, потому что ответ неверный.

4.14. *Укажите вариант ответа, дающий НЕВЕРНУЮ характеристику элементов HTML*:

Варианты ответов:

1. Объявление типа элемента обычно включает три части: начальный тег, содержимое и конечный тег.

2. Элемент может не иметь содержимого.

3. Встроенные элементы могут содержать только данные, а элементы уровня блока могут содержать данные и встроенные элементы.

4. Элемент может не иметь ни открывающего, ни закрывающего тегов.

Первый вариант дает классическую характеристику объявления типа элемента. Можно усомниться в точности обстоятельства «обычно», или разъяснить, что объявление типа включает модель содержимого, а не собственно содержимое элемента. Но в целом ответ дает верную характеристику.

Второй вариант. Да, ряд элементов HTML характеризуются моделью содержимого EMPTY, они принципиально пустые. Добавим, что есть группа других элементов, в которых допускается строго определенный тип содержимого, которое может либо присутствовать, иногда – произвольное число раз, либо отсутствовать вовсе. И тогда экземпляр элемента не имеет содержимого.

Верный «неверный» вариант третий. Вопреки ему, встроенные элемен-

ты *могут* содержать другие встроенные элементы и данные. Элементы уровня блока *могут* содержать другие элементы уровня блока, и встроенные элементы, и данные. Уровень блока – это абзац, таблица, объект. Все, что сосредоточивает в себе законченное описание. Встроенные элементы – это отдельные текстовые акценты и штрихи, делающие документ более удобным для восприятия.

Четвертый вариант нуждается в уточнениях. Какая группа элементов может не иметь обоих тегов? Какая – только закрывающих? В какой – закрывающий тег запрещен по определению? – А в целом – ошибки нет.

4.15. *Укажите назначения элемента TITLE и атрибута title*:

Варианты ответов:

1. Элемент TITLE определяет заголовок документа, атрибута title нет.
2. Атрибут title элемента HEAD определяет заголовок документа, элемента TITLE не существует.
3. Элемент TITLE определяет заголовок документа, атрибут title обычно используется для отображения всплывающих подсказок при попадании курсора мыши в поле элемента HTML.
4. Элемент TITLE определяет заголовок документа, атрибут title элементов H1, H2, H3, H4, H5, H6 определяет заголовки разделов документа.

Согласно спецификации, авторы должны использовать элемент TITLE для идентификации содержимого документа. Элемент не считается частью текста и располагается в разделе HEAD. Он отображается в качестве заголовка страницы или окна. В документе должен быть ровно один экземпляр элемента TITLE. В свою очередь, атрибут title может сопровождать любое количество элементов. Визуальные браузеры часто отображают его текст как подсказку (краткое сообщение, которое появляется, если курсор мыши попадает на объект). Синтезаторы речи могут проговаривать значение title.

Первые два варианта неверные, они утверждают, что HTML не поддерживает одновременно элемент и атрибут с именем TITLE, что в языке используется лишь один из них. Верный третий ответ резюмирует приведенную выше цитату из спецификации. Четвертый вариант неправильно определяет атрибут title.

4.16. *Ссылки на символы в HTML (укажите НЕВЕРНЫЙ ответ)*:

Варианты ответов:

1. Начинаются со знака "&" и заканчиваются точкой с запятой (;).
2. Являются подмножеством значений атрибута *link*.
3. Представляют собой числовые или символьные имена символов.
4. Удобны для обращения к редко используемым символам или к символам, которые трудно вводить в средствах разработки документов.

Первый вариант представляет собой цитату из спецификации HTML, следовательно, является верным, однако, он содержит в себе повод для дополнительного вопроса. Замыкающая точка с запятой в синтаксисе ссылки на символы не является обязательной. Она помогает нам визуально выделить ее

код в распечатке, но агентами пользователя не используется.

Второй вариант подразумевает знание русского перевода термина link – соединение, звено цепи, ссылка. В этом случае возникает ассоциация с понятием «ссылка на символ». Как результат, нетрудно заключить, что ссылка – это подмножество значений понятия link. Если не знать, что атрибут *link* устанавливает своим значением цвет отображаемой на экране гиперссылки, который задается в языке либо зарезервированным именем, либо шестнадцатеричным кодом.

Третий вариант, как и первый, заимствован из первоисточника. Синтаксис ссылки: лидирующий символ амперсанта “&”, решетка “#” и десятичный номер нужного символа Unicode. Если номер шестнадцатеричный, сразу после решетки следует поставить метку ‘x’. Комбинации ссылок на символы содержат мнемонические подсказки. Букву шведского алфавита å записывают “&aring”.

Не вызывает сомнений достоверность четвертого варианта. Ссылки на символы задуманы в языке разметки именно «для обращения к редко используемым символам или к символам, которые трудно (невозможно) вводить».

Верный вариант (содержащий ошибочный тезис) второй.

4.17. *Укажите назначения элемента LINK и атрибута link*:

Варианты ответов:

1. Элемент LINK не имеет содержимого, он определяет связь; атрибута *link* не существует.

2. Атрибут *link* элемента HEAD определяет заголовок документа, элемента LINK не существует.

3. Элемент LINK может присутствовать только в разделе HEAD документа, хотя может присутствовать неограниченное число раз; атрибут *link* обычно используется для указания цвета отображения активных, посещенных и неиспользованных ссылок.

4. Элемент LINK может представляться различными способами (например, в виде панели с выпадающим списком ссылок); атрибут *link* обычно используется для указания адресов связанных ресурсов.

Первый вариант неправильный, потому что атрибут *link* существует. Второй – потому что заголовок (элемент HEAD) не поддерживает атрибут *link*, не бывает в заголовке отображаемых гиперссылок. Элемент LINK предназначен для указания места текущего документа в упорядоченной подборке сгруппированных документов. Именно поэтому LINK выносится в заголовок. Для того, например, чтобы распознавать, не вникая в документ, каким именно «звеном цепи» он является в линейной последовательности родственных ресурсов.

Третий вариант настораживает выражением «неограниченное число», несвойственным для технических текстов. На самом деле ограниченным является множество возможных описаний положения документа в группе. Точнее говоря, элемент LINK может присутствовать в заголовке столько раз, сколькими способами можно описать связи документа в сборнике. Ответ

правильный.

Последний вариант неверен, потому что атрибут *link* используется не для указания адресов, а для изменения цвета подкраски гиперссылок в теле документа. Для различения активных, посещенных и неиспользованных ссылок. Его значения представляются не комбинацией *%uri*, а указателями цвета *%color*.

4.18. *Укажите ответ, дающий НЕВЕРНУЮ характеристику комментария HTML 4.0*:

Варианты ответов:

1. Строка символов переноса ("---") в комментарии не является ошибкой.

