Первое знакомство с тэгами

Основным понятием языка **HTML** является **"тэг"** (англ.: *tag*-ярлычок, этикетка). **Тэги** - это заключенные в угловые скобки "<,>" последовательности букв, (образованные, как правило, сокращением английских слов).

Код любого **HTML**-документа начинается тегом **<html>** и завершается тегом **</html>**. Пара "открывающий - закрывающий" тэги называется **контейнером**.

Каждый документ имеет контейнер **<head>...</head>**, содержащий **вспомогательную информацию**, и контейнер **<body>...</body>**, содержащий **сам текст документа**.

Контейнер **<title>...</title>**, расположенный в секции **<head>...</head>**, содержит текст, отображаемый в верхней строке окна браузера.

Задание 1: создайте Ваш первый HTML-документ.

Пусть он будет озаглавлен Вашей фамилией (в титульной строке) и содержит следующую информацию:

- Вашу фамилию, имя, отчество
- год рождения
- место рождения
- адрес
- номер школы, которую Вы закончили.

Большинство тэгов предназначено для **форматирования** документов:

Код	Вид
Шампунь для жирных котов	Шампунь для жирных котов
Спагетти, пицца, капуччино	Спагетти, пицца, капуччино
<strike>Вычеркнуть из памяти</strike>	~~Вычеркнуть из памяти~~
<u>Я подчеркиваю, что</u>	Я подчеркиваю, что
A_{i,j}	$A_{i,j}$
A²	A^2
<big>БОЛЬШОЙ</big> ТЕАТР	БОЛЬШОЙ ТЕАТР
Маленький Мук	Маленький Мук
ЦЕНТР	ЦЕНТР
неразрывный пробел	неразрывный пробел
перевод строки	перевод строки

Задание 2: Измените Ваш HTML-документ:

- фамилию, имя, отчество - наберите крупными жирными наклонными буквами, разместите по центру;
- год рождения - на следующей строке, подчеркните;
- место рождения - на следующей строке, мелкими буквами;
- адрес- на следующей строке, мелкими буквами.

В HTML -документах часто используются **списки: ненумерованные** или **нумерованные**.

Ненумерованный список	Нумерованный список
• бананы • анансы • апельсины • яблоки	1. тетради 2. авторучки 3. карандаши 4. ластики

Элементы **списков** выделяются с помощью тега **<li>** ***(list item)***.

Нумерованный список должен находиться в контейнере **
...
** ***(ordered list)***, **ненумерованный список может** находиться в контейнере **
...
** ***(unordered list)*** (наличие контейнера необязательно).

Так, для создания приведенного выше списка фруктов был использован код:

```
<li>бананы <li>анансы <li>апельсины <li>яблоки
```

С тем же успехом мы могли использовать код:

```
<ul> <li>бананы <li>анансы <li>апельсины <li>яблоки </ul>
```

Заменив контейнер **
...
** на **
...
**, мы получили бы список с номерами при каждом элементе. В частности, вот так выглядит код для списка канцелярских принадлежностей:

```
<ol> <li>тетради <li>авторучки <li>карандаши <li>ластики </ol>
```

Задание 3: Измените Ваш HTML-документ, добавив в него:

- нумерованный список дисциплин, которые Вы будете слушать в этом семестре
- ненумерованный список Ваших любимых кушаний.

Атрибуты тэгов

Большинство тэгов имеет **атрибуты**, т.е. свойства, которые могут принимать различные значения. Например, если мы хотим, чтобы текст документа находился на желтом фоне, нам нужно использовать следующий код:

```
<html>
<head>
...
</head>
<body bgcolor='yellow'>
      Наш документ на желтом фоне
</body>
</html>
```

Как нетрудно заметить, тот документ, что Вы сейчас читаете, находится на розоватом фоне. Посмотрите на его **HTML**-код и найдите название этого цвета.

Нередко документ размещают на фоне некоторого рисунка. Сам рисунок должен находиться в файле **графического формата (gif** или **jpg)**, а связь документа с ним обеспечивается атрибутом **background** тэга **<body>**:

```
<html>
<head>
...
</head>
<body background='sky.gif'>
      Наш документ на фоне неба
      </body>
</html>
```

Возможно, у Вас возник вопрос: *"какого размера должен быть фоновый рисунок?"* Дело в том, что если размер рисунка меньше размера окна браузера, то рисунок будет дублироваться по всему окну. Этот принцип лежит в основе создания **текстур** для background'а. В качестве примера посмотрим на файлы **'О пакете'** и **'Задания'**. (Они вызываются из главного меню). Обратите внимание - текст в файле **'О пакете'** словно

скользит над неподвижным фоном, в **'Заданиях'** же он прокручивается вместе с ним. Эффект "скольжения" достигается еще за счет одного атрибута тэга **<body>**:

```
<html>
<head>
...
</head>
<body  background='адрес  изобраения'bgproperties='fixed'>
Наш документ 'скользит' по фоновому рисунку
</body>
</html>
```

Итак, Вам уже знакомы:

Атрибуты тэга <body>	
background	адрес фонового изображения
bgcolor	цвет фона
bgproperties	fixed - для достижения эффекта скольжения текста по фону

Задание 4: Создайте два **HTML**-документа.(Используйте ту информацию, что Вы уже ввели в свой первый документ). Для одного из них выберите **фоновый рисунок**, для другого задайте **цвет фона**.

Метки и гиперссылки

Гиперссылки составляют самую "соль" любого HTML-документа. **Гиперссылки** - это фрагменты (слова, рисунки, кнопки и т.д.), щелкнув мышью на которых мы попадаем на новый документ

или на новой место в этом же документе.

Гиперссылки создаются с помощью тэга **<a>...** (англ.: ***anchor*** **- якорь**).

Гиперссылки имеют два основных атрибута: **href** и **name**.

Рассмотрим **два типа гиперссылок**:

1. Переход к другому документу.

Например, мы хотим, чтобы щелкнув на тексте **'Информация о пакете'** мы открыли в нашем окне документ **'About.html'**.

