

Формы и элементы форм в HTML.

Сейчас практически ни один сайт не обходится без элементов интерфейса вроде полей ввода текста, кнопок, переключателей и флажков. Они необходимы для взаимодействия с пользователем, чтобы он мог искать на сайте по ключевым словам, писать комментарии, отвечать на опросы, прикреплять фотографии и делать много других подобных вещей. Именно формы и обеспечивают получение данных от пользователя и передачу их на сервер, где они уже подвергаются анализу и обработке.

Формы являются одним из важных элементов любого сайта и предназначены для обмена данными между пользователем и сервером. Для получения и обработки данных форм используются языки веб-программирования, такие как **PHP**, **Perl**. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Обработка элементов формы производится с помощью скриптов, но они могут и вообще никак не обрабатываться.

В HTML 4.01 определены следующие типы управляющих элементов:

1. Кнопки - задаются с помощью элементов **BUTTON** и **INPUT**. Различают:

- кнопки отправки - при нажатии на них, они осуществляют отправку формы серверу;

- кнопки сброса - при нажатии на них, управляющие элементы принимают первоначальные значения;

- прочие кнопки - кнопки, для которых не указано действие, выполняемое по умолчанию при нажатии на них.

2. Зависимые переключатели (переключатели с зависимой фиксацией) - задаются элементом **INPUT** и представляют собой переключатели "вкл/выкл". Если несколько зависимых переключателей имеют одинаковые имена, то они являются взаимоисключающими. Это значит, что если одна из них ставится в положение "вкл", то все остальные автоматически - в положение "выкл". Именно это и является преимуществом их использования.

3. Независимые переключатели (переключатели с независимой фиксацией) - задаются элементом **INPUT** и представляют собой переключатели "вкл/выкл", но в отличие от зависимых, независимые переключатели могут принимать и изменять свое значение независимо от остальных переключателей. Даже если последние имеют такое же имя.

4. Меню - реализуется с помощью элементов **SELECT**, **OPTGROUP** и **OPTION**. Меню предоставляют пользователю список возможных вариантов выбора.

5. Ввод текста - реализуется элементами **INPUT**, если вводится одна строка, и элементами **TEXTAREA** - если несколько строк. В обоих случаях введенный текст становится текущим значением управляющего элемента.

6. Выбор файлов - позволяет вместе с формой отправлять выбранные файлы, реализуется HTML-элементом **INPUT**.

7. Скрытые управляющие элементы - создаются управляющим элементом **INPUT**.

Как видите, очень много элементов задаются с помощью универсального тега **INPUT**.

С появлением HTML5 формы получили новые типы полей ввода и новые атрибуты для полей и форм.

Формы		
<code><form>/</form></code>	html-форма	block
<code><input></code>	многофункциональные поля формы	inline-block
<code><textarea></code>	многострочное поле формы	inline-block
<code><label>/</label></code>	текстовая метка для элемента <code><input></code>	inline
<code><datalist>/</datalist></code>	контейнер для выпадающего списка элемента <code><input></code> с <code><option></code> -значениями	none
<code><select>/</select></code>	элемент управления с выбором значений из предложенных вариантов <code><option></code>	inline-block
<code><optgroup>/</optgroup></code>	контейнер с заголовком для группы элементов <code><option></code>	block
<code><option>/</option></code>	вариант (опция) в раскрывающемся списке	block
<code><fieldset>/</fieldset></code>	группирует связанные элементы в форме	block
<code><legend>/</legend></code>	заголовок элементов формы, сгруппированных с помощью <code><fieldset></code>	block
<code><button>/</button></code>	интерактивная кнопка	inline-block
<code><keygen></code>	генерирует закрытый и открытый ключи	inline-block
<code><progress>/</progress></code>	индикатор выполнения задачи	inline-block
<code><meter>/</meter></code>	индикатор измерения в заданном диапазоне	inline-block
<code><output>/</output></code>	поле для вывода результата вычисления	inline

Кроме того, **HTML5** предоставляет возможность изменять внешний вид элементов форм за счет CSS3.

Создание формы

Форма создаётся с помощью тега `<form>`, внутри которой могут быть любые необходимые теги. Элемент `<form>...</form>` не предусматривает ввод данных, так как является контейнером, удерживая вместе все элементы управления формы – поля. Атрибуты этого элемента содержат информацию, общую для всех полей формы, поэтому в одну форму нужно включать поля, объединенные логически.

Документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга.

Для отправки формы на сервер используется кнопка **Submit**, того же можно добиться, если нажать клавишу **Enter** в пределах формы. Если кнопка Submit отсутствует в форме, клавиша Enter имитирует ее использование.

Когда форма отправляется на сервер, управление данными передается программе, заданной атрибутом **action** тега `<form>`. Предварительно браузер подготавливает информацию в виде пары `«имя=значение»`, где имя определяется атрибутом **name** тега `<input>`, а значение введено пользователем или установлено в поле формы по умолчанию. Если для отправки данных используется метод **GET**, то адресная строка может принимать следующий вид.

```
http://www.htmlbook.ru/cgi-bin/handler.cgi?nick=%C2%E0%ED%FF+%D8%E0%EF%EE%F7%EA%E8%ED&page=5
```

Параметры перечисляются после вопросительного знака, указанного после адреса CGI-программы и разделяются между собой символом амперсанда (&). Нелатинские

символы преобразуются в шестнадцатеричное представление (в форме %HH, где HH — шестнадцатеричный код для значения ASCII-символа), пробел заменяется на плюс (+).

Допускается внутри контейнера **<form>** помещать другие теги, при этом сама форма никак не отображается на веб-странице, видны только ее элементы и результаты вложенных тегов.

Синтаксис

```
<form action="URL">
...
</form>
```

Начальный и конечный теги FORM задают границы формы, поэтому их указание является обязательным.

Если управляющие элементы указаны вне содержимого тега FORM, то они не создают форму, а используются для построения пользовательского интерфейса на веб-странице, то есть для привнесения в нее различных кнопок, флажков, полей ввода. Имена элементам формы присваиваются через их атрибут NAME.

Атрибуты тега </form>

Атрибут	Значение / описание
accept-charset	Значение атрибута представляет собой разделенный пробелами список кодировок символов , которые будут использоваться для отправки формы, например, <code><form accept-charset="ISO-8859-1"></code> .
action	<p>Обязательный атрибут, который указывает url обработчика формы на сервере, которому передаются данные. Представляет из себя файл (например, action.php), в котором описано, что нужно делать с данными формы. Если значение атрибута не будет указано, то после перезагрузки страницы элементы формы примут значения по умолчанию.</p> <p>В случае, если вся работа будет выполняться на стороне клиента сценариями JavaScript, то для атрибута action можно указать значение #.</p> <p>Также можно сделать так, чтобы заполненная посетителем форма приходила вам на почту. Для этого нужно внести следующую запись:</p> <pre><form action="mailto:адрес вашей электронной почты" enctype="text/plain"></form></pre>
autocomplete	<p>Отвечает за запоминание введенных в текстовое поле значений и автоподстановку их при последующем вводе:</p> <p>on — означает, что поле не защищено, и его значение можно сохранять и извлекать,</p> <p>off — отключает автозаполнение для полей форм.</p>
enctype	<p>Используется для указания MIME-типа данных, отправляемых вместе с формой, например, <code>enctype="multipart/form-data"</code>. Указывается только в случае <code>method="post"</code>.</p> <p>application/x-www-form-urlencoded — тип содержимого по умолчанию, указывает на то, что передаваемые данные представляют список URL-кодированных переменных формы. Символы пробела (ASCII 32) будут</p>

	<p>закодированы как <code>+</code>, а специальный символ, например, такой как <code>!</code> будет закодирован шестнадцатичной форме как <code>%21</code>.</p> <p>multipart/form-data — используется для отправки форм, содержащих файлы, не-ASCII данные и бинарные данные, состоит из нескольких частей, каждая из которых представляет содержимое отдельного элемента формы.</p> <p>text/plain — указывает на то, что передается обычный (не html) текст.</p>
method	<p>Задаёт способ передачи данных формы.</p> <p>Метод get передает данные на сервер через адресную строку браузера. При формировании запроса к серверу все переменные и их значения формируют последовательность вида <code>www.anysite.ru/form.php?var1=1&var2=2</code>. Имена и значения переменных присоединяются к адресу сервера после знака <code>?</code> и разделяются между собой знаком <code>&</code>. Все специальные символы и буквы, отличные от латинских, кодируются в формате <code>%nn</code>, пробел заменяется на <code>+</code>. Этот метод нужно использовать, если вы не передаете больших объемов информации. Если вместе с формой предполагается отправка какого-либо файла, этот метод не подойдет.</p> <p>Метод post применяется для пересылки данных больших объемов, а также конфиденциальной информации и паролей. Данные, отправляемые с помощью этого метода, не видны в заголовке URL, так как они содержатся в теле сообщения.</p> <pre><form action="action.php" enctype="multipart/form-data" method="post"></form></pre>
name	<p>Задаёт имя формы, которое будет использоваться для доступа к элементам формы через сценарии, например, <code>name="opros"</code>.</p>
novalidate	<p>Отключает проверку в кнопке для отправки формы. Атрибут используется без указания значения</p>
target	<p>Указывает окно, в которое будет направлена информация:</p> <ul style="list-style-type: none"> <code>_blank</code> — новое окно; <code>_self</code> — тот же фрейм; <code>_parent</code> — родительский фрейм (если он существует, если нет — то в текущий); <code>_top</code> — окно верхнего уровня по отношению к данному фрейму. Если вызов происходит не из дочернего фрейма, то в тот же фрейм.

Также для этого тега доступны универсальные атрибуты и события.

Пример:

```
<!DOCTYPE HTML>
<html>
<head>
  <meta charset=" windows-1251">
  <title>Данные формы</title>
</head>
<body>
```

```
<form action="/example/handler.php">
  <p><input name="login"> <input type="password" name="pass"></p>
  <p><input type="submit"></p>
</form>
</body>
</html>
```

В этом примере данные формы, обозначенные атрибутом *name* (*login* и *password*), будут переданы в файл по адресу */example/handler.php*. Если атрибут **action** не указывать, то передача происходит на адрес текущей страницы.

Часто бывает, что текущая страница написанная, допустим, на PHP, сама является обработчиком формы, в таком случае можно указать пустое значение атрибута *action* или вообще его опустить. В этом случае содержимое формы будет отправлено на URL-адрес, с которого загружалась данная веб-страница. В простейшем случае тег `<form>` не содержит никаких атрибутов.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=" windows-1251">
    <title>Формы</title>
  </head>
  <body>
    <form>
      <p><input name="a"> <input type="submit"></p>
    </form>
  </body>
</html>
```

В качестве значения атрибута **action** можно указать также адрес электронной почты, начиная его с ключевого слова **mailto:**. При отправке формы будет запущена почтовая программа, установленная по умолчанию. В целях безопасности в браузере задано, что отправить незаметно информацию, введенную в форме, по почте невозможно. Для корректной интерпретации данных используйте атрибут *enctype* со значением *text/plain* в теге `<form>`.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="windows-1251">
    <title>Формы</title>
  </head>
  <body>
    <form action="mailto:office@bsuir-mrk.by" enctype="text/plain">
      <p><input name="a"> <input type="submit"></p>
    </form>
  </body>
</html>
```

Браузеры неоднозначно работают с таким кодом.

Допускается использовать несколько форм на странице, но они не должны вкладываться одна в другую.

Передача на сервер происходит двумя разными методами: GET и POST, для задания метода в теге <form> используется атрибут method, а его значениями выступают ключевые слова get и post. Если атрибут method не задан, то по умолчанию данные отправляются на сервер методом GET. В таблице показаны различия между этими методами.

	GET	POST
Ограничение на объём	4 Кб	Ограничения задаются сервером.
Передаваемые данные	Видны сразу всем.	Видны только при просмотре через расширения браузера или другими методами.
Кэширование	Страницы с разными запросами считаются различными, их можно кэшировать как отдельные документы.	Страница всегда одна.
Закладки	Страницу с запросом можно добавить в закладки браузера и обратиться к ней позже.	Страницы с разными запросами имеют один адрес, запрос повторить нельзя.

Какой метод используется легко определить по адресной строке браузера. Если в ней появился вопросительный знак и адрес стал похож на этот, то это точно GET.

<http://www.google.ru/search?q=%D1%81%D0%B8%D1%81%D1%8C%D0%BA%D0%B8&ie=utf-8>

Уникальное сочетание параметров в адресной строке однозначно идентифицирует страницу, так что страницы с адресами ?q=node/add и ?q=node считаются разными. Эту особенность используют системы управления контентом (CMS, Content management system) для создания множества страниц сайта. В реальности же используется один файл, который получает запрос GET и согласно ему формирует содержимое документа.

Ниже перечислены типовые области применения этих методов на сайтах.

GET:

Передача небольших текстовых данных на сервер, поиск по сайту: поисковые системы, формы поиска по сайту всегда отправляются методом GET, это позволяет делиться результатами поиска с друзьями, слать ссылку по почте или выкладывать её на форуме.

