# Активные формы на веб-страницах

Помимо отображения текста, рисунков и другой информации, современные сайты позволяют пользователю вводить данные и обрабатывать их.

Активные формы – это элементы для ввода пользовательских данных. К ним относятся:

* кнопка
* переключатель (радиокнопка)
* флажок (чекбокс)
* поле для ввода (однострочное и многострочное)
* поле со списком
* полоса прокрутки
* и др.

Сами по себе активные формы не обрабатывают введенные в них данные. Этим занимаются программы.

Скрипты JavaScript обрабатывают введенные в поля данные **на стороне пользователя**, т.е. прямо в браузере. Если браузер не поддерживает JavaScript, то данные обработаны не будут.

Язык PHP встраивается в страницу, но обработка данных происходит **на сервере**. Т.е. после ввода данных и подтверждения (кнопка ОК), они отправляются на сервер. Там они обрабатываются и пользователю отправляется уже готовый результат в виде обычной HTML страницы.

## Элементы форм, ориентированные на ввод с клавиатуры

Наше знакомство с построением форм начнется с элементов, ориентированных на ввод с клавиатуры. Таких элементов два – текстовое поле (text box) и текстовая область (text area).

### Текстовое поле

В текстовое поле можно ввести только одну строку. В текстовых полях обычно вводится короткая текстовая информация – адрес электронной почты, почтовый адрес или имя. Синтаксис определения текстового поля:

<input type="text" nаme="имя\_переменной" size="N" maxlength="N" value="">

Определение текстового поля включает пять атрибутов:

* type — тип элемента (для текстовых полей — text);
* name — имя переменной, в которой сохраняются введенные данные, и одновременно имя самого поля;
* size — размер текстового поля в браузере;
* maxlength — максимальное количество символов, которые можно ввести в текстовом поле;
* value — значение, отображаемое в текстовом поле по умолчанию.

Пример текстового поля:

<p><b>Name:</b></p>

<input type="text" nаme="mytext" value="this is text box">



Особая разновидность текстовых полей – поле для ввода паролей. Оно работает точно так же, как обычное текстовое поле, однако вводимые символы заменяются звездочками или точками. Чтобы создать в форме поле для ввода паролей, достаточно указать type="password" вместо type="text".

### Текстовая область

Текстовая область (text area) используется для ввода больших объемов текста, состоящего из нескольких строк. Это может быть письмо администрации сайта, сообщение на форуме, отзыв о товаре. Синтаксис определения текстовой области:

<textarea name="имя\_переменной" rows="N" cols="N" value=""></textarea>

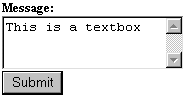
Определение текстового поля включает три атрибута:

* name – имя переменной, в которой сохраняются введенные данные;
* rows – количество строк в текстовой области;
* cols – количество столбцов в текстовой области.

Пример:

<p><b>Message:</b></p>

<textarea name="mytextarea" cols="20" value=""> This is a text box </textarea>



## Элементы форм, ориентированные на ввод с мыши

В других элементах форм пользователь выбирает один из заранее определенных вариантов при помощи мыши или стрелок клавиатуры.

### Флажок

Флажок (checkbox) используются в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора. Синтаксис определения флажка:

<input type="checkbox" name="имя\_переменной" valuе="начальное\_значение">

Определение флажка включает три атрибута:

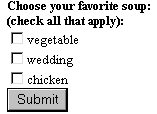
* type — тип элемента (для флажков — checkbox);
* name — имя переменной, в которой сохраняются введенные данные (отмечен или нет);
* value — значение переменной. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута value не используется.

<p><b>Choose your favourite soup (only one):</b></p>

<input type="checkbox" name="soup" value="vegetable"><br>vegetable

<input type="checkbox" name="soup" value="vegetable"><br>vegetable

<input type="checkbox" name="soup" value="vegetable"><br>vegetable



### Переключатель

Переключатель (radio button) похож на флажок, но одновременно в группе может быть установлен лишь один переключатель. Синтаксис определения переключателя:

<input type="radio" name="имя\_переменной" value="начальное\_значение">

Определение переключателя поля включает три атрибута:

* type — тип элемента (для переключателей — radio); переключатели из одной группы имеют одинаковое имя;
* name — имя переменной, в которой сохраняется выбранный вариант;
* value — значение, присваиваемое переменной.