Код	Результат
<html> <head> ... </head> <body> текст документа Информация о пакете </body> </html>	текст документа Информация о пакете

2. Перемещение в пределах документа.

Если документ большой, если в нем есть несколько смысловых разделов, желательно предусмотреть возможность перехода по гиперссылкам в пределах документа. Например, обеспечить возможность перехода на начало или конец документа. Для того, чтобы мы могли перейти на то или иное место в документе, его нужно предварительно отметить, бросить туда "якорь". Это делается с помощью того же тэга **<a>**, но с атрибутом **name** (а не **href**). Если Вы посмотрите на HTML-код этого документа, то в самом начале увидите фрагмент:

```
<body>
<a name="#nachalo">
...
```

Тем самым мы "отметили" начало документа. Визуально наличие этой метки незаметно - просматривая документ в браузере мы не можем узнать о ее существовании. Но предположим, нам нужно иметь возможность перейти на начало документа. Сделаем следующее:

Код	Результат
<html> <head> ... </head> <body> текст документа На начало документа </body> </html>	текст документа На начало документа

Итак, основными атрибутами тэга **<a>** являются:

name	Имя. Используется, если нужно отметить то или иное место в документе, чтобы потом "приходить" на него.
href	(hyper reference)- адрес документа, вызываемого при щелчке на гиперссылке, и/или имя метки для перехода (метка должна быть установлена заранее). Например, код все заново вызывает перемещение на начало данного документа, код о заказе вызывает перемещение на соответствующую метку в файле 'Books.html'.
target	Окно, в которое будет загружен указанный документ. Если этот атрибут опущен (или указан как _self), новый документ будет открыт в том же окне, что и текущий. Значение атрибута target="_blank" приведет к открытию нового документа в новом окне.

Важно! Говоря о гиперссылках, следует отметить, что адрес документа (значение атрибута **"href"**) не следует указывать, как **полный физический путь**, например, гиперссылка: **Введение в web-технологии**, конечно, сработает на Вашем компьютере, но если Вы перенесете Ваш файл на другой компьютер, то при его просмотре активизация гиперссылки приведет к тому, что браузер будет искать файл **LIntro.html** в папке **Tests**, вложеннной в папку **Eos**, вложенную в папку **Kek**, вложенную в папку **Sum**, находящуюся на диске **C**. Чтобы гиперссылка сработала на другом компьютере, он должен содержать такую же структуру папок. Если, к примеру, Вы перенесете свой файл не на диск **C:**, а на диск **D:**, то гиперсссылка работать не будет. Поэтому следует указывать **не абсолютный**, а **относительный путь**.

Так, если файл, на который Вы ссылаетесь, находится **в том же директории, что и текущий**, к нему следует обращаться просто **по имени**. Например:

Введение в web-технологии.

Если файл находится в какой либо папке, **вложенной в папку с текущим файлом**, следует указать имя папки и имя файла. Например:

Страничка Барби.

Если файл находится в папке, которая **содержит "текущую"**, обращение должно выглядеть следующим образом:

текст гиперссылки.

Если файл находится в папке, **вложенной в папку, содержащую "текущую"**, обращение должно выглядеть следующим образом:

текст гиперссылки.

Если нужно подняться на два уровня выше, обращение должно выглядеть следующим образом:

текст гиперссылки

и т.д...

К счастью, запоминать все эти правила не нужно - в HTML-редакторах есть "мастера" (или, как их еще называют "волшебники", wizards), которые ***автоматически определяют относительный путь*** к выбранному с помощью кнопки "Обзор" файлу. Следует учитывать, что ***автоматическое определение пути возможно лишь тогда, когда текущий файл уже сохранен, т.е. имеет адрес***. Иначе - непонятно, *относительно чего* следует вычислять путь.

Задание 5: Создайте в двух **HTML**-документах **взаимные гиперссылки**.

Форматирование текста

Форматирование текста, т.е. определение **шрифта**, **размера** и **цвета** букв, можно осуществлять с помощью атрибутов тэга **** (они перечислены ниже), хотя позднее Вы познакомитесь с другими, более гибкими способами **управления стилем** документа, в том числе и характеристиками текста.

Атрибуты		Примеры	
size	размер	 	Увеличили шрифт на 2 пункта Уменьшили шрифт на 1 пункт
color	цвет	Красный текст цвет в формате RGB и в 16-ичном виде	Красный текст цвет в формате RGB и в 16-ичном виде
face	шрифт	Arial Currier	Arial Currier

Изображения

Сейчас трудно представить себе web-документы без картинок, хотя первые текстовые браузеры "не поддерживали" графику. Чтобы "вставить" картинку в документ, используется тэг **<img>**.

Его основные атрибуты:

src	Источник(source) изображения, т.е. адрес графического файла.
width, height	Ширина и высота изображения. Эти атрибуты можно не указывать, тогда рисунок будет выглядеть "как есть", но лучше задавать их явно, чтобы браузер зарезервировал соответствующий объем памяти.
alt	Текст, появляющийся на экране вместо рисунка, если тот по какой-либо причине не может быть загружен. "Всплывает" при наведении мыши на рисунок.
border	Толщина рамки вокруг рисунка.

Теперь Вам вполне по силам выполнить первые 5 простых заданий.

Простые задания

1. **Создайте страничку с информацией о себе - выберите цвет фона и текста, начертание букв; разместите свою фотографию (можно условную). Используйте различные тэги форматирования текста.**
2. **Создайте небольшой сайт (4-5 страниц) с информацией о себе и своей семье (или своей учебной группе, друзьях и пр.). Обоснуйте выбор *структуры сайта* и способа *навигации* по нему. Используйте в качестве фона для HomePage подходящее изображение. Создайте для HomePage музыкальный фон. Создайте *гиперссылки* с помощью фрагментов текста и изображений.**

3. Создайте документ длиной 4-5 экранных страниц ,разбитый на смысловые абзацы. Для перехода от одного абзаца к другому используйте *метки*.