POST:

Пересылка файлов (фотографий, архивов, программ и др.); отправка комментариев; добавление и редактирование сообщений на форуме, блоге. Работа с формой по умолчанию происходит в текущей вкладке браузера, при этом допустимо при отправке формы изменить этот параметр и открывать обработчик формы в новой вкладке или во фрейме. Такое поведение задаётся через «имя контекста», которое выступает значением атрибута target тега <form>. Популярные значения это _blank для открытия формы в новом окне или вкладке, и имя фрейма, которое задаётся атрибутом name тега <iframe>.

Пример (открытие формы во фрейме):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="windows-1251">
    <title>Использование фрейма</title>
  </head>
  <body>
    <p><iframe name="area" width="500" height="200"></iframe></p>
    <form action="handler.php" target="area">
      <p><input placeholder="Введите текст" name="text">
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

В данном примере при нажатии на кнопку «Отправить» результат отправки формы открывается во фрейме с именем **area**.

Элементы формы традиционно располагаются внутри тега `<form>`, тем самым определяя те данные, которые будут передаваться на сервер. В то же время в HTML5 есть возможность отделить форму от её элементов. Это сделано для удобства и универсальности, так, сложный макет может содержать несколько форм, которые не должны пересекаться меж собой или к примеру, некоторые элементы выводятся с помощью скриптов в одном месте страницы, а сама форма находится в другом. Связь между формой и её элементами происходит в таком случае через идентификатор формы, а к элементам следует добавить атрибут `form` со значением, равным этому идентификатору.

Пример (связывание формы с полями):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=" windows-1251">
    <title>Форма</title>
  </head>
  <body>
    <form id="auth" action="handler.php" method="post"></form>
    <p>...</p>
    <p><input name="login" form="auth">
    <input type="password" name="pass" form="auth"></p>
    <p><input type="submit" form="auth"></p>
  </body>
</html>
```

В этом примере тег `<form>` однозначно отождествляется через идентификатор `auth`, а к полям, которые следует отправить с помощью формы, добавляется `form="auth"`. При этом поведение элементов не меняется, при нажатии на кнопку логин и пароль пересылаются на обработчик `handler.php`.

Хотя параметры передачи формы традиционно указываются в теге `<form>`, их можно перенести и в кнопки отправки формы (`<button>` и `<input type="submit">`). Для этого применяется набор атрибутов **formaction**, **formmethod**, **formenctype** и **formtarget**, которые являются аналогами соответствующих атрибутов без приставки `form`.

Пример (отправка формы):

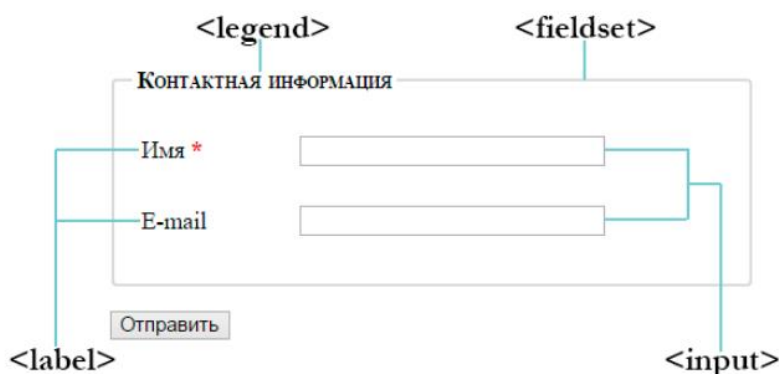
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=" windows-1251">
    <title>Отправка формы</title>
  </head>
  <body>
    <form>
      <p><input placeholder="Ваше имя" name="user"></p>
      <p><input type="submit" value="Отправить" formaction="handler.php" formmethod="post"></p>
    </form>
  </body>
</html>
```

Группировка элементов формы

Элемент **<fieldset>...</fieldset>** предназначен для группировки элементов, связанных друг с другом, разделяя таким образом форму на логические фрагменты.

Каждой группе элементов можно присвоить название с помощью элемента **<legend>**, который идет сразу за тегом **<fieldset>**. Название группы проявляется слева в верхней границе **<fieldset>**. Например, если в элементе **<fieldset>** хранится контактная информация:

```
<form>
  <fieldset>
    <legend>Контактная информация</legend>
    <p><label for="name">Имя <em>*</em></label><input type="text" id="name"></p>
    <p><label for="email">Е-mail</label><input type="email" id="email"></p>
  </fieldset>
  <p><input type="submit" value="Отправить"></p>
</form>
```



Атрибуты тега <fieldset>

Атрибут	Значение / описание
disabled	Если атрибут присутствует, то группа связанных элементов формы, находящихся внутри контейнера <fieldset> , отключены для заполнения и редактирования. Используется для ограничения доступа к некоторым полям формы, содержащих ранее введенные данные. Атрибут используется без указания значения — <fieldset disabled> .

Form	Значение атрибута должно быть равно атрибуту <code>id</code> элемента <code><form></code> в этом же документе. Указывает на одну или несколько форм, к которым принадлежит данная группа элементов. На данный момент атрибут не поддерживается ни одним браузером.
Name	Определяет имя , которое будет использоваться для ссылки на элементы в JavaScript, или для ссылки на данные формы после заполнения и отправки формы. Является аналогом атрибута <code>id</code> .

Тег INPUT и его методы

Элемент **INPUT** является наиболее употребительным тегом HTML-форм. С помощью этого тега реализуются основные функции формы.

Элемент `<input>` создает большинство полей формы. Атрибуты элемента отличаются в зависимости от типа поля, для создания которого используется этот элемент.

Обратите внимание на особенность **INPUT** - у него нет конечного (завершающего) тега.

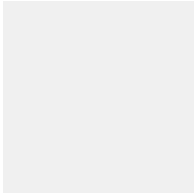
С помощью *CSS*-стилей можно изменить размер шрифта, тип шрифта, цвет и другие свойства текста, а также добавить границы, цвет фона и фоновое изображение. Ширина поля задается свойством `width`.

Атрибуты тега `<fieldset>`

Атрибут	Значение / описание
<code>accept</code>	Определяет тип файла, разрешенных для отправки на сервер. Указывается только для <code><input type="file"></code> . Возможные значения: <code>file_extension</code> — разрешает загрузку файлов с указанным расширением, например, <code>accept=".gif"</code> , <code>accept=".pdf"</code> , <code>accept=".doc"</code> <code>audio/*</code> — разрешает загрузку аудиофайлов <code>video/*</code> — разрешает загрузку видеофайлов <code>image/*</code> — разрешает загрузку изображений <code>media_type</code> — указывает на медиа-тип загружаемых файлов.
<code>Alt</code>	Определяет альтернативный текст для изображений, указывается только для <code><input type="image"></code> .
<code>autocomplete</code>	Отвечает за запоминание введенных в текстовое поле значений и автоподстановку их при последующем вводе: <code>on</code> — означает, что поле не защищено, и его значение можно сохранять и извлекать, <code>off</code> — отключает автозаполнение для полей форм.
<code>autofocus</code>	Позволяет сделать так, чтобы в загружаемой форме то или иное поле ввода уже имело фокус (было выбрано), являясь готовым к вводу значения.
<code>checked</code>	Атрибут проверяет, установлен ли флажок по умолчанию при загрузке страницы для полей типа <code>type="checkbox"</code> и <code>type="radio"</code> .
<code>disabled</code>	Отключает возможность редактирования и копирования содержимого поля.

<code>form</code>	Значение атрибута должно быть равно атрибуту <code>id</code> элемента <code><form></code> в этом же документе. Определяет одну или несколько форм, которым принадлежит данное поле формы.
<code>formaction</code>	Задаёт <code>url</code> файла, который будет обрабатывать введенные в поля данные при отправке формы. Задаётся только для полей типа <code>type="submit"</code> и <code>type="image"</code> . Атрибут переопределяет значение атрибута <code>action</code> самой формы.
<code>formenctype</code>	Определяет, как будут кодироваться данные полей формы при отправке на сервер. Переопределяет значение атрибута <code>enctype</code> формы. Задаётся только для полей типа <code>type="submit"</code> и <code>type="image"</code> . Варианты: <code>application/x-www-form-urlencoded</code> — значение по умолчанию. Все символы кодируются перед отправкой (пробелы заменяются на символ <code>+</code> , специальные символы преобразуются в значения ASCII HEX) <code>multipart/form-data</code> — символы не кодируются <code>text/plain</code> — пробелы заменяются на символ <code>+</code> , а специальные символы не кодируются.
<code>formmethod</code>	Атрибут определяет метод, который браузер будет использовать для отправки данных формы на сервер. Задаётся только для полей типа <code>type="submit"</code> и <code>type="image"</code> . Переопределяет значение атрибута <code>method</code> формы. Варианты: <code>get</code> — значение по умолчанию. Данные из формы (пара имя/значение) добавляются в <code>url</code> -адрес и отправляются на сервер: <code>URL?имя=значение&имя=значение</code> <code>post</code> — данные формы отправляются в виде <code>http</code> -запроса.
<code>formnovalidate</code>	Определяет, что данные полей формы не должны проверяться при отправке формы. Переопределяет значение атрибута <code>novalidate</code> формы. Можно использовать без указания значения атрибута.
<code>formtarget</code>	Определяет, где выводить ответ, полученный после отправки формы. Задаётся только для полей типа <code>type="submit"</code> и <code>type="image"</code> . Переопределяет значение атрибута <code>target</code> формы. <code>_blank</code> — загружает ответ в новое окно/вкладку <code>_self</code> — загружает ответ в то же окно (значение по умолчанию) <code>_parent</code> — загружает ответ в родительский фрейм <code>_top</code> — загружает ответ во весь экран <code>framename</code> — загружает ответ во фрейм с указанным именем.
<code>height</code>	Значение атрибута содержит количество пикселей без указания единицы измерения. Устанавливает высоту поля формы типа <code>type="image"</code> , например, <code><input type="image" src="img_submit.gif" height="50"></code> . Рекомендуется одновременно устанавливать как высоту, так и ширину поля.

list	Является ссылкой на элемент <code><datalist></code> , содержит его <code>id</code> . Позволяет предоставить пользователю несколько вариантов на выбор, когда он начинает вводить значение в соответствующем поле.
max	Позволяет ограничить допустимый ввод числовых данных максимальным значением, значение атрибута может содержать целое или дробное число. Рекомендуется использовать этот атрибут вместе с атрибутом <code>min</code> . Работает со следующими типами полей: <code>number</code> , <code>range</code> , <code>date</code> , <code>datetime</code> , <code>datetime-local</code> , <code>month</code> , <code>time</code> и <code>week</code> .
maxlength	Атрибут задает максимальное количество символов, вводимых в поле. Значение по умолчанию 524288 символов.
min	Позволяет ограничить допустимый ввод числовых данных минимальным значением.
multiple	Позволяет пользователю ввести несколько значений атрибутов, разделяя их запятой. Применяется в отношении файлов и адресов электронной почты. Указывается без значения атрибута.
name	Определяет имя, которое будет использоваться для доступа к элементу <code><form></code> , к примеру, в таблицах стилей <code>css</code> . Является аналогом атрибута <code>id</code> .
pattern	Позволяет определять с помощью регулярного выражения синтаксис данных, ввод которых должен быть разрешен в определенном поле. Например, <code>pattern="[a-z]{3}-[0-9]{3}"</code> — квадратные скобки устанавливают диапазон допустимых символов, в данном случае — любые строчные буквы, число в фигурных скобках указывает, что нужны три строчные буквы, после которых следует тире, далее — три цифры в диапазоне от 0 до 9.
placeholder	Содержит текст, который отображается в поле ввода до заполнения (чаще всего это подсказка).
readonly	Не позволяет пользователю изменять значения элементов формы, выделение и копирование текста при этом доступно. Указывается без значения атрибута.
required	Выводит сообщение о том, что данное поле является обязательным для заполнения. Если пользователь попытается отправить форму, не введя в это поле требуемое значение, то на экране отобразится предупреждающее сообщение. Указывается без значения атрибута.
size	Задаёт видимую ширину поля в символах. Значение по умолчанию — 20. Работает со следующими типами полей: <code>text</code> , <code>search</code> , <code>tel</code> , <code>url</code> , <code>email</code> и <code>password</code> .
src	Задаёт <code>url</code> изображения, используемого в качестве кнопки отправки данных формы. Указывается только для поля <code><input type="image"></code> .
step	Используется для элементов, предполагающих ввод числовых значений, указывает величину увеличения или уменьшения значений в