2. Комментарии в HTML имеют следующий синтаксис:

<!-- это комментарий -->, <!-- это тоже комментарий, он занимает несколько строк -->

3. Пробелы между открывающим разделителем разметки ("<!") и открывающим разделителем комментария ("--") недопустимы, но их можно использовать между закрывающими разделителями комментария ("--") и разметки (">")

4. Информация в комментариях не имеет специального значения (например, ссылки на символы не интерпретируются).

Первый вариант. Два следующих подряд символа переноса интерпретируются агентом пользователя как начало или окончание комментария, и если такая пара встретится внутри него, то логика обработки неизбежно переключится на распознавание в потоке байтов очередной HTML-конструкции. Тот самый вариант, «дающий НЕВЕРНУЮ характеристику», который мы ищем. Однако просто так очевидные ошибки в билеты не вводятся, они рассчитаны на уточнения, помогающие оценить уровень вашей подготовки. Допустим, вы записали:

```
<!ELEMENT BODY O O (%block;|SCRIPT)+ +(INS|DEL) --тело документа-
- >

<!ATTLIST   BODY
    %attrs;                                  -- %coreattrs; %i18n; %events; --
    onload      %Script;     #IMPLIED   -- документ загружен --
    onunload    %Script;     #IMPLIED   -- документ удален    --
>
```

Посчитаем, сколько раз между открывающей и закрывающей угловыми скобками списка атрибутов встречается строка из «двух подряд символов переноса». Как же браузер справляется с этим? Тот же подсчет способен смутить нас и со вторым вариантом. Комментарии к атрибутам в нем не заключены в скобки. А в варианте утверждается обратное, будь то одна строка пояснений или несколько. Что же записано правильно, ответы или примеры?

Третий ответ, та же ситуация: в примере достаточно много пробелов между открывающим разделителем разметки ("<!") и открывающим раздели-

телем комментария ("--"). Пробелы между конструкциями при анализе выбрасываются агентами пользователя, а в комментариях остаются. Этот прием сознательно используют для сокрытия скриптов. Браузеры первых лет, не способные обрабатывать программные коды, принимают их за комментарии и пропускают, не анализируя. Потому-то и справедливо утверждение третьего варианта.

Четвертый вариант: что значит «специальное значение»? – Так называют инструкцию браузеру по обработке, а не подсказку (комментарий) читателю разметки HTML. Кстати, и во втором ответе речь идет, читайте внимательно, о синтаксисе собственно комментария, а не о включении пояснений в строку размеченного текста. Говорится о самостоятельной конструкции, об аналоге уровня блока, а не о «встроенной» подсказке.

Верный вариант первый. Он дает неверную характеристику комментария.

4.19. *Определение комбинации параметров* HTML:

Варианты ответов:

1. Представляет собой макрос, на который можно ссылаться в DTD.
2. Комбинация параметров разворачивается в строку параметров и в таком виде отображается в документах HTML.
3. Определение комбинации параметров содержит три элемента; оно начинается с ключевого слова `<!ENTITY %`, продолжается строкой, в которую она разворачивается, и заканчивается обязательным символом ";".
4. Строка, в которую разворачивается комбинация параметров, не может содержать другие имена комбинаций параметров.

Рассмотрим пример комбинации параметров:

```
<!ENTITY % block
     “P | %heading; | %list; | %preformatted; | DL | DIV | NOSCRIPT |
    BLOCKQUOTE | FORM | HR | TABLE | FIELDSET | ADDRESS”>
```

Видим макрос и понимаем, что сослаться на него можно в любом месте, т.е. первому варианту можно верить. Развернули комбинацию в строку и заключили ее в кавычки, однако в таком виде сама собой она никогда, без сознательных усилий, не «отображается в документах HTML», значит, второй вариант отвергается. Мы начали с ключевого слова `<!ENTITY %`, продолжили строкой, но не закончили якобы «обязательным символом ";"», значит, третий вариант тоже нас не устроит. Наконец, три параметра заданы нами как «другие имена комбинаций параметров», и в этом – ошибка четвертого варианта.

Верный вариант первый.

Добавим, что сознательно привели в качестве примера эту комбинацию параметров. Никто из вас не ответил на экзамене, что такое элемент уровня блока. Надеемся, теперь это сделает каждый. Для полноты картины развернем встреченные в примере другие имена комбинаций параметров:

```
<!ENTITY % heading “ H1 | H2 | H3 | H4 | H5 | H6 “>
```

<!ENTITY % list “ UL | OL “>
<!ENTITY % preformatted “ PRE “>

В пояснениях к тесту 4.14 приведено неформальное определение элемента уровня блока. Здесь они перечислены без анализа их отличительных качеств.

4.20. *Укажите НЕВЕРНУЮ характеристику типа элемента*:

Варианты ответов:

1. Начинается с ключевого слова <!ELEMENT, заканчивается символом >.

2. Имя типа элемента обязательно указывается в его объявлении.

3. Два символа переноса после имени элемента означают, что начальный и конечный теги не являются обязательными. Один символ переноса, за которым следует буква "O", указывает, что начальный тег можно опустить. Две буквы "O" указывают, что можно опустить как начальный, так и конечный теги.

4. Модель содержимого для пустых элементов объявляется при помощи ключевого слова "EMPTY".

Первый вариант утверждает: «начинается с "<!ELEMENT" и заканчивается символом ">"». Действительно, именно таковы формальные признаки определения типа элемента. В таком виде они представлены во всех DTD – определениях типа документа.

Второй вариант. Нормально ли, если «имя типа элемента обязательно указывается в его объявлении»? Безусловно, да. Это самый простой способ идентификации. По указанному имени агент пользователя адресуется к DTD, чтобы выяснить допустимую модель содержимого и корректно интерпретировать описание встреченного в документе экземпляра элемента.

Переходим к следующему варианту. Первая фраза убеждает нас: символ переноса означает, что тег необязателен. Вторая фраза ей не противоречит: перенос по-прежнему символизирует необязательный тег, буква ‘O’ означает, что тег обязателен. Заключительная фраза выворачивает все наизнанку: буква ‘O’ означает теперь, что тег можно опустить. Противоречие очевидное. В четвертом варианте видим прямое соответствие нашего «пустого» иноязычному "EMPTY".

Верный вариант, как мы установили, третий.

4.21. *Укажите НЕВЕРНУЮ характеристику модели содержимого*

Варианты ответов:

1. Модель содержимого элемента указывается без использования синтаксических конструкций вида (…), A|B, A&B, A*, A+, A?, A,B.

2. Модель описывает, что может содержаться в экземплярах элемента.

3. Модель описывает имена допустимых и запрещенных типов элементов.