4. Создайте документ, содержащий выдержки из законодательных актов, регламентирующих приобретение и использование *пакетов прикладных программ*. Выдержите документ в следующем стиле: названия законодательных актов выведите жирным шрифтом темно-красного цвета размером 20pt. Текст выдержите черным цветом, курсивом, размер букв - 14 pt. Создайте метки на название каждого документа.

5. Создайте сайт из HomePage и 3-х документов. Предусмотрите взаимные гиперссылки:

HomePage - документы 1,2,3;

Документ 2 - документы 1,2,3.Обеспечьте единство *стилевого оформления* (цвета фона, букв, заголовков, типа

и размера шрифтов) за счет применения *каскадной таблицы стилей*.

6. Расположите на странице два изображения и кнопку ЗАМЕНА, при нажатии на которую изображения будут меняться местами.

7. Создайте три бегущие строки:

a. убегающий вправо и исчезающий за границей области текст;

b. колеблющуюся на оранжевом фоне последовательность слов и рисунков;

c. движущийся справа налево и останавливающийся по достижению границы текст.

Основы HTML

Как создаются таблицы?

"Скелет" таблицы выглядит следующим образом:

```
<table>
<tr>
<td>Содержимое ячейки</td>
<td>Содержимое ячейки</td>
...
</tr>
<tr>
<td>Содержимое ячейки</td>
<td>Содержимое ячейки</td>
...
</tr>
...
</table>
```

Здесь контейнер **<tr>...</tr>** определяет **строку** таблицы, контейнер **<td>...</td>** - **ячейку** в строке, контейнер **<table>...</table>** - собственно **таблицу.**

Например, код:

```
<table>
<tr>
<td>текст1</td>
<td>текст2</td>
</tr>
<tr>
<td>текст3</td>
<td>текст4</td>
</tr>
</table>
```

определяет вот такую таблицу:

Текст 1	Текст 2
Текст 3	Текст 4

Где же таблица? Без рамки она не видна, поэтому такие таблицы удобно использовать для **"разметки"** страницы, для расположения фрагментов документов в желаемом порядке.

С помощью **"мастера таблиц"** редактора **HomeSite** (или других современных HTML-редакторов) таблицы легко создаются в визуальном режиме.

Атрибуты таблиц и их элементов

Тэги **<table>, <tr>, <td>** допускает использование большого числа **атрибутов**, с помощью которых можно изменять размеры, положение и другие характеристики таблиц, строк или отдельных ячеек.

Значения большинства атрибутов можно указать непосредственно при создании таблицы с помощью мастера. Как правило, смысл атрибутов интуитивно понятен.

Рассмотрим их применение на **простых примерах.**

Пример 1.

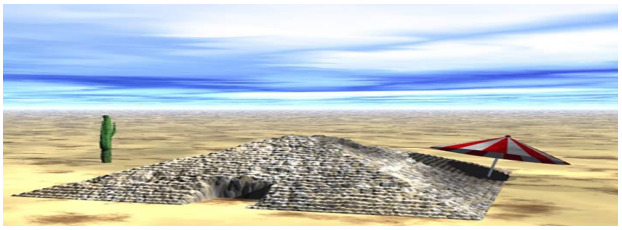
Довольно длинный текст	Текст покороче	Текстик
Текстик	Довольно длинный текст	Текст покороче

Эта таблица, в отличие от предыдущей, имеет фон (**bgcolor="#f0f8ff"**) и границу (**border="1"**) указанного цвета (**bordercolor="#6495ed"**). *Расстояние между ячейками* составляет 10 пикселов (**cellspacing="10"**), *отступ от границы ячейки до текста, находящегося внутри нее* - 5 пикселов (**cellpadding="5"**). Таблица расположена по центру документа (**align="center"**). Если мы будем изменять ширину окна браузера, то таблица будет "приспосабливаться", составляя 90% от ширины (**width="90%"**). (Если бы мы задали ширину таблицы не в процентах, а *в пикселах*, например: **width="600"**), то она не изменялась бы при изменении ширины экрана.

Задание 1: Создайте таблицу из 3-х строк и 2-х столбцов. Задайте для нее цвет фона (или фоновый рисунок) и двойной рамки. В клетках таблицы расположите рисунки или тексты. Обеспечьте отступ от границы ячейки до ее содержимого, равный 10 пикселам, расстояние между ячеек сделайте равным 3 пикселам. Таблицу "прижмите" к правой стороне окна браузера.

Некоторые из атрибутов тэга **<table>** применимы к отдельным *строкам* или *ячейкам* таблицы. Так, например, можно задать свой цвет или фоновый рисунок для отдельных ячеек. Атрибут **align**, заданный *для отдельной ячейки*, определяет *горизонтальное положение* текста внутри нее, атрибут **valign**, заданный *для ячейки*, определяет *вертикальное положение* ее содержимого (**top**, **middle** или **bottom**).

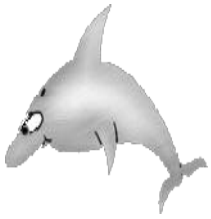
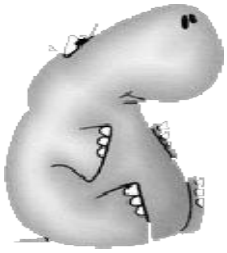
Пример 2.

	Для этой ячейки атрибут valign не задан
Для этой ячейки атрибут valign="top"	

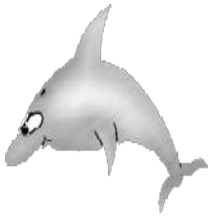
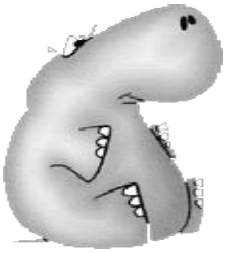
Обратите внимание на значение атрибутов **align** для ячеек этой таблицы. Кроме того, здесь граница выполнена в двух цветах - это достигается за счет использования атрибутов **bordercolorlight** и **bordercolordark**.

"Неправильные" таблицы

В таблицах, которые мы рассмотрели, *число ячеек в каждой строке было одинаковым*. Но как построить вот такую например таблицу?