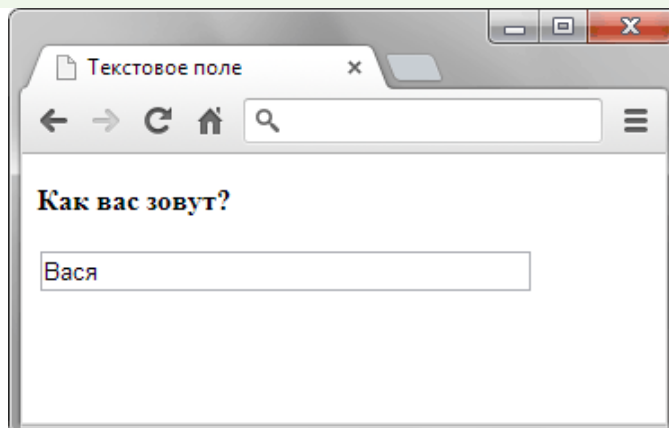
	процессе регулировки диапазона (шаг).
type	<code>button</code> — создает кнопку.
	<code>checkbox</code> — превращает поле ввода во флажок, который можно установить или очистить, например, <input type="checkbox"/> У меня есть автомобиль
	<code>color</code> — генерирует палитры цветов в поддерживающих браузерах, давая пользователям возможность выбирать значения цветов в шестнадцатеричном формате.
	<code>date</code> — позволяет вводить дату в формате дд.мм.гггг. День рождения:
	<code>datetime-local</code> — позволяет вводить дату и время, разделенные прописной английской буквой <code>T</code> по шаблону дд.мм.гггг чч:мм. День рождения — день и время:
	<code>email</code> — браузеры, поддерживающие данный атрибут, будут ожидать, что пользователь введет данные, соответствующие синтаксису адресов электронной почты. E-mail:
	<code>file</code> — позволяет загружать файлы с компьютера пользователя. Выберите файл:
	<code>hidden</code> — скрывает элемент управления, который не отображается браузером и не дает пользователю изменять значения по умолчанию.
	<code>image</code> — создает кнопку, позволяя вместо текста на кнопке вставить изображение. 
	<code>month</code> — позволяет пользователю вводить год и номер месяца по шаблону гггг-мм.
	<code>number</code> — предназначено для ввода целочисленных значений. Его атрибуты <code>min</code> , <code>max</code> и <code>step</code> задают верхний, нижний пределы и шаг между значениями соответственно. Эти атрибуты предполагаются у всех элементов, имеющих численные показатели. Их значения по умолчанию зависят от типа элемента. Укажите количество (от 1 до 5):
	<code>password</code> — создает текстовые поля в форме, при этом вводимые пользователем символы заменяются на звездочки, маркеры, либо

	<p>другие, установленные браузером значки.</p> <p>Введите пароль: <input type="password"/></p>
	<p>radio — создает переключатель — элемент управления в виде небольшого кружка, который можно включить или выключить.</p> <p>Вегетарианец: <input type="radio"/></p>
	<p>range — позволит создать такой элемент интерфейса, как ползунок, min / max — позволят установить диапазон выбора</p>
	<p>reset — создает кнопку, которая очищает поля формы от введенных пользователем данных.</p> <p><input type="reset"/></p>
	<p>search — обозначает поле поиска, по умолчанию поле ввода имеет прямоугольную форму.</p> <p>Поиск: <input type="search"/></p>
	<p>submit — создает стандартную кнопку, активизируемую щелчком мыши. Кнопка собирает информацию с формы и отправляет ее для обработки.</p> <p><input type="submit"/></p>
	<p>text — создает текстовые поля в форме, выводя однострочное текстовое поле для ввода текста.</p> <p><input type="text"/></p>
	<p>time — позволяет вводить время в 24-часовом формате по шаблону чч.мм. В поддерживающих браузерах оно отображается как элемент управления в виде числового поля ввода со значением, изменяемым с помощью мыши, и допускает ввод только значений времени.</p> <p>Укажите время: <input type="time"/></p>
	<p>url — поле предназначено для указания URL-адресов.</p> <p>Главная страница: <input type="url"/></p>
	<p>week — соответствующий инструмент-указатель позволяет пользователю выбрать одну неделю в году, после чего обеспечит ввод данных в формате гг-мм-дд. В зависимости от года число недель может быть 52 или 53.</p> <p>Укажите неделю: <input type="week"/></p>
value	<p>Определяет текст, отображаемый на кнопке, в поле или связанный текст. Не указывается для полей типа file.</p>
width	<p>Значение атрибута содержит количество пикселей. Позволяет задать ширину полей формы.</p>

Однострочное текстовое поле (поле ввода)

Однострочное текстовое поле предназначено для ввода строки символов с помощью клавиатуры. Синтаксис создания такого поля следующий:

```
<input maxlength="25" size="40" value="Вася">
```



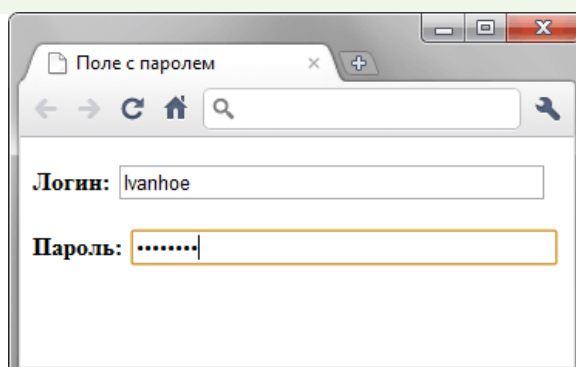
Значение атрибута **type** для тега **<input>** по умолчанию определено как **text**, поэтому его можно не указывать явно. Атрибуты текстового поля перечислены в таблице.

Атрибут	Описание
Size	Ширина текстового поля, которая определяется числом символов моноширинного шрифта. Иными словами, ширина задается количеством близстоящих букв одинаковой ширины по горизонтали (знакомест).
Maxlength	Устанавливает максимальное число символов, которое может быть введено пользователем в текстовом поле. Когда это количество достигается при наборе, дальнейший ввод становится невозможным. Если этот атрибут опустить, то можно вводить строку длиннее самого поля.
Name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
Value	Начальный текст отображаемый в поле (в поле будет изначально отображаться значение данного атрибута).

Поле для пароля

Поле для пароля — обычное текстовое поле, но отличается от него тем, что все вводимые символы отображаются звездочками, точками или другими знаками (это зависит от браузера). Поле предназначено для того, чтобы никто не подглядел вводимый пароль.

```
<form action="handler.php">
  <p><strong>Логин:</strong>
  <input maxlength="25" size="40" name="login"></p>
  <p><strong>Пароль:</strong>
  <input type="password" maxlength="25" size="40" name="password"></p>
</form>
```



Атрибуты совпадают с атрибутами текстового поля. Не рекомендуется устанавливать значение по умолчанию из соображений безопасности (value).

3. Скрытое текстовое поле

Часто возникает ситуация, когда требуется передать в форме некоторые промежуточные данные, которые не должны изменяться пользователем. Более того, такие данные не должны показываться пользователю, поскольку носят технический характер и обычно служат для передачи некоторой информации от страницы к странице. Для этой цели применяется скрытое поле, оно не отображается на странице и прячет своё содержимое от пользователя. Посетитель не может в него ничего внести или напечатать.

```
<input type="hidden" name="name" value="Vasya">
<input type="hidden" name="password" value="rupkin"></p>
```

Атрибуты:

Атрибут	Описание
Name	Имя поля для его идентификации обработчиком формы.
Value	Значение поля, определяющее, какая информация будет отправлена на сервер.

Такие поля передаются серверу, но на веб-странице не отображаются.

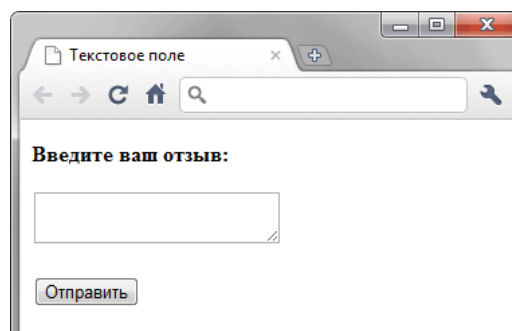
Многострочные текстовые поля ввода

Элемент формы **<textarea>** предназначен для создания области, в которой можно вводить несколько строк текста. В таком текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер. *Поле для многострочного текста незаменимо для добавления комментариев к статьям, написания сообщений форума, вставки и редактирования постов в блоге и во многих других случаях, когда одной строки текста явно недостаточно.*

В HTML многострочные текстовые поля создаются с помощью тега **<textarea>...</textarea>**. Поле, создаваемое этим тегом, позволяет вводить и отправлять не одну строку, а сразу несколько строк.

*Размеры поля устанавливаются при помощи атрибутов **cols** – размеры по горизонтали, **rows** – размеры по вертикали. Высоту поля можно задать свойством **height**. Все размеры считаются исходя из размера одного символа моноширинного шрифта.*

```
<form action="handler.php">
  <p><b>Введите ваш отзыв:</b></p>
  <p><textarea name="comment"></textarea></p>
  <p><input type="submit"></p>
</form>
```



Между тегами `<textarea>` и `</textarea>` можно поместить любой текст, который будет отображаться внутри поля. Если текста нет, то поле будет изначально пустым.

Допустимые атрибуты:

Атрибут	Значение / описание
autofocus	Устанавливает фокус на нужном начальном текстовом поле автоматически.
Cols	Устанавливает ширину через количество символов. Если пользователь вводит больше текста, появляется полоса прокрутки.
disabled	Отключает возможность редактирования и копирования содержимого поля.
Form	Значение атрибута должно быть равно значению атрибута <code>id</code> элемента <code><form></code> в этом же документе. Определяет одну или несколько форм, которым принадлежит данное текстовое поле.
maxlength	Значение атрибута задает максимальное число символов для ввода в поле.
Name	Задаёт имя текстового поля.
placeholder	Определяет короткую текстовую подсказку, которая описывает ожидаемое вводимое значение.
readonly	Отключает возможность редактирования содержимого поля.
required	Выводит сообщение о том, что данное поле является обязательным для заполнения.
Rows	Указывает число, которое означает, сколько строк должно отображаться в текстовой области.
Wrap	Определяет, должен ли текст сохранять переносы строк при отправке формы. Значение <code>hard</code> сохраняет перенос, а значение <code>soft</code> не сохраняет. Если используется значение <code>hard</code> , то должно указываться значение атрибута <code>cols</code> .

Необязательные параметры **cols** и **rows** желательно указывать.

Атрибут **wrap** определяет тип переноса текста, как будет выглядеть текст в поле ввода:

- **virtual** - справа от текстового поля выводится полоса прокрутки. Вводимый текст выглядит разбитым на строки, а символ новой строки вставляется при нажатии клавиши ENTER;
- **physical** - этот тип зависит от типа браузера и выглядит по-разному;
- **none** - текст выглядит в поле в том виде, в котором пользователь его вводит. Если текст не укладывается в одну строку, появляется горизонтальная полоса прокрутки.

Дополнительно поле может находиться в двух состояниях — заблокированном и только для чтения. Спецификация HTML5 не определяет вид поля и текста в подобных состояниях, поэтому браузеры по-разному его отображают.

Текст внутри заблокированного поля нельзя выделить и добавить, также содержимое такого поля не отправляется формой на сервер. Текст внутри поля для чтения доступен для копирования, но его нельзя отредактировать.

```
<form action="handler.php">
  <p>Скопируйте приведённый текст и вставьте его в
  поле запроса пароля.</p>
  <p><textarea name="comment" readonly>Мухаха</textarea></p>
</form>
```

Учтите, что поле для чтения по своему виду не отличается от обычного текстового поля, но пользователь не сможет в него ничего добавить. Так что используйте его осмотрительно, чтобы не вводить людей в заблуждение.

Зависимые переключатели

Переключатели (жарг. радиокнопки) используют, когда необходимо выбрать один единственный вариант из нескольких предложенных. Создаются следующим образом.

```
<input type="radio" name="имя" атрибуты>

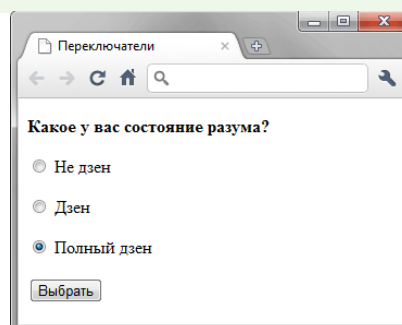
form action="http://localhost/script.php" method="GET">
<input type="radio" name="answer" value=yes checked>Да
<input type="radio" name="answer" value=no>Нет
<input type="submit" value=Отправить>
</form>
```

Атрибуты переключателей:

Атрибут	Описание
Checked	Предварительное выделение переключателя. По определению, набор переключателей может иметь только один выделенный пункт, поэтому добавление checked сразу к нескольким полям не даст особого результата. В любом случае, будет отмечен элемент, находящийся в коде HTML последним.
Name	Имя группы переключателей для идентификации поля. Поскольку переключатели являются групповыми элементами, то имя у всех элементов группы должно быть одинаковым.
Value	Задаёт, какое значение будет отправлено на сервер. Здесь уже каждый элемент должен иметь свое уникальное значение, чтобы можно было идентифицировать, какой пункт был выбран пользователем.