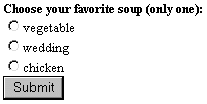
Как и у флажка, если переключатель не установлен, value не используется.

<p><b>Choose your favourite soup (only one):</b></p>

<input type="radio" name="soup" value="vegetable"><br>vegetable

<input type="radio" name="soup" value="vegetable"><br>vegetable

<input type="radio" name="soup" value="vegetable"><br>vegetable



### Раскрывающийся список

Раскрывающиеся списки особенно удобны в ситуации, когда у вас имеется длинный перечень допустимых вариантов, из которого пользователь должен выбрать один вариант. Как правило, раскрывающиеся списки применяются при работе с относительно большими наборами заранее известных вариантов – например, при выборе страны, города. Синтаксис определения раскрывающегося списка:

<select name="имя\_переменной">

<option valuе="имя\_переменной1 ">

<option value="имя\_переменной2">

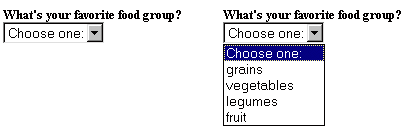
<option value="имя\_переменнойЗ">

<option value="имя\_переменнойN">

</select>

Определение переключателя поля включает три атрибута:

* name — имя переменной, в которой сохраняется выбранный вариант (в данном случае — строка, выбранная в списке);
* value — значение, отображаемое в списке по умолчанию.



## Кнопки

По нажатию кнопки выполняется какое-либо действие. Помимо обычных кнопок существуют и специальные – кнопка отправки данных (подтверждения ввода) и кнопка сброса (очистки данных). Эти кнопки работают только на форме и описаны ниже.

### Обычная кнопка

Обычно используется в связке с событием onclick:

<input type="button" value="текст\_на\_кнопке">

Определение кнопки включает два атрибута:

* type — тип элемента (для кнопки – button);
* value — текст, по умолчанию отображаемый на кнопке.

<input type="button" value="Моя кнопка">



### Кнопка отправки данных

Кнопка отправки данных запускает действие, заданное атрибутом action тега <form>. Синтаксис определения:

<input type="submit" value="текст\_на\_кнопке">

Определение кнопки включает два атрибута:

* type — тип элемента (для кнопки отправки данных — submit);
* value — текст, по умолчанию отображаемый на кнопке.

<input type="submit" value="Подтвердить">



### Кнопка сброса

Кнопка сброса отменяет все изменения, внесенные пользователем в элементы формы. Синтаксис определения:

<input type="reset" value=" текст \_на\_кнопке">

Определение кнопки включает два атрибута:

* type — тип элемента (для кнопки сброса — reset);
* value — текст, по умолчанию отображаемый на кнопке.

<input type="reset" value="Сброс">



## Форма

При вводе данных в форму используются различные управляющие элементы. В одних элементах пользователь вводит информацию с клавиатуры, в других он выбирает нужный вариант, щелкая кнопкой мыши.

Форма объединяет эти элементы в одно целое. Все элементы, расположенные на одной форме, обрабатываются одновременно.

Одна страница может содержать несколько форм, поэтому необходимы средства, которые позволяли бы отличить одну форму от другой. Более того, вы должны как-то сообщить форме, что следует делать, когда пользователь выполняет действие с формой (как правило, нажимает кнопку отправки данных).

Обе задачи решаются заключением форм в следующие теги HTML:

<form action = "действие" method = "метод"> Здесь располагаются элементы формы </form>

**Действие** указывает, какой сценарий должен обрабатывать форму.

Сценарий может быть функцией или кодом JavaScript. А может быть php-страницей, на которую осуществляется переход после ввода данных, и которая обрабатывает их.