			
ЭТО ЖИВОТНЫЕ			ЭТО ФРУКТ

Казалось бы, нужно в первой строке создать 4 ячейки, а во второй - всего 2. Но тогда получается вот что:

			
Это животные	Это фрукт		

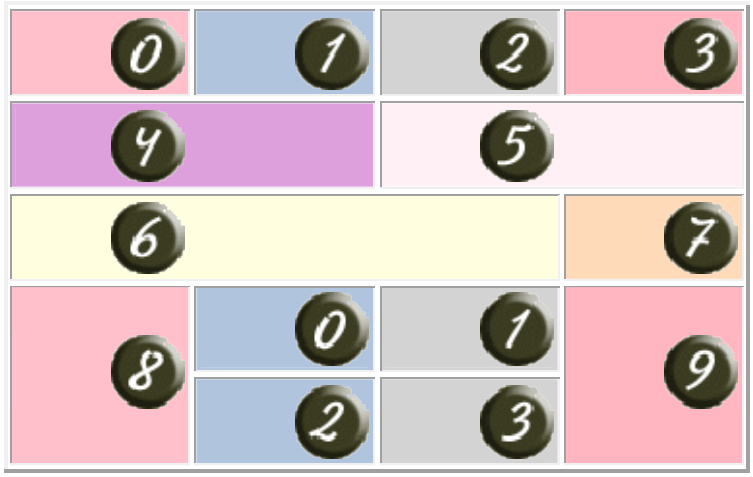
Чтобы первая ячейка "растянулась" на 3 ячейки, нужно добавить к ее коду атрибут **colspan**:

```
<td colspan=3><font size="+2"><b>ЭТО ЖИВОТНЫЕ</b></font></td>
```

Вторая ячейка должна остаться без изменений.

Аналогично, если ячейка должна "охватить" несколько строк, нужно добавить к ее коду атрибут **rowspan**.

С помощью **colspan**'ов и **rowspan**'ов можно создавать сколь угодно сложные таблицы, например, вот такую:



Задание 2: Создайте таблицу из 4-х строк и 4-х столбцов. С помощью атрибутов colspan и rowspan объедините отдельные ячейки. В объединенные ячейки разместите текст или изображение так, чтобы они располагались посередине ячейки и по горизонтали и по вертикали.

Задание 3: С помощью таблицы (без рамки) расположите текст в окне браузера в две колонки, над которыми имеется один общий фрагмент.

Теперь Вы сможете выполнить задание 1 из раздела "Создание таблиц".

Создание таблиц

1. **Создайте таблицу 3 х 4 клетки, расположенную по центру экрана. Задайте цвет фона или фоновый рисунок, цвет и толщину границы таблицы. Клетки таблицы должны содержать гиперссылки (текст, рисунки или кнопки) на соответствующие документы.**
2. **С помощью таблицы, занимающей 80% ширины экрана и 70% его высоты, разместите на странице следующие элементы:**
   - **заголовок (по центру),**
   - **баннер,**
   - **три рисунка,**
   - **общий для всех трех изображений текст,**
   - **4 одинаковых по размеру кнопки.**

3. Создайте документ, в центре которого находилось бы квадратное изображение размером 150 x 150 px, слева и справа от него (по высоте изображения) - текст, сверху и снизу - также текст по ширине экрана.

Формы

Основные понятия

Одним из важнейших свойств web-документов является возможность **получения данных от клиентов** и **отправки их на сервер**. Эта возможность обеспечивается с помощью так называемых **ФОРМ (FORMS)**.

Таким образом,

> Формы - это фрагменты HTML-документов, "ответственные" за ввод информации клиентом.

Как выглядят HTML-формы?

Вот один из простейших примеров:

Анкета кандидата в члены EMICS

Звездочкой (*) отмечены обязательные поля

1.	Фамилия *	
2.	Имя, отчество *	
3.	Место работы *	
4.	Должность *	
5.	Рабочий адрес *	
6.	Рабочий телефон *	
7.	Факс	
8.	E-mail *	
9.	Адрес личной web-страницы	
10.	Ученая степень	
11.	Ученое звание	
12.	Область научных интересов *	1. 2. 3. 4. 5. 6.

Отменить ввод

Отослать данные и перейти ко II этапу регистрации

Это очень простая форма. Она содержит лишь **текстовые поля** и две кнопки. Первая кнопка **(RESET)** позволяет **"сбросить"** все введенные пользователем данные, вторая **(SUBMIT)**- непосредственно **отсылает данные на сервер**. Ясно, что если забыть разместить в форме кнопку **SUBMIT**, то данные на сервер не будут отправлены. Поэтому

> В форме обязательно должна присутствовать кнопка SUBMIT, "ответственная" за отправку данных, введенных клиентом, на сервер.

Рассмотрим HTML-код, создающий форму.

> Формы размещаются в контейнере <form>...</form>.

Этот контейнер можно уподобить обычному бумажному почтовому конверту - на последнем обязательно присутствует адрес назначения и фамилия адресата, в тэге **<form>...</form>** непременно указывается **адрес серверного сценария**, который получит и обработает ввведенные клиентом данные. Для этого используется атрибут **action**. Таким образом, простейшая форма имеет такую структуру:

```
<form action="адрес серверного сценария">
....Всякие элементы форм (среди которых непременно присутствует SUBMIT).....
</form>
```

Атрибут **action** является обязательным (если, конечно, отправка данных на сервер предусмотрена), но не единственным атрибутом формы. О других атрибутах мы поговорим чуть позднее, а пока остановимся на основных элементах форм.

Элементы форм

Все элементы форм создаются тэгами, имеющими начало: **<input type="...**", где значения атрибута **type** и определяют тип элемента. Рассмотрим возможные значения этого атрибута.

