Набережночелнинский институт
ФГАОУ ВПО «Казанский (Приволжский) федеральный университет»

Кафедра «Математические методы в экономике»

Язык гипертекстовой разметки HTML5

Электронный образовательный ресурс по дисциплине
«Интернет-технологии»

Набережные Челны
2015

Печатается по решению кафедры «Математические методы в экономике».

Рецензент: канд. техн. наук, доцент Мулюков Р.И.

Язык гипертекстовой разметки HTML5: электронный образовательный ресурс по дисциплине «Интернет-технологии» / В.С. Фрикк, Д.М. Лысанов. - Набережные Челны: Изд-во НЧИ КФУ, 2015. - 60 с.

Работа содержит описание HTML5, приведена общая структура web-документа, показаны основные теги и атрибуты, подробно рассмотрены возможности работы с формами и отдельными элементами форм.

Предназаначено для студентов очной и заочной форм обучения экономических специальностей (направлений подготовки).

Содержание

1. Элементы HTML

Базовым кирпичиком веб-страницы выступает элемент. Они могут делиться по разным критериям, например, по типу или своему назначению.

1.1. Элементы по типу

Пустые элементы

К ним относятся элементы, у которых нет закрывающего тега: <area>, <base>,
, <col>, <command>, <embed>, <hr>, <img>, <input>, <keygen>, <link>, <meta>, <param>, <source>, <track>, <wbr>.

Необрабатываемые текстовые элементы

Предназначены для вывода скриптов или стилей, имеющих синтаксис отличный от HTML: <script>, <style>.

RCDATA

Эти элементы могут содержать любой текст или спецсимволы, за исключением нестандартных спецсимволов, которые называются сомнительным амперсандом, например: &copi; или &#38T. К этой группе элементов относятся <textarea> и <title>.

Инородные элементы

Элементы, относящиеся к MathML или SVG.

Обычные элементы

Все остальные элементы, которые не входят в предыдущие группы.

1.2. Элементы по назначению

Корневой элемент

Элемент <html>.

Метаданные документа

<head>, а также элементы, которые располагаются внутри него.

Скрипты

Скрипты позволяют добавлять интерактивности на веб-страницу, в эту группу входят элементы, управляющие скриптами.

Структурные элементы

Элементы, управляющие основными разделами веб-страницы, вроде <body>, <section>, <nav>, <article>, <aside> и др.

Группирование контента

Элементы, обрамляющие текст, списки, изображения.

Текст

Элементы, изменяющие вид текста, например, делающие его жирным или курсивным, а также выделяющие текст по смыслу — аббревиатура, цитата, переменная, код и т.д.

Рецензирование

Элементы <ins> и <del> показывающие редактирования в документе.

Внедряемый контент

Элементы, вставляемые на страницу в виде разных объектов — изображения, видео, аудио и др.

Табличные данные

Элементы для создания и управления видом таблиц.

Формы

Формы являются одним из важных элементов любого сайта и предназначены для обмена данными между пользователем и сервером. В эту группу входят элементы для создания формы и её полей.

Интерактивные элементы

Специальные виджеты, с помощью которых пользователь может получать дополнительную информацию или управление.

Ссылки

Элементы <a> и <area>.

Подобное группирование условно и может принимать другой вид, потому что одни и те же элементы могут принадлежать разным группам.

1.3. Теги

Для обозначения начала и конца элемента применяются теги. Внутри тегов могут быть атрибуты со своими значениями, расширяющими возможности тегов, а также содержимое (рис. 1).

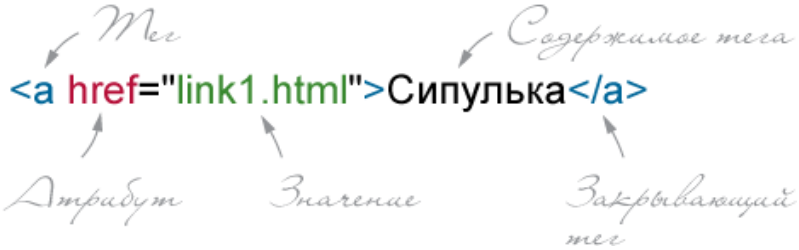


Рис. 1. Тег <a> с атрибутом href

Закрывающий тег похож на открывающий, но содержит слэш (/) внутри угловых скобок.

Пустые элементы не имеют закрывающего тега и содержимого (рис. 2).

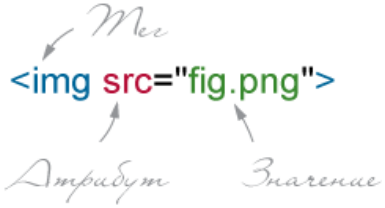


Рис. 2. Пустой тег <img>

Атрибуты тегов расширяют возможности самих тегов и позволяют гибко управлять различными настройками отображения элементов веб-страницы. Общее количество атрибутов достаточно велико, но их значения, как правило, можно сгруппировать по разным типам, например, задающих цвет, размер, адрес и др. Например, элемент <img> добавляет на веб-страницу изображение, при этом адрес графического файла мы указываем через атрибут src.

1.4. Доктайп

<!DOCTYPE> предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа) для того, чтобы браузер понимал, с какой версией HTML он имеет дело. Если доктайп не указан, браузеры переходят в режим совместимости, в котором не работают многие возможности HTML5, а также возникают ошибки с отображением документа. Доктайп не чувствителен к регистру и содержит всего два слова:

```
<!DOCTYPE html>
```

Это ключевой элемент и обычно он располагается в первой строке кода.

1.5. Комментарии

Некоторый текст можно спрятать от показа в браузере, сделав его комментарием. Хотя такой текст пользователь не увидит, он все равно будет передаваться в документе, так что, посмотрев исходный код, можно обнаружить скрытые заметки.

Комментарии нужны для внесения в код своих записей, не влияющих на вид страницы. Начинаются они тегом <!-- и заканчиваются тегом -->. Все, что находится между этими тегами отображаться на веб-странице не будет.

1.6. Необязательные теги

Если какой-то тег не указан, это не означает, что он не представлен вообще. Существуют определённые правила, позволяющие не писать некоторые теги. В табл. 1 представлены теги, которые можно не указывать и условие, при котором это происходит.

Табл. 1. Необязательные теги

Тег	Условие
<html>	
</html>	
<head>	Если внутри имеются другие элементы.
</head>	
<body>	Если пустой, а также содержит что-то кроме пробела или комментария.
</body>	
	Если после элемента следует или он последний у родителя.
</dt>	Если после элемента следует <dt> или <dd>.
</dd>	Если после элемента следует <dd>, <dt> или он последний у родителя.
	Если после элемента следует <address>, <article>, <aside>, <blockquote>, <dir>, , <dl>, <fieldset>, <footer>, <form>, <h1>,...,<h6>, <header>, <hgroup>, <hr>, <menu>, <nav>, , , <pre>, <section>, <table>, .
</rt>	Если после элемента следует <rt> или <rp>.
</rp>	Если после элемента следует <rt> или <rp>.
</optgroup>	Если после элемента следует <optgroup> или он последний у родителя.
</option>	Если после элемента следует <option>, <optgroup> или он последний у родителя.
<colgroup>	Если первым внутри идёт <col> и не следует перед другим элементом <colgroup>.
</colgroup>	
</thead>	Если после элемента следует <tbody> или <tfoot>.
<tbody>	Если первым внутри идёт <tr> и не следует перед <tbody>, <thead> или <tfoot> у которых опущен закрывающий тег.
</tbody>	Если после элемента следует <tbody> или <tfoot> или он последний у родителя.
</tfoot>	Если после элемента следует <tbody> или он последний у родителя.
</tr>	Если после элемента следует <tr> или он последний у родителя.
</td>	Если после элемента следует <td> или <th> или он последний у родителя.
</th>	Если после элемента следует <td> или <th> или он последний у родителя.