Метод определяет способ передачи данных обработчику. Существует два метода передачи данных:

**Метод get** передает все данные формы через адрес страницы URL. Из-за различных ограничений, связанных с длиной URL, этот метод применяется редко. Переменные и их значения перечисляются в конце адреса, после вопросительного знака:

<http://site.domain/?имя1=значение1&...имя2=значение2>

<http://www.sagmu.ru/index.php?option=com_content&view=article&id=14&Itemid=6>

**Метод post** передает все данные формы через http-запрос пользователя к серверу. Этот метод используется чаще, чем get.

Для JavaScript способ передачи данных не важен, обычно указывается post.

PHP взаимодействует с формой переменные. Если используется метод post, то переменные доступны явно. Если используется get, то их значения хранятся в специальном массиве.

## Все вместе: пример формы

Допустим, вы хотите создать форму, в которой пользователь может высказать мнение о вашем сайте. Данные, собранные формой будут обрабатываться на странице process.php.

<form action = "process.php" method = "post">

<b>Please take a moment to tell us what you think about our site:</b><p>

<b>Name:</b><br>

<input type="text" name="name" size="15" maxlength="25" value=""><br>

<b>Email:</b><br>

<input type="text" name="email" size="15" maxlength="45" value=""><br>

<b>How frequently do you visit our site?:</b><br>

<select name="frequency">

<option value="">Site frequency:

<option value="0">This is my first time

<option value="l">&lt; 1 time a month

<option value="2">Roughly once a month

<option value="3">Several times a week

<option value="4">Every day

<option va1ue-"5">I'm addicted

</select><br>

<b>I frequently purchase the following products from our site:</b><br>

<input type="checkbox" name="software" value="software">Software<br>

<input type="checkbox" name="cookware" value="cookware">Cookware<br>

<input type="checkbox" name="hats" value="hats">Chef's Hats<br>

<b>0ur site's greatest asset is:</b><br>

<input type="radio" name="asset" value="products">Product selection<br>

<input type="radio" name="asset" value="design">Cool design<br>

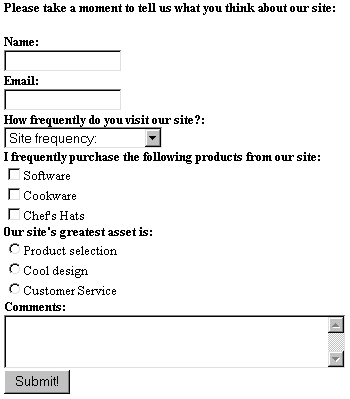
<input type="radio" name="asset" value="service">Customer Service<br>

<b>Comments:</b><br>

<textarea name="comments" rows="3" cols="40"></textarea><br>

<input type="submit" value="Submit!">

</form>



## События

События возникают при каком-либо действии пользователя на сайте: нажатии на кнопку, наведении кнопки мыши и т.д.

События используются, чтобы запустить JavaScript. Например: вычислить сумму двух полей при нажатии на кнопку. Увеличить картинку при наведении кнопки мыши.

Наиболее часто используемые события:

* *onLoad* - выполнение скрипта или функции при загрузке;
* *onChange* - порождается при изменении значения элемента формы;
* *onClick* - порождается при выборе объекта (button, checkbox и т.п.);
* *onSelect*- порождается при выборе текстового объекта (text, textarea);
* *onSubmit* - при нажатии на кнопку Submit;
* *onUnload* - при переходе к другой странице.

События можно привязать как к элементам формы, так и к обычным тегам.

Например,

<input type=text name=weight onchange="change\_weight()">

при вводе каждого символа в поле будет вызываться функция change\_weight().

Другой пример:

<p>Нажмите <font color=red onclick="show\_pic()">сюда</font>, чтобы увидеть картинку.

При нажатии на красный текст «сюда» будет вызываться функция show\_pic, которая, очевидно, отображает картинку на экране.

## Задание. Обработка форм с помощью JavaScript

JavaScript обрабатывает все данные, введенные в форму, сразу же после нажатия кнопки Submit. Эти данные никуда не отправляются.

Обработка данных с формы в PHP описана в лабораторной работе по PHP.

Разработаем страничку с простым тестом, который будет содержать разные типы вопросов:

* с одним правильным вариантом ответа (radio);
* с несколькими правильными вариантами ответа (checkbox);
* со свободным ответом (text);
* на соответствие (select).