- **Текстовое поле (text field):**

(Мы только что видели их в нашем примере)

Текстовое поле определяет код:

```
<input type="Text" name="имя поля" value="начальный текст" align="LEFT" size="число символов" maxlength="максимальное число символов">
```

Здесь

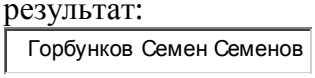
- **name** - имя поля (впоследствии мы узнаем, как к элементам форм можно обращаться просто по имени);
- **value** - то, что будет записано в текстовое поле "по умолчанию", как только документ появится в окне браузера;
- **align** - этот атрибут "отвечает" за расположение текстового поля; он не является специфическим для элементов форм и (как и в других тэгах) может принимать значения: **left**, **right**, **center**;
- **size** - длина поля в символах;
- **maxlength** - максимальное число символов, которое можно ввести в поле. Если этот атрибут не указывать, ограничений на ввод не будет.

Например:

код:

```
<input name="fio" value="Горбунков Семен Семенович" align="LEFT" size="28"
maxlength="35">
```

результат:



Значением текстового поля должен быть только текст. Если, к примеру, поместить в него html-код, он будет отображаться как текст (а не так, как он выглядел бы в окне браузера). Но вот пример размещения в текстовом поле бегущей строки, выполненный Сергеем Бурылиным. Как ему это удалось - попробуйте разобраться сами.

- **Текстовая область (text area):**

Текстовые области удобно использовать для ввода большого количества текстовой информации

Код текстовой области выглядит следующим образом:

```
<textarea name="имя" cols="число столбцов" rows="число строк" wrap="способ переноса строк"> ТЕКСТ, РАЗМЕЩЕННЫЙ В ТЕКСТОВОЙ ОБЛАСТИ
</textarea>
```

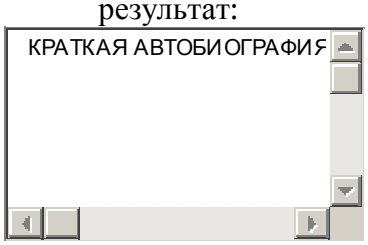
Здесь атрибут **"wrap"** указывает на способ разрыва строки: если присвоить ему значение **off**, вводимый текст будет **"уходить" за пределы области**, в противном случае (**on**, **soft** или даже просто **wrap** без присвоения) текст автоматически переносится на новую строку.

Например:

код:

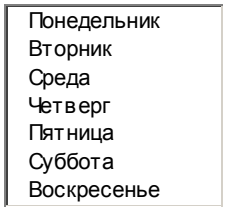
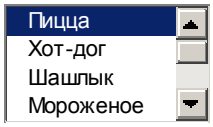
```
<TEXTAREA name=bio cols=30 rows=6 wrap>
КРАТКАЯ АВТОБИОГРАФИЯ </TEXTAREA>
```

результат:



- **Список:**

Списки позволяют сделать **единственный** или **множественный** выбор из набора предлагаемых **опций (options).**

Например:	
Список с единственным выбором: Какой сегодня день недели?  Показаны все имеющиеся опции, ни одна не выделена.	Список с множественным выбором: Сделайте заказ продуктов:  Показаны лишь 4 из 8-ми опций, первая опция выделена.

Структура HTML-кода, определяющего список, такова:

```
<select name="имя списка" size="число видимых опций" multiple>
<option value="значение" selected>текст
<option value="значение">текст
..... и так далее.....
<option value="значение">текст
<option value="значение">текст
</select>
```

Коды списков, приведенных нами в качестве примеров, имеют вид:

```
<select name="week" size="7">
<option value="1">Понедельник</option>
<option value="2">Вторник</option>
<option value="3">Среда</option>
<option value="4">Четверг</option>
<option value="5">Пятница</option>
<option value="6">Суббота</option>
<option value="7">Воскресенье</option>
      </select>
```

Список с единственным выбором, показаны все имеющиеся опции, ни одна не выделена.

```
<select name="food" size="4" multiple>
<option value="1" selected>Пицца</option>
<option value="2">Хот-дог</option>
<option value="3">Шашлык</option>
<option value="4">Мороженое</option>
<option value="5">Шампанское</option>
<option value="6">Ананас</option>
<option value="7">Шоколад</option>
<option value="8">Кофе</option>
</select>
```

Список с множественным выбором, показаны лишь 4 из 8-ми опций, первая опция выделена.

- **"Флажок" (Checkbox)**

"Единичный" флажок создается с помощью кода:

```
<input type="Checkbox" name="имя флажка" value="значение" checked> текст, который обычно помещается рядом с флажком
```

Атрибут **checked** указывает на то, что "флажок" по умолчанию будет отмеченным. Часто используется сразу несколько "флажков", например:

```
Укажите Ваши навыки:

<input type="checkbox" name="skill" value="1"checked>HTML

<input type="checkbox" name="skill" value="2">JavaScript

<input type="checkbox" name="skill" value="3">ASP

<input type="checkbox" name="skill" value="4">Perl

<input type="checkbox" name="skill" value="5">PHP

<input type="checkbox" name="skill" value="6">XML

<input type="checkbox" name="skill" value="7">FLASH
```

Укажите Ваши навыки:

- [x] HTML
- [ ] JavaScript
- [ ] ASP
- [ ] Perl
- [ ] PHP
- [ ] XML
- [ ] FLASH

- **"Радиокнопка" (Radiobutton)**

"Единичная" радиокнопка создается с помощью кода:

```
<input type="radio" name="имя радиокнопки" value="значение" checked> текст, который обычно помещается рядом с радиокнопкой
```

Атрибут **checked** указывает на то, что радиокнопка по умолчанию будет отмеченной. Часто используется сразу несколько радиокнопок, например:

```
Укажите Ваши навыки:

<input type="radio" name="skill" value="1" checked>HTML

<input type="radio" name="skill" value="2">JavaScript

<input type="radio" name="skill" value="3">ASP

<input type="radio" name="skill" value="4">Perl

<input type="radio" name="skill" value="5">PHP

<input type="radio" name="skill" value="6">XML

<input type="radio" name="skill" value="7">FLASH
```

Укажите Ваши навыки:

- (•) HTML
- ( ) JavaScript
- ( ) ASP
- ( ) Perl
- ( ) PHP
- ( ) XML
- ( ) FLASH

Здесь все радиокнопки имеют одно имя - skill. Это обеспечивает единственность выбора. Если присвоить радиокнопкам разные имена, будет возможен множественный выбор, как и в случае использования checkbox'ов.