Пример создания переключателей:

```
<form action="handler.php">
  <p><b>Какое у вас состояние разума?</b></p>
  <p><input name="dzen" type="radio" value="nedzen"> Не дзен</p>
  <p><input name="dzen" type="radio" value="dzen"> Дзен</p>
  <p><input name="dzen" type="radio" value="pdzen" checked> Полный дзен</p>
  <p><input type="submit" value="Выбрать"></p>
</form>
```

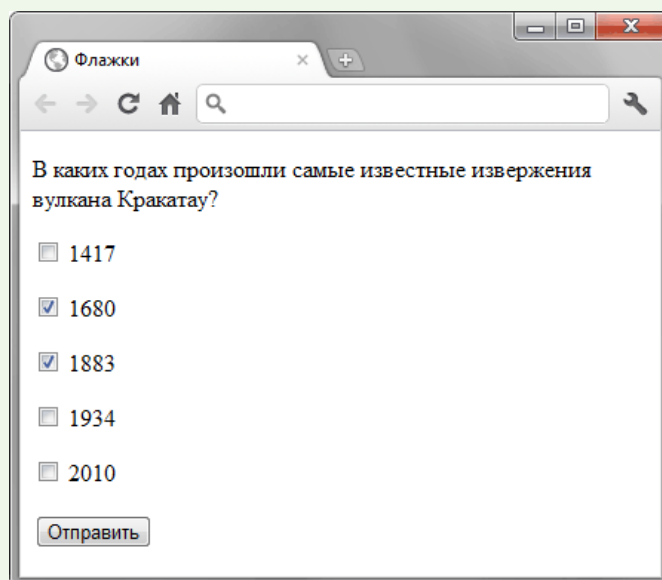


Заметьте, что в данном примере значение атрибута `name` для всех переключателей одинаково, именно в таком случае браузер понимает, что переключатели связаны между собой и помечает только один пункт из предложенных. Значение атрибута **`value`** же должно различаться, чтобы обработчик формы мог понять, какой вариант выбран пользователем.

Независимые переключатели (флажки)

Флажки (жарг. чекбоксы) используют, когда необходимо выбрать любое количество вариантов из предложенного списка. Если требуется выбор лишь одного варианта, то для этого следует предпочесть переключатели (`radiobutton`). Флажок создается следующим образом.

```
<form action="handler.php">
  <p>В каких годах произошли самые известные извержения вулкана Кракатау?</p>
  <p><input type="checkbox" name="a" value="1417"> 1417</p>
  <p><input type="checkbox" name="a" value="1680" checked> 1680</p>
  <p><input type="checkbox" name="a" value="1883" checked> 1883</p>
  <p><input type="checkbox" name="a" value="1934"> 1934</p>
  <p><input type="checkbox" name="a" value="2010"> 2010</p>
  <p><input type="submit" value="Отправить"></p>
</form>
```



Атрибуты флажков:

Атрибут	Описание
Checked	Предварительное выделение флажка.
Name	Имя флажка для его идентификации обработчиком формы.
Value	Задаёт, какое значение будет отправлено на сервер.

Если переключатель был включен на момент нажатия кнопки отправки данных, то скрипту будет передан параметр `имя=значение`. Если же флажок выключен, то сценарию вообще ничего не будет передано - как будто переключателя вообще нет.

Переключатель по умолчанию либо включен, либо выключен. Чтобы переключатель был по умолчанию включен, необходимо для него указать атрибут **`checked`**.

Переключатель **`checkbox`** называется независимым, так как его состояние не зависит от состояния других переключателей **`checkbox`**. Таким образом, в одной форме может быть одновременно выбрано несколько переключателей.

Элемент **<label>** применяется при реализации выбора с помощью переключателей и флажков. Можно выбрать нужный пункт, просто щелкая кнопкой мыши на тексте, связанном с ним. Для этого нужно поместить **<input>** внутрь элемента **<label>**.

```
<label>
  <input type="checkbox" name="newsletter" />
  Sign up for our newsletter
</label>
```

9. Поле со списком (раскрывающийся список)

Поле со списком, называемое еще ниспадающее меню, один из гибких и удобных элементов формы. В зависимости от настроек, в списке можно выбирать одно или несколько значений. Преимущество списка в его компактности и универсальности, список может занимать одну или несколько строк, в нём можно выбирать одно или несколько значений. Поле со списком создается следующим образом.

```
<select атрибуты>
  <option атрибуты>Пункт 1</option>
  <option атрибуты>Пункт 2</option>
</select>
```

Пример списка с единственным выбором:

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

Пример списка с множественным выбором:

```
<select name=day size=7 multiple>
  <option value=1>Понедельник</option>
  <option value=1>Вторник</option>
  <option value=1>Среда</option>
  <option value=1>Четверг</option>
  <option value=1>Пятница</option>
  <option value=1>Суббота</option>
  <option value=1>Воскресенье</option>
</select>
```

Тег **<select>** выступает контейнером для пунктов списка и определяет его вид, будет ли это раскрывающийся список или же список с одним или множественным выбором. Вид зависит от использования атрибута **size** тега **<select>**, который устанавливает высоту списка, ширина списка при этом определяется автоматически исходя из длины текста внутри **<option>**. Ниже представлен список множественного выбора, в котором пункты выделяются с помощью клавиши Ctrl и Shift и раскрывающийся список.

*Список множественного
выбора*

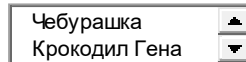
Раскрывающийся список

Атрибуты тега <select>, с помощью которых можно изменять представление списка:

multiple: наличие multiple сообщает браузеру отображать содержимое элемента <select> как список множественного выбора. В таких списках можно выбрать не одно, а сразу несколько вариантов значений. Конечный вид списка зависит от используемого атрибута size. Если он отсутствует, то высота списка равна количеству пунктов, если значение size меньше числа пунктов, то появляется вертикальная полоса прокрутки.



*Атрибут size
отсутствует*



Атрибут size равен 2

Для выбора нескольких значений списка применяются клавиши *Ctrl* и *Shift* совместно с курсором мыши.

name: определяет уникальное имя элемента <select>. Как правило, это имя используется для доступа к данным через скрипты или для получения выбранного значения серверным обработчиком;

size: устанавливает высоту списка. Если значение size равно единице, то список превращается в раскрывающийся. Значение по умолчанию зависит от атрибута multiple. Если он присутствует, то размер списка равен числу элементов. Когда multiple нет, то значение атрибута size равно 1.

Пример создания списка множественного выбора:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Список</title>
  </head>
  <body>
    <form>
      <p><select name="select" size="3" multiple>
        <option selected value="s1">Чебурашка</option>
        <option value="s2">Крокодил Гена</option>
        <option value="s3">Шапокляк</option>
        <option value="s4">Крыса Лариса</option>
      </select>
      <input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

Атрибуты тега <OPTION>, влияющие на вид списка:

selected: делает текущий пункт списка выделенным. Если у тега <select> добавлен атрибут multiple, то можно выделять более одного пункта;

value: определяет значение пункта списка, которое будет отправлено на сервер. На сервер отправляется пара «имя/значение», где имя задаётся атрибутом name тега <select>, а значение — атрибутом value выделенных пунктов списка. Значение может как совпадать с текстом пункта, так быть и самостоятельным;

label: предназначен для указания метки пункта списка, сокращённой по сравнению с текстом внутри `<option>`. Если атрибут `label` присутствует, то текст внутри тега `<option>` игнорируется и в списке выводится значение `label`. Браузер Firefox не поддерживает этот атрибут.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Список</title>
  </head>
  <body>
    <form>
      <p><strong>Выбери персонажа</strong></p>
      <p><select name="hero">
        <option value="s1">Чебурашка</option>
        <option value="s2" selected>Крокодил Гена</option>
        <option value="s3">Шапокляк</option>
        <option value="s3" label="Лариса">Крыса Лариса</option>
      </select>
      <input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

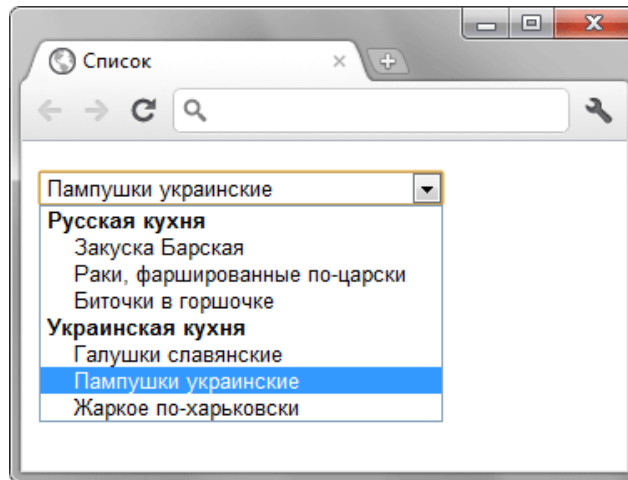
Группирование элементов списка

При достаточно обширном списке имеет смысл сгруппировать его элементы по блокам, чтобы обеспечить наглядность списка и удобство работы с ним. Для этой цели применяется тег `<optgroup>`. Он представляет собой контейнер, внутри которого располагаются теги `<option>` объединённые в одну группу. Особенностью тега `<optgroup>` является то, что он не выделяется как обычный элемент списка, акцентируется с помощью жирного начертания, а все элементы, входящие в этот контейнер, смещаются вправо от своего исходного положения. Для добавления заголовка группы используется атрибут `label`.

Пример группирования элементов списка:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Список</title>
  </head>
  <body>
    <form>
      <p><select name="food">
        <optgroup label="Русская кухня">
          <option value="r1">Закуска Барская</option>
          <option value="r2">Раки, фаршированные по-царски</option>
          <option value="r3">Биточки в горшочке</option>
        </optgroup>
        <optgroup label="Украинская кухня">
          <option value="u1">Галушки славянские</option>
          <option value="u2">Пампушки украинские</option>
          <option value="u3">Жаркое по-харьковски</option>
        </optgroup>
      </select></p>
```

```
<p><input type="submit" value="Отправить"></p>
</form>
</body>
</html>
```



6. Кнопки

Кнопки являются одним из самых понятных и интуитивных элементов интерфейса. По их виду сразу становится понятно, что единственное действие, которое с ними можно производить — это нажимать на них. За счёт этой особенности кнопки часто применяются в формах, особенно при их отправке и очистке.

Кнопку на веб-странице можно создать двумя способами — с помощью тега `<input>` и тега `<button>`.

Рассмотрим вначале добавление кнопки через `<input>` и его синтаксис.

```
<input type="button" атрибуты>
```

Атрибуты кнопки:

Атрибут	Описание
Name	Имя кнопки, предназначено для того, чтобы обработчик формы мог его идентифицировать.
Value	Значение кнопки и одновременно надпись на ней.

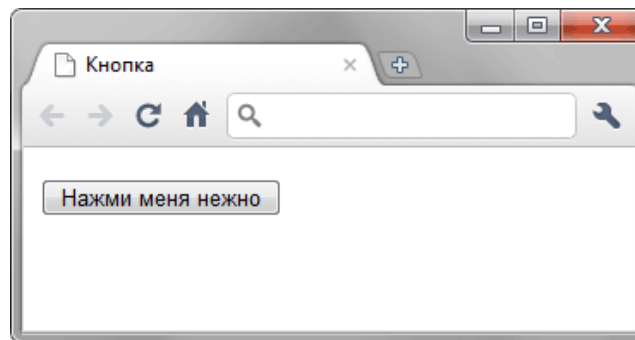
Пример создания кнопки:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

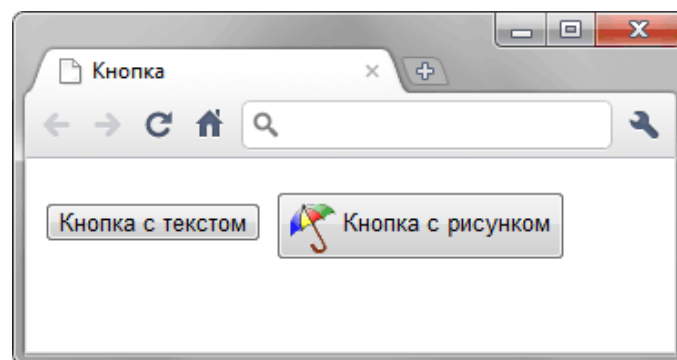


```
<title>Кнопка</title>
</head>
<body>
  <form>
    <p><input type="button" value=" Нажми меня нежно "></p>
  </form>
</body>
</html>
```

Пробелы в надписи на кнопке, в отличие от текста HTML, учитываются, поэтому можно ставить любое количество пробелов, которые в итоге влияют на ширину кнопки.



Второй способ создания кнопки основан на использовании тега <button>. Он по своему действию напоминает результат, получаемый с помощью тега <input>. Но в отличие от него предлагает расширенные возможности по созданию кнопок. Например, на подобной кнопке можно размещать любые элементы HTML включая изображения и таблицы.



Синтаксис создания такой кнопки следующий.

```
<button атрибуты>Надпись на кнопке</button>
```

Атрибуты такие же, как и при добавлении кнопки через <input>, но в отличие от кнопки <input> атрибут value определяет только отправляемое на сервер значение, а не надпись на кнопке. Если требуется вывести на кнопке изображение, то тег добавляется внутрь <button>.

Пример добавления рисунка на кнопку:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Кнопка</title>
```

```
</head>
<body>
  <form>
    <p><button>Кнопка с текстом</button>
    <button>
      
      Кнопка с рисунком
    </button></p>
  </form>
</body>
</html>
```

В данном примере показано создание обычной кнопки с текстом, а также кнопки с одновременным использованием текста и рисунка. Размер кнопки зависит от содержимого контейнера <button>, но пробелы игнорируются, поэтому простым увеличением их количества, как в случае использования <input>, ширину кнопки изменить не удастся.

6.1. Кнопка отправки формы Submit

Для отправки данных на сервер предназначена специальная кнопка Submit. Её вид ничем не отличается от обычных кнопок, но при нажатии на нее происходит выполнение серверной программы, указанной атрибутом action тега <form>. Эта программа, называемая еще обработчиком формы, получает данные, введенные пользователем в полях формы, производит с ними необходимые манипуляции, после чего возвращает результат в виде HTML-документа.

Что именно делает обработчик, зависит от автора сайта, например, подобная технология применяется при создании опросов, форумов, тестов и многих других вещей.

Синтаксис создания кнопки Submit зависит от используемого тега <input> или <button>.

```
<input type="submit" атрибуты>
<input type=submit [name=go] value=Отправить>

<button type="submit">Надпись на кнопке</button>
```

Атрибут value определяет текст, который будет написан на кнопке отправки. Атрибут name определяет имя кнопки и является необязательным. *Атрибуты те же, что и у рядовых кнопок.*

Пример отправки данных на сервер:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Кнопка</title>
  </head>
  <body>
    <form>
      <p><input name="login"></p>
      <p><input type="submit"></p>
    </form>
  </body>
</html>
```

Атрибут `name` для этого типа кнопки можно не писать. Если не указать значение `value`, то браузер автоматически добавит текст, он различается в зависимости от браузера. Так, Firefox пишет «Отправить запрос», IE — «Подача запроса», Opera и Chrome — «Отправить». Сам текст надписи никак на функционал кнопки не влияет.

6.2. Кнопка Reset

Кнопка сбрасывает параметры формы, а точнее, устанавливает для всех элементов формы значения по умолчанию. При нажатии на кнопку Reset данные формы возвращаются в первоначальное значение. Как правило, эту кнопку применяют для очистки введенной в полях формы информации.

Синтаксис создания прост и похож на другие кнопки:

```
<input type="reset" атрибуты>
<input type="reset" value="Сброс">

<button type="reset">Надпись на кнопке</button>
```

В примере показана форма с одним текстовым полем, которое уже содержит предварительно введенный текст с помощью атрибута `value` тега `<input>`. После изменения текста и нажатия на кнопку «Очистить», значение поля будет восстановлено и в нём снова появится надпись «Введите текст».

Пример кнопки для очистки формы:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Кнопка</title>
  </head>
  <body>
    <form>
      <p><input value="Введите текст"></p>
      <p><input type="submit" value="Отправить">
        <input type="reset" value="Очистить"></p>
    </form>
  </body>
</html>
```

Значение кнопки Reset никогда не пересылается на сервер. Если надпись на кнопке опустить, иными словами, не задавать атрибут `value`, на кнопке по умолчанию будет добавлен текст «Очистить».

6.3. Кнопка отправки с рисунком

Вместо кнопки `submit` можно использовать рисунок для отправки данных, чтобы не ограничивать дизайн и расширить возможности по оформлению форм. Клик на этом рисунке дает то же самое, что и нажатие на кнопку `submit`. При нажатии на рисунок данные формы отправляются на сервер и обрабатываются программой, заданной атрибутом `action` тега `<form>`. Однако, кроме этого, сценарию будут переданы координаты места клика на рисунке. Координаты будут переданы в формате `имя.x=коор_X, y=коор_Y`.

Если добавить к полю с изображением уникальное имя через атрибут **name**, например, **ok**, тогда координаты передаются в виде **ok.x** и **ok.y**, где впереди через точку стоит имя поля.

Синтаксис кнопки отправки с рисунком:

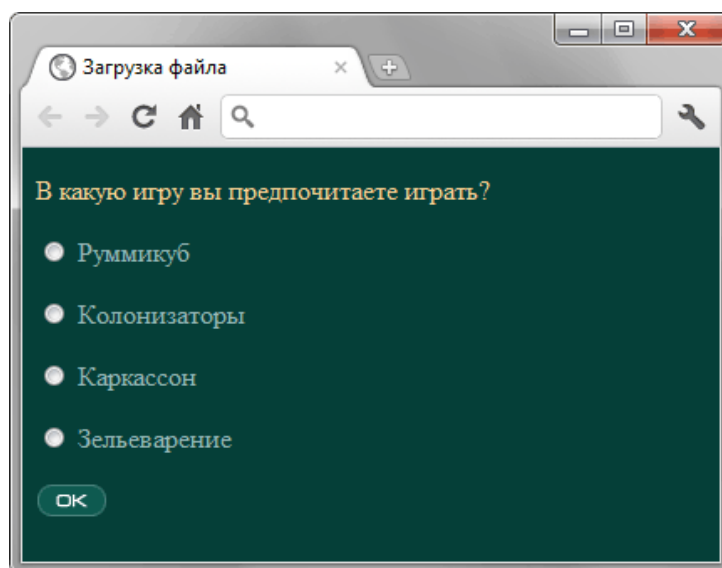
```
<input type="image" src="URL" alt="альтернативный текст">

<input type=image name=имя src=рисунок>
```

Здесь URL это адрес изображения в формате JPEG, PNG или GIF, alt указывает альтернативный текст, который виден при отключении картинок в браузере. Вообще, это поле похоже на добавление в код изображения и работает подобно элементу .

Пример кнопки с изображением:

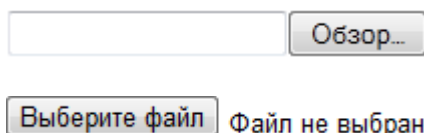
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Загрузка файла</title>
    <style>
      body { background: #053f38; color: #98baba; }
      p.question { color: #ffd595; }
    </style>
  </head>
  <body>
    <form action="handler.php">
      <p class="question">В какую игру вы предпочитаете играть?</p>
      <p><input type="radio" name="game" value="1"> Руммикуб</p>
      <p><input type="radio" name="game" value="2"> Колонизаторы</p>
      <p><input type="radio" name="game" value="3"> Каркассон</p>
      <p><input type="radio" name="game" value="4"> Зельеварение</p>
      <p><input type="image" src="images/okbutton.png" alt="OK">
    </form>
  </body>
</html>
```



7. Загрузка файлов на сервер

Тег *INPUT* позволяет реализовать еще одну возможность форм, а именно создавать поле выбора файла для его отправки на сервер.

Для того чтобы можно было загружать на сервер один или несколько файлов, в форме применяется специальное поле. В браузерах *Firefox*, *IE* и *Opera* такой элемент отображается как текстовое поле, рядом с которым располагается кнопка с надписью «Обзор...». В *Safari* и *Chrome* доступна только кнопка «Выберите файл».



При нажатии на кнопку открывается окно для выбора файла, где можно указать, какой файл необходимо использовать. Синтаксис поля для отправки файла следующий:

```
<input type="file" атрибуты>
```

```
<input type=file name=имя [value=имя_файла]>
```

Атрибуты:

Атрибут	Описание
Accept	Устанавливает фильтр на типы файлов, которые вы можете отправить через поле загрузки файлов.
Size	Ширина текстового поля, которое определяется числом символов моноширинного шрифта.
Multiple	Позволяет выбирать и загружать сразу несколько файлов.
Name	Имя поля, используется для его идентификации обработчиком формы.

Прежде, чем использовать данное поле, в форме необходимо сделать следующее:

- 1) задать метод отправки данных POST (**method="post"**);
- 2) установить у атрибута **enctype** значение **multipart/form-data**.

Пример создания поля для отправки файла:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Отправка файла на сервер</title>
  </head>
  <body>
    <form enctype="multipart/form-data" method="post">
      <p><input type="file" name="f">
        <input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

Хотя можно установить ширину поля через атрибут *size*, в действительности ширина никак не влияет на результат работы формы. В браузерах *Safari* и *Chrome* этот атрибут вообще никакого воздействия не оказывает.

Атрибут **multiple** более важен, он позволяет не ограничиваться одним файлом для выбора, а указать их сразу несколько для одновременной загрузки.

Если атрибут **accept** не указывать, тогда добавляются и загружаются файлы любого типа. Наличие ассерт позволяет ограничить выбор файла, что особенно важно, когда требуется загрузить только изображение или видео. В качестве значения выступает **MIME-тип**, несколько значений разделяются между собой запятой. Также можно использовать следующие ключевые слова:

audio/* — выбор музыкальных файлов любого типа;

image/* — графические файлы;

video/* — видеофайлы.

В таблице приводятся MIME-типы наиболее употребимых типов файлов:

Тип файлов	MIME-тип
HTML	text/html
XHTML	application/xhtml+xml
Изображения	image/gif
	image/png
	image/jpeg
	image/svg+xml
PDF	application/pdf
Flash	application/x-shockware-flash
MP4 видео	video/mp4
QuickTime	video/quicktime
Windows Media	video/x-ms-wmv
	audio/x-ms-wma
OGG	video/ogg
	audio/ogg
	application/ogg
MP3 аудио	audio/mpeg
WAV	audio/wav
	audio/x-wav
	audio/x-pn-wav

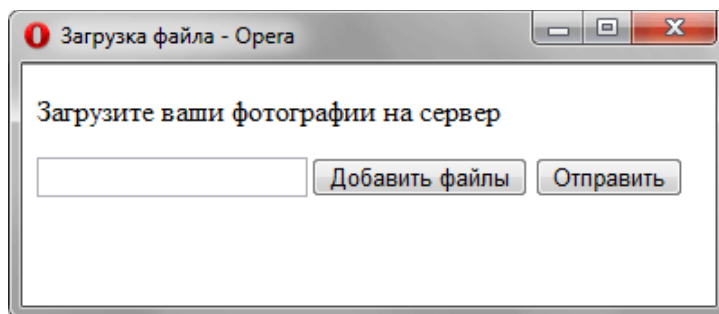
В таблице показаны некоторые допустимые значения атрибута ассерт:

Значение	Описание
image/jpeg	Только файлы в формате JPEG.
image/jpeg,image/png	Только файлы в формате JPEG и PNG.
image/*	Любые графические файлы.
image/*,video/*	Любые графические и видеофайлы.

Пример использования дополнительных атрибутов (загрузка фотографий):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Загрузка файла</title>
  </head>
  <body>
    <form enctype="multipart/form-data" method="post">
```

```
<p>Загрузите ваши фотографии на сервер</p>
<p><input type="file" name="photo" multiple accept="image/*,image/jpeg">
<input type="submit" value="Отправить"></p>
</form>
</body>
</html>
```



Не все браузеры поддерживают новые атрибуты. IE полностью игнорирует *multiple* и *accept*, Safari не поддерживает *accept*, а Firefox не работает с MIME-типом, только с ключевыми словами. Поэтому в примере выше специально для Firefox установлено значение *image/*,image/jpeg*. Также учтите странную ошибку в Опере, она не допускает пробелы после запятой внутри *accept*.

10. Адрес электронной почты

В формах часто требуется ввод адреса электронной почты, для чего обычно применяется однострочное текстовое поле. Однако в отличие от него специальное поле для ввода адреса почты позволяет проверять корректность записи введённого адреса. Синтаксис создания поля следующий.

```
<input type="email" атрибуты>
```

Атрибуты поля для почтового адреса по большей части совпадают с текстовым полем:

Атрибут	Описание
Maxlength	Устанавливает максимальное число символов, которое может быть введено пользователем в поле. Когда это количество достигается при наборе, дальнейший ввод становится невозможным. Если этот атрибут опустить, то можно вводить строку длинее самого поля.
Multiple	Позволяет указывать несколько адресов через запятую.
Name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
Size	Ширина поля, которая определяется числом символов моноширинного шрифта. Иными словами, ширина задается количеством близстоящих букв одинаковой ширины по горизонтали.
Value	Начальный почтовый адрес отображаемый в поле.

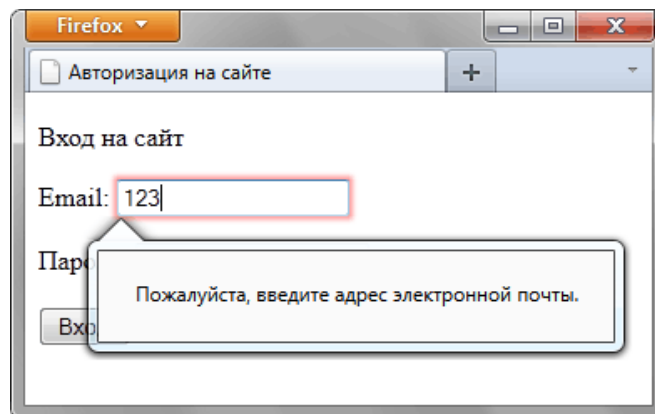
По сравнению со значением **text**, добавлен атрибут **multiple**, который позволяет вводить сразу несколько почтовых адресов.