Кроме того, должны присутствовать кнопки для проверки и очистки ответов, а также поле, куда мы выведем результат теста.

Вопросы для теста:

1. Как называется язык гипертекстовой разметки документов? (открытый вопрос)
2. Какой тег служит для оформления гиперссылок? (один вариант ответа)
   * H
   * P
   * A
   * LINK
3. Какой тег используется для оформления строк таблицы? (один вариант ответа)
   * TD
   * TR
   * HR
   * BR
4. Укажите атрибуты тега IMG. (несколько вариантов ответа)
   * SIZE
   * SRC
   * BORDER
   * HREF
5. Какие теги используются для оформления списков? (несколько вариантов ответа)
   * OL
   * UL
   * EL
   * LI
6. Дан фрагмент исходного кода страницы:

<p> <font color=green> Варкалось.<b> Хливкие <i> шорьки </i> Пырялись </b> </font> по наве. <b> И хрюкотали <font color=red> зелюки </b> <i>Как мюмзики </font> в мове.</i></p>

<p align=right>Л.Кэррол «Алиса в стране чудес»</p>

Каким шрифтом отобразится фрагмент «И хрюкотали»? (несколько вариантов ответа)

* + курсив
  + полужирный
  + красный
  + зеленый

1. В тексте документа index.htm поставлена отметка:

<a name="my\_anchor">

По какой ссылке можно перейти к этой метке из внешнего документа? (один вариант ответа)

* index.htm#my\_anchor
* index.htm&my\_anchor
* index.htm?my\_anchor
* index.htm.my\_anchor

1. Укажите количество строк в таблице. Ответ введите цифрами. (открытый вопрос)

<table width=60% rows=5 cols=4>

<tr> <td></td> <td></td> <td></td> </tr>

<tr> <td rowspan=2></td> <td colspan=2></td> </tr>

<tr> <td colspan=2></td> </tr>

</table>

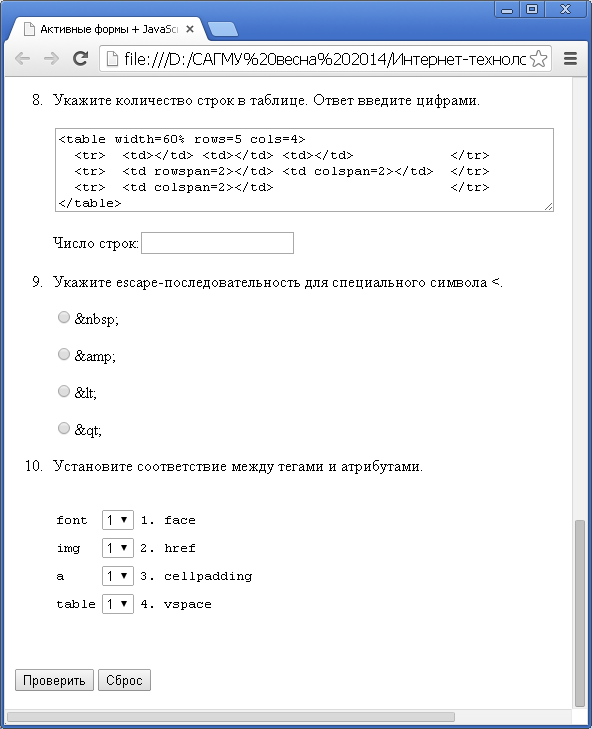
1. Укажите escape-последовательность для специального символа <. (один вариант ответа)

* &nbsp;
* &amp;
* &lt;
* &qt;

1. Установите соответствие между тегами и атрибутами (выбрать номер атрибута из выпадающего списка)

|  |  |
| --- | --- |
| font | 1. face |
| img | 1. href |
| a | 1. cellpadding |
| table | 1. vspace |

Примерный вид в браузере:



На странице HTML необходимо разместить форму для ввода данных:

<HTML>

<HEAD>

<title>Активные формы + JavaScript</title>

</HEAD>

<BODY>

<h1>Тест-тренажер по языку гипертекстовой разметки</h1>

<form id=test\_form>

<ol>

</ol>

</form>

</BODY>

</HTML>

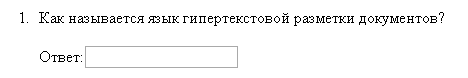
Для нее задан идентификатор test\_form, чтобы была возможность обращаться к элементам на форме из скриптов. Все элементы размещаются внутри пары тегов <form></form>.