Если открывающий тег содержит один или несколько атрибутов, то тег должен указываться обязательно.

Из-за того, что многие теги можно не указывать, т.к. они подразумеваются по умолчанию, любой документ сводится к следующим частям.

1. Необязательная метка порядка байтов (byte order mark, BOM).
2. <!DOCTYPE html>.
3. <title>.

До и после доктайпа разрешается вставлять любое количество пробелов или комментариев. Таким образом, доктайп не обязательно должен располагаться в первой строке кода.

В примере 1 показан минимальный код HTML для вывода традиционного приветствия.

Пример 1. Минимальный HTML

```
<!DOCTYPE html>
<title> </title>
Привет, мир!
```

Метка порядка байтов состоит из кода символа U+FEFF в начале документа, где она используется для определения кодировки. Рекомендуется убирать этот символ, поскольку его наличие приводит к ошибкам отображения документа в некоторых браузерах. Для этого можно использовать редактор Notepad++, в меню «Кодировки» выбрать пункт «Кодировать в UTF-8 (без BOM)» (рис. 3).

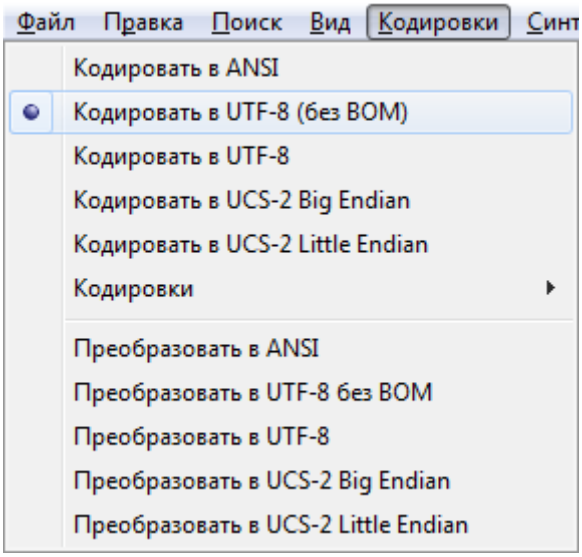


Рис. 3. Выбор кодировки

2. Устаревшие теги и атрибуты

HTML5 активно взаимодействует с CSS, поэтому запрет на многие атрибуты, начатый в HTML4 в пользу стилей, только усилился. Включение новых элементов вроде <video> и <audio> также сводит на нет теги, на которые возлагалась функция мультимедиа.

Устаревшие теги

Эти теги более не поддерживаются и должны исключаться автором из кода.

- <applet> — добавляет Java-апплет в документ. Вместо него следует использовать <embed> или <object>.
- <acronym> — этот тег вызывал постоянные вопросы, что такое акроним и чем он отличается от аббревиатуры. Для упрощения остался единственный тег <abbr>.
- <bgsound> — определяет музыкальный файл, который будет проигрываться на веб-странице при её открытии. Для воспроизведения музыки используйте новый элемент <audio>.
- <dir> — создает список, содержащий названия директорий, вместо него используйте <ul>.
- <frame>, <frameset>, <noframe> — фреймы более не поддерживаются. Если они вам требуются, используйте другую версию HTML или <iframe> совместно со стилями.
- <isindex> — предназначен для поискового индекса в текущем документе. Комбинация <form> и <input> лучше справляется с этой задачей.
- <listing>, <xmp> — для вывода листинга программы предназначены <pre> и <code>.
- <nextid> — этот тег не предназначен для людей и указывает идентификатор следующего документа для автоматических редакторов HTML. Полностью исключён.
- <noembed> — предназначен для отображения информации на веб-странице, если браузер не поддерживает работу с плагинами. В качестве альтернативы используйте <object>.
- <plaintext> — отображает содержимое контейнера «как есть», любые теги выводятся как текст. Вместо тега используйте MIME-тип text/plain.
- <rb> — определяет базовый текст внутри <ruby>. Этот тег полностью исключён.
- <strike> — для зачёркнутого текста применяется <s>, а для указания редакторской правки <del>.
  <basefont>, <big>, <blink>, , <font>, <marquee>, <multicol>, <nobr>, <spacer>, <tt>, <u> — вместо этих тегов управляющих видом текста применяются стили.

Обратите внимание, что тег <small> допустим, хотя относится к той же группе тегов, что и <big>.

Устаревшие, но поддерживаемые атрибуты

Атрибут http-equiv тега <meta> для указания языка должен заменяться атрибутом lang.

Было:

```
<meta http-equiv="content-language" content="ru" />
```

Стало:

```
<html lang="ru">
```

Атрибут name тега <a> должен заменяться атрибутом id. Если name присутствует, то должен содержать пустую строку или совпадать со значением id.

Было:

```
<a name="p12"></a>
```

Стало:

```
<a id="p12"></a>
```

Атрибут language тега <script> должен быть опущен. Если он присутствует, значение совпадает с JavaScript или другим типом, но в таком случае язык скрипта следует указать через type.

Было:

```
<script type="text/javascript" language="JavaScript">
```

Стало:

```
<script>
```

Атрибут border тега <img> не указывается, а толщина границы задаётся через стили. Если этот атрибут присутствует, его значение должно быть 0.

Было:

```
<img border="1">
```

Стало:

```
<img style="border: 1px solid black">
```

Наличие атрибута summary тега <table> приведёт к предупреждению.

Устаревшие атрибуты

- Элемент <a>: charset, coords, shape, methods, name, rev, urn.
- Элемент <area>: nohref.
- Элемент <body>: alink, bgcolor, link, marginbottom, marginheight, marginleft, marginright, margintop, marginwidth, text, vlink.
- Элемент
: clear.
- Элемент <embed>: name.
- Элемент <head>: profile.
- Элемент <html>: version.
- Элемент <iframe>: longdesc.
- Элемент <img>: longdesc, lowsrc, name.
- Элемент <input>: usemap.
- Элемент <link>: charset, methods, rev, target, urn.
- Элемент <meta>: scheme.
- Элемент <option>: name.
- Элемент <object>: archive, classid, code, codebase, codetype, declare, standby
- Элемент <param>: type, valuetype.
- Элемент <script>: event, for, language.
- Элемент <table>: datapagesize.
- Элемент <td> и <th>: abbr, axis.

Также запрещён атрибут datasrc для всех элементов.

3. Атрибуты

Любые атрибуты нечувствительны к регистру, пишутся внутри открывающего тега и содержат имя и значение. В именах атрибутов нельзя использовать пробелы, кавычки, знак больше (>), слэш (/) и равно (=), а также любые символы не определенные в Юникоде. На практике, все имена атрибутов у тегов известны и вставлять «отсебятину» не имеет смысла. В значениях атрибутов допустимо писать текст и спецсимволы за исключением амперсанда (&), который должен заменяться на &.

Различают четыре разных способа написания атрибутов и их значений.

Пустой атрибут (логический, булев)

Этот атрибут не имеет значения, поведение тега определяет наличие этого атрибута. Если такой атрибут указан, подразумевается, что установлено значение «истина», а отсутствие атрибута означает «ложь». Само значение можно не указывать, достаточно написать один лишь атрибут. Также разрешается в качестве значения писать пустую строку или имя атрибута. Ниже показаны разные варианты с равнозначным результатом.

```
<input disabled>
<input disabled="">
<input disabled="disabled">
```

Браузеры включают атрибут даже при наличии недопустимых значений вроде true или 1. Но лучше избегать таких решений, поскольку они противоречат спецификации HTML5 и не гарантируют правильную работу.