- **Кнопка (button):**

Как мы уже говорили, каждая форма должна иметь кнопку, нажатие на которую обеспечивает **отправку данных на сервер**. Такая кнопка создается с помощью кода:

```
<input type="submit" name="имя кнопки" value="то, что на этой кнопке написано">
```

Например:

```
<input type="submit" name="subm" value="Отправить данные">
```

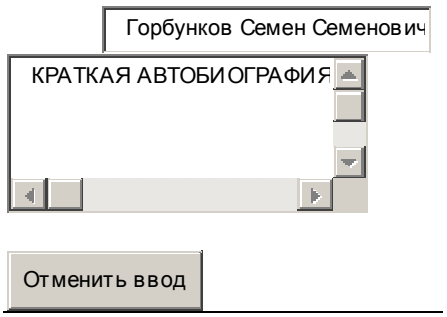
Результат: [Отправить данные]

При нажатии кнопки **submit** данные передаются на сервер по адресу указанному в качестве значения атрибута **action**.

Чтобы предусмотреть возможность **сброса** всех введенных пользователем данных (и восстановления данных, указанных по умолчанию), используется кнопка **reset**. Код ее выглядит следующим образом:

```
<input type="reset" name="имя кнопки" value="то, что на этой кнопке написано">
```

Например:



Наряду с кнопками **submit** и **reset** используются и "простые" кнопки, с которыми обычно связывается то или иное действие. "Простая кнопка" создается с помощью кода:

```
<input type="button" name="имя кнопки" value="то, что на этой кнопке написано" onclick="действие, связанное с кнопкой">
```

Обратите внимание на атрибут **onclick**. Он представляет собой **обработчик события** - нажатия на кнопку. Для обработки этого события (и множества других событий) используются клиентские сценарии, написанные на языке **JavaScript** или **VBScript**. (Язык **JavaScript** мы вскоре будем изучать.) Как правило, эти сценарии оформляются в виде **функций**, вызов которых является значением обработчика события **onclick**.

Например:

```
<input type="button" value="Показать сегодняшнее число" onclick="alert(Date());">
```

Результат:

Забегая вперед, скажем, что здесь вызывается встроенный JavaScript'овский метод `alert()`, который создает системное окно для размещения в нем той или иной информации (значения его аргумента). В нашем случае аргументом метода `alert()` является встроенный JavaScript'овский объект `Date()`, содержащий информацию о текущей дате и времени. Об этом объекте мы впоследствии поговорим более подробно.

Кнопки могут иметь и более "экзотический" вид. Так, на них можно разместить не только текст, но и рисунок, в том числе и анимированный, и вообще, **любой фрагмент HTML-кода**. Вот общий вид кода "нестандартных" кнопок:

```
<button name="имя кнопки onclick="действие, связанное с кнопкой">HTML-код фрагмента, размещаемого на кнопке"></button>
```

Вот несколько образчиков:

	`<button style="width:124" onclick="alert('Это кнопка с картинкой');"><img src="pictures/pencil.gif" width=118></button>`
Ошибка! 	Здесь рисунок уменьшен в размере (за счет атрибутов `width` и `height` тэга `img`) и помещен в бегущую строку, находящуюся в контейнере `<button ...>...</button>`

Кнопку с размещенной на ней картинкой можно создать и так:

```
<input type="image" src="адрес рисунка" onclick="действие, связанное с кнопкой">
```

Однако в этом случае она мало напоминает кнопку. Сравните, например:

	`<button style="width:124" onclick="alert('Это кнопка с картинкой');"><img src="pictures/pencil.gif" width=118></button>`
	`<input type="image" src="pictures/pencil.gif" onclick="alert('Это тоже кнопка с картинкой');">`

- **Поле для ввода пароля (password):**

Поле для ввода пароля отличается от обычного текстового поля тем, что вводимое в него значение отображается с помощью звездочек. Синтаксис такого поля имеет вид:

```
<input type="password" value="значение по умолчанию" size="длина поля (в символах)" maxlength="максимально допустимое число вводимых символов">
```

Например:

Код:

```
<input type="password" value="123456" size="6" maxlength="8">
```

Результат:

- **Скрытое поле (hidden):**

Иногда в формах используются поля, которые вообще никак не обнаруживают своего присутствия - это так называемые "скрытые" поля. Возникает вопрос: как же пользователь введет в такое поле свои данные? Дело в том, что пользователю этого делать и не нужно - эти данные будут размещаться в скрытых полях с помощью **клиентского сценария** (язык клиентских сценариев мы будем изучать чуть позднее).

Но зачем нужны скрытые поля? С их помощью можно передать на сервер (и сохранить там для последующего использования) какую-либо информацию о клиенте, например, дату посещения данного документа, настройки браузера и пр.

Синтаксис скрытого поля имеет вид:

```
<input type="hidden" value="значение по умолчанию">
```

Пример скрытого поля приводить бессмысленно - его просто не будет видно.

- **Прикрепление файлов:**

Использование форм позволяет пользователю **прикрепить файл** и отправить его на сервер вместе с содержимым текстовых полей и других элементов формы. Например, многие байты, где предусмотрена регистрация пользователей (скажем, агентства по трудоустройству) предусматривают отсылку фото, текстовых файлов с резюме, архивных файлов с текстами статей и пр. Для поиска файла на компьютере клиента (или в сетевом окружении, если клиентский компьютер находится в локальной сети) служит элемент:

```
<input type="file" name="имя элемента" size="длина адресного поля в символах" accept="допустимый тип файла">
```

Например:

Код:

```
<input type="file" size="50">
```

Результат:

Здесь допустимым является файл любого типа.

В следующем примере мы ограничиваем выбор прикрепляемого файла форматами: **txt**, **html**, **avi** (видео), **bmp** (графика)

```
<input type="file" size="50" accept="text/plain,text/html,video/x-msvideo,image/x-portablebitmap">
```

ВАЖНО! Форма, содержащая поле типа **FILE** (для прикрепления файлов), должна иметь атрибут **enctype="multipart/form-data"**.
"Ответственным" за **сохранение файла на сервере** является, конечно же, **серверный сценарий**. Он будет использовать атрибут **name** как **временное имя файла**.