Вопросы нумеруются по порядку, используется тег <ol>.

Вопрос с открытым ответом, ответ вводится через поле ввода text:

<li>Как называется язык гипертекстовой разметки документов?

<p>Ответ:<input type=text name=answ1 value=""></p>



Текстовому полю присвоен уже не идентификатор, а имя answ1. Если задать идентификатор, то появится всплывающая подсказка с ранее введенными вариантами.

Тестовый вопрос с одним вариантом ответа, вводится через radio:

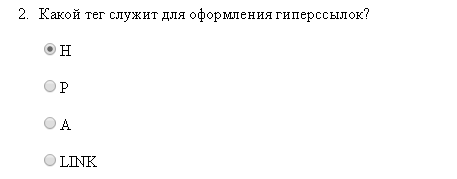
<li>Какой тег служит для оформления гиперссылок?

<p><input type=radio name=answ2>H</p>

<p><input type=radio name=answ2>P</p>

<p><input type=radio name=answ2>A</p>

<p><input type=radio name=answ2>LINK</p>



Все варианты ответа на один вопрос имеют одинаковое имя (answ2). Это показывает, что эти переключатели не могут быть выбраны одновременно.

Вопрос с несколькими вариантами ответа, элемент ввода checkbox:

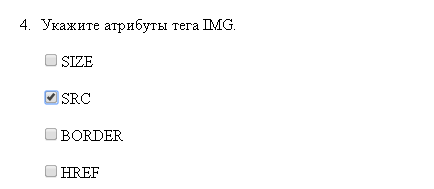
<li>Укажите атрибуты тега IMG.

<p><input type=checkbox name=answ4>SIZE</p>

<p><input type=checkbox name=answ4>SRC</p>

<p><input type=checkbox name=answ4>BORDER</p>

<p><input type=checkbox name=answ4>HREF</p>



В этом случае также желательно использовать одинаковое имя для вариантов ответа на один вопрос, хотя разные имена не вызовут ошибок.

Вопрос на соответствие можно оформить с помощью выпадающего списка:

<li>Установите соответствие между тегами и атрибутами.

<pre>

<table border=0>

<tr><td>font</td><td><select name=answ10\_font>

<option>1<option>2<option>3<option>4</select></td>

<td>1. face</td>

</tr>

<tr><td>img</td><td><select name=answ10\_img>

<option>1<option>2<option>3<option>4</select></td>

<td>2. href</td>

</tr>

<tr><td>a</td><td><select name=answ10\_a>

<option>1<option>2<option>3<option>4</select></td>

<td>3. cellpadding</td>

</tr>

<tr><td>table</td><td><select name=answ10\_table>

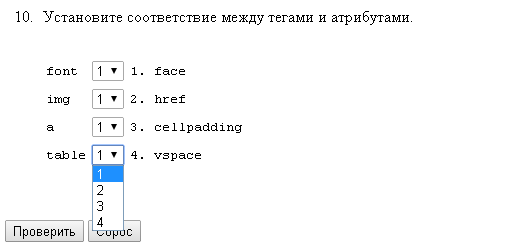
<option>1<option>2<option>3<option>4</select></td>

<td>4. vspace</td>

</tr>

</table>

</pre>



В данном случае каждый выпадающий список имеет свое имя: answ10\_font, answ10\_img, answ10\_a, answ10\_table, поскольку нам потребуется выбранное значение для каждого из них.

Тег <pre></pre> обрамляет предварительно отформатированный текст (листинг и т.п.). Отображается моноширинным шрифтом, наподобие Courier New.

В два столбца ответы распределены с помощью таблицы с нулевой шириной обводки: <table border=0>.

В нескольких вопросах мы сталкиваемся с проблемой: нужно отобразить теги на странице. Для отображения специальных символов используются escape-последовательности. Но без специальной программы заменить каждый знак < > & на &lt; &gt; &amp; не очень удобно, а полученный текст теряет читабельность.

Как вариант, можно разместить вставки с HTML-кодом внутрь поля для ввода textarea. В нем теги не учитываются.

Например:

<p><textarea cols=80 rows=3 readonly><p> <font color=green> Варкалось.<b> Хливкие <i> шорьки </i> Пырялись </b> </font> по наве. <b> И хрюкотали <font color=red> зелюки </b> <i>Как мюмзики </font> в мове.</i></p></textarea></p>



Атрибут readonly запрещает редактирование текста, cols и rows задают размер поля для ввода в символах.

В конце теста необходимо добавить две кнопки: для подтверждения и сброса результатов.

<input type=button value="Проверить" onclick="CheckAnswers()">

<input type=reset value="Сброс">

Первая кнопка имеет тип button, а не submit, поскольку submit очищает форму от введенных результатов и перезагружает страницу.

Теперь рассмотрим скрипт для проверки результатов теста. Количество правильных ответов будем хранить в локальной переменной RightAnswers, выводить пока через alert:

function CheckAnswers()

{

var RightAnswers = 0;

//...

alert('Результат: ' + RightAnswers);

};

Проверка вопроса с открытым ответом – простая проверка на равенство:

if (test\_form.answ1.value.toUpperCase() == 'HTML')

{ RightAnswers += 1 };

К элементу мы обращаемся через форму: test\_form.answ1. Просто answ1.value работать не будет.

Метод toUpperCase() переводит строку в верхний регистр. Таким образом, ответ становится нечувствительным к регистру символов, и html, Html будут правильными ответами.

Проверка вопроса с одним правильным ответом:

if (test\_form.answ2[2].checked)

{ RightAnswers += 1 };

Все элементы с одинаковым именем помещаются в массив, нумерация идет с 0. Поэтому answ2[2] – это третий вариант ответа на второй вопрос. Свойство checked равно true, если вариант выбран и false в противном случае.

Проверка нескольких вариантов ответа выполняется аналогично, ну нужно для каждого варианта проверить, выбран он или нет:

if (!test\_form.answ4[0].checked &&

test\_form.answ4[1].checked &&

test\_form.answ4[2].checked &&

!test\_form.answ4[3].checked)

{ RightAnswers += 1 };

Восклицательный знак ! означает отрицание. Т.е. первый [0] и последний [3] варианты должны быть **не** отмечены.

Проверка ответа на вопрос на соответствие аналогична открытому ответу для каждой строки:

if ((test\_form.answ10\_font.value == '1') &&

(test\_form.answ10\_img.value == '4') &&

(test\_form.answ10\_a.value == '2') &&

(test\_form.answ10\_table.value == '3'))

{ RightAnswers += 1 };

Самостоятельно оформите остальные вопросы по аналогии.

Теперь проверка нашего теста работает. Но вывод результата через alert – не лучшее решение. Кроме того, неплохо бы помешать редактировать ответы после получения результата.

Для этого добавим на страницу элемент div, в который будем выводить результат тестирования. Причем кнопки «Проверить» и «Сброс» поместим внутрь того же div – после выполнения проверки они будут удаляться со страницы.

<div id=test\_result>

<input type=button value="Проверить" onclick="CheckAnswers()">

<input type=reset value="Сброс">

</div>

В JavaScript отредактируем последнюю строчку:

test\_result.innerHTML = 'Результат: ' + RightAnswers + '<br><input type=reset value="Заново" onclick="ResetAnswers()">';

Обратите внимание, поскольку в текстовой строке присутствуют символы " ", сама она обрамляется в одинарные кавычки ' '.

Теперь вместо вывода результата в alert, он отобразится на странице, кнопки «Проверить» и «Сброс» исчезнут, а вместо них появится кнопка «Заново». Для нее нужно написать еще одну функцию ResetAnswers(), которая очистит ответы и вернет кнопки назад.

function ResetAnswers()

{

test\_result.innerHTML = '<input type=button value="Проверить" onclick="CheckAnswers()"><input type=reset value="Сброс">';

test\_form.reset();

}

Самостоятельно доработайте тест так, чтобы возле каждого вопроса выводилось, правильный дан ответ или нет.