4. Определения модели содержимого могут включать: текст документа ("#PCDATA"), комбинации DTD, ссылки на символы и комбинации симво-

лов.

В любом DTD в объявлениях типов элементов встречаются указанные в первом варианте конструкции. Два примера:

```
<!ELEMENT MAP – – ((%block;)+ | AREA+) -- клиентская карта -- >
<!ELEMENT TABLE – –
(CAPTION?, (COL* | COLGROUP*), THEAD?, TFOOT?, TBODY+)>
```

В первом примере модель содержимого элемента заключена в круглые скобки, в ней присутствуют конструкции A|B и A+. Во втором дополнительно представлены конструкции A*, A?, A,B. Не охваченной осталась A&B, попробуйте самостоятельно вспомнить, где она встречается.

Второй вариант тоже не дает поводов сомневаться. Модель содержимого для того и предназначена, она «описывает, что может содержаться в экземплярах элемента». Это по определению «допустимое содержимое элемента, называемое *моделью содержимого*». Агент пользователя сверяется с ней и воспроизводит в документе. Обратимся к приведенным примерам. В обоих случаях перед нами исчерпывающее описание возможной начинки элемента. Клиентская карта – это хотя бы один элемент уровня блока, либо заданная координатами область экрана, либо то и другое вместе. Таблица может содержать «шапку» – заголовок, а может и не содержать его. Она может многократно включать объединения свойств столбцов и структурное разделение столбцов на группы, либо и то, и другое вместе, либо не иметь ни того, ни другого. В таблице могут предусматриваться верхний и нижний заголовки, в ней должен присутствовать хотя бы один раздел горизонтального объединения – группа строк.

Третий вариант рассмотрен выше (якорь не может содержать якорь, форма – форму, абзац – абзац). Запрещенные моделью содержимого типы элементов помечаются знаком '-'. Например:

```
<!ELEMENT LABEL – – (%inline;)* –(LABEL) -- текст метки поля формы -- >
```

Наконец, определения модели содержимого, действительно, могут включать все, что перечислено в четвертом ответе.

Верный «неверный» вариант первый.

4.22. *Объявления атрибутов в HTML 4.0*:

Варианты ответов:

1. Объявление атрибутов, которые может иметь элемент, начинается с ключевого слова <!ATTLIST. За ним следуют имя элемента, список определений атрибутов и закрывающий символ >.

2. Указание имен типов элементов в объявлении атрибутов необязательно.

3. Тип атрибута объявляется явным набором его допустимых значений, которые учитывают регистр.

4. Объявление атрибута не задает его значение по умолчанию только в случаях, когда это объявление не связывается с конкретными элементами.

Первый вариант формализует возможный способ определения атрибутов, т.е. свойств элемента. Утверждается, что в DTD список атрибутов отыскивается по ключевому слову, что следом указывается имя типа элемента, который поддерживает перечисляемые атрибуты, что завершается объявление закрывающим символом. Все строго и логично. Ответ верный.

Второй вариант предлагает объявлять атрибуты независимо от элементов. Принципиально такой способ возможен, но он требует введения в DTD нерациональной избыточности. В любом случае тип элемента должен описываться его свойствами, даже если определены они в другом месте. При этом любой атрибут может оказаться обязательным (как *name* в PARAM, *alt* в MAP) для одной группы элементов и необязательным для другой (как *name* в OBJECT, *alt* в INPUT). В HTML эта избыточность не вводится. Атрибуты по определению суть свойства тех элементов, которые их поддерживают. Никак не объявишь их, если не известно, чьи они. Многие атрибуты способны характеризовать разные элементы, но в DTD они всякий раз повторяются как строки в индивидуальных списках типов элементов. Ответ неправильный.

Третий ответ содержит противоположную ошибку. Предлагается лишь небольшая часть возможных способов объявления типа атрибута, причем та самая, для которой учет регистра безразличен. Вот примеры:

align (left | center | right | justify | char)
dir (ltr | rtl)
shape (rect | circle | poly | default).

Четвертый вариант. Известно, что SGML-объявления атрибутов вводятся списком, в котором обязательно указывается тип элемента, поддерживающего перечисленные атрибуты. Одного этого достаточно, чтобы квалифицировать вариант как неверный. Второй ошибкой является предубеждение, будто объявления атрибутов предназначены для установления их значений по умолчанию. На самом деле это справедливо только для фиксированных значений атрибутов объединения (например, *span*), в определениях которых сознательно задано единичное фиксированное значение (#FIXED). Вариант неверный.

4.23. *Комбинации DTD в определениях атрибутов*:

Варианты ответов:

1. Определения атрибутов не могут содержать комбинации ссылок.
2. Комбинации параметров не могут быть вложенными (не могут содержать другие комбинации параметров).
3. Комбинации параметров определены для всех элементов HTML.
4. DTD определяет комбинацию "%URI;" расширением строки "CDATA": <!ENTITY % URI "CDATA" -- Универсальный идентификатор ресурсов -->

Первый вариант неверный. Комбинации ссылок (мнемонические ссылки) допускаются для указания значений ряда атрибутов. В случаях, когда эти значения содержат кавычки или другие управляющие символы языка, их

употребление предпочтительнее явного указания.

Аналогично по второму варианту: «комбинации комбинаций разворачиваются рекурсивно». Следовательно, они «могут быть вложенными». Примеры использования приведены выше: *%attrs; %coreattrs; %i18n; %events;*.

Ошибка третьего ответа в том, что комбинации определены не для всех типов элементов. BASE, к примеру, вполне обходится без них.

Правильный вариант четвертый. В нем содержится объявление комбинации параметров %URI. Ответьте на более общий вопрос: а какие еще комбинации расширяются той же строкой символов "CDATA".

4.24. *Логические атрибуты в HTML 4.0.*

Варианты ответов:

1. Логические атрибуты представляются именами, например: <OPTION selected="selected">; минимизированной формы <OPTION selected> нет.

2. В HTML 4.0 логические атрибуты предназначены для организации переходов и циклов в скриптах.

3. Наличие логических атрибутов в начальном теге элемента подразумевает, что значением атрибута является "истина". Их отсутствие означает "ложь". Многие агенты пользователей распознают только минимизированную форму логических атрибутов и не распознают полную.

4. Логические атрибуты представляются значениями *true* или *false*.

Первый вариант неверен благодаря ненужному отрицанию. Минимизированная форма <OPTION *selected*>, как и у других логических атрибутов, существует. Более того, в разделе 2 мы объявили, что ее применение предпочтительно, так как не все браузеры воспринимают полную форму.