Итак, мы рассмотрели все элементы форм и знаем принцип связи документа, содержащего форму, с серверным сценарием, "принимающим эстафету" от него (его адрес указывается в качестве значения атрибута **action**). Однако атрибут **action** является не единственным атрибутом формы. Забегая вперед, назовем еще два важных атрибута.

Атрибуты форм

- **Action** (о нем мы уже говорили) - **адрес серверного сценария**, получающего и обрабатывающего данные, вводимы пользователем при помощи формы.
- **Name** - **имя формы**. Этот атрибут не является обязательным, но его наличие позволяет обращаться к форме (и ее отдельным элементам) "фамильярно", т.е. просто по имени.
- **Method** - **способ передачи данных**. Этот атрибут может принимать два значения: **GET** или **POST**. Подробнее об этих методах мы будем говорить позднее, когда будем изучать средства создания серверных сценариев, а пока скажем лишь только, что метод **GET** используется в тех случаях, когда объем передаваемых данных невелик (например, значение одной-двух переменных), если же передается большой объем данных (или его трудно оценить заранее), то применяется метод **POST**. И еще: если передается пароль, то метод **GET** использовать нельзя, так как в этом случае текст пароля будет "светиться" в адресной строке.
- **Enctype** - **тип передаваемых данных**. Если форма не содержит прикрепляемых файлов, этот атрибут можно не указывать (по умолчанию он примет значение: **"application/x-www-form-urlencoded"**); если же форма содержит поля типа **file**, значение этого атрибута должно быть **"multipart/form-data"**.

Как правило, перед отправкой данных на сервер производится их предварительный анализ клиентскими средствами. Например, проверяется, заполнены ли обязательные поля, правилен ли формат адреса электронной почты (содержит знак '@') и пр. Для этого на языке **JavaScript** (или **VBScript**) создаются специальные сценарии. С ними мы обязательно познакомимся, но чуть позднее, когда изучим объектную модель **JavaScript**. А еще позднее научимся писать серверные сценарии, которые будут получать и обрабатывать данные клиентов ("классическими" примерами таких сценариев являются **гостевые книги**, **форумы**, **счетчики посещений** со всевозможной **статистикой** и пр.)

Задание: Создайте форму, содержащую все перечисленные выше элементы.

Фреймы

Структура фреймсодержащих документов

Фреймы - это **фрагменты окна браузера**, в каждый из которых может загружаться отдельный **HTML**-документ.

Благодаря фреймам создается возможность одновременного просмотра нескольких документов. Посмотрим пример фреймсодержащего документа. Открыв его источник через панель инструментов браузера, мы увидим HTML-код, определяющий структуру документа:

```
<frameset cols="30%,*">
<frame name="first"
src="ExFrame1.html">
<frame name="second"
src="ExFrame2.html">
</frameset>
```

Здесь **first** и **second** - имена фреймов, **ExFrame1.html** и **ExFrame2.html** - имена документов, загружаемых во фреймы. Атрибут **cols** означает, что фреймы являются вертикальными (иначе мы использовали бы атрибут **rows**), согласно присвоенному нами значению: **30%,*** левый фрейм будет занимать 30% ширины окна браузера, а остальная часть окна будет занята правым фреймом.

Атрибуты тэга <frame>

src	Адрес документа, загружаемого в данный фрейм.
name	Имя фрейма (используется для обращения к фрейму из операторов JavaScript).
scrolling	Указывает на наличие (yes) или отсутствие (no) линеек прокрутки; значение auto указывает на то, что линейки прокрутки появятся автоматически, если в этом возникнет необходимость.
frameborder	Указывает на наличие (yes) или отсутствие (no) границы между фреймами.

Обратите внимание на то, что web-сайт кафедры экономической кибернетики Казанского университета (http://kek.ksu.ru) также имеет фреймсодержащую структуру.

Чтобы посмотреть код документов, загружаемых во фреймы, нужно, находясь в соответствующем фрейме, **правой кнопкой мыши** открыть меню, из которого выбрать **"Просмотр в виде HTML" ("View source")**.

Нередко фреймы используют для того, чтобы в одном из них размещать некоторую постоянную часть, например, **навигационное меню**. При этом гиперссылки, расположенные в одном фрейме, вызывают смену документов в другом. Это достигается с помощью атрибута **target** тэга **<a...>** (англ.: ***target - цель, мишень***), значением которого является **имя фрейма**, в который загружается открываемый по гиперссылке документ.

Задание: Создайте фреймсодержащий документ, в одном из фреймов которого находится меню - гиперссылки этого меню должны открывать разные документы во втором фрейме.

Пример 2. Создадим документ из двух горизонтальных фреймов одинакового размера. В верхний загрузим документ "osysteme.html", в нижний - "saits.html". Код документа должен выглядеть следующим образом:

```
<html>
<head>
</head>
<frameset rows="50%,*">
<frame name="verh" src="osysteme.html" scrolling="auto" frameborder="yes">
<frame name="nisznij" src="saits.html" scrolling="auto" frameborder="yes">
</frameset>
</html>
```

Посмотрим результат.

Для создания **сложных фреймсодержащих документов** используются **вложенные контейнеры <frameset>...</frameset>** или же во фреймы загружаются документы, которые, в свою очередь, также являются **фреймсодержащими**.

Средствами языка **JavaScript** можно изменять содержимое фреймов, отменять или назначать фреймы, а также динамически ("на лету") создавать документы, загружаемые во фреймы. Но об этом позднее...

Плавающие фреймы

Плавающие фреймы - это **фрагменты окна браузера**, занимающие произвольное

положение на экране.
Плавающий фрейм создается с помощью контейнера **<iframe>...</iframe>**.