Пример создания поля для адреса электронной почты:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Авторизация на сайте</title>
  </head>
  <body>
    <form action="handler.php">
      <p>Вход на сайт</p>
      <p>Email: <input type="email" name="login"></p>
      <p>Пароль: <input type="password" name="pass"></p>
      <p><input type="submit" value="Вход"></p>
    </form>
  </body>
</html>
```

В примере показано применение значения `email` для создания формы авторизации на сайте, где в качестве логина указывается адрес электронной почты.

По своему виду поле для ввода адреса ничем не отличается от текстового поля. Различия проявляются, если указать некорректный адрес, в этом случае браузер выведет замечание об ошибке. Сама форма на сервер не отправляется, пока ошибка не будет исправлена.



11. Веб-адрес

Для ввода адресов сайтов или, как их ещё называют веб-адресов, предназначено значение `url` тега `<input>`, которое делает проверку на правильность ввода данных. Каждый веб-адрес должен начинаться с протокола (`http://`, `https://`, `ftp://`), больше ограничений нет — адрес может быть набран латинскими символами, кириллицей, содержать точку или наоборот, писаться без неё. Браузер Орега не требует даже наличие протокола, подставляя «`http://`» перед текстом автоматически в случае его отсутствия. Синтаксис написания поля для веб-адреса следующий.

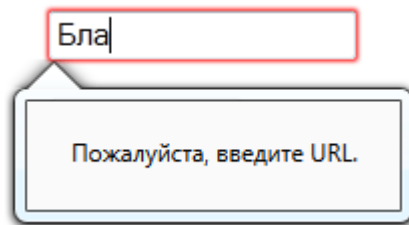
```
<input type="url" атрибуты>
```

Атрибуты совпадают с текстовым полем (`<input type="text">`).

Пример использования поля для веб-адреса:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Веб-адрес</title>
  </head>
  <body>
    <form>
      <p><input placeholder="Ваше имя" name="user"></p>
      <p><input type="url" placeholder="Сайт" name="site" required></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

В данном примере при нажатии на кнопку «Отправить» браузер будет выводить сообщение, что необходимо правильно ввести URL. Вид и текст сообщения зависит от браузера.



12. Выбор цвета

Для выбора шестнадцатеричного значения цвета в формы HTML5 включено специальное поле, которое позволяет указать желаемый цвет. Синтаксис создания такого поля следующий:

```
<input type="color" value="цвет" name="имя">
```

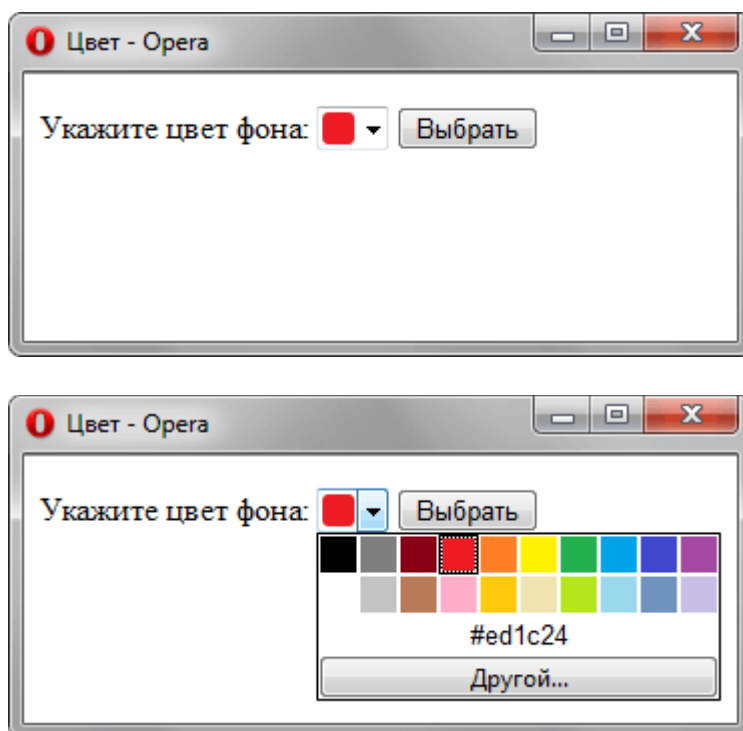
Атрибут **value** необходим для установки исходного цвета и может быть опущен, **name** используется для идентификации получаемого значения.

Вид поля для выбора цвета возложен на браузеры и может различаться по своему оформлению. В действительности же пока только Opera и Chrome корректно работает с выбором цвета, остальные браузеры покажут стандартное текстовое поле.

Пример создания поля для указания желаемого цвета:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Цвет</title>
  </head>
  <body>
    <form action="handler.php">
      <p>Укажите цвет фона: <input type="color" name="bg" value="#ff0000">
      <input type="submit" value="Выбрать"></p>
    </form>
  </body>
```

```
</html>
```



12. Ввод чисел

Для ввода чисел предназначено специальное поле, которое допускает ограничения по нижней и верхней границе, а также устанавливает шаг приращения. Само поле для ввода чисел похоже на обычное текстовое поле, но со стрелками, которые позволяют увеличивать и уменьшать значение.



Синтаксис создания поля следующий:

```
<input type="number" атрибуты>
```

Допустимые атрибуты:

Атрибут	Описание
Min	Минимальное значение.
Max	Максимальное значение.
Size	Ширина поля.
Step	Шаг приращения числа. Может быть как целым (2), так и дробным (0.2).
Name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
Value	Начальное число, которое выводится в поле.

Для ограничения введённого числа предназначены атрибуты min и max, они могут принимать отрицательное и положительное значение. При достижении верхнего или

нижнего порога стрелки в поле в зависимости от браузера блокируются или не дают никакого эффекта. Несмотря на такие запреты, в любом случае в поле самостоятельно можно вводить любые значения, включая текст. Атрибуты `min` и `max` работают только при использовании стрелок в поле.

Пример ограничения ввода чисел:

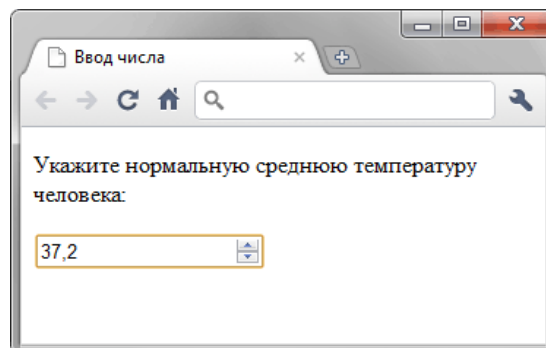
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ввод числа</title>
  </head>
  <body>
    <form action="handler.php">
      <p>Введите число от 1 до 10:</p>
      <p><input type="number" size="3" name="num" min="1" max="10" value="1"></p>
    </form>
  </body>
</html>
```

Если значение **min** превышает **max**, то атрибут **min** игнорируется.

Атрибут **step** задаёт шаг приращения и по умолчанию равен 1. В то же время значение может быть и дробным числом.

Пример: шаг приращения

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ввод числа</title>
  </head>
  <body>
    <form action="handler.php">
      <p>Укажите нормальную среднюю температуру человека:</p>
      <p><input type="number" name="t" value="35" min="35" max="40" step="0.2"></p>
    </form>
  </body>
</html>
```



Браузеры плохо поддерживают это поле, пока лишь это делает Chrome и Opera. В остальных браузерах поле для ввода числа приобретает вид обычного текстового поля.

13. Ползунок

Ползунок предназначен для ввода чисел в указанном диапазоне, но в отличие от поля `<input type="number">` имеет другой интерфейс и применяется в тех случаях, когда не особенно важно указывать точное значение. На рисунке продемонстрирован вид ползунка в разных браузерах.



Синтаксис создания ползунка следующий:

```
<input type="range" min="0" max="100" step="1" value="50">
```

Здесь **min** — минимальное число в диапазоне (по умолчанию 0), **max** — максимальное число (по умолчанию 100), **step** — шаг изменения чисел (по умолчанию 1), **value** — текущее значение. По умолчанию value вычисляется по формуле:

$$value = \frac{max+min}{2}$$

Если значение **max** меньше, чем значение **min**, то **value** равно **min**.

Атрибуты не являются обязательными, их можно опустить, в таком случае они принимают значения по умолчанию.

Независимо от минимального и максимального числа ширина ползунка остаётся одинаковой.

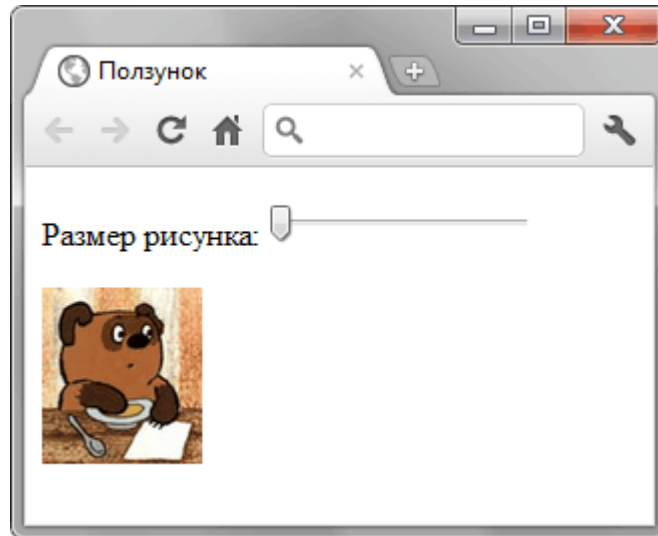
Сами ползунки редко применяются в «чистом» виде, поскольку не обеспечивают необходимую обратную связь с пользователем, а вот в сочетании с JavaScript это становится мощным и удобным элементом интерфейса.

Пример использования ползунка:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Ползунок</title>
  <script>
    function sizePic() {
      size = document.getElementById("size").value;
      img = document.getElementById("pic");
      img.width = 60 + 20*size;
    }
  </script>
</head>
<body>
  <p>Размер рисунка: <input type="range" min="1" max="5" id="size"
    oninput="sizePic()" value="3"></p>
  <p></p>
</body>
</html>
```

В примере с помощью ползунка изменяется размер изображения. При управлении ползунком срабатывает событие `oninput`, которое вызывает функцию `sizePic`. Эта функция

изменяет размер изображения в зависимости от установленного пользователем значения ползунка. Тем самым ширина картинке при желании уменьшается или наоборот, увеличивается.



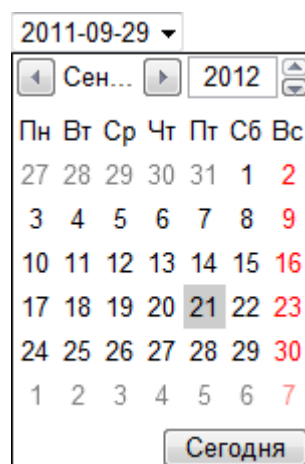
Старые версии браузеров, которые не поддерживают значение `range` для атрибута `type`, отображают поле формы как текстовое.

14. Календарь

Выбор даты применяется на сайтах, торгующих авиа и железнодорожными билетами, ведь посетителя интересует заказ билета на определённый день. Календари также применяются в блогах, где записи систематизируются по дате, и сайтах, связанных с разными событиями, например, спортивными. Так или иначе, календарь востребован и может быть добавлен следующим образом.

```
<input type="date" атрибуты>
```

На сервер данные передаются в формате ГГГГ-ММ-ДД, ?**например, 22.12.2014?**, а вид календаря может различаться в зависимости от браузера. Полностью поддерживает календарь пока только Орега, выводя виджет для выбора любой даты.



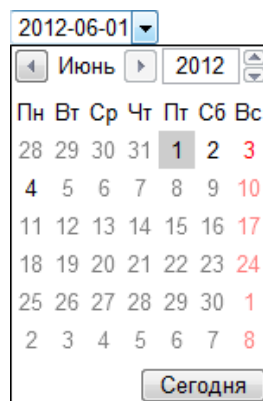
Пример создания календаря для выбора произвольной даты:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Календарь</title>
  </head>
  <body>
    <form>
      <p>Выберите дату: <input type="date" name="calendar">
      <input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

Допустимо ограничить ввод даты заданным значением через атрибуты *min* и *max*, они соответственно указывают нижнюю и верхнюю дату. Так, если требуется сузить диапазон ввода до ± 3 дней от даты 01.06.2012, то код запишется следующим образом:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Календарь</title>
  </head>
  <body>
    <form>
      <p>Выберите дату:
      <input type="date" name="calendar" value="2012-06-01"
      max="2012-06-04" min="2012-05-29">
      <input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

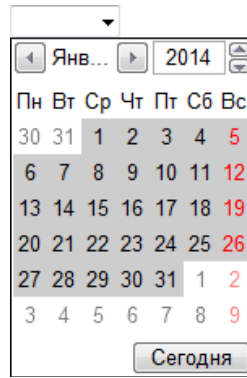
Текущая дата, заданная через атрибут *value* подсвечивается фоном, неактивные дни, которые нельзя выбрать — серым цветом.



Кроме традиционного календаря, в котором можно выбрать дату, месяц и год, существует и календарь только для ввода месяца и недели. Они записываются в таком виде.

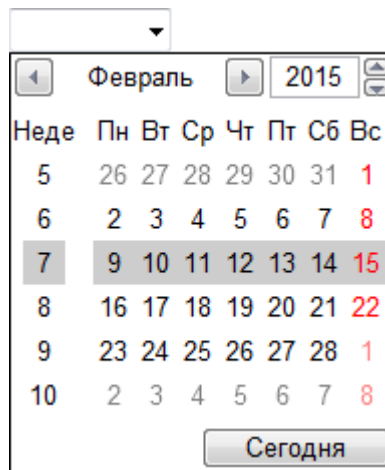
```
<input type="month">
<input type="week">
```

Выбор месяца в Opera происходит через аналогичный виджет, но в этом случае нельзя указать конкретную дату.



На сервер данные поля `type="month"` пересылаются как ГГГГ-ММ, например, 2014-10.

Похожим образом выглядит и виджет для выбора недели, но дополнительно выводится номер недели и выбрать можно только её. На сервер при этом значение отправляется как 2014-W38, где вначале указывается год, затем через дефис *W* и после него номер недели от начала года.



Пример создания поля для ввода месяца:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Календарь</title>
</head>
<body>
  <form>
    <p>Укажите месяц:
    <input type="month" name="month">
    <input type="submit" value="Отправить"></p>
  </form>
</body>
</html>
```

15. Дата и время

Наряду с календарём, предназначенном для указания даты, месяца или недели, иногда возникает необходимость вводить ещё и время, например, для точной публикации сообщения. Для подобных ситуаций используется поле **`datetime`** и **`datetime-local`**, синтаксис у них следующий.

```
<input type="datetime" атрибуты>
```

```
<input type="datetime-local" атрибуты>
```

Данные на сервер по умолчанию пересылаются в виде ГГГГ-ММ-ДДТч:ммZ (например: 2015-09-25T12:15Z), где вначале указывается год, месяц и день, затем после латинской буквы T идут часы с минутами. В теории вместо Z указывается часовой пояс (например: +08:00 или -04:00), а также по желанию секунды с дробной частью, но на практике всё ограничивается только минутами и без часового пояса.