Второй вариант ошибочен. В разветвляющихся программах выполнение циклов и переходов управляется результатами проверки логических условий, и скрипты наследуют эту схему. Логические атрибуты здесь не при чем – это переключатели свойств типов элементов. В конкретном экземпляре элемента они подтверждают, либо исключают объявленное в DTD свойство типа элемента.

Верный вариант третий. Он представляет цитату из спецификации.

Четвертый вариант, как и второй, рассчитан на небрежную аналогию с элементами смежных дисциплин.

4.25. *Укажите среди перечисленных НЕВЕРНУЮ формулировку*:

Варианты ответов:

1. Автор – это человек или программа, пишущие или генерирующие документы в формате HTML. Средство разработки – это отдельный случай автора, а именно программа, генерирующая код HTML.

2. Пользователь – это человек, взаимодействующий с агентом пользователя для просмотра, прослушивания или другого использования документа.

3. Агент пользователя – это любое устройство, интерпретирующее документы в формате HTML. Агенты пользователя – это визуальные браузеры, невизуальные браузеры (аудио, Брайля), поисковые машины и т.д.

4. Нежелательный элемент или атрибут – это объект HTML, использующий внешние таблицы стилей для представления текстовых документов.

Первые три ответа содержат формулировки, заимствованные из спецификации HTML. Правильный вариант (неверная формулировка) четвертый. «Нежелательный элемент или атрибут – это объект HTML, устаревший вследствие появления новых конструкций». И чаще всего в роли "новых конструкций" выступают именно «таблицы стилей для представления текстовых документов», причем не только внешние, но и объявленные в документе.

4.26. *Набор символов документа HTML включает в себя*:

Варианты ответов:

1. Репертуар, т.е. набор абстрактных символов (латинская буква "A", кириллическая буква "I", китайский иероглиф "вода") и коды, т.е. набор целочисленных ссылок на символы репертуара.
2. Средства кодировки символов «на лету» для выполнения запросов агентов пользователей (транскодирование).
3. Последовательность байтов из набора символов ASCII репертуара универсального набора символов UCS, определенного в [ISO10646]. Этот стандарт определяет репертуар тысяч символов, используемых во всем мире.
4. Способ преобразования последовательности байт в символы.

Верный первый вариант утверждает, что набор символов документа содержит в себе все те и только те символы, которые в этом документе встречаются. Даже если эти символы представлены не своим начертанием, а ссылкой на него. Против такой формулировки трудно найти возражения, она тривиальна.

Второй вариант содержит слова и выражения, порознь представляющие конкретный технический смысл. Однако в своей совокупности они этот смысл утрачивают. Можно согласиться с тем, что некий набор символов служит одним из средств кодировки других символов. Уже упоминалась азбука Морзе, в которой всего два символа – точка и тире – кодируют любой символ алфавита. Но для выполнения этого назначения не важно, исполняется ли кодировка предварительно или «на лету». Не важно, выполняются ли при этом запросы агента пользователя или любое другое преобразование информации.

Третий вариант: в безупречное определение вкраплена популярная аббревиатура ASCII, искажающая смысл цитаты. Удалите ее – и все станет на свои места. Результат окажется более точным, чем формулировка первого варианта.

Четвертый вариант объявляет множество символов методом преобразования. Очевидно, что любой символ может являться либо исходной величиной, либо результатом, но никак не способом преобразования. Ответ ошибочный.

4.27. *Укажите вариант, НЕ ПРЕДСТАВЛЯЮЩИЙ тип данных спецификации HTML 4.0*:

Варианты ответов:

1. Информация о регистре.
2. Дата и время.
3. Целое без знака.
4. Данные сценариев.

Все варианты теста 4.27 представляют типы данных компьютерных языков, есть среди них универсальные, поддерживаемые всеми языками, есть специальные. В HTML не предусмотрены вычислительные процедуры, и нет нужды в типе «целое без знака». Верный вариант третий. Целое без знака не представляет собой тип данных языка разметки гипертекста.

4.28. *Глобальная структура документа HTML определяется как последовательность*:

Варианты ответов:

1. Элементы HTML, HEAD, BODY.
2. Информация о версии HTML, элементы HEAD и BODY.
3. Элементы HEAD, FRAMESET и BODY.
4. Элементы TITLE, HEAD и BODY.

Первый вариант неправильный, потому что не определяет последовательность. На самом деле элементы HEAD (заголовок) и BODY (тело документа) вложены внутрь элемента HTML (страница). Другими словами, страница состоит из заголовка, содержащего общую информацию о документе, и из размеченного текста, упакованного в тело документа.

Правильный второй вариант свободен от этого недостатка. Перед нами строгая последовательность: указание операционной системе, какой программный продукт и как будет обрабатывать поступающий поток байтов, далее – общая информация для агента пользователя о настройках, необходимых для успешного представления документа, и, наконец, воспроизводимое содержимое.

Третий вариант неверный. Элементы BODY и FRAMESET не могут составлять конкатенации, потому что они взаимоисключающие. BODY – это чистый лист для оформления однооконного документа, FRAMESET – это разметка чистого листа для представления документа в нескольких окнах. Так установлено в HTML: документ может представляться либо в одном, либо в нескольких окнах, и для достижения этих разных целей предусмотрены разные элементы.

В четвертом варианте вложенный (по определению) TITLE удален из оболочки HEAD и ошибочно преподносится как самостоятельный элемент глобальной структуры документа.

4.29. *Информация о языке и направлении текста задается в HTML*:

Варианты ответов:

1. Атрибутами lang и dir.
2. Атрибутом lang и элементом DIR.
3. Элементом LANG и атрибутом dir.

4. Элементами LANG и DIR.

Варианты теста 4.29 сосредоточивают допускаемые здравым смыслом сочетания начальных букв слов language (язык) и direction (направление). Коротко рассмотрим их. Элемент LANG пока не существует, следовательно, два последних утверждения неверные. Элемент DIR (второй ответ) существует, но определяет многостраничный список каталогов (directory). Никакого отношения ни к языку, ни к направлению текста он не имеет. Правильный вариант первый.

4.30. *Укажите тип элемента HTML 4.0, НЕ ПОЗВОЛЯЮЩИЙ форматировать абзацные отступы в тексте*:

Варианты ответов:

1. Элемент TABLE (таблица документа HTML).
2. Элемент BLOCKQUOTE (двойные кавычки документа HTML).
3. Элемент P (абзац документа HTML).
4. Элемент PRE (предварительно форматированный фрагмент документа).

Многие авторы используют TABLE для форматирования документов или фрагментов, а элемент-контейнер PRE по замыслу предназначен для этой цели. Добавьте в строку таблицы слева ячейку с шириной, равной отступу, отмените отображение сетки – и абзацный отступ подготовлен. Менее определенная ситуация с PRE. Разные браузеры настроены на различную ширину пробелов и табуляции, и по-разному воспроизводят задуманный нами отступ. О типе элемента BLOCKQUOTE в брошюре не упоминалось, но каждый знает, что он определен как длинная цитата, которая обычно воспроизводится с отступом. И даже без кавычек, чтобы удобнее было с его помощью форматировать абзацы.

А вот абзац (тип элемента P) отступа не создает. Считается, что наличие отступа не повышает информативности текста, и пользователю он не нужен. Цитируем более общий тезис спецификации HTML. «Организация информации в абзацы не влияет на смысл абзаца: абзацы с двойным выравниванием содержат те же мысли, что и абзацы с выравниванием влево».

Правильный ответ третий.

4.31. *Укажите НЕВЕРНОЕ утверждение об элементах представления списков в HTML*:

Варианты ответов:

1. Элемент DIR предназначен для создания многостраничных списков каталогов. Элемент MENU предназначен для использования в списках меню, состоящих из одного столбца.

2. Элементы DL для представления списков определений состоят из двух частей: термина и определения. Термин обозначается с помощью элемента DT и может иметь только встроенное содержимое. Описание указывается с помощью элемента DD, имеющего содержимое уровня блока.

3. Упорядоченные (OL) и неупорядоченные (UL) списки генерируются одинаково за исключением того, что элементы упорядоченных списков ну-

меруются, а элементы неупорядоченного списка не нумеруются.

4. Списки могут быть вложенными (и тогда строки сдвигаются вправо соответственно уровню вложенности), но комбинации списков разных типов (например, OL и UL), а также пустые списки (без элементов LI) не допускаются.

Вариант первый претензий не вызывает, формулировка верная. Как обычно, тривиальность вопроса предполагает уточнения: являются ли элементы DIR и MENU устаревшими или нежелательными и почему? Чем отличается их генерация агентами пользователя от генерации неупорядоченных списков UL?

Второй вариант утверждает, что элемент DD имеет содержимое уровня блока. Уточним, что уровень блока допускает встроенные элементы. Модель содержимого типа элемента DD описывается комбинацией (*%flow;*)*, которая разворачивается в «*%block;*» и «*%inline;*», причем их может оказаться сколько угодно, в том числе и ноль. В целом ответ правильный.

Третий вариант цитирует спецификацию. Ответ верный.

Правильный (неверный) вариант четвертый. Ошибка исчезнет, если удалить утверждение, будто бы «комбинации списков разных типов не допускаются» на всех уровнях вложенности. Допускаются, причем сознательно. Классический пример – кулинарный рецепт, представленный охватывающим списком определений, в термины которого (ингредиенты, процедура, примечания) вложены упорядоченный и неупорядоченный списки. Неупорядоченный содержит перечень ингредиентов, помеченных одинаковыми метками (например, кружком). Это подчеркивает равноценность компонент списка. Упорядоченный список описывает процедуру приготовления теста и выпечки, где важна нумерованная последовательность действий. Примечание, в силу необязательности сведений, обычно содержит пояснительный текст.

4.32. *Модель таблиц HTML позволяет авторам*:

Варианты ответов:

1. Представлять с помощью атрибутов dir и valign текст в вертикальном направлении (сверху вниз или снизу вверх).

2. Упорядочивать данные – текст, форматированный текст, изображения, ссылки, формы, поля форм, другие таблицы и т.д. – в строки и столбцы ячеек.

3. Использовать ячейки таблиц как целевые кадры для фреймов с помощью атрибутов frame и rules в случаях, когда агенты пользователя не сконфигурированы для поддержания фреймов.

4. В модели таблиц HTML атрибута frame не существует; существует элемент FRAME для резервирования заданной части экрана в целях отображения документа или его именованного фрагмента.

Вариант первый. Атрибут *dir* регулирует горизонтальное направление текста: слева направо или наоборот. Атрибут *valign* управляет величиной отступа от верхней кромки ячейки. Ни тот, ни другой не дают возможности

представлять текст в вертикально. Заметим, что ввести в ячейку строчку снизу вверх и обратно все же можно. Используйте элемент OBJECT. Ответ неправильный.

Верный второй вариант исходит из способности таблиц структурировать информацию. Важнейшее свойство – упорядоченное представление двумерных массивов, результатов эксперимента, отчетности. В ячейки таблиц можно вводить заранее отформатированные данные для достижения необходимых визуальных эффектов. Изображения и якоря тоже легко встраиваются внутрь таблиц. В то же время, наложение на форму скрытой таблицы позволяет без дополнительных усилий выравнивать поля ввода и метки по горизонтали и вертикали.

Третий (ошибочный) вариант проверяет наше понимание различий между обычными и многооконными документами. Предлагается следующая схема. Если в ячейки можно вставлять рисунки, то не попробовать ли использовать их в качестве фреймов и загружать в них автономные ресурсы. Обратите внимание на способ реализации идеи. Атрибуты *frame* и *rules* отвечают за наличие рамки и сетки таблицы, а не за содержимое ячеек, следовательно, ничем помочь не смогут. И последнее: «когда агенты пользователя не сконфигурированы для поддержания фреймов», собрать воедино независимые ресурсы можно только с помощью механизма внедрения объектов. Это элементы OBJECT и IFRAME.

Четвертый вариант неправильный: атрибут *frame* своими значениями устанавливает граничную рамку таблицы, следовательно, он существует, и ответ неверный. Элемент FRAME никакого отношения к таблицам не имеет. И задача его состоит не в резервировании «части экрана», этим занимается FRAMESET, а в описании источника и свойств загружаемого ресурса.

4.33. *Ссылки и якоря в HTML 4.0*.

Варианты ответов:

1. Ссылка имеет два конца, называемых якорями (anchors), и направление. Ссылка начинается в "исходном" якоре (источнике) и указывает на "целевой" якорь. Последний может быть любым ресурсом Web (изображением, видеоклипом, звуковым файлом, документом HTML, элементом в документе и т.д.).

2. Активизация ссылок производится с помощью щелчка мыши, ввода с клавиатуры, голосовых команд и т.д. Пользователи могут перейти к этим ресурсам благодаря атрибуту link в исходном якоре, который указывает адрес целевого якоря с использованием URI.

3. Ссылки, определяемые элементом LINK, не описывают положение (вперед, назад) документа в последовательности документов. Они генерируются с содержимым документа, и агенты пользователей могут отображать их, например, в виде средств перехода.

4. Элемент LINK может присутствовать только в заголовке документа. Элемент A может присутствовать только в теле документа. Никакие другие элементы и атрибуты языка HTML не создают ссылки на другие ресурсы.

Первый вариант правильный. Он не дает строгого определения ссылки, но точно описывает ее структуру.