Рассмотрим код плавающего фрейма:

```
<iframe ID='okno' src="osysteme.html"
 scrolling='yes'
 frameborder="yes"
 width='300' height='250'>
```

Фрейм оказался как бы "врезанным" в документ.
Здесь:
ID-идентификатор фрейма (он играет ту же роль, чо атрибут name), src-адрес документа, загружаемого во фрейм, width, height-размеры фрейма в пикселах.

Динамическое создание плавающего фрейма

Документ, загружаемый в плавающий фрейм, можно формировать ***динамически***. Для этого следует присвоить некоторой переменной строку, содержащую HTML-код. ¶ В этом случае в качестве значения атрибута **src** следует указать:
"javascript:имя переменной, содержащей HTML-код".

Например, рассмотрим код:

```
<script>
var s="<html><body><h1>ПРИВЕТ!</h1>";
s+="<img src='Pictures/w1.gif' align='middle' alt='паучок'>";
s+="</body></html>";
</script>
...
<iframe ID='pauk'
     src="javascript:parent.s"
     scrolling='no'
     align='center'
     valign='middle'
     frameborder="yes"
     width='190' height='190'>
</iframe>
```

переменная s-строка, содержащая HTML-код загружаемого во фрейм документа. Поскольку по отношению к фрейму она является внешней ("родительской"), к ней следует обращаться с помощью префикса parent.

Плавающие фреймы использованы, например, в документах: События, Фильтры, Бегающие строки.

Бегущие строки

Тэг <marquee>

Слово "***marquee***" буквально означает "***шатер***", "***палатка***". В качестве компьютерного термина оно используется для обозначения некоторого ***выделенного фрагмента*** документа или изображения. В языке HTML этот тэг позволяет не просто выделить область web-страницы, но и заставить ее "бегать" по экрану.
Бегущие строки создаются с помощью контейнера:

```
<marquee>содержимое бегущей строки</marquee>
```

"Заставить бегать" не только текст, но и любые фрагменты HTML-документов

Атрибуты бегущей строки

Тэг **<marquee>... </marquee>** имеет множество атрибутов, задающих стиль строки и характер движения.

Некоторые атрибуты используются и в других тэгах, их названия говорят сами за себя. Это, например, атрибуты:

- **width** - ширина области;
- **hight** - высота области;
- **bgcolor** - цвет фона.

Другие атрибуты специфичны для бегущих строк.

Атрибут behaviour

Атрибут **behaviour** (т.е. "***поведение***") отвечает за **способ "пробега"** строки.

Атрибут direction

Атрибут **direction** (т.е. "***направление***") определяет **направление движения** строки.

Атрибут loop

Атрибут **loop** ("***цикл, петля***") задает **число пробегов** строки. Если этот атрибут не указан или имеет значение "**-1**", то строка будет "бегать" бесконечно.

Атрибут scrolldelay

Атрибут **scrolldelay** (т.е. "***запаздывание пробега***") определяет **время между прорисовками** строки (в миллисекундах). По умолчанию этот атрибут имеет значение **85**.

scrolldelay="85"

Сенсорные изображения - что это такое?

Сенсорное изображение - это рисунок, разделенный на отдельные участки ("горячие области", "hot spots"), каждый из которых, как правило, является гиперссылкой на отдельный документ или связан с какой-либо процедурой.

Например, поместив курсор в различные точки изображения:

Вы увидите различные надписи. Это достигается с помощью кода:

```
<map name='имя карты'>
<area shape='rect' coords='0, 0 150, 150' href='#' title='Текст,
всплывающий при наведении мыши на область'>
. . .
<area shape='poly' coords='420,100 520,130, 620,260 740,200
740,100 420,100' href='#' title='Текст, всплывающий при наведении
мыши на область'>
</map>
<img src="Адрес файла с рисунком" width=743 height=499 border=0
usemap='#имя карты'>
```

Как видно из приведенного фрагмента, тэгу **<img>**, обеспечивающему вывод изображения, предшествует контейнер **<map>...</map>**, задающий "карту разбивки" рисунка на "горячие области".
Карта должна иметь уникальное **имя**, задаваемое атрибутом **name**. Каждая область задается тэгом **<area>**, их может быть сколько угодно.

Атрибуты тэга <area>:

shape	Задает форму области: • прямоугольник (rect, rectangle), • многоугольник (poly, polygon) или • круг (circle).
coords	Задает координаты (левый верхний и правый нижний углы прямоугольника; вершины многоугольника (начальная и конечная точки должны совпадать); точка на окружности и центр круга).
href	Адрес документа, вызываемого при щелчке на данной области или '#', если

	загружать новый документ не нужно.
`title`	Текст, всплывающий при наведении мыши на соответствующую область.

Рисунок загружается **после** разбивки карты с помощью контейнера **<map>...</map>**. В тэг **<img>** добавляется атрибут **usemap=имя карты.**

Еще один пример сенсорного изображения - здесь "обработаны события": 'щелчок', 'двойной щелчок' и 'наведение мыши' - поищите "горячие области" на этом изображении. (Этот пример создан студентом гр. 960В Салтанаевым Эдиком.	

А вот пример **Рустама Гильмутдинова** (гр.970В).

Надеемся, **Вы** тоже внесете свой вклад в развитие и совершенствование пакета!) Попробуйте теперь выполнить **задание 1** из раздела "Работа с изображениями".

Работа с изображениями

Минимальное число баллов:

1. Создайте сенсорное изображение с областями произвольной формы при наведении курсора на которые появлялся бы соответствующий текст.

2. Создайте документ, содержащий изображение и три кнопки: "Уменьшить рисунок", "Увеличить рисунок", "Исходные размеры". Нажатие на первую кнопку должно уменьшать длину и ширину рисунка вдвое по сравнению с исходными размерами, нажатие на вторую кнопку - вдвое увеличивать, третья кнопка должна обеспечивать восстановление размеров.

3. Создайте простейшую двухкадровую анимацию посредством периодической смены изображений, находящихся в папке "Pictures".

4. Создайте функцию, выводящую (в документ) свойства *передаваемого ей* объекта. Документ (создаваемый динамически) поместить в плавающий фрейм: