МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Государственное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

ЯЗЫКОВЫЕ СРЕДСТВА СОЗДАНИЯ ГИПЕРДОКУМЕНТОВ

Учебно – м*етодическое пособие по курсу «Языковые средства создания гипердокументов» по выполнению лабораторных работ и самостоятельной работы для студентов ВУЗа*

Томск
2018

Пособие составлено в соответствии с тематикой лабораторных работ и самостоятельной работы «Языковые средства создания гипердокументов». Особое внимание уделено основам практического применения языков HTML и PHP для создания веб-сайтов. Пособие содержит темы и содержание лабораторных работ, методические указания к их проведению.
Для преподавателей, аспирантов, студентов и магистрантов.

СОСТАВИТЕЛЬ: Е.А. Шельмина

СОДЕРЖАНИЕ

Краткое содержание тем и результатов их освоения

Темы лабораторных работ	Деятельность студента. Выполняя задания, студент:
1. Создание Web-страниц.	• *использует* основные теги языка HTML; • *выбирает* стиль оформления web – страницы и необходимые для этого теги; • *использует* знания, полученные ранее и самостоятельно создает свою веб-страницу;
2. Гипертекстовые ссылки и иллюстрации на веб-страницах. Построение таблиц.	• *изучает* возможности создания гиперссылок, размещения изображений и таблиц на веб-страницах; • *учится* создавать веб-страницы используя изученные теги;
3. Фреймы и формы	• *изучает* принципы создания веб-страниц с использованием фреймов и форм; • *получает навыки* создания веб-страниц с использованием фреймов и форм;
4. Каскадные таблицы стилей (CSS)	• *изучает* возможности создания веб-страниц с использованием технологии каскадных таблиц стилей;
5. Основы синтаксиса языка PHP	• *знакомится* с синтаксисом и возможностями языка PHP; • *получает* практические навыки программирвоания на языке PHP;
6. Обработка запросов с помощью PHP	• *учиться* создавать веб-страницы с обработкой запросов посредством форм и языка PHP;
7. Функции в PHP	• *разрабатывает* веб-сайт с использованием языков HTML и PHP;

Раздел 1. Лабораторные работы

Лабораторная работа №1
Создание Web-страниц

Цель работы: знакомство с основными тегами языка HTML, связанными с заголовками различного уровня, форматированием текста, заданием цвета.

Теоретическое обоснование

HyperText Markup Language (HTML) - язык разметки гипертекста - предназначен для написания гипертекстовых документов, публикуемых в World Wide Web.

Гипертекстовый документ - это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера. С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок - специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

Существует два способа создания гипертекстовых документов. Можно воспользоваться одним из WYSIWYG HTML-редакторов, для работы с которыми не требуется специальных знаний о внутренней структуре создаваемого документа. Этот способ позволяет создавать документы для WWW без знания языка HTML. HTML-редакторы автоматизируют создание гипертекстовых документов, избавляют от рутинной работы. Однако их возможности ограничены, они сильно увеличивают размер получаемого файла и не всегда полученный с их помощью результат соответствует ожиданиям разработчика. Но, безусловно, этот способ незаменим для новичков в деле подготовки гипертекстовых документов.

Альтернативой служит создание и разметка документа при помощи обычного редактора текста. При этом способе в текст вручную вставляются команды языка HTML. Создавая документы таким способом, вы точно знаете, что делаете.

Как уже отмечалось, HTML-документ содержит символьную информацию. Одна ее часть - собственно текст, т. е. данные, составляющие содержимое документа. Другая -теги, называемые также флагами разметки, - специальные конструкции языка HTML, используемые для разметки документа и управляющие его отображением. Именно теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ. Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (браузеры) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом. Для файлов, содержащих HTML-документы, приняты расширения .htm или .html.

Прописные и строчные буквы при записи тегов не различаются. В большинстве случаев теги используются парами. Пара состоит из открывающего и закрывающего тегов. Синтаксис открывающего тега: <имя_тега [атрибуты]>

Прямые скобки, используемые в описании синтаксиса, означают, что данный элемент может отсутствовать. Имя закрывающего тега отличается от имени открывающего лишь тем, что перед ним ставится наклонная черта: </имя_тега>.

Атрибуты тега записываются в следующем формате: имя[="значение"]. Кавычки при задании значения аргумента не обязательны и могут быть опущены. Для некоторых атрибутов значение может не указываться. У закрывающего тега атрибутов не бывает.

Действие любого парного тега начинается с того места, где встретился открывающий тег и заканчивается при встрече соответствующего закрывающего тега. Часто пару, состоящую из открывающего и закрывающего тегов, называют контейнером, а часть текста, окаймленную открывающим и закрывающим тегом, **- элементом.**

Структура HTML-документа

Самым главным из тегов HTML является одноименный тег - <HTML>. Он должен всегда открывать ваш документ, так же, как тег </HTML> должен непременно стоять в последней его строке. Эти теги обозначают, что находящиеся между ними строки представляют единый гипертекстовый документ. Без этих тегов браузер или другая программа просмотра не в состоянии идентифицировать формат документа и правильно его интерпретировать.

Закрывающий тег так же важен, как и открывающий. Если, например, документ включен в электронное письмо, тег </HTML> дает команду программе просмотра прекратить интерпретацию текста, как HTML-кода.

HTML-документ состоит из двух частей: заголовок (head) и тело (body), расположенных в следующем порядке:

```
<HTML>
 <HEAD>
    ...
 </HEAD>

 <BODY>
    ...
 </BODY>
</HTML>
```

В HTML-документ можно включать комментарии, позволяющие скрыть часть текста от браузера. Все, что заключено между последовательностями символов <!- и ->, при просмотре страницы остается невидимым. Комментарии не могут быть вложенными друг в друга.

Заголовок документа

Включение в документ заголовочной части не является обязательным. Задачей заголовка является представление необходимой информации для браузера и сервера HTTP. Информация, размещенная внутри заголовка документа, обычно не выводится на экран (кроме названия документа).

Заголовок документа открывается тегом <HEAD>, который обычно следует сразу же за тегом <HTML>. Закрывающий тег </HEAD> показывает конец этого раздела, между ними располагаются остальные теги заголовка документа.

Чаще всего в заголовок документа включают парный тег <TITLE> ... </TITLE>, определяющий название документа. Многие программы просмотра используют его как заголовок окна, в котором выводят документ. Программы, индексирующие документы в сети Интернет, используют название для идентификации страницы. Хорошее название должно быть достаточно длинным для того, чтобы можно было корректно указать соответствующую страницу, и в то же время оно должно помещаться в заголовке окна. Название документа вписывается между открывающим и закрывающим тегами.

Тело документа

В отличии от заголовка, тело документа является обязательным элементом, так как в нем располагается весь материал вашего документа (есть только одно исключение, с которым мы познакомимся далее, - когда документ содержит вместо тела группу фреймов).

Тело документа размещается между тегами <BODY> и </BODY>. Все, что размещено между этими тегами, интерпретируется браузером в соответствии с правилами языка HTML, позволяющими корректно отображать страницу на экране монитора.

Тег <BODY> не только обозначает начало содержимого документа, но и задает его основные свойства: цвет фона, текста и многое другое. Эти свойства определяются с помощью атрибутов, которые приведены ниже:

Атрибут	Назначение
ALINK	определяет цвет активной ссылки
BACKGROUND	указывает на URL-адрес изображения, которое используется в качестве фонового
BGCOLOR	определяет цвет фона документа
LINK	определяет цвет непосещенной ссылки
TEXT	определяет цвет текста
VLINK	определяет цвет посещенной ссылки

Цветовое оформление документа

Определение цвета составных частей документа - один из первых шагов в его создании. Если этого не сделать, то будут использоваться цвета по умолчанию, определяемые установками браузера. Не существует каких-либо правил создания хорошо сбалансированной цветовой палитры документа. Нужно лишь заботиться о том, чтобы можно было прочитать текст, не испытывая при этом неудобств. При выборе цветовой палитры старайтесь поддерживать высокую контрастность текста и фона и избегайте соседства областей с близкими цветами.

Цвет может быть задан названием или шестнадцатеричным числом, определяющим цвет в модели RGB. Эта цветовая модель базируется на определении цвета как композиции трех основных оттенков цвета: красного (Red), зеленого (Green) и синего (Blue). Каждая компонента задается двузначным шестнадцатеричным числом (т. е. изменяется от 00 до FF). Затем эти значения объединяются в одно число, перед которым ставится символ # (большинство современных браузеров может распознать цвет и без указания символа #).

Следует также отметить, что в записи шестнадцатеричного значения цвета можно использовать как большие, так и маленькие латинские буквы, например, запись #00FF00 равнозначна записи #00ff00.

Ниже представлена таблица 16 стандартных цветов вместе с их шестнадцатеричными кодами.

Цвет	Код	Цвет	Код
black (черный)	#000000	silver (серебряный)	#C0C0C0
maroon (темно-бордовый)	#800000	red (красный)	#FF0000
green (зеленый)	#008000	lime (известь)	#00FF00
olive (оливковый)	#808000	yellow (желтый)	#FFFF00
navy (темно-синий)	#000080	blue (синий)	#0000FF
purple (фиолетовый)	#800080	fuchsia (фуксия)	#FF00FF
teal (сине-зеленый)	#008080	aqua (аква)	#00FFFF
gray (серый)	#808080	white (белый)	#FFFFFF

Разделение текста на абзацы

Язык HTML предполагает, что автор документа ничего не знает о компьютере своего читателя. Читатель вправе установить любой размер окна и пользоваться любым из имеющихся у него шрифтов. Это означает, что место переноса в строке определяется только программой просмотра и установками конечного пользователя. Символы перевода строки оригинального документа игнорируются, в результате чего текст, отлично смотревшийся в окне вашего редактора, может превратиться в сплошной неудобочитаемый текст в окне программы просмотра.

Избежать этой неприятности позволяет разделение на абзацы при помощи тега <P>. Разместите его в начало каждого абзаца, и программа просмотра отделит абзацы друг от друга пустой строкой. Использование закрывающего тега </P> необязательно. Несколько стоящих подряд тегов <P> не дают дополнительного пространства между абзацами.

Тег абзаца имеет один атрибут, поддерживаемый большинством браузеров. Это атрибут ALIGN, задающий выравнивание текста в абзаце. Если этот атрибут не задан, то текст выравнивается по левому краю окна браузера. В таблице представлены возможные значения этого атрибута:

Значение	Функция
LEFT	Выравнивание текста по левой границе окна браузера

CENTER	Выравнивание по центру окна браузера
RIGHT	Выравнивание по правой границе окна браузера
JUSTIFY	Выравнивание текста по ширине окна браузера

Разрыв строки

Иногда требуется "разорвать" текст, перенеся его остаток на новую строку, при этом не выделяя нового абзаца. Для этого используется тег разрыва строки
. Он заставляет программу просмотра выводить стоящие после него символы с новой строки. В отличие от тега абзаца, тег
 не добавляет пустую строку. У этого тега нет парного закрывающего тега.

Некоторые браузеры интерпретируют несколько стоящих рядом тегов
 как один тег, поэтому не стоит использовать его для вставки пустых строк.

Бывают случаи, когда возникает надобность в операции противоположного назначения - запретить перевод строки. Текст, заключенный между тегами <NOBR> и </NOBR>, будет гарантированно располагаться в одной строке без переноса на другую.

Горизонтальные линии

Другим методом разделения документа на части является проведение горизонтальных линий. Они визуально подчеркивают законченность той или иной области страницы. Тег <HR> позволяет провести рельефную горизонтальную линию в окне большинства программ просмотра. Этот тег не требует закрывающего тега. До и после линии автоматически вставляется пустая строка. Атрибуты тега <HR> представлены в таблице.

Атрибут	Назначение
ALIGN	выравнивает по краю или центру, имеет значения LEFT, CENTER, RIGHT
WIDTH	устанавливает длину линии в пикселах или процентах от ширины окна браузера (в последнем случае добавляется символ %)
SIZE	устанавливает ширину линии в пикселах
NOSHADE	отменяет рельефность линии
COLOR	указывает цвет линии

Форматирование гипертекста

Язык HTML поддерживает как логический (logical), так и физический (physical) стили форматирования содержимого документа. Использование логического форматирования указывает на назначение данного фрагмента текста, а при физическом форматировании досконально задается его внешний вид. По возможности стоит использовать логические стили, так как они позволяют браузеру выбрать наиболее подходящий документу вид. Использование логических стилей также поможет читателю разобраться в структуре документа. Физический стиль используется в основном программами, конвертирующими текстовые файлы, содержащие физическое форматирование, в HTML, так как логическое форматирование документа невозможно выполнить автоматически.

Логическое форматирование

Хотя язык HTML включает теги для достижения различных шрифтовых эффектов (полужирный шрифт, курсив, подчеркнутый шрифт), не все браузеры их поддерживают. Однако все браузеры поддерживают тот или иной способ выделения текста. Поэтому использование логического форматирования текста в любом случае приведет к выделению программой просмотра различных частей текста и выявит структуру документа.

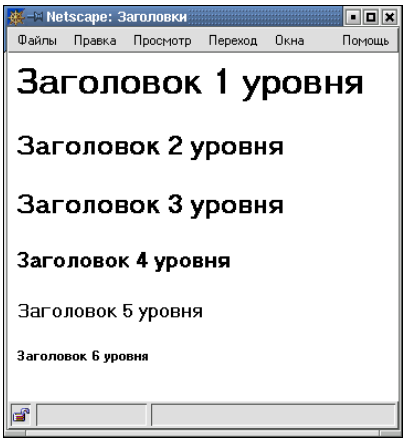
Говоря о логической разметке текста, можно выделить две основные части:

- выделение заголовков в документе;
- логическое выделение элементов текста.

Название документа, задаваемое с помощью тега <TITLE>, не выводится на экран как часть документа. Чтобы отобразить название используется один из тегов заголовка. Заголовки в типичном документе разделяются по уровням. Язык HTML позволяет задать шесть уровней заголовков: H1 (заголовок первого уровня), H2, H3, H4, H5 и H6. Заголовок первого уровня имеет обычно больший размер и насыщенность по сравнению с заголовком второго уровня.

Пример

```
<HTML>
 <HEAD>
  <TITLE> Заголовки </TITLE>
 </HEAD>
 <BODY BGCOLOR=white>
  <H1> Заголовок 1 уровня</H1>
  <H2> Заголовок 2 уровня</H2>
  <H3> Заголовок 3 уровня</H3>
  <H4> Заголовок 4 уровня</H4>
  <H5> Заголовок 5 уровня</H5>
  <H6> Заголовок 6 уровня</H6>
  </BODY>
</HTML<
```



Помните, что если вы забудете поставить закрывающий тег заголовка, вид страницы будет искажен: любой из тегов заголовка автоматически вставляет пустую строку до и после себя.

Теги заголовков поддерживают атрибут ALIGN, действие которого аналогично действию такого же атрибута тега выделения абзаца.

Элементы логического выделения фрагментов текста, а также возможное оформление каждого из них приведены в таблице.

Теги	Результат
<CITE> </CITE>	используется для выделения цитат или названий книг и статей
<CODE> </CODE>	применяется для вывода небольшого куска программного кода
<EM> </EM>	используется для выделения важных фрагментов текста
<KBD> </KBD>	выделяет текст, вводимый пользователем с клавиатуры
<SAMP> </SAMP>	используется для выделения текста примера
<STRONG> </STRONG>	используется для выделения очень важных фрагментов текста
<VAR> </VAR>	используется для отметки имен переменных
<STRIKE> </STRIKE>	используется для отметки удаленного текста

Физическое форматирование

Одним из отличий HTML-документа от документа, подготовленного на печатной машинке, является возможность форматирования текста. Язык HTML позволяет автору документа выбрать понравившийся ему шрифт, подходящий размер букв, их цвет и начертание. За все эти параметры отображения текста отвечают теги физического форматирования. Они действуют на все символы, стоящие между открывающим и закрывающим тегами.

Теги	Результат
<B> </B>	полужирный
<I> </I>	курсив
<U> </U>	подчеркнутый
<S> </S>	зачеркнутый
<BIG> </BIG>	большой
<SMALL> </SMALL>	маленький
	верхний индекс
	нижний индекс

Элементы физического форматирования могут быть вложенными друг в друга, хотя конечный результат зависит от браузера. При этом нужно внимательно следить, чтобы один контейнер находился целиком в другом контейнере.

Кроме вышеперечисленных тегов в документе может использоваться тег <FONT>, позволяющий непосредственно задать размер и цвет шрифта. Элемент <FONT> представляет собой контейнер, т. е. требует как открывающего, так и закрывающего тегов, и сам может использоваться внутри любого другого текстового контейнера.

После стартового тега обязательно указание атрибутов, без которых элемент не оказывает никакого влияния на текст, помещенный в контейнер.

Атрибут FACE позволяет указать тип шрифта, которым программа просмотра выведет ваш текст (если таковым располагает пользователь). Если нужного шрифта нет,

программа проигнорирует запрос и будет использовать шрифт, установленный по умолчанию.

Этот атрибут позволяет указать как один, так и несколько шрифтов (через запятую). Весь список будет просмотрен слева направо и первый из имеющихся на машине пользователя будет использован для вывода документа.

Атрибут SIZE служит для указания размера шрифта в условных единицах от 1 до 7. Считается, что размер "нормального" шрифта соответствует числу 3. Размер может быть как абсолютной величиной (SIZE=5), так и относительной (SIZE=+2). Во втором примере текущий размер шрифта увеличивается на 2.

Атрибут COLOR устанавливает цвет шрифта, который может быть задан как в формате RGB, так и указанием имени.

Списки

В настоящее время стандарты HTML поддерживают теги для списков трех различных видов: нумерованных (упорядоченных), маркированных (неупорядоченных) и списков определений. Списки и элементы списков являются блочными элементами. Это означает, что перед ними и после них автоматически добавляются пустые строки.

Язык HTML допускает вложенность любых видов списков. Для этого размещают одну пару тегов (стартовый и завершающий) внутри другой. Следует помнить о том, что все имеющиеся списки должны завершаться закрывающим тегом.

Нумерованные списки

Нумерованные (упорядоченные) списки используют, когда важен порядок вывода элементов списка. Браузер автоматически вставляет номера элементов по порядку, в исходном HTML-тексте номера не печатаются. Если количество элементов списка изменится (в результате удаления или добавления новых элементов), то нумерация автоматически обновится.

Весь нумерованный список заключается между парой тегов <OL> и </OL>, а каждый элемент списка расположен между тегами <LI> и </LI> (закрывающий тег </LI> может отсутствовать). Тег <OL> может имеет атрибуты TYPE и START:

<OL START=n TYPE=вид_счетчика>

Атрибут TYPE задает вид счетчика, возможные значения которого приведены в таблице, а START - начальное значение.

Значение	Функция
A	большие латинские буквы (A,B,C...)
a	маленькие латинские буквы (a,b,c...)
I	большие римские цифры (I,II,III...)
i	маленькие римские цифры (i,ii,iii...)
1	арабские цифры (1,2,3...); используется по умолчанию

Маркированные списки

Маркированный (неупорядоченный) список используется для представления коллекции элементов, порядок вывода которых не важен. При выводе маркированных списков браузер автоматически вставляет специальные значки (маркеры), отмечающие каждый элемент списка.

Маркированный список начинается стартовым тегом <UL> и завершается тегом </UL>. Каждый элемент списка начинается с тега <LI> и завершается

(необязательным) тегом </LI>. Тег <UL> имеет атрибут TYPE, определяющий внешний вид маркера: <UL TYPE=тип_маркера>.

Значение атрибута TYPE может быть одним из следующих: disc (круг - форма по умолчанию), circle (окружность) или square (квадрат).

Атрибут TYPE применяется и в теге <LI> для изменения формы маркера перед конкретным элементом списка.

Тег <LI> обеспечивает вывод маркера и разделение элементов списка. Если хочется использовать нестандартные маркеры, то тег <LI> не указывается. Для выделения элементов списка в этом случае используются какие-либо картинки или символы, а тег
 обеспечивает переход к следующему элементу списка.

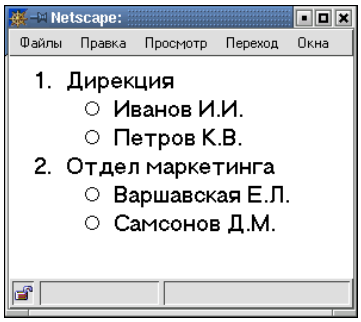
Список определений

Список описаний (список определений) начинается с тега <DL> и завершается тегом </DL>. Данный список служит для создания списков типа "термин" - "описание". Термин автоматически размещается у левой границы страницы, а их определения смещены относительно них вправо. Каждый термин обозначается тегом <DT>, а его описание - тегом <DD>.

Вложенные списки

Любой список может быть частью другого списка, вложен в другой список. Считается полезным использование сдвигов при подготовки текста исходного HTML-документа, чтобы четко представлять уровни вложенности списков.

```
<OL>
 <LI>Дирекция</LI>
 <UL>
   <LI>Иванов И.И.</LI>
   <LI>Петров К.В.</LI>
 </UL>
 <LI>Отдел маркетинга</LI>
 <UL>
  <LI>Варшавская Е.Л.</LI>
  <LI>Самсонов Д.М.</LI>
 </UL>
</OL>
```



При выводе вложенных маркированных списков браузер автоматически проставляет маркеры перед элементами, находящимися на разных уровнях вложенности.

Для вложенных нумерованных списков браузер, к сожалению, не изменяет тип нумерации. По умолчанию каждый уровень внутри такого списка будет отмечен арабскими цифрами. Для получения списка в другом формате потребуется вручную проставить каждую метку, используя атрибут TYPE.

Задания для лабораторной работы

Задание 1. Создать страницу о себе. Продемонстрировать все возможности форматирования текста, использование списков, задания цвета.

Задание 2. Создать Web-страницу следующего содержания, с использованием HTML, соблюдая форматирование:

Радуга

Радуга — атмосферное оптическое и метеорологическое явление, наблюдаемое обычно после дождя или перед ним. Оно выглядит как ***дуга*** или ***окружность***, составленная из цветов спектра (глядя снаружи — внутрь дуги: красный, оранжевый, жёлтый, зелёный, голубой, синий, фиолетовый).

Причина радуги — преломление света

Радуга возникает из-за того, что солнечный свет испытывает преломление в капельках воды, взвешенных в воздухе. Эти капельки по-разному отклоняют свет разных цветов (**Красный** свет отклоняется на 137 градусов 30 минут, а **фиолетовый** на 139°20’), в результате чего белый свет разлагается в спектр. Наблюдателю кажется, что из пространства по концентрическим кругам (дугам) исходит разноцветное свечение (при этом источник яркого света всегда находится за спиной наблюдателя).

Хотя многоцветный спектр радуги непрерывен, по традиции в нем выделяют 7 цветов. Считают, что первым выбрал число 7 **Исаак Ньютон**, для которого число 7 имело специальное символическое значение (по пифагорейским, богословским или нумерологическим соображениям). Причём первоначально он различал только пять цветов - красный, жёлтый, зелёный, голубой, фиолетовый, о чём и написал в своей "*Оптике*".Но вспоследствии, стремясь создать соответствие между числом цветов спектра и числом основных тонов музыкальной гаммы, Ньютон добавил к пяти перечисленным цветам спектра еще два.

Для запоминания их последовательности есть мнемонические фразы, первые буквы каждого слова в которых соответствуют первым буквам названия цвета:

Как однажды Жак-звонарь головой сломал фонарь.
Каждый охотник желает знать где сидит фазан.

Лабораторная работа №2
Гипертекстовые ссылки и иллюстрации на Web-страницах. Построение таблиц

Цель работы: научиться использовать теги HTML для создания гиперссылок, таблиц и размещения изображений на веб-странице.

Использование графики в HTML-документах

Рисунки и анимация могут сделать HTML-документ более привлекательным и интересным. Они не только украшают страницу, но и помогают лучше передать содержание документа. Для правильного использования графики в HTML-документе необходимо учитывать следующие факторы: многие браузеры поддерживают только графические форматы GIF и JPEG, файлы, содержащие графику, передаются медленно, некоторые пользователи не имеют графических браузеров или намерено отключают загрузку изображений.

Тег <IMG> вставляет изображение в документ, как если бы оно было просто одним большим символом. Синтаксис тега: <IMG атрибут1=значение1 ... [атрибутN=значениеN]>.

Атрибуты тега <IMG> и их значения приведены в таблице.

Атрибут	Назначение
SRC="файл"	задает URL-адрес изображения (можно указывать как абсолютный, так и относительный URL-адрес; если файл с изображением находится в той же директории, что и HTML-документ, то достаточно просто указать имя файла); этот атрибут является обязательным
ALT="текст"	задает альтернативный текст для браузеров, не поддерживающих работу с изображениями
ALIGN="тип"	задает расположение картинки относительно текста, тип может принимать следующие значения: TOP, MIDDLE, BOTTON, LEFT, RIGHT
BORDER=n	устанавливает толщину обрамления вокруг изображения в пикселах
HEIGHT=n(%)	устанавливает высоту изображения в пикселах или в процентах от высоты окна браузера
WIDTH=n(%)	устанавливает ширину изображения в пикселах или в процентах
HSPACE=n	задает свободное пространство слева и справа от изображения (в пикселах)
VSPACE=n	задается свободное пространство над и под изображением (в пикселах)

Обратите внимание, что ширина и высота изображения могут быть заданы не только в пикселах, но и в процентах от размеров окна браузера. Многие компоненты, включаемые в состав Web-страниц (изображения, таблицы, аплеты и т. д.), позволяют задавать размер в относительных единицах (т. е. в процентах). Это позволяет уменьшить зависимость внешнего вида документа от текущих установок конкретного браузера и особенностей операционной системы. Рекомендуется задавать только один из атрибутов пары "ширина-высота" изображения, иначе рисунок может быть непропорционально деформирован и изменит свой вид.

Язык HTML позволяет задать расположение изображения относительно окружающего его текста. Атрибут ALIGN может принимать следующие значения.

Значение	Функция
TOP	выравнивает одну строку по верху изображения, остальные помещает после рисунка
MIDDLE	выравнивает одну строку по середине изображения, остальные помещает после рисунка
BOTTOM	выравнивает одну строку по низу изображения, остальные помещает после рисунка
LEFT	прижимает обтекаемое текстом изображение к левой стороне окна браузера
RIGHT	прижимает обтекаемое текстом изображение к правой стороне окна браузера

Гиперссылки

Несмотря на то, что в состав HTML-документа входят самые различные компоненты, можно сказать, что гипертекстовые ссылки - основа WWW. Если бы Web-страницы не ссылались друг на друга, содержимое Web превратилось бы в обычный набор файлов, не связанных между собой.

Для создания гипертекстовой ссылки используется пара тегов <A>... </A>. Фрагмент текста, изображение или любой другой объект, расположенный между этими тегами, отображается в окне браузера как гипертекстовая ссылка. Активация такого объекта приводит к загрузке в окно браузера нового документа или к отображению другой части текущей Web-страницы. Гипертекстовая ссылка формируется с помощью выражения: <A HREF="URL_ресурса"> фрагмент документа </A>, HREF здесь является обязательным атрибутом, значение которого и есть URL-адрес запрашиваемого ресурса.

Текстовые указатели, т. е. фрагменты текста, являющиеся ссылками, не отличаются разнообразием внешнего вида. Обычно такой указатель представляет собой слово или слова, подчеркнутые прямой линией. Цвет указателя может регулироваться автором и установками программы просмотра.

Использование таблиц в HTML

Теги HTML для создания таблиц первоначально предназначались для представления строк и столбцов табулированных данных. Однако дизайнеры научились с их помощью управлять разметкой веб-страниц: создавать столбцы текста, задавать интервалы между элементами и изменять внешний вид текста способами, недоступными другим тегам форматирования HTML.

Таблицы в языке HTML всегда имеют прямоугольный вид и состоят из строк, которые в свою очередь состоят из ячеек. Все языковые конструкции, описывающие компоненты создаваемой таблицы, заключаются между тегами<TABLE> и </TABLE>.

Заполнение таблицы происходит построчно. Для обозначения строки используется пара тегов <TR>... </TR>. Строка состоит из ячеек, для задания которых используют либо теги <TH>... </TH>, если эти ячейки содержат заголовки столбцов, либо теги <TD>... </TD>. Заголовки выводятся полужирным шрифтом и располагаются по центру ячейки. Данные имеют обычный шрифт и выравниваются по левой стороне ячейки. Для задания заголовка всей таблицы используются теги <CAPTION> и </CAPTION>.

Основные атрибуты тега <TABLE>

Назначение основных атрибутов тега <TABLE> и значения, которые они могут принимать перечислены в таблице.

Атрибут	Назначение
BORDER=n	определяет ширину рамки таблицы (в пикселах), например, BORDER=1 ; значение, равное нулю, означает отсутствие рамки
WIDTH=n	определяет ширину всей таблицы в пикселах, либо в процентах от ширины окна браузера
HEIGHT=n	определяет высоту всей таблицы в пикселах, либо в процентах от высоты окна браузера
ALIGN	задает горизонтальное выравнивание таблицы в окне браузера; имеет значения left, center и right (по умолчанию - left)
CELLPADDING=n	добавляет свободное пространство между данными внутри ячейки и ее границами; по умолчанию значение равно 2
CELLSPACING=n	добавляет свободное пространство между ячейками внутри всей таблицы (по умолчанию значение равно 2)

HSPACE=n	задает области свободного пространства указанной ширины (в пикселах) слева и справа от таблицы
VSPACE=n	задает области свободного пространства заданной высоты (в пикселах) сверху и снизу от таблицы
BGCOLOR=цвет	устанавливает цвет фона всей таблицы
BACKGROUND=URL	указывает фоновое изображение для таблицы, где URL - адрес источника (имя файла с изображением)

Объединение ячеек

Смежные ячейки таблицы могут объединяться. Например, в таблице из нескольких столбцов все ячейки первой строки можно объединить и поместить в этой строке красивый заголовок таблицы. Возможно также объединение нескольких строк или создание пустой прямоугольной области.

Для соединения двух смежных ячеек в одном столбце нужно использовать атрибут ROWSPAN тега <TH> или <TD>, например, <TD ROWSPAN=2>.

Для объединения двух смежных ячеек в одной строке нужно использовать атрибут COLSPAN тех же тегов, например, <TD COLSPAN=2>.

Цвет в таблицах

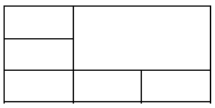
В HTML не предусмотрено специальных средств раскрашивания таблиц. Вы можете изменить цвет фона ячейки при помощи атрибута BGCOLOR перед размещением в ней текста или изображения, а также использовать атрибут BORDERCOLOR для изменения цвета рамки ячейки. Теги <TABLE>, <TD>, <TH> и <TR> также допускают использование в них указанных атрибутов. Таким образом, вы можете изменить цвет всей таблицы, отдельной ячейки или строки таблицы.

Значения цветов, установленные на уровне ячейки, будут перекрывать значения, установленные на уровне строки, которые в свою очередь, будут перекрывать значения, заданные на уровне всей таблицы.

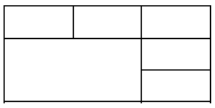
Задание для лабораторной работы 2

Задание 1. Создать веб-страницу на языке HTML, содержащую следующие таблицы:

1.



2.



Задание 2. В текстовом файле создать HTML-страницу, которая должна содержать таблицы, рисунки, ссылки, различное оформление текста (начертание, размер шрифта, выравнивание и т.д.).

Лабораторная работа №3

Фреймы и формы

Цель работы: научиться создавать веб-страницы на языке HTML, содержащие фреймы и формы.

Фреймы

Один из способов выдать сразу несколько файлов HTML на экран пользователя - это открыть несколько окон браузера. Другой путь состоит в том, чтобы разбить окно на несколько разделов. Эти разделы называются фреймами или кадрами. В каждом фрейме показывается свой HTML-документ. Каждый фрейм может иметь свои полосы прокрутки, ссылки, графические изображения и т. д. Фреймы могут функционировать независимо или влиять друг на друга, используя ссылки, указывающие на другие фреймы.

Контейнер <FRAMESET>. Web-страница, которая разделена на фреймы, называется документом группы фреймов. Документы группы фреймов содержат стандартный заголовок, задаваемый тегом <HEAD>, но в отличии от стандартных HTML-документов, они не содержат тега BODY. Вместо него используется контейнер (т. е. парный тег) <FRAMESET>, который применяется для определения строк и столбцов отдельных фреймов, каждый из которых обозначается тегом <FRAME>.

Если включить контейнер <BODY> в документ, где используется контейнер <FRAMESET>, то кадры будут проигнорированы программой просмотра, и информация, содержащаяся в документах, задаваемых тегами <FRAME>, не будет выведена. Будет показана только информация, содержащаяся в контейнере BODY.

Внутри контейнера <FRAMESET> ... </FRAMESET> могут располагаться только теги <FRAME> или другие контейнеры FRAMESET.

Определение параметров кадров. Тег <FRAMESET> имеет два главных атрибута: ROWS и COLS, задающих разбиение на строки и столбцы соответственно. Ниже приведен вид контейнера FRAMESET:

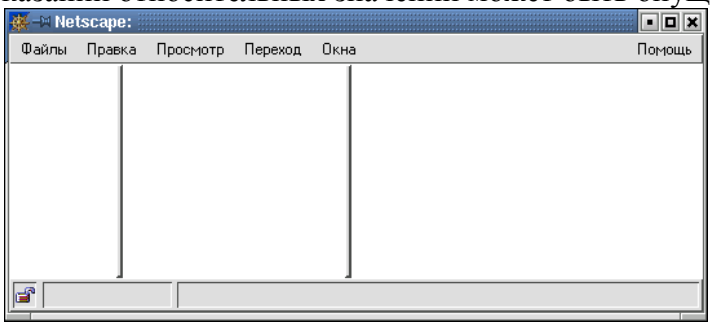
<FRAMESET ROWS="список_значений" COLS="список_значений">
...
</FRAMESET>

Можно определить любое число рядов и столбцов, необходимым условием является указание хотя бы одного из атрибутов ROWS или COLS.

Кадр не может быть единственным: если вы определили единственный ряд и единственный столбец, то программа просмотра проигнорирует контейнер FRAMESET, и экран останется пустым.

Значение атрибута ROWS или COLS представляет собой строку, содержащую список значений в пикселах, процентах или относительных единицах, разделенных запятыми. Количество рядов или столбцов кадров определяется числом этих значений.

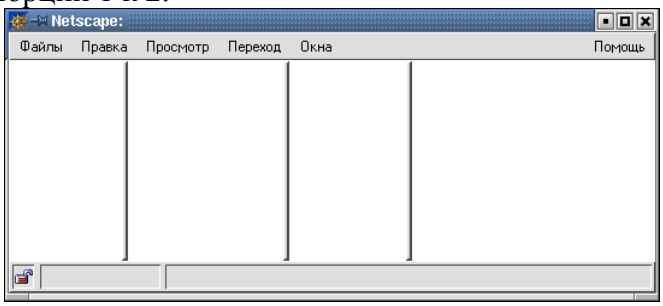
Символ * обозначает пропорциональное деление окна программы просмотра. В данном примере окно будет разделено на три вертикальных кадра, первый из которых будет иметь ширину в 1/6, второй - в 2/6 (или 1/3) и третий - в 3/6 (или 1/2) от ширины окна браузера. Единица при указании относительных значений может быть опущена.



Указание значений атрибутов ROWS и COLS может быть и смешанным, включающим любое сочетание абсолютных размеров, процентных отношений и относительных значений, например,

<FRAMESET COLS="100,25%,*,2*">

Здесь первому кадру присвоено абсолютное значение в 100 пикселов по ширине, второму - 25% от ширины окна. Оставшееся пространство делится между третьим и четвертым кадрами в пропорции 1 к 2.



Приоритеты в указаниях значений атрибутов таковы: в первую очередь (слева направо) отводится место для кадра с абсолютным значением, затем - для кадра со значением в процентах, и в последнюю очередь - для кадров с относительными величинами.

Тег <FRAME>. Тег <FRAME> определяет отдельный кадр. Он должен располагаться внутри контейнера FRAMESET.

Пример

```
<FRAMESET ROWS="*,2*">
 <FRAME>
 <FRAME>
</FRAMESET>
```

Заметьте, что этот тег не является контейнером и, в отличие от тега <FRAMESET>, не имеет закрывающего тега.

Число тегов <FRAME> обязательно должно быть равно числу кадров, определенных в теге <FRAMESET>.

Атрибуты тега <FRAME>. Строка <FRAME SRC="URL"> определяет URL-адрес содержимого кадра. Это обычно файл HTML-документа, расположенный в том же каталоге, что и документ, содержащий контейнер FRAMESET.

Атрибут MARGINWIDTH=n задает размещение слева и справа от содержимого кадра областей свободного пространства высотой по n пикселов, а MARGINHEIGHT=n, соответственно, сверху и снизу. Значения этих атрибутов всегда должны указываться в абсолютных значениях (пикселах).

К построенным вами кадрам автоматически добавляются полосы прокрутки, если содержание кадра больше его размера. Иногда это может нарушить эстетику страницы, поэтому в HTML предусмотрен атрибут SCROLLING тега <FRAME>, имеющий следующий формат: <FRAME SCROLLING="yes|no|auto">.

Этот атрибут может принимать одно из трех значений: yes, no и auto. Последнее значение подразумевается по умолчанию, т. е. когда атрибут не определен.

По умолчанию размеры кадров могут легко изменяться читателями, однако понятно, что это может сильно нарушить авторский замысел. Поэтому вы, скорее всего, захотите использовать атрибут NORESIZE тега <FRAME>, запрещающий возможность "перекраивания" вашей страницы: <FRAME NORESIZE>. Этот атрибут не имеет значений.

Для определения рамки кадра в HTML существуют три атрибута: BORDER, FRAMEBORDER и BORDERCOLOR. Первый из этих атрибутов используется только с тегом <FRAMESET> и устанавливает ширину всех рамок для всех кадров контейнера FRAMESET. Эта величина указывается в пикселах, например, <FRAMESET BORDER="10">.

Если этот атрибут нулевой, то все кадры контейнера будут без рамок. По умолчанию атрибут BORDER имеет значение 5.

Атрибут FRAMEBORDER используется с тегами <FRAMESET> и <FRAME> и может принимать два значения: yes или no. В случае yes рамка имеет трехмерную форму. Если FRAMEBORDER="no", рамка невидима, т. е. она имеет цвет фона окна браузера, устанавливаемого по умолчанию.

По умолчанию атрибут FRAMEBORDER имеет значение yes, т. е. предусматривает наличие трехмерной рамки. Рамка кадра будет невидимой, если значение FRAMEBORDER="no" установлено для всех кадров, смежных с ним.

Атрибут BORDERCOLOR может использоваться с тегами <FRAMESET> и <FRAME>. Ему может быть присвоено стандартное имя цвета или RGB-значение.

Формы в HTML-документах

Формы HTML позволяют получать информацию от читателей. До сих пор мы обсуждали только способы вывода данных, теперь речь пойдет об обратном действии. Формы дают возможность запрашивать информацию в виде свободного текста, получать ответы типа "да/нет" или делать выбор из нескольких опций.

Тегом <FORM> начинается каждая форма. В нем нужно определить два атрибута, указывающих используемый скрипт и метод посылки данных.

Атрибут	Назначение
ACTION	определяет URL, который примет и обработает данные формы. Если этот атрибут не определен, данные отправляются по адресу страницы, на которой помещена форма
METHOD	указывает форме, как послать информацию соответствующей программе обработки (скрипту). Обычно он получает значение post, тогда информация формы посылается отдельно от URL. Значению get соответствует посылка вместе с URL

Работа с тегами форм

В HTML существует три тега для создания различного типа полей в форме: <TEXTAREA>, <SELECT> и <INPUT>. Любое их количество может быть размещено в контейнере между тегами <FORM> и </FORM>. Ниже дано их краткое описание, а подробнее они будут рассмотрены чуть позже.

Тег	Назначение
<TEXTAREA>	определяет поле, в которое пользователь вводит многострочную текстовую информацию
<SEECT>	позволяет пользователю сделать выбор в окне с полосой прокрутки, либо в раскрывающемся меню
<INPUT>	обеспечивает некоторые другие виды ввода информации: ввод одной строки текста, установку и сброс флажков (check boxes), выбор переключателя (radio buttons) и нажатие кнопки для отправки данных или очистки формы

Тег <TEXTAREA>. Этот тег предназначен для построения поля с целью ввода многострочный текстовой информации. В контейнере TEXTAREA допускается размещать любой текст, который будет выведен в поле ввода по умолчанию. Перечислим атрибуты этого тега.

Атрибут	Назначение
NAME	определяет название поля

ROWS	устанавливает высоту поля, т. е. число строк в нем
COLS	устанавливает ширину поля, т. е. длину строки

При помощи атрибутов ROWS и COLS можно задать поле любого размера. Хотя эти атрибуты не являются обязательными, они не имеют определенных значений по умолчанию (для каждого браузера эти значения различны), поэтому лучше их всегда указывать явно.

Тег <SELECT>. Этот тег используется для создания всплывающего меню или списка опций с полосой прокрутки. Список опций и пункты меню располагаются внутри контейнера SELECT. Аналогично тегу <TEXTAREA>, <SELECT> требует обязательного определения имени в атрибуте NAME. Количество опций указывается в атрибуте SIZE. Ниже перечислены атрибуты тега <SELECT>.

Атрибут	Назначение
NAME	определяет название информации
SIZE	определяет вертикальный размер окна для опций выбора. Если атрибут опущен или его значение равно 1, выводится всплывающий список опций. Если указано число больше единицы, то опции выводятся в окне с полосой прокрутки. Если значение атрибута больше, чем фактическое количество элементов списка, добавляются пустые строки. При их выборе пользователем возвращаются пустые поля
MULTIPLE	позволяет выбирать сразу нескольких опций

Список опций включается в контейнер <SELECT> при помощи тега <OPTION>. Этот тег имеет два атрибута.

Атрибут	Назначение
VALUE	указывает значение, возвращаемое программе обработки (скрипту), в случае выбора опции пользователем
SELECTED	указывает на опцию, выбранную по умолчанию

Тег <INPUT>. Тег <INPUT>, в отличие от <TEXTAREA> и <SELECT>, является одиночным тегом. Он предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных (Submit) и для очистки формы (Reset, Clear). Тег <INPUT> располагает следующими атрибутами.

Атрибут	Назначение
SIZE	указывает размер поля ввода в символах
MAXLENGTH	определяет максимально возможное число символов, вводимых в поле
VALUE	для текстового поля определяет текст, выводимый по умолчанию. Для флажков и переключателей указывает значение, возвращаемое программе обработки. Для кнопок отправки и очистки формы определяет надпись на кнопке
CHECKED	устанавливает флажок или переключатель во включенное состояние по умолчанию. С другими типами тегов <INPUT> не употребляется
TYPE	устанавливает тип поля ввода

Тип поля ввода, атрибут TYPE. Атрибут TYPE тега <INPUT> может принимать следующие значения.

Атрибут	Назначение
TEXT	является значением по умолчанию и предполагает создание одной строки для ввода данных. Для этого типа поля ввода употребляются атрибуты NAME (обязательный), SIZE, VALUE и MAXLENGTH
PASSWORD	позволяет заменять вводимые символы пароля звездочками. Для этого типа поля ввода используются атрибуты NAME (обязательный), SIZE, MAXLENGTH и VALUE
CHECKBOX	позволяет вывести поле для установки флажка в виде маленького квадратика, в котором может быть произведена отметка опции "галочкой". Может использоваться совместно с атрибутами NAME(обязательный), VALUE и CHECKED (определяет установленный по умолчанию флажок). Флажки обычно употребляются, когда можно выбрать сразу несколько опций из числа предложенных. Нужно быть очень осторожным в использовании флажков и переключателей, если цвет фона страницы определяется не документом, а пользователем при помощи установок программы просмотра. Не допускайте, чтобы опции сливались с фоном страницы
RADIO	позволяет выбрать только одну из представленного числа опций. Переключатели можно группировать, задавая одно и то же значение атрибута NAME (обязательный). Так же используются атрибуты VALUE иCHECKED
RESET	позволяет создать кнопку для очистки формы. Атрибут VALUE может быть использован здесь для наименования этой кнопки (по умолчанию кнопка имеет надпись Reset)
SUBMIT	используется для создания кнопки, по нажатию которой введенные данные отправляются на сервер для обработки программой-скриптом. В атрибуте VALUE может быть указано название для этой кнопки (по умолчанию - Submit Query)

Задания для лабораторной работы

Задание 1. Создать html-страницы с использованием фреймов на одну из тем: цветы, кино, музыка, известные личности, любимый город, животные, IT-технологии или на другую самостоятельно выбранную тему.

Задание 2. Создать html-страницы с использованием форм согласно варианту.

Номер варианта	**Задание**
1	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта автомобилей. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
2	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта мебели продукции. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов

	списков.
3	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта бытовой техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
4	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта компьютерной техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.
5	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта молочной продукции. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.
6	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта компьютерной техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.
7	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта аудио и видео техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
8	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта бытовой техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
9	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта автомобилей. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
10	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта мебели продукции. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
11	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта компьютерной техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.
12	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта молочной продукции. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.
13	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта компьютерной техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit.

14	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта аудио и видео техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
15	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта бытовой техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.

Лабораторная работа №4
Каскадные таблицы стилей (CSS)

Цель работы: научиться использовать каскадные таблицы стилей при создании веб-сайтов.

Каскадные таблицы стилей (CSS)

Каскадные таблицы стилей (Cascading Style Sheets, CSS) — это стандарт, определяющий представление данных в браузере. Если HTML предоставляет информацию о структуре документа, то таблицы стилей сообщают как он должен выглядеть.

Стиль — это совокупность правил, применяемых к элементу гипертекста и определяющих способ его отображения. Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля и расположение объектов на странице.

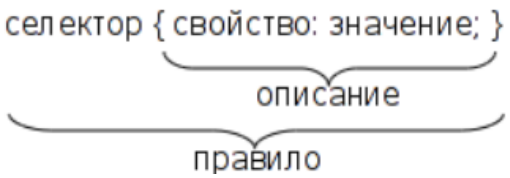
Таблица стилей — это совокупность стилей, применимых к гипертекстовому документу.

Каскадирование — это порядок применения различных стилей к веб-странице. Браузер, поддерживающий таблицы стилей, будет последовательно применять их в соответствии с приоритетом: сначала связанные, затем внедренные и, наконец, встроенные стили. Другой аспект каскадирования — наследование (inheritance), — означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементам гипертекстового документа.

Использование каскадных таблиц дает возможность разделить содержимое и его представление и гибко управлять отображением гипертекстовых документов путем изменения стилей.

Общий синтаксис таблиц стилей

Таблицы стилей строятся в соответствии с определенным порядком (синтаксисом), в противном случае они не могут нормально работать. Таблицы стилей составляются из определенных частей (рис. 1):



Селектор — это элемент, к которому будут применяться назначаемые стили. Это может быть тег, класс или идентификатор объекта гипертекстового документа. Свойство определяет одну или несколько характеристик селектора. Свойства задают формат отображения селектора: отступы, шрифты, выравнивание, размеры и т.д. Значения — это фактические числовые или строковые константы, определяющие свойство селектора.

Таким образом, таблица стилей — это набор правил, задающих значения свойств селекторов, перечисленных в этой таблице. Общий синтаксис описания правила выглядит так:

```
селектор[, селектор[, ...]] {свойство: значение;}
```

Регистр символов значения не имеет, порядок перечисления селекторов в таблице и свойств в определении не регламентирован.

Правила CSS

Итак, каскадная таблица стилей — это набор правил форматирования тегов HTML. Приведем несколько примеров написания таких правил:

Основной текст с выравниванием по ширине, абзацный отступ 30px, гарнитура (шрифт) — Serif, кегль (размер шрифта) — 14px:

```
p {
text-align: justify;
text-indent: 30px;
font-family: Serif;
font-size: 14px;
}
```

Это правило будет применено ко всем тегам <p>.

1. Синий цвет для заголовков с первого по третий уровень:

```
h1, h2, h3 {color: blue; /* тоже самое, что и #0000FF */}
```

2. Таблицы и изображения выводить без обрамления:

```
table, img {border: none;}
```

3. Ссылки в элементах списков показывать без подчеркивания:

```
li a {text-decoration: none;}
```

4. Внутренние отступы слева и справа для блоков ( ), заголовков таблиц и ячеек таблиц установить в 10px и залить фон желтым цветом:

```
div, th, td {
padding-left: 10px;
padding-right: 10px;
background-color: yellow;}
```

5. Все ссылки в документе отображать черным цветом и полужирным шрифтом, а в основном тексте и списках — обычным, а также выделять их зеленым цветом и подчеркивать только при наведении курсора (в описании правил использован псевдоэлемент a:hover).

```
a {color: black; font-weight: bold;}
p a, li a {font-weight: normal; text-decoration: none;}
p a:hover, li a:hover {
color: #00FF00; text-decoration: underline;}
```

Классы

Стандарт CSS представляет возможности создания именованных стилей — стилевых классов. Это позволяет ответить на такой, например, вопрос: Как применить разные стили к одному и тому же селектору?

Предположим, что в документе вам нужны два различных вида основного текста — один без отступа, второй — с левым отступом и шрифтом красного цвета.

Для этого нужно создать правила для каждого из них, например так:

```
p {margin-left: 0;}
p.warn {margin-left: 40px; color: #FF00;}
```

Для применения созданного класса его имя нужно указать в атрибуте class для выбранных абзацев:

Красный текст с отступом слева

Общий синтаксис описания класса: селектор.имя_класса {описание}.

При создании класса селектор можно не указывать, тогда это правило можно применять к любому селектору, поддерживающему тот же набор свойств.

Идентификаторы

В качестве селектора может выступать идентификатор элемента гипертекста, указанный в атрибуте id. Для назначения стилей таким элементам используется синтаксис, аналогичный описанию классов, но вместо точки ставится знак # (“решетка”). Например:

```
div#content {
position: absolute;
top: 10px;
left: 10%;
right: 10%;
border: solid 1px silver;}
...
<div id="content">Текст</div>
```

Следует помнить, что идентификаторы элементов должны быть уникальны в пределах документа.

Группировка свойств

Группировка (grouping) состоит в объединении значений родственных свойств. При этом таблица стилей становится более компактной, но предъявляются более жесткие требования к описанию правил. Ниже приведен пример обычного стиля, задающего отступы:

```
div {margin-left: 10px;
margin-top: 5px;
margin-right: 40px;
margin-bottom: 15px;}
```

Это же правило можно переписать с группировкой в следующем виде:

```
div {margin: 5px 40px 15px 10px;} /*порядок: top right bottom left*/
```

Оба стиля будут отображаться одинаково.

Группировка может применяться для таких свойств, как padding, font, border, background и еще некоторых (см. документацию CSS)

Использование в веб-страницах

Существует три способа применения таблицы стилей к документу HTML:

Встраивание (Inline). Этот метод позволяет применить стиль к заданному тегу HTML.

Внедрение (Embedded). Внедрение позволяет управлять стилями страницы целиком.

Связывание (Linked или External). Связанная таблица стилей позволяет вынести описание стилей во внешний файл, ссылаясь на который можно контролировать отображение всех страниц сайта.

Встроенные стили

Встраивание стилей предоставляет максимальный контроль над всеми элементами веб-страницы. Встроенный стиль применяется к любому тегу HTML с помощью атрибута style следующим образом:

Это текст с кеглем 12 точек и гарнитурой Courier</P>

Пример:

<div style="font-family: Garamond; font-size: 18 pt;>"

Весь текст в этом разделе имеет размер 18 точек и шрифт Garamond.

А этот фрагмент еще и выделен красным цветом.

Встроенные стили полезны, когда необходима тонкая настройка отображения некоторого элемента страницы или небольшой веб-страницы.

Внедренные стили

Внедренные стили используют тег <style>, который обычно размещают в заголовке HTML-документа (<head>...</head>):

<html>
<head>
...
<style>
правила CSS
</style>
...
</head>
<body>
...

Связанные таблицы стилей

Связанные (linked), или *внешние (external)* таблицы стилей — наиболее удобное решение, когда речь идет об оформлении целого сайта. Описание правил помещается в отдельный файл (обычно, но не обязательно, с расширением *.css*). С помощью тега <link> выполняется связывание этой таблицы стилей с каждой страницей, где ее необходимо применить, например так: <link rel=stylesheet href="sample.css" type="text/css">

Любая страница, содержащая такую связь, будет оформлена в соответствии со стилями, указанными в файле sample.css. Следует отметить, что файл со стилями физически может находиться на другом веб-сервере, тогда в href нужно указать абсолютный путь к нему.

CSS-свойства: размеры, цвета, шрифты, текст

Размеры. Размеры в CSS можно задавать в различных единицах измерения: **pt** – пункты (типографская единица измерения шрифта), **px** – пиксель, **%** – процент. Гораздо реже используется указание размеров в миллиметрах (mm), сантиметрах (cm) и дюймах (in).

Единица измерения записывается сразу за значением без пробела: TABLE {font-size: 12pt}.

Цвета. В CSS цвет задается как и в HTML – 6 шестнадцатеричными цифрами или стандартными названиями цветов на английском.

URL. URL задаются конструкцией url(…). Например, следующий CSS-код добавляет фоновое изображение для страницы: BODY {background-image: url(images/bg.jpg);}

Шрифты. Шрифт – набор начертаний букв и знаков. В компьютере шрифт представляет собой файл, в котором описано, как должны отображаться на мониторе или принтере различные символы: буквы, цифры, знаки пунктуации и др. Типы шрифтов:

serif – шрифты с засечками (антиквенные), например: Times New Roman, Georgia.

sans-serif – рубленные шрифты (шрифты без засечек или гротески), типичные представители – Arial, Impact, Tahoma, Verdana;

cursive – курсивные шрифты: Comic Sans MS;

fantasy – декоративные шрифты, например: Curlz MT.

monospace – моноширинные шрифты, ширина каждого символа одинакова.

Текст. CSS позволяет управлять свойствами шрифта и текста.

font-family – задает начертание шрифта. Можно указать несколько значений через запятую. Браузер проверит первый шрифт из списка: если шрифт установлен на компьютере пользователя, то браузер применит его, если нет – перейдет ко второму шрифту и т.д. Последним в списке обычно указывается общий тип шрифта serif, sans-serif, cursive, fantasy или monospace.

font-size – размер шрифта. Может задаваться абсолютным значением в пунктах (pt) или пикселях (px) или в процентах (%) или в em.

font-style – задает начертание текста: normal (обычное), italic (курсивное) или oblique (наклонное). Курсивное начертание является специальной измененной версией шрифта, имитирующей рукописный текст с наклоном вправо. Наклонное начертание получается из обычного наклоном букв.

font-weight – позволяет изменить уровень жирности текста: normal (обычная), bold (полужирная). Действие аналогично тегу <b>.

color – задает цвет текста.

line-height – межстрочный интервал (интерлиньяж), указывает расстояние между строками текста. Может задаваться числом как множитель от текущего размера шрифта, в процентах, а также в пунктах (pt), пикселях (px) и других единицах измерения CSS.

text-decoration – задает оформление текста. Варианты: line-through (перечеркнутый), overline (линия над текстом), underline (подчеркивание), none (отключение эффектов).

text-align – выравнивание текста в блоке: left (по левому краю), center (по центру), right (по правому краю) или justify (по ширине).

text-indent – отступ первой строки. Длина отступа может задаваться в процентах (%) от ширины текстового блока, пикселях (px), пунктах (pt) и др.

Ширина полей и заполнения задается следующими CSS свойствами:

margin-top, margin-right, margin-bottom, margin-left – для верхней, правой, нижней, левой стороны поля.

Если для margin указать два значения через пробел, то первое из них будет задавать ширину верхнего и нижнего поля, а второе – левого и правого. Если указать три значения, то первое будет присваиваться верхнему полю, второе – левому и правое, а третье – нижнему. Наконец, при указании четырех значений, они поочередно будут указывать верхнее, правое, нижнее и левое поля.

padding-top, padding-right, padding-bottom, padding-left – устанавливают ширину заполнения1 сверху, справа, снизу и слева от содержимого соответственно.

padding – устанавливает значение сразу для всех сторон. Padding может принимать не только одно, но и 2, 3 или 4 значения.

Для margin и padding можно задавать значение auto. В этом случае браузер сам автоматически рассчитает величину полей и заполнения.

Для границ можно задать толщину, цвет и стиль:

border-width – толщина границы;

border-color – цвет границы (по умолчанию – черный);

border-style – стиль границы. Может принимать значения solid (по умолчанию), dotted, dashed, double, groove, ridge, inset или outset.

CSS-свойства: фон, оформление таблиц

Фон. Как и в языке HTML, в CSS фоном служит заливка цветом или изображение. Фоновое изображение может быть повторяющимся.

background-color – устанавливает цвет фона. Пример: TD.head {background-color: #ffff00}

background-image – устанавливает в качестве фона изображение: BODY {background-image: url(images/bg.jpg)}

background-attachment – задает поведение фонового изображения при прокрутке. По умолчанию задается значение scroll – фон прокручивается вместе с содержимым. Значение fixed делает фон неподвижным.

background-position – начальное положение фонового изображения по горизонтали (left, center, right) и вертикали (top, center, bottom). Вместо ключевых слов можно указывать расстояние в пикселях или процентах.

background-repeat – указывает, в каком направлении должно размножаться фоновое изображение:

repeat – по горизонтали и вертикали (по умолчанию);

repeat-x – только по горизонтали;

repeat-y – только по вертикали;

no-repeat – отключить повторение.

Теги DIV и SPAN

До сих пор мы применяли стили CSS к тегам, уже имеющим заранее заданную функцию: таблицам, заголовкам, параграфам и т.д. Но иногда нужно применить стили к фрагменту содержимого, не включенного в отдельный тег. Например, выделить фоном несколько слов в тексте.

Теги … и … используются там, где не подходит никакой другой тег. Сами по себе они не определяют никакого форматирования, но удобны для привязки к ним стилей. При этом DIV является блочным элементом, а SPAN – строчным.

Основное различие между блочными и строчными элементами заключается в следующем: строчные элементы идут друг за другом в строке текста, а блочные – располагаются один по другим. К строчным элементам относятся такие теги, как <a>, <img>, <input>, <select>, , <sub>, <sup> и др. К блочным: , <form>, <h1>…<h6>, <ol>, <p>, <table>, <ul> и некоторые другие.

Блочные элементы располагаются друг под другом, многие занимают всю возможную ширину. Блочные элементы могут включать в себя строчные и другие блочные. Но строчные элементы не могут содержать блочные.

Еще одним отличием является то, что для строчных элементов не работают такие свойства, как margin-top, margin-bottom, padding-top и paddingbottom.

Исключением являются теги <img>, <input>, <textarea> и <select> – для них можно задавать отступы padding-top и paddingbottom.

Задание для лабораторной работы

Задание. Модифицировать разработанные в рамках лабораторной работы №3 страницы (страницы с фреймами), внедрив в них поддержку CSS. Использовать три способа подключения каскадных таблиц стилей: связанные, внедренные и встроенные.

С помощью CSS задайте следующие параметры:

- размер, цвет, шрифт текста на странице, межстрочный интервал, красная строка, выравнивание;
- фон страницы;
- при оформлении страницы использовать теги DIV и SPAN.

Лабораторная работа №5
Основы синтаксиса языка PHP

Цель работы: познакомиться с синтаксисом языка PHP.

Возможности PHP

В первую очередь PHP используется для создания скриптов, работающих на стороне сервера, для этого его, собственно, и придумали. PHP способен решать те же задачи, что и любые другие CGI - скрипты, в том числе обрабатывать данные html-форм, динамически генерировать html страницы и т.п. Но есть и другие области, где может использоваться PHP . Всего выделяют три основные области применения PHP.

Первая область, как уже говорилось, – это создание приложений (скриптов), которые исполняются на стороне сервера.

Вторая область – это создание скриптов, выполняющихся в командной строке. То есть с помощью PHP можно создавать такие скрипты, которые будут исполняться, вне зависимости от web-сервера и браузера, на конкретной машине.

И последняя область – это создание GUI -приложений (графических интерфейсов), выполняющихся на стороне клиента.

Первая PHP-программа
Рассмотрим пример:

```
<html>
  <head>
  <title>Пример</title>
  </head>
  <body>
  <?php
  echo "<p>Привет, я – скрипт PHP!</p>";
  ?>
</body>
</html>
```

Это простой HTML-файл, в который встроен с помощью специальных тегов код, написанный на языке PHP.

Основной синтаксис

Первое, что нужно знать относительно синтаксиса PHP, – это то, как он встраивается в HTML-код, как интерпретатор узнает, что это код на языке PHP. В предыдущей лекции мы уже говорили об этом. Повторяться не будем, отметим только, что в примерах мы чаще всего будем использовать вариант <?php ?>, и иногда сокращенный вариант<? ?>.

Разделение инструкций

Программа на PHP (да и на любом другом языке программирования) – это набор команд (инструкций). Обработчику программы необходимо как-то отличать одну команду от другой. Для этого используются специальные символы – разделители. В PHP инструкции разделяются так же, как и в Си или Perl, – каждое выражение заканчивается точкой с запятой.

Закрывающий тег " ?> " также подразумевает конец инструкции, поэтому перед ним точку с запятой не ставят.

Комментарии

Часто при написании программ возникает необходимость делать какие-либо комментарии к коду, которые никак не влияют на сам код, а только поясняют его. Это важно при создании больших программ и в случае, если несколько человек работают над одной программой. При наличии комментариев в программе в ее коде разобраться гораздо проще. Кроме того, если решать задачу по частям, недоделанные части решения также удобно комментировать, чтобы не забыть о них в дальнейшем. Во всех языках программирования предусмотрена возможность включать комментарии в код программы.

PHP поддерживает несколько видов комментариев: в стиле Си, C++ и оболочки Unix. Символы // и # обозначают начало однострочных комментариев, /* и */ – соответственно начало и конец многострочных комментариев.

Переменные, константы и операторы

Важным элементом каждого языка являются переменные, константы и операторы, применяемые к этим переменным и константам. Рассмотрим, как выделяются и обрабатываются эти элементы в PHP.

Переменные

Переменная в PHP обозначается знаком доллара, за которым следует ее имя. Например: $my_var. Имя переменной чувствительно к регистру, т.е. переменные $my_var и $My_var различны.

Имена переменных соответствуют тем же правилам, что и остальные наименования в PHP: правильное имя переменной должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

В PHP возможно присваивание значения переменной по значению и по ссылке. Для того, чтобы присвоить значение переменной по ссылке, это значение должно иметь имя, т.е. оно должно быть представлено какой-либо переменной. Чтобы указать, что значение одной переменной присваивается другой переменной по ссылке, нужно перед именем первой переменной поставить знак амперсанд &.

Константы

Для хранения постоянных величин, т.е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются константы. Такими величинами могут быть математические константы, пароли, пути к файлам и т.п. Основное отличие константы от переменной состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у константы нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения. Для определения констант существует специальная функция define() . Ее синтаксис таков:

define("Имя_константы", "Значение_константы", [Нечувствительность_к_регистру]).

По умолчанию имена констант чувствительны к регистру. Для каждой константы это можно изменить, указав в качестве значения аргумента Нечувствительность_к_регистру значение True.

Получить значение константы можно, указав ее имя. В отличие от переменных, не нужно предварять имя константы символом. Кроме того, для получения значения константы можно использовать функцию constant() с именем константы в качестве параметра.

Операторы

Операторы позволяют выполнять различные действия с переменными, константами и выражениями.

Арифметические операторы

Обозначение	Название	Пример
+	Сложение	$a + $b
-	Вычитание	$a - $b
*	Умножение	$a * $b
/	Деление	$a / $b
%	Остаток от деления	$a % $b

Строковые операторы

Обозначение	Название	Пример
.	Конкатенация (сложение строк)	$c = $a . $b (это строка, состоящая из $a и $b)

Операторы присваивания

Обозначение	Название	Описание	Пример
=	Присваивание	Переменной слева от оператора будет присвоено значение, полученное в результате выполнения каких-либо операций или переменной / константы с правой стороны	$a = ($b = 4) +5; ($a будет равна 9,$b будет равна 4)
+=		Сокращение. Прибавляет к переменной число и затем присваивает ей полученное значение	$a += 5; (эквивалентно $a = $a + 5;)
.=		Сокращенно обозначает комбинацию операций конкатенациии присваивания (сначала добавляется строка, потом полученная строка записывается в переменную)	$b = "Привет "; $b .= "всем"; (эквивалентно $b = $b . "всем";) В результате:$b="Привет всем"

Логические операторы

Обозначение	Название	Описание	Пример
and	И	$a и $b истинны (True)	$a and $b
&&	И		$a&&$b
or	Или	Хотя бы одна из переменных $a или $b истинна (возможно, что и обе)	$a or $b
||	Или		$a || $b
xor	Исключающее или	Одна из переменных истинна. Случай, когда они обе истинны, исключается	$a xor $b
!	Инверсия (NOT)	Если $a=True, то !$a=False и наоборот	! $a

Операторы инкремента и декремента

Обозначение	Название	Описание
++$a	Пре- инкремент	Увеличивает $a на единицу и возвращает $a
$a++	Пост- инкремент	Возвращает $a, затем увеличивает $a на единицу
--$a	Пре- декремент	Уменьшает $a на единицу и возвращает $a

$a--	Пост- декремент	Возвращает $a, затем уменьшает $a на единицу

Типы данных

PHP поддерживает восемь простых типов данных.

Четыре скалярных типа:

- boolean (логический) ;
- integer (целый) ;
- float (с плавающей точкой) ;
- string (строковый).

Два смешанных типа:

- array (массив) ;
- object (объект).

И два специальных типа:

- resource (ресурс) ;
- NULL.

В PHP не принято явное объявление типов переменных. Предпочтительнее, чтобы это делал сам интерпретатор во время выполнения программы в зависимости от контекста, в котором используется переменная. Рассмотрим по порядку все перечисленные типы данных.

Тип array (массив)

Массив в PHP представляет собой упорядоченную карту – тип, который преобразует значения в ключи. Этот тип оптимизирован в нескольких направлениях, поэтому вы можете использовать его как собственно массив, список (вектор), хеш-таблицу (являющуюся реализацией карты), стэк, очередь и т.д. Поскольку вы можете иметь в качестве значения другой массив PHP, можно также легко эмулировать деревья.

Определить массив можно с помощью конструкции array () или непосредственно задавая значения его элементам.

Определение при помощи array(): array ([key] => value, [key1] => value1, ...).

Языковая конструкция array() принимает в качестве параметров пары ключ => значение, разделенные запятыми. Символ => устанавливает соответствие между значением и его ключом. Ключ может быть как целым числом, так и строкой, а значение может быть любого имеющегося в PHP типа. Числовой ключ массива часто называют индексом. Индексирование массива в PHP начинается с нуля. Значение элемента массива можно получить, указав после имени массива в квадратных скобках ключ искомого элемента. Если ключ массива представляет собой стандартную запись целого числа, то он рассматривается как число, в противном случае – как строка. Поэтому запись $a["1"] равносильна записи $a[1], так же как и $a["-1"] равносильно $a[-1].

Если для элемента ключ не задан, то в качестве ключа берется максимальный числовой ключ, увеличенный на единицу. Если указать ключ, которому уже было присвоено какое-то значение, то оно будет перезаписано. Начиная с PHP 4.3.0, если максимальный ключ – отрицательное число, то следующим ключом массива будет ноль.

Определение с помощью синтаксиса квадратных скобок. Создать массив можно, просто записывая в него значения. Как мы уже говорили, значение элемента массива можно получить с помощью квадратных скобок, внутри которых нужно указать его ключ, например, $book["php"]. Если указать новый ключ и новое значение, например,$book["new_key"]="new_value", то в массив добавится новый элемент. Если мы не укажем ключ, а только присвоим значение $book[]="new_value", то новый элемент массива будет иметь числовой ключ, на единицу больший максимального

существующего. Если массив, в который мы добавляем значения, еще не существует, то он будет создан.

Для того чтобы изменить конкретный элемент массива, нужно просто присвоить ему с его ключом новое значение. Изменить ключ элемента нельзя, можно только удалить элемент (пару ключ / значение) и добавить новую. Чтобы удалить элемент массива, нужно использовать функцию unset () .

Заметим, что, когда используются пустые квадратные скобки, максимальный числовой ключ ищется среди ключей, существующих в массиве с момента последнего переиндексирования. Переиндексировать массив можно с помощью функции array_values ().

Условные операторы

Оператор if

Это один из самых важных операторов многих языков, включая PHP. Он позволяет выполнять фрагменты кода в зависимости от условия. Структуру оператора if можно представить следующим образом: if (выражение) блок_выполнения.

Здесь выражение есть любое правильное PHP-выражение (т.е. все, что имеет значение). В процессе обработки скрипта выражение преобразуется к логическому типу. Если в результате преобразования значение выражения истинно (True), то выполняется блок_выполнения. В противном случае блок_выполнения игнорируется. Если блок_выполнения содержит несколько команд, то он должен быть заключен в фигурные скобки { }.

Правила преобразования выражения к логическому типу:

1. В FALSE преобразуются следующие значения:
   - логическое False
   - целый ноль (0)
   - действительный ноль (0.0)
   - пустая строка и строка "0"
   - массив без элементов
   - объект без переменных (подробно об объектах будет рассказано в одной из следующих лекций)
   - специальный тип NULL
2. Все остальные значения преобразуются в TRUE.

Оператор else

Мы рассмотрели только одну, основную часть оператора if . Существует несколько расширений этого оператора. Оператор else расширяет if на случай, если проверяемое в if выражение является неверным, и позволяет выполнить какие-либо действия при таких условиях.

Структуру оператора if , расширенного с помощью оператора else , можно представить следующим образом:

if (выражение) блок_выполнения else блок_выполнения1.

Эту конструкцию if...else можно интерпретировать примерно так: если выполнено условие (т.е. выражение=true), то выполняем действия изблока_выполнения, иначе – действия из блока_выполнения1. Использовать оператор else не обязательно.

Оператор elseif

Еще один способ расширения условного оператора if – использование оператора elseif . elseif – это комбинация else и if . Как и else , он расширяет if для выполнения различных действий в том случае, если условие, проверяемое в if , неверно. Но в отличие от else , альтернативные действия будут выполнены, только если elseif-условие

является верным. Структуру оператора if , расширенного с помощью операторов else и elseif, можно представить следующим образом:

if (выражение) блок_выполнения
elseif(выражение1) блок_выполнения1
...
else блок_выполненияN

Операторов elseif может быть сразу несколько в одном if-блоке. Elseif-утверждение будет выполнено, только если предшествующее if-условие является False, все предшествующие elseif-условия являются False, а данное elseif-условие – True.

Оператор switch

Еще одна конструкция, позволяющая проверять условие и выполнять в зависимости от этого различные действия, – это switch . На русский язык название данного оператора можно перевести как "переключатель". И смысл у него именно такой. В зависимости от того, какое значение имеет переменная, он переключается между различными блоками действия. switch очень похож на оператор if...elseif...else или набор операторов if . Структуру switch можно записать следующим образом:

```
switch (выражение или переменная){
case значение1:
   блок_действий1
break;
case значение2:
   блок_действий2
break;
...
default:
   блок_действий_по_умолчанию}
```

В отличие от if , здесь значение выражения не приводится к логическому типу, а просто сравнивается со значениями, перечисленными после ключевых слов case (значение1, значение2 и т.д.). Если значение выражения совпало с каким-то вариантом, то выполняется соответствующий блок_действий – от двоеточия после совпавшего значения до конца switch или до первого оператора break , если таковой найдется. Если значение выражения не совпало ни с одним из вариантов, то выполняются действия по умолчанию (блок_действий_по_умолчанию), находящиеся после ключевого слова default. Выражение в switch вычисляется только один раз, а в операторе elseif – каждый раз, поэтому, если выражение достаточно сложное, то switch работает быстрее.

Циклы

В PHP существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы while , do..while , foreach и for . Рассмотрим их более подробно.

while

Структура:

while (выражение) { блок_выполнения }

либо

while (выражение): блок_выполнения endwhile;

while – простой цикл. Он предписывает PHP выполнять команды блока_выполнения до тех пор, пока выражение вычисляется как True (здесь, как и в if , происходит приведение выражения к логическому типу). Значение выражения проверяется каждый раз в начале цикла, так что, даже если его значение изменилось в процессе

выполнения блока_выполнения, цикл не будет остановлен до конца итерации (т.е. пока все команды блока_выполнения не будут исполнены).

do... while

Циклы do..while очень похожи на циклы while , с той лишь разницей, что истинность выражения проверяется в конце цикла, а не в начале. Благодаря этому блок_выполнения цикла do...while гарантированно выполняется хотя бы один раз.

Структура: do {блок_выполнения} while (выражение).

for

Это самые сложные циклы в PHP. Они напоминают соответствующие циклы C.

Структура:

for (выражение1; выражение2; выражение3) {блок_выполнения}

либо

for (выражение1; выражение2; выражение3): блок_выполнения endfor;

Здесь, как мы видим, условие состоит сразу из трех выражений. Первое выражение выражение1 вычисляется безусловно один раз в начале цикла. В начале каждой итерации вычисляется выражение2. Если оно является True, то цикл продолжается и выполняются все команды блока_выполнения. Если выражение2 вычисляется как False, то исполнение цикла останавливается. В конце каждой итерации (т.е. после выполнения всех команд блока_выполнения) вычисляется выражение3.

Каждое из выражений 1, 2, 3 может быть пустым. Если выражение2 является пустым, то это значит, что цикл должен выполняться неопределенное время (в этом случае PHP считает это выражение всегда истинным). Это не так бесполезно, как кажется, ведь цикл можно останавливать, используя оператор break .

foreach

Еще одна полезная конструкция и предназначена исключительно для работы с массивами.

Синтаксис:

foreach ($array as $value) {блок_выполнения}

либо

foreach ($array as $key => $value) {блок_выполнения}

В первом случае формируется цикл по всем элементам массива, заданного переменной $array. На каждом шаге цикла значение текущего элемента массива записывается в переменную $value, и внутренний счетчик массива передвигается на единицу (так что на следующем шаге будет виден следующий элемент массива). Внутри блока_выполнения значение текущего элемента массива может быть получено с помощью переменной $value. Выполнение блока_выполнения происходит столько раз, сколько элементов в массиве $array.

Вторая форма записи в дополнение к перечисленному выше на каждом шаге цикла записывает ключ текущего элемента массива в переменную $key, которую тоже можно использовать в блоке_выполнения.

Когда foreach начинает исполнение, внутренний указатель массива автоматически устанавливается на первый элемент.

Операторы передачи управления

Иногда требуется немедленно завершить работу цикла либо отдельной его итерации. Для этого используют операторы break и continue .

break

Оператор break заканчивает выполнение текущего цикла, будь то for , foreach , while , do..while или switch . break может использоваться с числовым аргументом, который говорит, работу скольких управляющих структур, содержащих его, нужно завершить.

continue

Иногда нужно не полностью прекратить работу цикла, а только начать его новую итерацию. Оператор continue позволяет пропустить дальнейшие инструкции из блока_выполнения любого цикла и продолжить выполнение с нового круга. Continue можно использовать с числовым аргументом, который указывает, сколько содержащих его управляющих конструкций должны завершить работу.

Операторы включения

include

Оператор include позволяет включать код, содержащийся в указанном файле, и выполнять его столько раз, сколько программа встречает этот оператор. Включение может производиться любым из перечисленных способов:

include 'имя_файла';
include $file_name;
include ("имя_файла");

Задание для лабораторной работы

Задание 1. Написать программу на PHP для решения задач согласно варианту.

№ варианта	**Задание**
1	Вычислить факториал заданного числа (n!)
2	Дана квадратная матрица A(N,N). Составить программу замены положительных элементов, расположенных выше главной диагонали и кратных 5, на 100. Исходную и скорректированную матрицы напечатать.
3	Дана квадратная матрица A(N.N). Составить программу нахождения количества четных элементов, расположенных на главной и побочной диагоналях.
4	Дана вещественная матрица A(N,M). Составить программу замены всех отрицательных элементов матрицы на элемент, имеющий максимальное абсолютное значение. Исходную и скорректированную матрицы напечатать.
5	Дана вещественная матрица A(N,M). Составить программу замены всех положительных элементов матрицы на элемент, имеющий минимальное значение. Исходную и скорректированную матрицы напечатать.
6	Дана квадратная матрица A(N,N). Составить программу замены положительных элементов, расположенных выше побочной диагонали, на число p. Значение p задать самостоятельно. Исходную и скорректированную матрицы напечатать.
7	Дана квадратная матрица A(N,N). Составить программу замены отрицательных элементов, расположенных ниже главной диагонали, на 0. Исходную и скорректированную матрицы напечатать.
8	Дана вещественная квадратная матрица A(N,N). Составить программу вычисления количества и суммы элементов, расположенных выше главной диагонали.
9	Дана вещественная матрица A(N,M). Составить программу нахождения

	максимального значения элементов матрицы и выделения элементов, имеющих это максимальное значение.
10	Дана квадратная матрица A(N,N). Составить программу подсчета количества нечетных элементов, расположенных ниже главной диагонали.
11	Дана квадратная матрица A(N,N). Составить программу замены отрицательных элементов, расположенных ниже главной диагонали и кратных 7, на 2. Исходную и скорректированную матрицы напечатать.
12	Дана квадратная матрица A(N.N). Составить программу нахождения произведения четных элементов.
13	Дана вещественная матрица A(N,M). Найти минимальный и максимальный элементы.
14	Дана вещественная матрица A(N,M). Составить программу замены всех нулевых элементов матрицы на элемент, имеющий максимальное значение. Исходную и скорректированную матрицы напечатать.
15	Дана квадратная матрица A(N,N). Составить программу замены отрицательных элементов на число p. Значение p задать самостоятельно. Исходную и скорректированную матрицы напечатать.

Лабораторная работа №6
Обработка запросов с использованием PHP

Цель работы: научиться обрабатывать запросы с использованием языка PHP.

Использование HTML-форм для передачи данных на сервер

Для передачи данных серверу в языке HTML есть специальная конструкция – формы. Формы предназначены для того, чтобы получать от пользователя информацию. Например, вам нужно знать логин и пароль пользователя для того, чтобы определить, на какие страницы сайта его можно допускать. Или вам необходимы личные данные пользователя, чтобы была возможность с ним связаться. Формы как раз и применяются для ввода такой информации. В них можно вводить текст или выбирать подходящие варианты из списка. Данные, записанные в форму, отправляются для обработки специальной программе (например, скрипту на PHP) на сервере. В зависимости от введенных пользователем данных эта программа может формировать различные web-страницы, отправлять запросы к базе данных, запускать различные приложения и т.п.

Обработка запросов с помощью PHP

До сих пор мы упоминали только, что запросы клиента обрабатываются на сервере с помощью специальной программы. На самом деле эту программу мы можем написать сами, в том числе и на языке PHP, и она будет делать с полученными данными все, что мы захотим. Для того, чтобы написать эту программу, необходимо познакомиться с некоторыми правилами и инструментами, предлагаемыми для этих целей PHP.

Внутри PHP-скрипта имеется несколько способов получения доступа к данным, переданным клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ к таким данным осуществлялся по именам переданных переменных (напомним, что данные передаются в виде пар "имя переменной, символ "=", значение переменной"). Таким образом, если, например, было передано first_name=Nina, то внутри скрипта появлялась переменная $first_name со значением Nina. Если требовалось различать, каким методом были переданы данные, то использовались ассоциативные массивы $HTTP_POST_VARS и $HTTP_GET_VARS , ключами которых являлись имена переданных переменных, а значениями – соответственно значения этих переменных. Таким

образом, если пара first_name =Nina передана методом GET , то $HTTP_GET_VARS["first_name"]="Nina".

Использовать в программе имена переданных переменных напрямую небезопасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданным с помощью HTTP-запросов, специальный массив – $_REQUEST . Этот массив содержит данные, переданные методами POST и GET , а также с помощью HTTP cookies. Это суперглобальный ассоциативный массив, т.е. его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

После введения массива $_REQUEST массивы $HTTP_POST_VARS и $HTTP_GET_VARS для однородности были переименованы в $_POST и $_GET соответственно, но сами они из обихода не исчезли из соображений совместимости с предыдущими версиями PHP. В отличие от своих предшественников, массивы $_POST и $_GET стали суперглобальными, т.е. доступными напрямую и внутри функций и методов.

Для того, чтобы сохранить возможность обработки скриптов более ранних версий, чем PHP 4.1.0, была введена директива register_globals , разрешающая или запрещающая доступ к переменным непосредственно по их именам. Если в файле настроек PHP параметр register_globals=On, то к переменным, переданным серверу методами GET и POST , можно обращаться просто по их именам (т.е. можно писать $first_name). Если же register_globals=Off, то нужно писать $_REQUEST["first_name"] или $_POST["first_name"], $_GET["first_name"],$HTTP_POST_VARS["first_name"], $HTTP_GET_VARS["first_name"]. С точки зрения безопасности эту директиву лучше отключать (т.е. register_globals=Off). При включенной директиве register_globals перечисленные выше массивы также будут содержать данные, переданные клиентом.

Задание для лабораторной работы

Задание 1. Разработать приложение, в котором создается форма для введения пользователем данных, PHP-сценарий получает данные с формы, отображает извлеченные из формы данные в окне браузера, сценарий генерирует отправку на ещё один PHP-файл, который отображает персональное приветствие пользователю.

Лабораторная работа №7
Функции в PHP

Цель работы: повторение основных тегов языка HTML, получение навыков создания пользовательских функций и использования встроенных.

Функции, определяемые пользователем

Посмотрим, как в общем виде выглядит задание (объявление) функции в PHP. Функция может быть определена с помощью следующего синтаксиса:

```
function Имя_функции (параметр1, параметр2,
  ... параметрN){
  Блок_действий
  return "значение, возвращаемое функцией";
}
```

Если прямо так написать в php-программе, то работать ничего не будет. Во-первых, Имя_функции и имена параметров функции (параметр1, параметр2 и т.д.) должны соответствовать правилам наименования в PHP (и русских символов в них лучше не использовать). Имена функций нечувствительны к регистру. Во-вторых, параметры функции – это переменные языка, поэтому перед названием каждого из них должен стоять

знак $. Никаких многоточий ставить в списке параметров нельзя. В-третьих, вместо слов блок_действий в теле функции должен находиться любой правильный PHP-код (не обязательно зависящий от параметров). И наконец, после ключевого слова return должно идти корректное php-выражение (что-либо, что имеет значение). Кроме того, у функции может и не быть параметров, как и возвращаемого значения.

Если функция однажды определена в программе, то переопределить или удалить ее позже нельзя. Несмотря на то, что имена функций нечувствительны к регистру, лучше вызывать функцию по тому же имени, каким она была задана в определении.

Аргументы функций

У каждой функции может быть, как мы уже говорили, список аргументов. С помощью этих аргументов в функцию передается различная информация (например, значение числа, факториал которого надо подсчитать). Каждый аргумент представляет собой переменную или константу.

С помощью аргументов данные в функцию можно передавать тремя различными способами. Это передача аргументов по значению (используется по умолчанию), по ссылке и задание значения аргументов по умолчанию. Рассмотрим эти способы подробнее.

Когда аргумент передается в функцию по значению, изменение значения аргумента внутри функции не влияет на его значение вне функции. Чтобы позволить функции изменять ее аргументы, их нужно передавать по ссылке. Для этого в определении функции перед именем аргумента следует написать знак амперсанд "&".

В функции можно определять значения аргументов, используемые по умолчанию. Само значение по умолчанию должно быть константным выражением, а не переменной и не представителем класса или вызовом другой функции.

Если у функции несколько параметров, то те аргументы, для которых задаются значения по умолчанию, должны быть записаны после всех остальных аргументов в определении функции. В противном случае появится ошибка, если эти аргументы будут опущены при вызове функции.

Использование переменных внутри функции

Глобальные переменные

Чтобы использовать внутри функции переменные, заданные вне ее, эти переменные нужно объявить как глобальные. Для этого в теле функции следует перечислить их имена после ключевого слова global.

Статические переменные

Чтобы использовать переменные только внутри функции, при этом сохраняя их значения и после выхода из функции, нужно объявить эти переменные как статические. **Статические переменные** видны только внутри функции и не теряют своего значения, если выполнение программы выходит за пределы функции. Объявление таких переменных производится с помощью ключевого слова static.

Возвращаемые значения

Любая функция может возвращать как результат своей работы какое-нибудь значение. Это делается с помощью утверждения return. Возвращаемое значение может быть любого типа, включая списки и объекты. Когда интерпретатор встречает команду return в теле функции, он немедленно прекращает ее исполнение и переходит на ту строку, из которой была вызвана функция.

Когда функция возвращает несколько значений для их обработки в программе, удобно использовать языковую конструкцию list (), которая позволяет одним действием присвоить значения сразу нескольким переменным.

Возвращение ссылки

В результате своей работы функция также может возвращать ссылку на какую-либо переменную. Это может пригодиться, если требуется использовать функцию для того, чтобы определить, какой переменной должна быть присвоена ссылка. Чтобы получить из функции ссылку, нужно при объявлении перед ее именем написать знак амперсанд (&) и каждый раз при вызове функции перед ее именем тоже писать амперсанд (&). Обычно функция возвращает ссылку на какую-либо глобальную переменную (или ее часть – ссылку на элемент глобального массива), ссылку на статическую переменную (или ее часть) или ссылку на один из аргументов, если он был также передан по ссылке.

Внутренние (встроенные) функции

Говоря о функциях, определяемых пользователем, все же нельзя не сказать пару слов о встроенных функциях. С некоторыми из встроенных функций, такими как echo(), print(), date(),include(), мы уже познакомились. На самом деле все перечисленные функции, кроме date(), являются языковыми конструкциями. Они входят в ядро PHP и не требуют никаких дополнительных настроек и модулей. Функция date() тоже входит в состав ядра PHP и не требует настроек. Но есть и функции, для работы с которыми нужно установить различные библиотеки и подключить соответствующий модуль. Например, для использования функций работы с базой данных MySql следует скомпилировать PHP с поддержкой этого расширения. В последнее время наиболее распространенные расширения и соответственно их функции изначально включают в состав PHP так, чтобы с ними можно работать без каких бы то ни было дополнительных настроек интерпретатора.

Задания для лабораторной работы

Задание 1. Создать сайт (не менее 5 страниц) и наполнить его текстом с изображениями в соответствии с выбранной тематикой. Необходимо использовать HTML (теги оформления текста, списков, таблиц, изображений, гиперссылок, фреймов и форм) PHP (обработка запросов).

Раздел 2. Самостоятельная работа

2.1. Проработка лекционного материала по темам лекций:

- Основные понятия языковых средств создания гипертекстовых документов
- Создание Web-страниц
- Гипертекстовые ссылки и иллюстрации на Web-страницах. Построение таблиц
- Фреймы и формы
- Каскадные таблицы стилей (CSS)
- Основы синтаксиса языка PHP
- Обработка запросов с использованием PHP
- Функции в PHP

2.2. Оформление отчетов по лабораторным работам

Темы лабораторных работ:

- Создание Web-страниц
- Гипертекстовые ссылки и иллюстрации на Web-страницах. Построение таблиц
- Фреймы и формы
- Каскадные таблицы стилей (CSS)
- Основы синтаксиса языка PHP
- Обработка запросов с использованием PHP
- Функции в PHP

Контрольные вопросы

1. Что такое HTML? Что такое гипертекстовый документ?
2. Что такое тег? Структура тега HTML. Формат записи.
3. Привести структуру HTML документа. Описать назначение тегов <html>, <head>, <meta>, <body>.
4. Что такое атрибут тега? Формат записи атрибутов.
5. Перечислить теги для представления текстового содержимого и дать их описание.
6. Как представляются гиперссылки в HTML документе?
7. Перечислить теги для создания списков в HTML.
8. Вложенные списки в HTML.
9. Как включаются графические объекты в HTML документы?
10. Куда будет указывать ссылка, если атрибут href оставить пустым (анкор)?
11. Чем отличаются операторы циклов в PHP?
12. Как определить пользовательскую функцию?
13. Как в PHP задаются массивы?
14. Чем отличаются методы GET и POST?
15. Перечислите основные типы данных в PHP.

Список литературы

1. Кручинин В. В. Разработка сетевых приложений: Учебное пособие [Электронный ресурс] / В. В. Кручинин — Томск: ТУСУР, 2013. — 120 с. — Режим доступа: https://edu.tusur.ru/publications/2835 , дата обращения: 13.05.2018
2. Ехлаков Ю. П. Основы гипертекстового представления интернет-контента: учебное пособие [Электронный ресурс] / Ю. П. Ехлаков, Э. К. Ахтямов — Томск: ТУСУР, 2017. — 181 с. — Режим доступа: https://edu.tusur.ru/publications/7086 , дата обращения: 13.05.2018