Второй вариант объявляет активизацию ссылки следствием «щелчка мыши, ввода с клавиатуры, голосовых команд». Но в качестве средства перехода ошибочно называет «атрибут *link* в исходном якоре, который указывает адрес». На самом деле *link* определяет цветовую гамму исходных якорей в документе, подкрашивая активные, посещенные и невостребованные ссылки соответствующими красками. Адресами заведуют другие атрибуты – *href, src, data*.

Третий вариант утверждает, что «элементы LINK не описывают положение (вперед, назад) документа в последовательности». Не важно, генерируется ли такая ссылка с содержимым документа или при анализе заголовка, можно ли «отображать их в виде средств перехода», или нельзя. Вариант неверный, потому что элемент LINK предназначен именно для определения места документа в группе документов, объединенных общей темой.

Вариант четвертый выдвигает на передний план элементы A и LINK, но исключает другие объекты HTML, образующие ссылки: формы и большинство их наследников, карты и области. Ответ неверный.

4.34. *Элемент OBJECT в HTML 4.0 предназначен*:

Варианты ответов:

1. Только для внедрения в документ изображений и апплетов.
2. Для предоставления агенту пользователя возможности управлять генерацией данных без специальных указаний авторов документов.
3. Для реализации объекта HTML, определения его местоположения и указания начальных рабочих значений (инициализации).
4. Для включения в составные документы внедренных документов (независимо от типов устройств, воспроизводящих документы).

Первый вариант начинается дополнением «только». На самом деле элемент OBJECT задуман в HTML «как универсальное средство внедрения в документ любых существующих и будущих структур». Пусть об OBJECT говорят, что он перекрывает все свойства и функции элементов IMG и APPLET, это не значит, что в объект нельзя вставить абзац или таблицу. Равно как и объект можно вставить в них. Вариант неверный.

Второй вариант утверждает, что назначение типа элемента OBJECT состоит в обучении агента пользователя методам генерации данных. Вариант, как и предыдущий, неверный. Без указаний автора браузер не разберется с управлением генерацией данных в объекте. Объекты – это воспроизводимые элементы документа, а не инструкторы по их созданию.

Третий вариант аналогичен второму, он наделяет объект не свойственными ему функциями. Да, OBJECT задуман создателями для «реализации и определения местоположения», но не инициализации. Для этого предназначен другой тип элемента – PARAM. Вариант неверный.

Правильный вариант четвертый.

4.35. *Таблицы стилей в HTML предназначены для следующих целей*:

Варианты ответов:

1. Использование собственных расширений HTML и написание программ разметки документов вместо использования разметки HTML.

2. Унифицированный механизм представления в документах HTML интервалов между строками текста, отступов, цветов, используемых для текста и фона, размера и стиля шрифтов и другой информации.

3. Использование таблиц HTML для размещения объектов на странице.

4. Преобразование текста в изображения и использование изображений для управления пустым пространством.

Вариант первый. Известно, что «собственные расширения» любого компьютерного языка представляют собой конструкции, не принадлежащие этому языку, созданные энтузиастами для получения дополнительных возможностей. Ответ неправильный. Стандартный аппарат таблиц стилей предназначен для унификации и единообразия инструментов представления документа, а не для доморощенных усовершенствований программного продукта.

Верный второй вариант называет таблицы стилей механизмом оформления документов. Ответ правильный. Таблица стилей представляет собой перечень атрибутов целого класса элементов и набор установленных для этих атрибутов значений. Это рецепт, механизм, средство достижения цели.

Третий вариант: «использование таблиц HTML для размещения объектов на странице». Вспомним, что таблицы HTML позволяют структурировать текст документа, хотя делать это не рекомендуется. А таблицы стилей управляют начертанием элементов визуальных документов. Ответ ошибочный.

Вариант четвертый: таблица стилей, т.е. меню значений атрибутов, объявляется инструментом «преобразования текста в изображения и использование изображений для управления пустым пространством». Обе названные проблемы не только не имеют ничего общего с таблицами стилей, но вообще не нашли еще своего полноценного решения. Ответ неверный.

4.36. *Выравнивание, стили шрифтов и горизонтальные разделители в документах HTML*:

Варианты ответов:

1. Управляются только с помощью таблиц стилей.

2. Управляются только с помощью атрибутов форматирования.

3. Используются только для визуального представления документов.

4. Применяются только к встроенным, прикрепляемым и обтекаемым объектам.

Вариант первый: выравнивание, стили шрифтов и разделители, безусловно, управляются «с помощью таблиц стилей», но не «только». Дополнительный вопрос – в чем состоит управление горизонтальными разделителями?

Второй вариант: и «с помощью атрибутов форматирования» они управляются тоже, и тоже не «только». Дополнительные вопросы – какие типы элементов языка разметки обеспечивают горизонтальное разделение

страниц? Какие атрибуты форматирования признаны нежелательными и почему?

Верный третий ответ: управление форматированием, действительно, используется «для визуального представления документов» потому что стили элементов синтезированной речи, как и движения штырьков тактильных устройств, управляются не форматом. Попробуйте ответить, какими средствами.

Четвертый ответ: правильно, применяются к «встроенным, прикрепляемым и обтекаемым объектам». И снова не «только». По умолчанию производится управление форматированием всей страницы.

4.37. *Фреймы в HTML позволяют авторам*:

Варианты ответов:

1. Представлять документы в нескольких разделах экрана, которые могут быть независимыми или вложенными окнами.
2. Указывать с помощью атрибута target альтернативное содержимое для агентов пользователей, не поддерживающих фреймы или сконфигурированных так, чтобы не отображать их.
3. Делать документы более доступными для людей, использующих невизуальные агенты пользователя.
4. Вводить в ячейки таблиц HTML внешние документы (в том числе изображения) с помощью элемента FRAMESET.

Правильный первый ответ тривиален, и это порождает сомнения: фреймы ли (тип элемента FRAME) позволяют представлять документы в нескольких полях экрана, или наборы фреймов (тип элемента FRAMESET)? Многие путаются в функциях этих элементов. Который из них пустой, а какой имеет строго определенное содержимое? Кто из них отвечает за отображение альтернативного текста? Как обстоят дела с вложенностью этих типов элементов? – Подобные вопросы не звучат в варианте прямо, но к ним нужно быть готовыми.

Второй вариант. Известно, что атрибут «*target*» указывает окно, в которое нужно загрузить документ. А в ответе: указывает альтернативное содержимое. Допустим, альтернативный документ (или фрагмент), но *target* указывает *куда*, а не *что* загружать. Далее: для агентов пользователей, не поддерживающих фреймы или сконфигурированных так, чтобы не отображать их. Рекомендуется грузить кусочки *фреймов* (рамок) в *документы* (портреты, тексты и т.д.) в тех случаях, если компьютер вообще не работает с фреймами. Ответ неверный.

Третий вариант подсказывает: «делать документы более доступными для людей, использующих невизуальные агенты пользователя». Нужны ли комментарии? Многооконное представление позволяет совмещать разные документы на одном экране, видеть их одновременно, сопоставлять друг с другом. А как быть речевому браузеру? Воспроизводить словесное описание всех окон разными голосами? Понятно, что документы не станут доступнее, если несколько монологов сольются в одну какофонию. Ответ ошибочный.

Вариант четвертый: «вводить в ячейки таблиц внешние документы (в

том числе изображения) с помощью элемента FRAMESET». Элемент FRAMESET, как известно, предназначен для деления экрана на окна. Он не имеет очертаний, он представляет доли неизвестной заранее площади. С его помощью нельзя вводить документы, но можно распланировать плоскость под другие элементы языка (FRAME), способные внедрять изображения или текст.

4.38. *Укажите правильное определение формы HTML*:

Варианты ответов:

1. Метод отправки данных на сервер (атрибут method).
2. Совокупность изобразительных средств для организации обмена информацией в Web.
3. Раздел документа, в котором содержатся: обычная информация, разметка и специальные управляющие элементы.
4. Контейнер для программы, которая будет обрабатывать заполненную и переданную форму (атрибут action).

Вариант первый: форма HTML (т.е. элемент FORM) есть «метод отправки данных». Противоречие очевидно: метод – это технологический способ достижения цели, а элемент – это конструкция языка. Упомянутый в тексте атрибут *method*, действительно, определяет технологию передачи информации из формы на сервер, но это совпадение не дает определения формы. Ответ неверный.

Вариант второй использует непривычные термины, которые могут быть интерпретированы достаточно произвольно. В широком смысле все элементы компьютерной графики можно назвать «изобразительными средствами», а образование гиперссылки, посылку запроса и загрузку целевого ресурса – обменом информацией. Однако формы не являются организующим началом процесса. Этот тип элемента представляет собой лишь удобный инструмент реализации одной из фаз обмена. Ответ ошибочный.

Третий вариант тоже отличается свободой определений. В спецификации не определен термин «раздел». Так в зависимости от контекста называют части страницы (HEAD, BODY), группы строк таблицы (THEAD, TBODY, TFOOT), параграфы текста. В этом смысле форму тоже можно назвать разделом документа, имеющим самостоятельное значение, конкретным инструментом взаимодействия с сетью, содержащим обычную информацию, разметку и специальные управляющие элементы. Ответ правильный.

Четвертый вариант объявляет форму контейнером для программы. По определению, модель ее содержимого включает скрипты, но их назначение шире, чем «обрабатывать заполненную и переданную форму». Ситуация разрешается уточнением, заключенным в вопросе в скобки: атрибут *action* преподносится как инструмент работы с программой. Его истинное назначение, однако, состоит в том, чтобы задать адрес агента пользователя на сервере для обработки формы. Это может быть URI с указанием HTTP (для передачи формы программе) или mailto (для отправки по электронной почте). Ответ неверный.

4.39. *Укажите вариант, правильно характеризующий возможности скриптов в HTML*:

Варианты ответов:

1. Скрипты не могут динамически изменять содержимое документа во время загрузки.
2. Скрипты не используются в форме для обработки вводимых данных.
3. Скрипты могут включаться событиями, оказывающими влияние на документ, например, загрузкой, выгрузкой, фокусом элемента, перемещением мыши и т.д., однако не могут связываться с управляющими элементами формы.
4. Скрипты в HTML предназначены для выполнения всех действий, упомянутых в вариантах 1, 2 и 3.

Скрипт по определению – это программный код, предназначенный для управления представлением документа. Для адаптации к происходящим при воспроизведении изменениям. Для осуществления задуманной автором реакции на события: загрузку, нажатие клавиш, движение мыши и прочие. Первый вариант отрицает определение, значит, он неверный. Как и второй. Третий подтверждает, что «скрипты могут включаться событиями», однако отрицает, что такими событиями могут управлять элементы формы. Это неправильно. Большинство элементов формы поддерживают атрибуты внутренних событий, которые реализуют обсуждаемую связь.

Верный вариант четвертый. Скрипты способны на все перечисленные действия.

4.40. *Типы управляющих элементов, создаваемых с помощью элемента INPUT*:

Варианты ответов:

1. Элемент FORM для ввода, разметки и рассылки документов в Web.
2. Элемент PASSWORD для ввода паролей и представления вводимых символов в закодированном виде (например, звездочками).
3. Элемент TABLE для ввода информации в упорядоченном виде.
4. Элемент BUTTON для создания графических кнопок (атрибут image).

Тип элемента INPUT представляет собой поле ввода данных для диалога пользователя с системой. Он используется для создания целого ряда управляющих элементов формы, что узаконено следующей комбинацией элементов:

```
% InputType      " (TEXT | PASSWORD | CHECKBOX | RADIO |
                  SUBMIT | RESET | FILE | HIDDEN | IMAGE | BUTTON) "
```

Вариант первый утверждает, что из таких элементов создаются формы HTML. Ответ неверный. Элемент FORM служит контейнером для управляющих элементов и определяет: макет формы, обрабатывающую программу, метод отправки на сервер, кодировку символов. Эти атрибуты INPUT не

поддерживает.

Правильный второй вариант существенно обедняет комбинацию «%*InputType*;», однако, никаких противоречий не содержит.

Третий вариант развивает идею первого. Предлагается создавать из элементов INPUT таблицы HTML. Конечно, внутрь ячейки таблицы можно вставить поле ввода. Можно изобразить на форме ряды таких полей вплотную друг к другу так, чтобы придать результату сходство с табличным представлением данных. Но элементом TABLE такое построение не станет. Ответ ошибочный.

Последний, четвертый вариант не так прямолинеен. Элемент BUTTON, действительно, может быть получен как потомок элемента INPUT для создания кнопок графического доступа. Достигается это присвоением атрибуту *type* значения *image*. Однако составители сознательно допустили в тексте неточность. Они назвали *image* атрибутом, что и сделало вариант неверным.

4.41. Какие функции выполняют атрибуты Border и FrameBorder у элемента FRAME?

Варианты ответов:

1. FrameBorder отвечает за наличие границы фрейма, атрибут Border элементом FRAME не поддерживается.

2. Атрибут Border отвечает за толщину внутренней границы фрейма, FrameBorder – за толщину внешней.

3. Атрибут Border отвечает за расстояние между границей фрейма и объектами, входящими в этот фрейм, FrameBorder – за толщину границы фрейма.

4. Атрибут Border отвечает за толщину границы фрейма, FrameBorder – за толщину обрамления каждого элемента страницы, отображенной во фрейме.

Наличие в языке разметки сочетаний терминов frame и border зачастую приводит к путанице этих понятий. Первый вариант дает правильный ответ.

Второй вариант неверный. Допустим, что *FrameBorder* ограждает фрейм от других, а *border* отвечает за внутреннюю рамку – ее можно, например, сделать другого цвета. Но кто объяснит, что такое «внутренняя граница»?

Третий вариант: *border* управляет расстоянием между границей фрейма и объектами, входящими в этот фрейм. Интерпретация ошибочная, обычно эти расстояния называются горизонтальными и вертикальными отступами.

Атрибут *border* известен по таблицам, он отвечает за толщину их границ. *FrameBorder* – граница фрейма – выполняет объявляемую названием функцию. Четвертый вариант, навязывающий иную точку зрения, некорректен.

4.42. *Что означает комбинация символов *?

Варианты ответов:

1. Подсказка браузеру, что последующий текст начнется с нового абза-

ца.

2. Комментарий в виде набора символов между ограничителями '&' и ';'.

3. На экране отобразится неразрывный пробел.

4. На экране отобразится китайский иероглиф «вода».

Первый вариант неверный. Согласно спецификации переход к следующему абзацу управляется очередным открывающим тегом элемента P, встреченным браузером в потоке данных. Все другие способы перехода на следующую строку формально не являются новым абзацем. Ни возврат каретки, ни символ перевода строки, ни элемент BR, ни даже очередной элемент уровня блока.

Второй вариант утверждает, что текст, заключенный между символами '&' и ';' является комментарием языка разметки. Ответ ошибочный. Комментарий HTML ограничивается с обеих сторон двумя подряд идущими дефисами '--'. Ограничители, указанные в ответе, согласно спецификации, содержат в себе комбинацию ссылок на символы, причем замыкающий знак ';' не обязателен.

Правильный вариант – третий. Комбинация символов «* *» означает неразрывный пробел. Кстати, а в чем заключается его (пробела) неразрывность? Можно ли расположить его по частям, например, на разные строки? Нельзя. А обычный пробел? Тоже нельзя. Так в чем отличие? Ответ прост – если вы в тексте HTML-документа поставите несколько пробелов, чтобы заметнее развести символы, браузер превратит все ваши пробелы в один. А если идут друг за другом подряд комбинации « », то все они сохранятся при воспроизведении.

Четвертый ответ неверный, мнемонически иероглиф «вода» не определен.

4.43. *Что такое веб-сайт*?

Варианты ответов:

1. Совокупность HTML-страниц, таблиц стилей, скриптов, графических файлов, музыкальных файлов, и т.д., объединенных общей темой, целью и местом размещения.

2. Совокупность файлов различного типа, связанных между собой гиперссылками, и размещенных на одном логическом хосте.

3. Совокупность HTML-страниц, таблиц стилей, скриптов, графических файлов, музыкальных файлов, и т.д., связанных между собою гиперссылками и объединенных общей темой, целью и местом размещения.

4. Совокупность HTML-страниц, таблиц стилей, скриптов, графических файлов, музыкальных файлов, и т.д., связанных между собою гиперссылками и объединенных общей темой и целью.

В определении ошибочного первого варианта недостает гиперссылок, а без них перед нами не сайт, а простая совокупность файлов.

Второй вариант тоже неверный. Если хост содержит hlp-файлы, связанные гиперссылками, тогда это не веб-сайт, а справочная система.

Отличие третьего варианта от четвертого – дополнение «местом раз-

мещения». Если вы зафиксируете часть файлов своего сайта на логический диск C:, а часть – на логический диск D:, связав их гиперссылками, это будет один сайт. Даже если эти диски будут не логическими, а физическими. Даже если вы разместили свои файлы на одном сервере, а гостевую книгу создали на другом. Место размещения компонентов сайта не является существенным признаком.

Правильный вариант – четвертый.

4.44. *Назначение элемента META. Выберите НЕВЕРНЫЙ вариант.*

Варианты ответов:

1. Для внесения в HTML-документ информации об его авторе и кодировке символов, в которой должен отображаться документ.
2. Для указания документа, на который необходимо перейти с текущего.
3. Для организации слайд-шоу в документе.
4. Для указания даты окончания действия документа (срока годности).

Метаданные позволяют указывать общую информацию о документе. Элемент META поддерживает атрибуты *name*, объявляющие имя свойства документа, и *content*, присваивающие этому свойству определенное значение. В одном и том же элементе META могут равноправно сосуществовать несколько *name* и соответствующих им *content*. Можно записать:

```
<META name=”author” lang=”ru” content=”Пушкин”
name=”keywords” lang=”ru” content=”отпуск, Сочи, солнцепек”
http-equiv=”Content-Type” content=”text/html; charset = EUC-JP”>
```

Первый вариант правильный, переходим ко второму. Для указания документа, на который необходимо перейти, вообще говоря, предпочтительно использовать элемент LINK. Однако элемент META тоже предоставляет такую возможность. Второй ответ не содержит противоречий.

Третий вариант выходит за рамки спецификации HTML, термина «слайд-шоу» в ней нет. Нет его и в описании элемента META. Ответ ошибочный.

Четвертый вариант правильный, элемент META можно использовать для указания срока действия документа. Можно устанавливать продолжительность демонстрации документа. Многие авторы применяют этот прием для создания кратковременных заставок на экране.

Искомый неверный вариант третий.

ЛИТЕРАТУРА

1. Дарнелл Р., Бэсори-Киц Д., Брайан Дж. Г., Кемпбелл Б. “HTML. Энциклопедия пользователя”. Киев: - Диасофт, 1998;

2. Матросов А.В., Сергеев А.О., Чаунин М.П. “HTML 4.0: Новый уровень создания HTML-документов. Серия «В подлиннике»”. Спб.: - BHV, 2001;

3. Коржинский С.Н. “Настольная книга Web-мастера: Эффективное

применение HTML, CSS и JavaScript”. М.: - Кнорус, 2001;
4. Шапошников И. В. “Самоучитель HTML”. Спб.: - BHV, 2001;
5. Вайскопф Дж. “Microsoft FrontPage 2000. Учебный курс”. Спб.: - «Питер», 2000;
6. Лехто К., Полонски В. “FrontPage 98. Официальное руководство Microsoft”. Спб.: - BHV, 1998.
7. Омельченко Л.Н., Федоров А.Ф. “Microsoft FrontPage 2002. Самоучитель”. Спб.: - BHV, 2002.

ОГЛАВЛЕНИЕ

На начало документа
На содержание