Пример создания поля для ввода даты и времени:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Дата и время</title>
</head>
<body>
  <form>
    <p>Время создания публикации</p>
    <p><input type="datetime" name="created"></p>
    <p><input type="submit"></p>
  </form>
</body>
</html>
```

Разница между значениями `datetime` и `datetime-local` заключается в добавлении часового пояса для `datetime`. На деле же пересылаются те же самые данные, только без Z в конце (2015-09-25T12:15).

Для ввода только времени без даты применяется `<input type="time">`.

Пример создания поля для ввода времени в указанном диапазоне:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
```

```
<title>Время</title>
</head>
<body>
  <form>
    <p>В какое время запускать Cron?</p>
    <p><input type="time" name="cron" value="03:15" min="00:01" max="06:00"></p>
    <p><input type="submit"></p>
  </form>
</body>
</html>
```

В данном примере значение изначально задано с помощью атрибута value, а минимальное и максимальное время установлено через min и max.

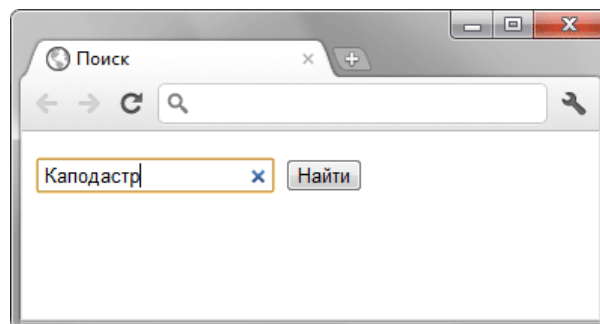
16. Поле для поиска

На сайтах часто востребован поиск по ключевым словам, для ввода которых используется текстовое поле. В HTML5 для поиска добавлено новое поле, синтаксис его следующий:

```
<input type="search" атрибуты>
```

Используемые атрибуты совпадают с текстовым полем.

Разница между текстовым полем и полем для поиска состоит в стилистическом оформлении.



Пример создания формы с полем для поиска:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Поиск</title>
  </head>
  <body>
    <form>
      <p><input type="search" name="q" placeholder="Поиск по сайту">
        <input type="submit" value="Найти"></p>
    </form>
  </body>
</html>
```

17. Поле для телефона

Чтобы указать телефонный номер используйте для этого специальное поле. По своему виду и работе оно совпадает с текстовым полем. Синтаксис создания этого поля следующий:

```
<input type="tel" атрибуты>
```

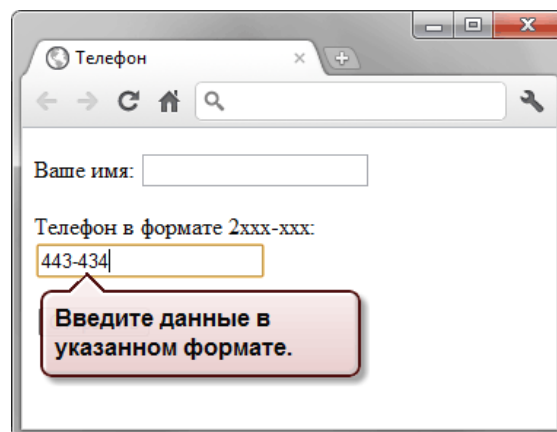
Атрибуты полностью совпадают с текстовым полем.

В отличие от веб-адреса или адреса электронной почты, поле для телефона не проверяется на правильность синтаксиса. Это связано с тем, что телефонные номера в разных местах мира могут иметь самые разнообразные формы написания, включать цифры, буквы и другие символы. Смысл применения этого поля в настройке шаблона ввода телефона, который используется в определённом городе или стране, а также в некоторых расширенных возможностях по управлению стилями.

Пример создания поля для телефона:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Телефон</title>
  </head>
  <body>
    <form>
      <p>Ваше имя: <input name="login"></p>
      <p>Телефон в формате 2xxx-xxx: <input type="tel" name="tel"
        pattern="2[0-9]{3}-[0-9]{3}"></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

Требуется ввести телефон в указанном формате, иначе браузер выведет сообщение об ошибке и не станет отправлять форму, пока поле не будет заполнено корректно.



18. Переход между полями с помощью табуляции

При достаточно большом количестве полей формы, которые необходимо заполнить, переходить между ними с помощью курсора мыши становится неудобно. При этом требуется навести курсор на соответствующее поле, нажать кнопку мыши, и только

после этого вводить нужное значение. Как альтернатива, используется клавиша **Tab**, которая позволяет быстро переключать фокус с одного поля на другое. Атрибут **tabindex** определяет последовательность перехода между полями при нажатии на Tab.

Фокусом называется активность поля, иными словами, поле доступно для того, чтобы в него вводили информацию или использовали какое другое действие. Например, кнопки, флажки, переключатели можно активизировать с помощью пробела.

Следующие элементы формы могут иметь атрибут **tabindex**: **<button>**, **<input>**, **<select>**, **<textarea>**. В качестве значения принимается число, которое задаёт шаг перехода. Так, номер 1 означает, что это поле первое получит фокус, номер 2 будет идти следующим и т.д.

Пример использования tabindex:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Порядок полей в форме</title>
  <style>
    fieldset {
      width: 200px; /* Ширина */
      padding: 0 10px; /* Поля */
    }
    fieldset:nth-child(odd) {
      float: left; /* Обтекание для всех нечётных элементов */
    }
  </style>
</head>
<body>
  <form>
    <fieldset>
      <p>Имя:</p>
      <p><input name="name" tabindex="1"></p>
    </fieldset>
    <fieldset>
      <p>Фамилия:</p>
      <p><input name="lastname" tabindex="3"></p>
    </fieldset>
    <fieldset>
      <p>Телефон:</p>
      <p><input name="tel" type="tel" tabindex="2"></p>
    </fieldset>
    <fieldset>
      <p>Пол:</p>
      <p><select name="gender" tabindex="4">
        <option selected>Мужской</option>
        <option>Женский</option>
      </select></p>
    </fieldset>
  </form>
</body>
</html>
```

В примере показано применение **tabindex** когда поля формы размещаются в ячейках таблицы. Если значение **tabindex** не указано, то по умолчанию переход по элементам формы происходит так, как они расположены в коде HTML, т.е. сверху вниз.

Обратите внимание, что с помощью табуляции фокус вначале получает поле, где надо ввести имя, а затем поле для ввода телефона.

Порядок полей в форме

Имя: Ваня

Фамилия:

Телефон:

Пол: Мужской

19. Блокирование элементов форм

У любого элемента формы есть два состояния, которые ограничивают доступ к элементу или ввод данных, — **блокирование (disabled)** и **только для чтения (readonly)**.

19.1. Блокирование

Блокирование элемента не позволяет вообще производить с ним каких-либо действий, в том числе выделять содержимое текстового поля, изменять его или активизировать. К тому же такие поля не отправляются на сервер.

Некоторые браузеры позволяют выделять и копировать содержимое заблокированного текстового поля, но все остальные действия недоступны.

На рисунке представлены разные элементы форм в заблокированном состоянии.

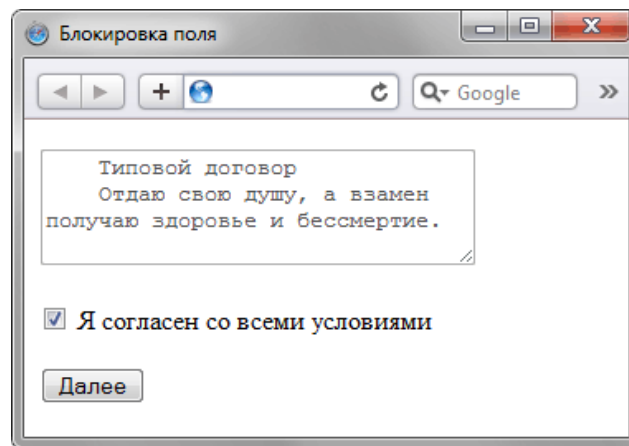
<p>Текстовое поле</p> <p>Ваше имя</p>	<p>Многострочное текстовое поле</p> <p>Не печалься, друг мой нежный, будет нам еще с тобой.</p>
<p>Кнопка</p> <p>Submit</p>	<p>Флажки</p> <p><input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/></p>
<p>Список</p> <p>Чай</p>	<p>Переключатели</p> <p><input type="radio"/> <input type="radio"/> <input type="radio"/></p>

Для блокирования элемента формы используется атрибут `disabled`. Добавление этого атрибута разрешает отображать элемент формы, но не позволяет изменять его.

Блокирование элементов форм обычно используется для того, чтобы динамически с помощью скриптов изменять значение поля. Пользователь не должен в подобном случае иметь доступ к полю, поэтому оно блокируется.

Пример применения скриптов для изменения блокировки кнопки:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Блокировка поля</title>
  <script>
    function agreeForm(f) {
      // Если поставлен флажок, снимаем блокирование кнопки
      if (f.agree.checked) f.submit.disabled = 0
      // В противном случае вновь блокируем кнопку
      else f.submit.disabled = 1
    }
  </script>
</head>
<body>
  <form>
    <p><textarea cols="30" rows="4" disabled>
      Типовой договор
      Отдаю свою душу, а взамен получаю здоровье и бессмертие.
    </textarea></p>
    <p><input type="checkbox" name="agree" onclick="agreeForm(this.form)">
      Я согласен со всеми условиями</p>
    <p><input type="submit" name="submit" value="Далее" disabled></p>
  </form>
</body>
</html>
```



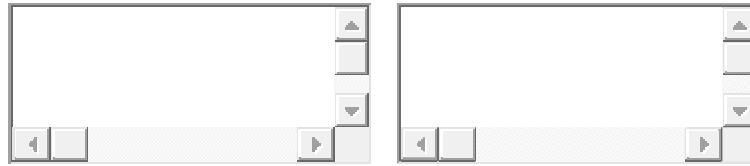
В данном примере применяется блокирование кнопки, но оно снимается, как только пользователь поставит флажок возле текста «Я согласен со всеми условиями».

19.2. Только для чтения

Поля формы можно не только блокировать, но и переводить их в режим только для чтения. В этом случае доступ к ним сохраняется, но изменять значения заданные по умолчанию нельзя. Разумеется, речь идёт только о полях, где требуется вводить текст. Выделять и копировать текст можно, но изменить не получится.

Для установки режима «только для чтения» используется атрибут **readonly**, он добавляется к тегу **<input>** или **<textarea>**. На вид элемента формы это никак не влияет, но как было уже замечено, модифицировать значение поля не удастся.

Ниже представлены два поля с многострочным текстом, одно из которых находится в обычном режиме, а второе — «только для чтения».



Пример создания поля для чтения:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Поле для чтения</title>
  </head>
  <body>
    <form>
      <p><textarea>Обычное текстовое поле</textarea>
        <textarea readonly>Поле только для чтения</textarea></p>
    </form>
  </body>
</html>
```

20. Автофокус

Фокус это активность элемента формы, позволяющая производить с ним какие-то действия. Для текстового поля можно вводить текст, для списка выбирать пункт с помощью клавиатуры и др. Автофокус — это автоматически установленный фокус поля формы. К примеру, при открытии google можно сразу набирать текст в строке поиска без лишних манипуляций с мышью и клавиатурой.

Автофокус создаётся с помощью атрибута **autofocus**, который можно добавлять к следующим тегам: **<button>**, **<input>**, **<keygen>**, **<select>**, **<textarea>**. Для текстового поля синтаксис следующий:

```
<input autofocus>
```

На странице должен быть только один элемент с автофокусом.

Пример создания формы авторизации с автофокусом:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Автофокус</title>
  </head>
  <body>
    <form>
      <fieldset>
        <legend>Вход на сайт</legend>
        <p><input name="login" autofocus></p>
      </fieldset>
    </form>
  </body>
</html>
```



```

<p><input type="password" name="pass"></p>
<p><input type="submit" value="Вход"></p>
</fieldset>
</form>
</body>
</html>

```



20. Подсказывающий текст

В дизайне часто требуется вставить пояснение к текстовому полю, но не всегда для этого имеется место. Решением в таком случае является добавление подсказывающего текста непосредственно внутрь поля, при получении фокуса исходный текст должен пропадать. Это делается с помощью атрибута **placeholder**, значением которого служит любой текст. Подсказка делается для полей `<input type="text">`, `<input type="password">`, `<input type="search">`, `<input type="email">`, `<input type="tel">`, `<input type="url">` и `<textarea>`, иными словами, везде, где вводится текст.

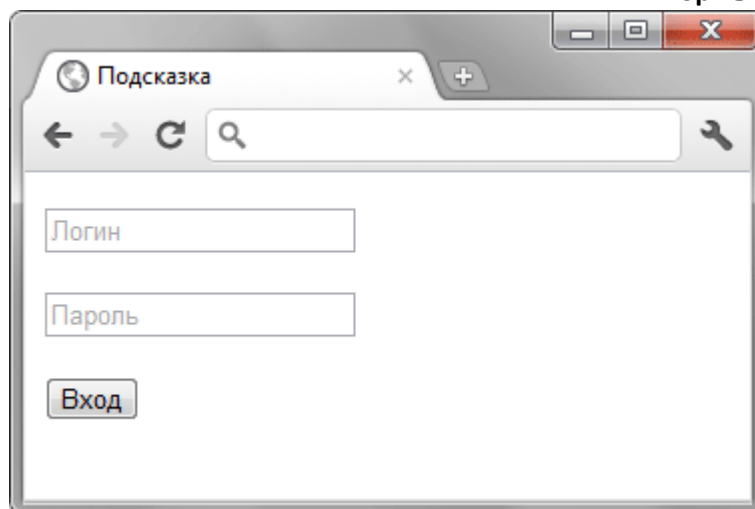
Подсказывающий текст отображается серым цветом, исчезает при получении фокуса и снова появляется, если фокус теряется. Если в поле с подсказкой ввести любой текст, то подсказка больше не появится. Также её не будет при наличии атрибута `value` с непустым значением.

Пример добавления подсказки к полям формы для создания авторизации:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Подсказка</title>
  </head>
  <body>
    <form>
      <p><input name="login" placeholder="Логин"></p>
      <p><input name="pass" type="password" placeholder="Пароль"></p>
      <p><input type="submit" value="Вход"></p>
    </form>
  </body>
</html>

```



21. Шаблон ввода данных

Текстовые поля никак не ограничивают ввод данных, хотя часто возникает необходимость задать для них параметры ввода, например, вводить только буквы, цифры или текст в определённом формате. Такие ограничения позволяют снизить ошибки пользователя и добиться от него ввода данных в нужном виде. Сам шаблон устанавливается для тега `<input>` с помощью атрибута **pattern** значением которого выступают регулярные выражения.

Выражение	Описание
<code>\d [0-9]</code>	Одна цифра от 0 до 9.
<code>\D [^0-9]</code>	Любой символ кроме цифры.
<code>\s</code>	Пробел.
<code>[A-Z]</code>	Только заглавная латинская буква.
<code>[A-Za-z]</code>	Только латинская буква в любом регистре.
<code>[А-Яа-яЁё]</code>	Только русская буква в любом регистре.
<code>[A-Za-zА-Яа-яЁё]</code>	Любая буква русского и латинского алфавита.
<code>[0-9]{3}</code>	Три цифры.
<code>[A-Za-z]{6,}</code>	Не менее шести латинских букв.
<code>[0-9]{,3}</code>	Не более трёх цифр.
<code>[0-9]{5,10}</code>	От пяти до десяти цифр.
<code>^[a-zA-Z]+\$</code>	Любое слово на латинице.
<code>^[А-Яа-яЁё\s]+\$</code>	Любое слово на русском включая пробелы.
<code>^[0-9]+\$</code>	Любое число.

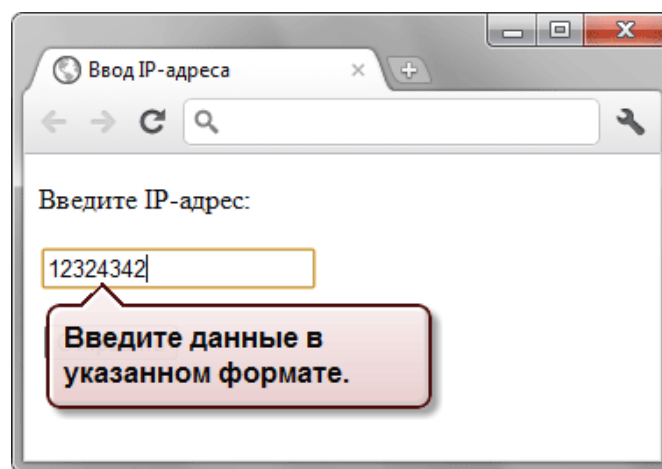
Пример создания шаблона ввода IP-адреса:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ввод IP-адреса</title>
  </head>
  <body>
    <form>
      <p>Введите IP-адрес:</p>
      <p><input name="ip" pattern="\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>

```

При вводе значения, не соответствующего шаблону будет выводиться сообщение об ошибке.



22. Защита от дурака

«Защитой от дурака» называется комплекс мер по пресечению ввода неправильной информации в форме.

Например, если в поле требуется ввести положительное число от 0 до 10, то следует проверить, чтобы пользователь не ввёл текст или число, которое не лежит в указанном диапазоне, т.е. число не должно быть меньше нуля и больше десяти.

Почему происходит ввод неправильной информации? Это в основном совершается по трём причинам:

- 1) пользователь ошибся случайно, например, невнимательно прочитал, что ему требуется указать;
- 2) на веб-странице неоднозначно просят ввести данные, поэтому пользователю приходится гадать и делать предположение, что же в действительности от него хотят. При этом не всегда происходит совпадение мнений разработчика и пользователя;
- 3) есть ряд людей, которые воспринимают инструкции как вызов и стараются поступить наоборот. Такие пользователи рассуждают примерно так: «Ага, меня просят ввести число. А что будет, если я укажу буквы?». После чего задают явно неправильную информацию и смотрят, к чему это приведёт.

Следует понимать, что точные и правильные формулировки хотя и снижают вероятность возникновения ошибок, но никак не спасают от них. Только технические средства на стороне сервера позволяют получить требуемый результат и избежать ввода неправильной информации. Тем не менее, ревизия или, как её ещё называют, валидация на стороне клиента позволяет быстро проверить данные, вводимые пользователем, на корректность, без отправки формы на сервер. Таким образом, экономится время пользователя и снижается нагрузка на сервер. С позиции юзабилити тоже имеются плюсы — пользователь сразу получает сообщение о том, какую информацию он указал неверно и может исправить свою ошибку.

22.1. Обязательное поле

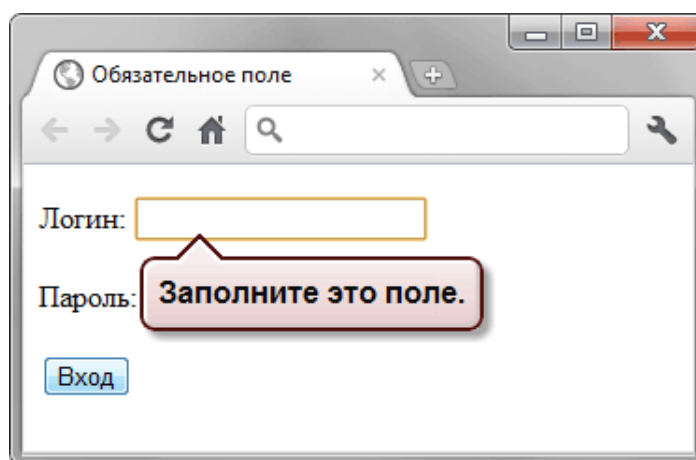
Некоторые поля формы должны быть обязательно заполнены перед их отправкой на сервер. Это, к примеру, относится к форме регистрации, где требуется ввести логин и пароль.

Для указания обязательных полей используется атрибут **required**.

Пример указания обязательных полей:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Обязательное поле</title>
  </head>
  <body>
    <form>
      <p>Логин: <input name="login" required></p>
      <p>Пароль: <input type="password" name="login" required></p>
      <p><input type="submit" value="Вход"></p>
    </form>
  </body>
</html>
```

Обязательные поля должны быть заполнены перед отправкой формы, иначе форма на сервер не отправится и браузер выдаст об этом предупреждение. Вид сообщения зависит от браузера.



22.2. Корректность данных

Исходно имеется два поля, в котором вводимые пользователем данные проверяются автоматически. Это веб-адрес и адрес электронной почты.

Веб-адрес (`<input type="url">`) должен содержать протокол (`http://`, `https://`, `ftp://`).

Адрес электронной почты (`<input type="email">`) должен содержать буквы или цифры до символа `@`, после него, затем точку и домен первого уровня.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Корректность данных</title>
  </head>
  <body>
    <form>
      <p>Заполните форму (все поля обязательны)</p>
      <p>Имя: <input name="name" required></p>
      <p>Email: <input type="email" name="email" required></p>
      <p>Сайт: <input type="url" name="site" required></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

В примере показана форма с обязательными полями, в которой два поля проверяется браузером.

22.3. Шаблон ввода

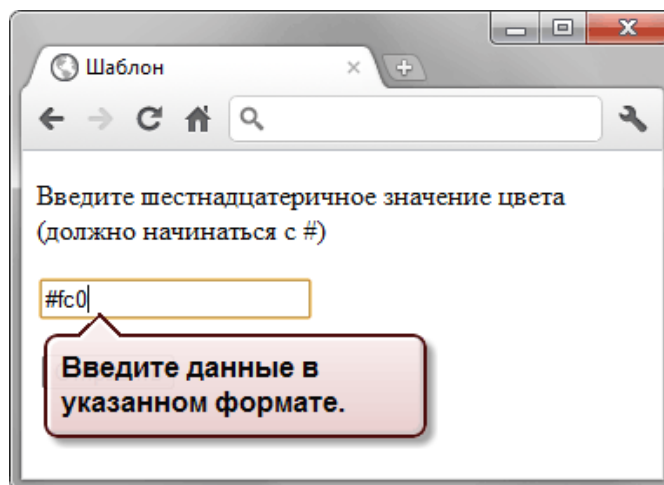
Некоторые данные нельзя отнести к одному из видов элементов формы, поэтому для них приходится использовать текстовое поле. При этом их ввод происходит по определённому стандарту. Так, IP-адрес содержит четыре числа разделённых точкой (192.168.0.1), почтовый индекс ограничен шестью цифрами (124007), телефон содержит код города и конкретное количество цифр часто разделяемых дефисом (391 555-341-42) и др. Браузеру необходимо указать шаблон ввода, чтобы он согласно нему проверял вводимые пользователем данные. Для этого используется атрибут **pattern**, а его значением выступает регулярное выражение. Некоторые типовые значения перечислены в таблице.

Шаблон	Описание
<code>[a-zA-Z]+\$</code>	Любые латинские буквы.
<code>[0-9]+\$</code>	Любое количество цифр.
<code>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}</code>	IP-адрес.
<code>[0-9]{6}</code>	Почтовый индекс.
<code>\d+(\,\d{2})?</code>	Цена в формате 1,34 (разделитель запятая).
<code>\d+(\.\d{2})?</code>	Цена в формате 2.10 (разделитель точка).

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ввод цвета</title>
  </head>
  <body>
    <form>
      <p>Введите шестнадцатеричное значение цвета
        (должно начинаться с #)</p>
      <p><input name="digit" required pattern="#[0-9A-Fa-f]{6}"></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

В примере предлагается ввести шестнадцатеричное значение цвета (#ffcc00) и если оно не лежит в этом диапазоне, браузер выводит сообщение об ошибке.



22.4. Отмена валидации

Валидация не всегда требуется для формы, к примеру, разработчик пожелает использовать универсальное решение на JavaScript и дублирующая проверка браузером ему уже ни к чему. В подобных случаях необходимо отключить встроенную валидацию. Для этого применяется атрибут **novalidate** тега **<form>**.

Пример отмены валидации:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Атрибут novalidate</title>
  </head>
  <body>
    <form novalidate>
      <p><input name="user" required placeholder="Ваше имя"></p>
      <p><input type="submit" value="Отправить"></p>
    </form>
  </body>
</html>
```

Для аналогичной цели применяется и атрибут **formnovalidate**, который добавляется к кнопке для отправки формы, в данном случае к тегу **<input type="submit">**. В этом случае форма из примера будет иметь следующий вид.

```
<form>
  <p><input name="user" required placeholder="Ваше имя"></p>
  <p><input type="submit" value="Отправить" formnovalidate></p>
</form>
```