Значения без кавычек

Значение пишется непосредственно после знака равно идущим вслед за именем атрибута. До и после знака равно можно вставлять любое количество пробелов или обойтись без них.

```
<img src=link.html alt=Картинка>
```

Здесь атрибутами являются src и alt, а после знака = идёт их значение без кавычек. Поскольку атрибуты разделяются между собой одним или несколькими пробелами, то при отсутствии кавычек легко допустить ошибку, когда браузер воспримет предлагаемое нами значение как атрибут.

```
<img src=link.html alt=Картинка в тексте>
```

Здесь значением атрибута alt будет слово «Картинка», остальные слова воспринимаются как неверные атрибуты.

Использование двойных кавычек

Значение берётся в двойные кавычки, обычно такая форма указывается для текста.

```
<input type="checkbox">
```

Использование одинарных кавычек

Вместо двойных кавычек также допустимо писать одинарные.

```
<input type='checkbox'>
```

Значения атрибутов разделяются между собой пробелом, поэтому если у вас в качестве значения указывается предложение, обязательно берите его в одинарные или двойные кавычки.

Кавычки внутри значений

Внутри значений атрибутов не разрешается применять те же кавычки, в которых взято само значение. Но допустимо сочетать разные типы кавычек между собой. Если внутри текста необходимы одинарные кавычки или апостроф, то сам текст следует взять в двойные кавычки.

```
<img src="r.jpg" alt="Город Р'льех">
```

Соответственно, текст содержащий внутри двойные кавычки надо взять в одинарные.

```
<img src='c.jpg' alt='Такой вид кладки называется "циклопическим"'>
```

Также вместо двойной кавычки в тексте можно использовать спецсимвол ", а вместо апострофа — '.

4. Глобальные атрибуты

Наряду с атрибутами, характерными для конкретных тегов, в HTML5 существует и ряд атрибутов, который можно добавлять к любым тегам, поэтому входящие в эту группу атрибуты называются глобальными или универсальными. Ниже они перечислены с кратким описанием. По ссылке доступно подробное описание атрибута.

accesskey

Атрибут accesskey позволяет активировать ссылку с помощью некоторого сочетания клавиш с заданной в коде ссылки буквой или цифрой. Браузеры при этом используют различные комбинации клавиш. Например, для accesskey="s" работают следующие сочетания.

- Internet Explorer: Alt + S
- Chrome: Alt + S
- Opera: Shift + Esc, S
- Safari: Alt + S
- Firefox: Shift + Alt + S

class

Задаёт стилевой класс, который позволяет связать определенный тег со стилевым оформлением. В значении допускается указывать сразу несколько классов, разделяя их между собой пробелом.

contenteditable

Сообщает, что элемент доступен для редактирования пользователем — допускается удалять текст, и вводить новый. Также работают стандартные команды вроде отмены, вставки текста из буфера и др.

contextmenu

Устанавливает контекстное меню для элемента. В качестве значения указывается идентификатор меню созданного с помощью тега <menu>.

dir

Задает направление и отображение текста — слева направо или справа налево. Браузеры обычно самостоятельно различают направление текста, если он задан в кодировке Юникод, но с помощью атрибута dir можно указать, в каком направлении отображать текст. Для арабских и еврейских символов приоритетным является направление, заложенное в Юникод, поэтому на них атрибут dir действовать не будет.

draggable

Позволяет перетаскивать элемент для дальнейшего манипулирования с ним.

dropzone

Указывает, что делать с перетаскиваемым элементом.

hidden

Скрывает содержимое элемента от просмотра. Такой элемент не отображается на странице, но доступен через скрипты.

id

Задаёт идентификатор — уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты. Идентификатор в коде документа должен быть в единственном экземпляре, иными словами, встречаться только один раз.

itemid, itemprop, itemref, itemscope, itemtype

Группа атрибутов, предназначенная для работы с микроданными.

lang

Текст документа может быть набран как на одном языке, так и содержать вставки на других языках, которые могут различаться по своим правилам оформления текста. Например, для русского, немецкого и английского языка характерны разные кавычки, в которые берется цитата. Чтобы указать язык, на котором написан текст внутри текущего элемента и применяется атрибут lang. Браузер использует его значение для правильного отображения некоторых символов.

spellcheck

Указывает браузеру проверять или нет правописание и грамматику в тексте. Хотя атрибут можно устанавливать практически для всех элементов, результат будет заметен только для полей форм (<input>, <textarea>), а также редактируемых элементов (у них установлен атрибут contenteditable).

style

Применяется для определения стиля элемента с помощью правил CSS.

tabindex

Атрибут tabindex устанавливает порядок получения фокуса при переходе между элементами с помощью клавиши Tab. Переход происходит от меньшего значения к большему, например от 1 к 2, затем к 3 и так далее. При этом строгая последовательность не важна, допускается пропускать какие-то числа и начинать с любой цифры. Если значения tabindex у элементов совпадают, тогда учитывается их порядок появления в коде.

title

Создаёт всплывающую текстовую подсказку, которая появляется при наведении курсора на элемент. Вид такой подсказки зависит от браузера, настроек операционной системы и не может быть изменен с помощью HTML-кода или стилей.

5. Атрибуты data-*

В HTML5 для любого элемента можно использовать собственные атрибуты, начинающиеся с префикса data-. Это позволяет хранить разную информацию, которая может помочь в работе скриптов, а также для оформления элементов через CSS.

Атрибут должен иметь хотя бы один символ в нижнем регистре. Буквы в верхнем регистре хотя и допустимы, но они принудительно будут переведены в нижний регистр, поэтому не дают никакого эффекта. В именах атрибутов можно использовать дополнительные дефисы, как показано в примере 1.

```
Пример 1. Использование атрибута data-*
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>data-*</title>
 </head>
 <body>
  <ul>
    <li data-damage-resistance="40" data-weight="45"
        data-effect="+10 Сопротивляемость радиации">
        Силовая броня Клёна</li>
    <li data-damage-resistance="40" data-weight="45"
        data-effect="+15 Сопротивляемость радиации">
        Силовая броня Анклава</li>
    <li data-damage-resistance="50" data-weight="40"
        data-effect="+25 Сопротивляемость радиации">
        Силовая броня T-51b</li>
  </ul>
 </body>
</html>
```

Сами атрибуты никак не выводятся в браузере, поэтому мы увидим обычный список.

6. Значения атрибутов

Каждый атрибут тега содержит значение определённого типа и формата написания. Различают следующие типы значений: ключевые слова, числа, дата и время, цвета, URL.

Для всех атрибутов характерны следующие принципы.

- Если значение явно не указано, применяется значение по умолчанию или атрибут не учитывается.
- Если указано недопустимое значение атрибута, например, вместо числа добавлен текст, то значение такого атрибута игнорируется и используется значение по умолчанию.

Не все атрибуты имеют значение по умолчанию, в случае отсутствия такого атрибута или при его некорректном значении, атрибут пропускается.

6.1. Ключевые слова

Это заданный фиксированный набор определённых слов, которые допустимо подставлять в качестве значений атрибутов. Ключевые слова не чувствительны к регистру и их можно писать любым удобным способом. Так, значения handheld, Handheld и HANDHELD по своему действию одинаковы.

В примере 1 показано создание формы, которая отправляется на сервер методом POST с помощью значения post атрибута method.

```
Пример 1. Метод отправки формы
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Ключевые слова</title>
 </head>
 <body>
  <form method="post">
   <input name="user" placeholder="Введите имя">
   <input type="submit" value="Отправить">
  </form>
 </body>
</html>
```

В данном примере в элементе <form> используется атрибут method со значением post. Если этот атрибут не указать, то будет подставляться значение get, применяемое по умолчанию. Для первого элемента <input> атрибут type не задан, поскольку он подставляется автоматически со значением text, а для второго <input> атрибут type уже указан для создания кнопки отправки формы.

6.2. Числа

К числам относятся: положительные целые числа, содержащих одну или несколько цифр от 0 до 9; отрицательные числа; а также вещественные или дробные (например, 0.5).

Положительные целые числа

Активно применяются когда надо задать количество колонок, ограничить размер поля формы и количество вводимых символов, установить ширину и высоту рисунка и др. В примере 2 показано создание нумерованного списка, начинающегося с 11.

```
Пример 2. Использование целого числа
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Целое число</title>
 </head>
 <body>
  <ol start="11">
    <li>Чебурашка</li>
    <li>Крокодил Гена</li>
```

```
    <li>Шапокляк</li>
  </ol>
 </body>
</html>
```

Отрицательные целые числа

Отрицательные целые числа применяются реже по сравнению с положительными по понятным причинам. С их помощью нельзя задать перечисление или ширину, хотя бы потому, что рисунков с отрицательной шириной не бывает. Так что в HTML такие числа применяются в основном в формах для ограничения ввода.

Для указания отрицательного числа перед ним ставится знак минус (-) без пробелов (например: -15). В примере 3 показано использование отрицательных значений.

```
Пример 3. Число со знаком минус
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Отрицательные числа</title>
 </head>
 <body>
  <form>
   <p>Введите число с шагом 2
   <input type="number" min="-50" max="50" step="2" value="-10"></p>
  </form>
 </body>
</html>
```

Браузер Firefox не поддерживает тип number и вместо него отображает обычное текстовое поле.

Вещественные числа

Сюда входят десятичные дроби, для разделения целой и дробной части используется точка. Число может быть и отрицательным, тогда перед ним ставится знак минус (-).

Примеры вещественных чисел: -0.84, 3.1415926, 1.717.

Есть и вторая форма записи вещественных чисел, когда после числа добавляется латинский символ e или E, а затем число означающее степень, причём оно может быть и со знаком минус. Это аналогично тому, что число перед E умножают на 10 в степени, указанной после символа E. Вот несколько примеров для наглядности.

$3.1415926e5 = 3.1415926 \times 10^5 = 3.1415926 \times 100000 = 314159.26$
$5e\text{-}2 = 5 \times 10^{-2} = 5 / 100 = 0.05$
$78e2 = 78 \times 10^2 = 7800$

Браузер Opera некорректно работает с вещественными числами, в которых есть заглавная буква E, так что вставляйте строчную букву e.

6.3. Дата и время

Используется для указания года, месяца и года или полной даты, а также времени. Дата и время задается в особом формате, пример которого показан в табл. 1.

Табл. 1. Форматы даты и времени

Значение	Формат	Пример
Год	ГГГГ	2014
Месяц и год	ГГГГ-ММ	2014-12
Полная дата	ГГГГ-ММ-ДД	2014-12-23
Дата и время с минутами	ГГГГ-ММ-ДДТчч:мм	2014-07-24T18:18
Дата и время с секундами	ГГГГ-ММ-ДДТчч:мм:сс	2014-07-24T18:18:18
Дата и время с часовым поясом	ГГГГ-ММ-ДДТчч:мм:сс±чч:мм	2014-07-24T18:18:18+04:00

Для каждой единицы существует своя заданная форма и ограничения.

- Год — задается четырьмя цифрами (1860).
- Месяц — две цифры (01 — январь, 02 — февраль, 12 — декабрь).
- День — две цифры от 01 до 31.
- Час — две цифры от 00 до 23.
- Минуты — две цифры от 00 до 59.
- Секунды — две цифры от 00 до 59.
- Часовой пояс — часы и минуты с указанием знака плюс или минус.

Дата и время разделяются между собой заглавной латинской буквой T. Часовой пояс при необходимости пишется после времени со знаком плюс или минус. К примеру, для Москвы часовой пояс будет +03:00.

6.4. Цвета

Значение цвета представляет собой три числа в диапазоне от 0 до 255 в шестнадцатеричном представлении описывающие красную, зелёную и синюю компоненту в цветовом пространстве sRGB. Каждое значение цвета должно начинаться с символа решётки (#), после которого могут идти следующие шесть цифр или букв в любом регистре: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Другие символы не допускаются. Подробнее о цвете в HTML смотрите здесь.

6.5. URL

URL — это адрес документа или файл. Содержит в себе несколько частей, не все из которых являются обязательными. Это протокол, имя хоста, порт, путь, строка запроса и хэш. В табл. 2 перечислены параметры URL с их описанием.

Табл. 2. Параметры URL

Параметр	Описание	Пример
протокол	Сетевой протокол. Для гипертекстовых документов это HTTP.	http:// https://
имя хоста	Адрес сайта.	htmlbook.ru www.google.com
порт	Системный ресурс, выделяемый веб-серверу. По умолчанию имеет значение 80, его можно не указывать.	:80
путь	Путь к документу на сайте.	/open/doc.html
строка запроса	Строка, в которой передаются параметры GET-запроса со значениями. Пишется после знака вопроса (?).	?name=vasya
хэш	Строка после знака решётки (#).	#top

В зависимости от наличия тех или иных параметров различают абсолютные и относительные адреса. Абсолютный адрес содержит порт и имя хоста, относительный — путь к документу. Строка запроса и хэш допустимо добавлять к адресу любого типа. Если URL содержит только хэш, то в текущем документе произойдёт переход к элементу, у которого задано id="имя_хэша". Решётка в атрибуте id в таком случае не пишется.

На URL влияет элемент <base> с атрибутом href, адрес документа с его учётом может оказаться другим, чем тот, что задан явно.

7. Корневой элемент

В HTML корневым называется элемент <html>, внутри которого располагаются элементы <body> и <head>. Само присутствие открывающего тега <html> и закрывающего </html> является необязательным и их можно опустить (пример 1).

```
Пример 1. Страница без корневого элемента
<!DOCTYPE html>
 <head>
  <meta charset="utf-8">
  <title>html</title>
```

```
 </head>
 <body>
  <p>Текст</p>
</body>
```

В качестве атрибутов используются глобальные атрибуты, например, lang для указания языка документа. В примере 2 показана страница на английском языке, поэтому к тегу <html> добавлен атрибут lang со значением en.

```
Пример 2. Атрибут lang
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <title>Welcome</title>
 </head>
 <body>
  <p>Lets me speak from my heart in English.</p>
 </body>
```

Уникальным для корневого элемента является атрибут manifest, который указывает манифест кэша приложения, что позволяет создавать автономный сайт, работающий даже без подключения к Интернету.

8. Метаданные документа

Вид документа заданный с помощью стилей, поведение отдельных элементов, а также отношения текущего документа с другими задаются с помощью метаданных. В качестве контейнера для них выступает элемент <head>. Единственным обязательным элементом в метаданных выступает <title>, он устанавливает заголовок страницы. В зависимости от сложности документа объём метаданных может быть маленьким или большим. В примере 1 показан минимальный объём метаданных.

```
Пример 1. Заголовок документа
<!DOCTYPE html>
<html>
 <head>
  <title>Заголовок страницы</title>
 <body>
  <p>Текст</p>
 </body>
</html>
```

Открывающий тег <head>, а также закрывающий тег </head> являются необязательными и могут быть опущены.

8.1. Элемент title

Элемент <title> определяет заголовок или имя документа. Это один из важных элементов, поскольку заголовок используется поисковыми системами при выдаче результатов поиска, указывается при сохранении в закладках и в журнале браузера. Сам заголовок представляет собой текст и отображается во вкладке и в заголовке браузера или только в одном месте (рис. 1).

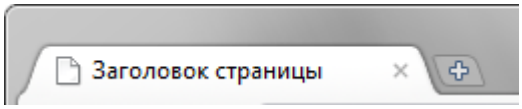


Рис. 1. Заголовок в браузере Chrome

<title> обязательный элемент в документе, поэтому он должен присутствовать в любом коде хотя бы формально. В примере 1 показан пустой тег <title>.

```
Пример 1. Пустой заголовок
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title> </title>
 </head>
 <body>
```

```
  <p>Текст</p>
 </body>
</html>
```

На деле заголовок не пустой, поскольку содержит пробел. Пустой заголовок не содержащий ни одного символа, включая пробел, не допустим.

Также запрещено включать в код два и более элемента <title>, он должен быть только один. Любые теги внутри <title> отображаются как обычный текст со всеми угловыми скобками < и >, также можно использовать спецсимволы (например: ©) или текст в юникоде. В примере 2 показаны примеры корректных заголовков.

```
Пример 2. Допустимые заголовки
<title>H<sub>2</sub>O</title>
<title>&copy; Copyright Ваня Шапочкин</title>
<title>Компания Арес&reg;</title>
<title>♉ Быки и коровы</title>
```

8.2. Элемент base

Элемент <base> не выводит никакого контента и выполняет исключительно служебную функцию — позволяет указать базовый URL, относительно которого будут устанавливаться другие адреса, например, для изображений и ссылок. Также <base> задаёт значение атрибута target, которое по умолчанию применяется ко всем ссылкам.

В документе разрешается иметь только один <base>.

Атрибут href

Предположим, все изображения для наших документов хранятся в папке assets/images. Чтобы постоянно не указывать этот путь перед именем файла, его можно вынести в значение атрибута href тега <base>, как показано в примере 1.

```
Пример 1. Использование href
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Адрес изображения</title>
  <base href="/example/images/">
 </head>
 <body>
  <p><img src="figure.jpg" alt=""></p>
 </body>
</html>
```

Хотя в данном примере путь к файлу задан как figure.jpg в действительности его надо понимать как /example/images/figure.jpg. Обратите внимание на последний слэш (/), если его не указать, то путь к файлу будет /example/imagesfigure.jpg, что приведёт к ошибке и рисунок выводиться не будет.

В данном случае атрибут href будет действовать на все адреса, используемые в документе, включая ссылки, поэтому на них путь, заданный в <base> также будет распространяться. Если в пример 1 добавить ссылку ссылка, то браузер будет искать документ по адресу /example/images/link.html.

Из-за того, что <base> влияет на все используемые ниже него адреса, его рекомендуется ставить в <head> раньше элементов со ссылками. Если перед <base> стоит, к примеру, <link>, то базовый адрес на него действовать не будет. Также базовый адрес не влияет на атрибут manifest в корневом элементе <html>.

В качестве значения можно использовать и адрес сайта, как показано в примере 2.

```
Пример 2. Адрес сайта
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Адреса</title>
```

```
  <base href="http://htmlbook.ru/">
 </head>
 <body>
  <p><img src="example/images/figure.jpg" alt=""></p>
  <p><a href="example/1.html">Ссылка</a></p>
 </body>
</html>
```

Полный адрес изображения будет http://htmlbook.ru/example/images/figure.jpg. При сохранении страницы на локальный компьютер при её открытии все файлы будут загружаться непосредственно с сайта htmlbook.ru.

Хотя в примерах выше в <base> использовались адреса папок, допустимо указывать и адрес документа:

```
<base href="example/index.php">
```

Такой путь не надо подставлять прямолинейно, как это делалось в случае с папками. Адрес документа в href говорит, что относительно него следует задавать пути к файлам. В качестве примера рассмотрим следующую структуру файлов на сайте (рис. 1).

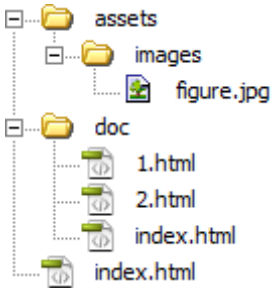


Рис. 1. Файлы и папки на сайте

Если задать базовый адрес как doc/index.html, тогда в файле 1.html пути к остальным файлам поменяются, как показано в примере 3.

```
Пример 3. Файл 1.html
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Адрес относительно файла</title>
  <base href="doc/index.html">
 </head>
 <body>
  <p><img src="../../assets/images/figure.jpg" alt=""></p>
  <p><a href="../2.html">Соседний документ</a></p>
  <p><a href="../../index.html">Вернуться на главную страницу</a></p>
 </body>
</html>
```

Поскольку перед doc нет дополнительных слэшей, то браузер считает, что папка doc находится в том же месте, что и файл 1.html, т.е. полный путь к базовому файлу относительно корня будет doc/doc/index.html. Не имеет значения, что в действительности такой папки и файла не существует, главное указать относительно неё путь. Поэтому в примере 3 и появляются дополнительные конструкции вида ../. Поскольку адрес в примере фиктивный, замена doc внутри href на любое другое значение (скажем, doc123) не изменит путь к ссылкам.

Хотя пример 3 работоспособный, использование относительных адресов вносит путаницу, поэтому рекомендуем включать перед адресом слэш, который указывает на корень сайта (пример 4).

```
Пример 4. Базовый адрес относительно папки doc
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Адрес относительно doc</title>
  <base href="/doc/index.html">
 </head>
```

```
 <body>
  <p><img src="../assets/images/figure.jpg" alt=""></p>
  <p><a href="2.html">Соседний документ</a></p>
  <p><a href="../index.html">Вернуться на главную страницу</a></p>
 </body>
</html>
```

Конструкция /doc указывает на папку doc находящуюся в корне нашего сайта. Учтите, что адреса со слэшем впереди работают только под управлением сервера, но никак не локально.

Допустимо использовать только один базовый адрес на странице. Если ошибочно добавлено несколько <base> с разными href, то применяется только первый, остальные игнорируются.

Атрибут target

Значение атрибута target определяет имя контекста, которое применяется для ссылок (<a>) и форм (<form>). В качестве имени контекста обычно указывается имя фрейма или ключевое слово: _blank, _self, _parent или _top.

В качестве примера рассмотрим добавление на страницу фрейма с именем frame, оно определяется атрибутом name тега <iframe>. Чтобы ссылки за пределами фрейма открывались внутри него, в элементе <base> укажем target="frame", как показано в примере 5.

```
Пример 5. Имя контекста
<!DOCTYPE html>
<html>
  <head>
  <meta charset="utf-8">
  <title>Ссылки во фрейме</title>
  <base target="frame">
 </head>
 <body>
  <p><a href="2.html">2</a> <a href="3.html">3</a></p>
  <p><iframe name="frame" src="1.html"></iframe></p>
 </body>
</html>
```

Использование <base> в таком качестве равнозначно тому, что для каждой ссылки добавляется target="frame".

Кроме использования имен допустимы следующие ключевые слова.

- _blank — открывает ссылку в новом окне или вкладке;
- _self — открывает ссылку в текущем окне или вкладке, равнозначно значению "" (пустая строка);
- _parent — открывает ссылку в родительском контексте. Например, если такая ссылка нажата внутри фрейма, то она открывается не во фрейме, а на странице содержащей фрейм;
- _top — открывает ссылку в контексте верхнего уровня. Результат будет заметен при сложной фреймовой структуре, при наличии на странице одного фрейма, открытие ссылки внутри него подобно действию _parent.

Если одновременно требуются атрибуты href и target, они объединяются в пределах одного тега:

```
<base href="http://htmlbook.ru/content" target="_blank">
```

9. Формы

Формы являются одним из важных элементов любого сайта и предназначены для обмена данными между пользователем и сервером. С помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

9.1. Создание формы

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.

Сама форма создаётся с помощью тега <form>, внутри которой могут быть любые необходимые теги, и характеризуется следующими необязательными параметрами:

1. адрес программы на веб-сервере, которая будет обрабатывать содержимое данных формы;
2. элементы формы, которые представляют собой стандартные поля для ввода информации пользователем;
3. кнопка отправки данных на сервер.

Допускается использовать несколько форм на странице, но они не должны вкладываться одна в другую (пример 1).

```
Пример 1. Ошибочное использование форм
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Формы</title>
 </head>
 <body>
  <form action="handler.php">
   <p><input name="a"> <input type="submit"></p>
   <form action="handler.php">
    <p><input name="b"> <input type="submit"></p>
   </form>
  </form>
 </body>
</html>
```

Перед отправкой данных браузер подготавливает информацию в виде пары «имя=значение», где имя определяется атрибутом name тега <input> или другим тегом допустимым в форме, а значение введено пользователем или установлено в поле формы по умолчанию. После нажатия пользователем кнопки Submit, происходит вызов обработчика формы, который получает введенную в форме информацию, а дальше делает с ней то, что предполагает разработчик. В качестве обработчика формы обычно выступает программа, заданная атрибутом action тега <form>. Программа может быть написана на любом серверном языке вроде PHP, Python, C# и др.

В простейшем случае тег <form> не содержит никаких атрибутов и представлен в примере 2.

```
Пример 2. Простая форма
<!DOCTYPE html>
 <html>
  <head>
  <meta charset="utf-8">
  <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. Для корректной интерпретации данных используйте атрибут enctype со значением text/plain в теге <form> (пример 3).

```
Пример 3. Отправка формы по почте
<!DOCTYPE html>
 <html>
  <head>
  <meta charset="utf-8">
  <title>Формы</title>
 </head>
 <body>
  <form action="mailto:vlad@htmlbook.ru" enctype="text/plain">
   <p><input name="a"> <input type="submit"></p>
  </form>
 </body>
</html>
```

Браузеры неоднозначно работают с таким кодом. Firefox предложит список подходящих приложений для отправки почты (рис. 1), Internet Explorer выведет предупреждение и попытается запустить программу, сопоставленную с почтой, Opera подобно Firefox предложит список вариантов подходящих для отправки почты.

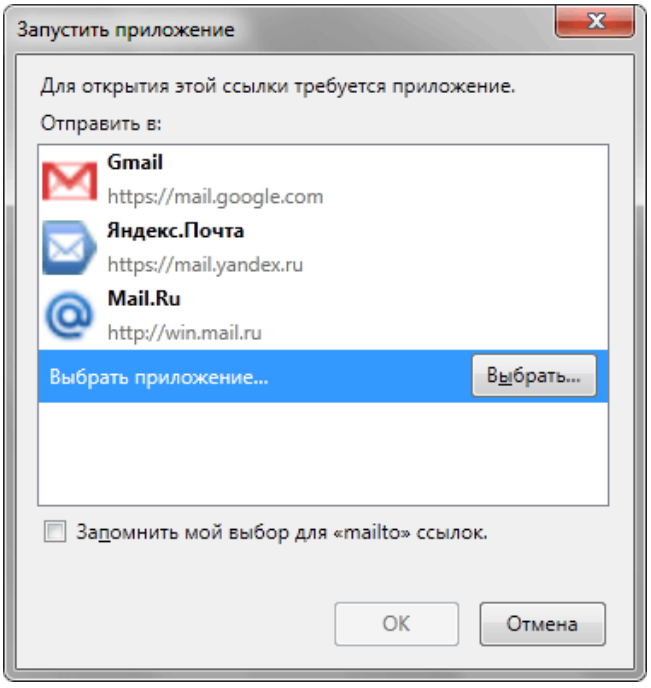


Рис. 1. Запуск приложения в Firefox

9.2. Отправка данных формы

Сама форма обычно предназначена для получения от пользователя информации для дальнейшей пересылки её на сервер, где данные формы принимает программа-обработчик. Такая программа может быть написана на любом серверном языке программирования вроде PHP, Perl и др. Адрес программы указывается в атрибуте action тега <form>, как показано в примере 1.

```
Пример 1. Отправка данных формы
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <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.

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 табл. 1 показаны различия между этими методами.

Табл. 1. Различия между методами GET и POST

	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 2).

```
Пример 2. Открытие формы во фрейме
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <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 3).

```
Пример 3. Связывание формы с полями
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <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. В примере 4 показано использование этих атрибутов.

```
Пример 4. Отправка формы
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <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>
```

Все новые атрибуты форм не поддерживаются некоторыми браузерами, в частности, Internet Explorer и Safari.

9.3. Однострочное текстовое поле

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

```
<input атрибуты>
```

Значение атрибута type для тега <input> по умолчанию определено как text, поэтому его можно не указывать явно. Атрибуты текстового поля перечислены в табл. 1.

Табл. 1. Атрибуты однострочного текстового поля

Атрибут	Описание
size	Ширина текстового поля, которая определяется числом символов моноширинного шрифта. Иными словами, ширина задается количеством близстоящих букв одинаковой ширины по горизонтали.
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 длинее самого поля.
name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
value	Начальный текст отображаемый в поле.

Создание текстового поля показано в примере 1.

```
Пример 1. Текстовое поле
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текстовое поле</title>
 </head>
 <body>
  <form action="handler.php">
   <p><strong>Как вас зовут?</strong></p>
   <p><input maxlength="25" size="40" value="Вася"></p>
  </form>
 </body>
</html>
```

В результате получим следующее (рис. 1).

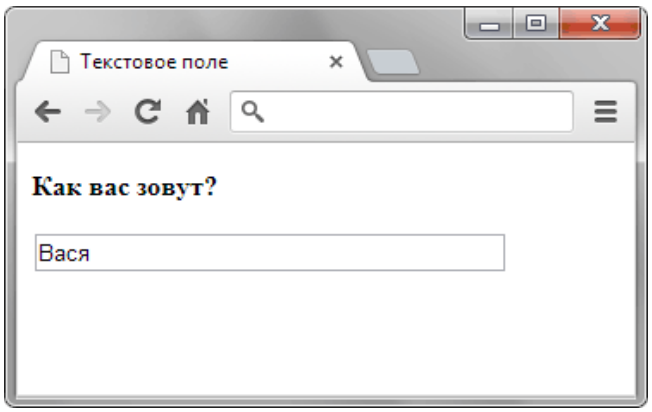


Рис. 1. Вид текстового поля в браузере Chrome

9.4. 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.

Синтаксис создания следующий.

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

Атрибуты совпадают с атрибутами текстового поля и перечислены в табл. 1.

Табл. 1. Атрибуты поля с паролем

Атрибут	Описание
size	Ширина поля с паролем, которая определяется числом звездочек моноширинного шрифта.
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.
name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
value	Начальный текст, который выводится в поле. Этот текст не отображается и заменяется звездочками.

Создание поля для пароля показано в примере 1.

```
Пример 1. Поле с паролем
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Поле с паролем</title>
 </head>
 <body>
  <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>
 </body>
</html>
```

В результате получим следующее (рис. 1).

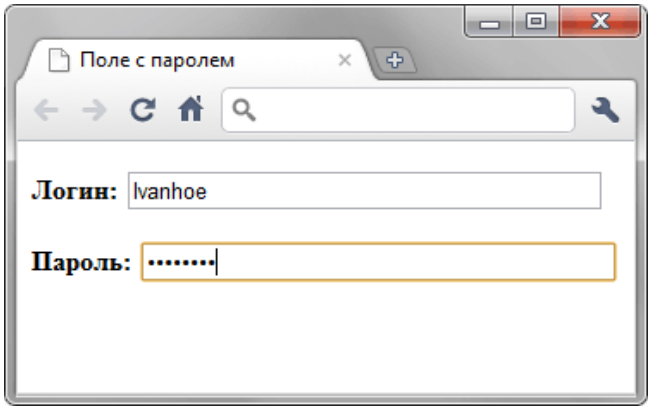


Рис. 1. Вид поля с паролем в браузере Chrome

9.5. Многострочный текст

Этот элемент формы 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.

Синтаксис создания поля следующий.

```
<textarea атрибуты>
```

```
  Текст
</textarea>
```

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

Допустимые атрибуты перечислены в табл. 1.

Табл. 1. Атрибуты тега <textarea>

Атрибут	Описание
cols	Ширина поля в символах.
disabled	Блокирует доступ и изменение элемента.
maxlength	Максимальное число символов текста, которое можно ввести.
name	Имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать.
readonly	Устанавливает, что поле не может изменяться пользователем.
rows	Высота поля в строках текста.
wrap	Параметры переноса строк.

Создание поля многострочного текста показано в примере 6.1.

Ни один из этих атрибутов не является обязательным, поэтому простая форма для отправки текста выглядит так (пример 1).

```
Пример 1. Текстовое поле
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текстовое поле</title>
 </head>
 <body>
 <form action="handler.php">
  <p><b>Введите ваш отзыв:</b></p>
  <p><textarea name="comment"></textarea></p>
  <p><input type="submit"></p>
 </form>
 </body>
</html>
```

Результат примера в браузере Chrome показан на рис. 1.

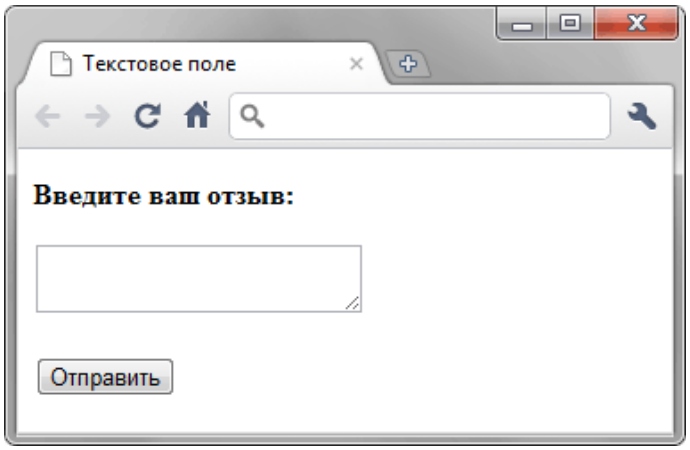
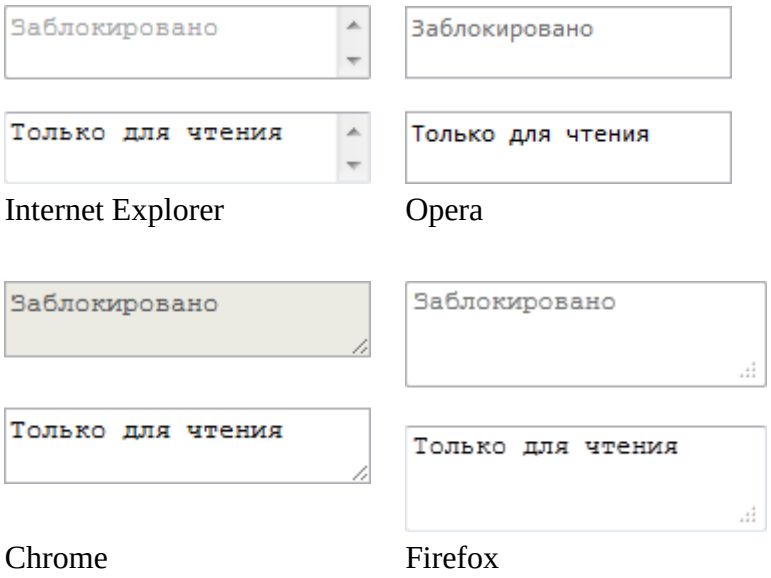


Рис. 1. Вид текстового поля по умолчанию

Дополнительно поле может находиться в двух состояниях — блокированном и только для чтения. Спецификация HTML5 не определяет вид поля и текста в подобных состояниях, поэтому браузеры по-разному его отображают (рис. 2).



Текст внутри блокированного 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. В примере 2 показан способ создания поля для чтения.

```
Пример 2. Поле для чтения
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Поле для чтения</title>
 </head>
 <body>
 <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>
 </body>
</html>
```

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.

9.6. 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 перечислены в табл. 1.

Табл. 1. Атрибуты кнопок

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

Создание кнопки показано в примере 1.

```
Пример 1. Добавление 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. Результат примера показан на рис. 1.

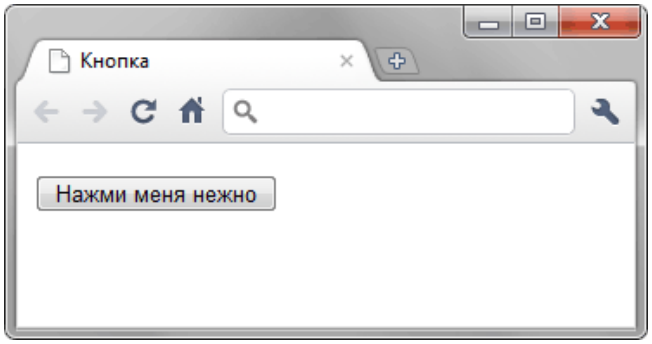


Рис. 1. Вид кнопки

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. На рис. 2 показаны разные виды кнопок, полученные с помощью <button>.

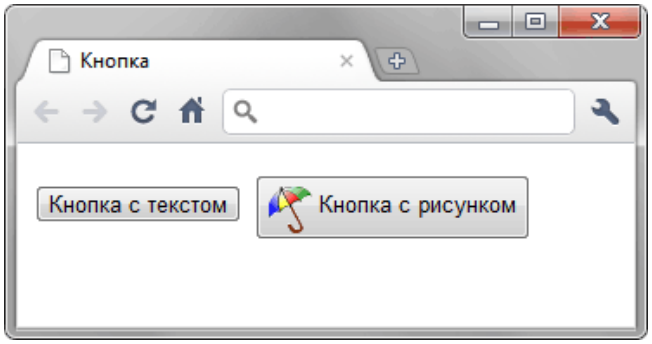


Рис. 2. Кнопки, созданные с помощью <button>

Синтаксис создания такой кнопки следующий.

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

Атрибуты перечислены в табл. 1, 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 <img> добавляется внутрь <button>, как показано в примере 2.

```
Пример 2. Рисунок на кнопке
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Кнопка</title>
 </head>
 <body>
  <form>
   <p><button>Кнопка с текстом</button>
   <button>
```

```
    <img src="images/umbrella.gif" alt="" style="vertical-align:middle">
    Кнопка с рисунком
   </button></p>
  </form>
 </body>
</html>
```

В данном примере показано создание обычной кнопки с текстом, при этом , 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.

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

Атрибуты те же, что и у рядовых кнопок (пример 3).

```
Пример 3. Отправка 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.

Кнопка Reset

При нажатии на кнопку Reset данные формы возвращаются в первоначальное значение. Как правило, эту кнопку применяют для очистки введенной в полях формы информации. Для больших форм от использования кнопки Reset лучше вообще отказаться, чтобы по ошибке на нее не нажать, ведь тогда придётся заполнять форму заново.

Синтаксис создания указанной кнопки прост и похож на другие кнопки.

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

В примере 4 показана форма с одним текстовым полем, которое уже содержит предварительно введённый 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 4. Кнопка 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».

9.7. Переключатели

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

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

Атрибуты переключателей перечислены в табл. 1.

Табл. 1. 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.

Создание группы переключателей показано в примере 1.

```
Пример 1. Создание переключателей
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Переключатели</title>
 </head>
 <body>
  <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>
 </body>
</html>
```

В результате получим следующее (рис. 1).

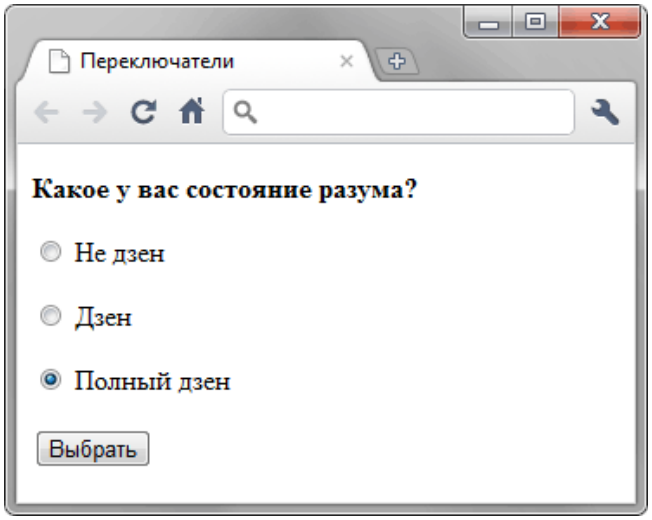


Рис. 1. Вид переключателей в браузере

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.

9.8. 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.

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

Атрибуты флажков перечислены в табл. 1.

Табл. 1. Атрибуты флажков

Атрибут	Описание
checked	Предварительное выделение флажка.
name	Имя флажка для его идентификации обработчиком формы.
value	Задаёт, какое значение будет отправлено на сервер.

Использование флажков показано в примере 1.

```
Пример 1. Создание флажков
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Флажки</title>
 </head>
 <body>
  <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>
 </body>
</html>
```

В результате получим следующее (рис. 1).

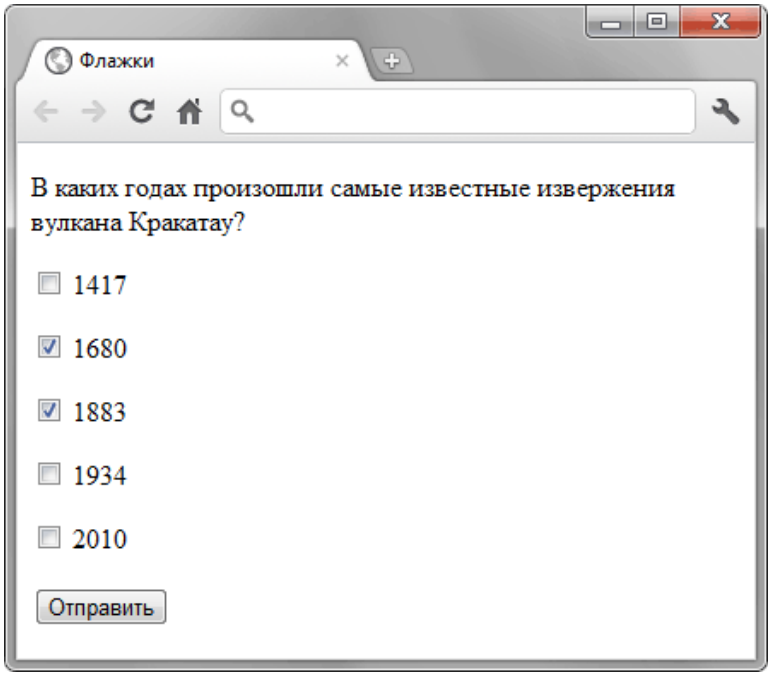


Рис. 1. Вид флажков в браузере

9.9. Поле со списком

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>
```

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>

Рассмотрим атрибуты тега <select>, с помощью которых можно изменять представление списка.

multiple

Наличие multiple сообщает браузеру отображать содержимое элемента <select> как список множественного выбора. 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.

В примере 1 показано создание списка множественного выбора.

```
Пример 1. Список 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>
```

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.

Атрибуты тега <OPTION>

Тег <option> также имеет атрибуты, влияющие на вид списка, они представлены далее.

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.

Создание списка показано в примере 2.

```
Пример 2. Использование списка
<!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, как показано в примере 3.

```
Пример 3. Группирование 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>
```

Результат примера показан на рис. 1.

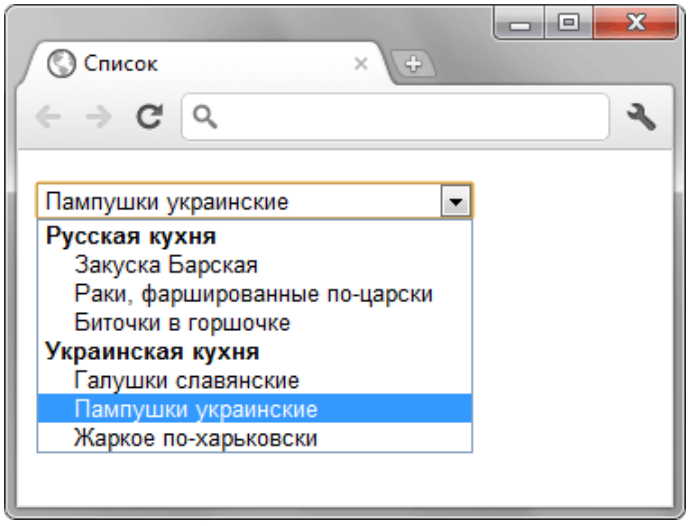


Рис. 1. Группированный список

9.10. Скрытое 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.

Синтаксис создания скрытого поля.

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

Атрибуты перечислены в табл. 1.

Табл. 1. Атрибуты скрытого поля

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

Пример использования скрытых полей приведен в примере 1.

```
Пример 1. Использование скрытого поля
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Скрытое поле</title>
 </head>
 <body>
  <form action="handler.php" method="post">
   <p><b>Напишите любимое слово и нажмите кнопку Отправить
   (никакие данные не будут передаваться на сервер!):</b></p>
   <p><input size="25" name="word">
   <input type="hidden" name="name" value="Vasya">
   <input type="hidden" name="password" value="pupkin"></p>
   <p><input type="submit" value="Отправить"></p>
  </form>
 </body>
</html>
```

В данном примере показано создание двух скрытых полей, одно из них носит имя name и получает значение Vasya, а второе именуется password со значением pupkin. В результате отправки формы обработчику, указанному в атрибуте action, программа может легко прочитать эти данные и интерпретировать их по усмотрению разработчика.

9.11. Поле с изображением

Для отправки формы на сервер применяется кнопка Submit, которая имеет слабые средства по настройке своего вида. Чтобы не ограничивать дизайн и расширить возможности по оформлению форм, используйте поле с изображением, аналогичное по своему действию кнопке Submit. При нажатии на рисунок данные формы отправляются на сервер и обрабатываются программой, заданной атрибутом action тега <form>.

Поле с изображением создаётся следующим образом.

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

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 <img>.

В примере 1 показано использование поля с изображением.

```
Пример 1. Кнопка 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>
```

Результат данного примера показан на рис. 1.

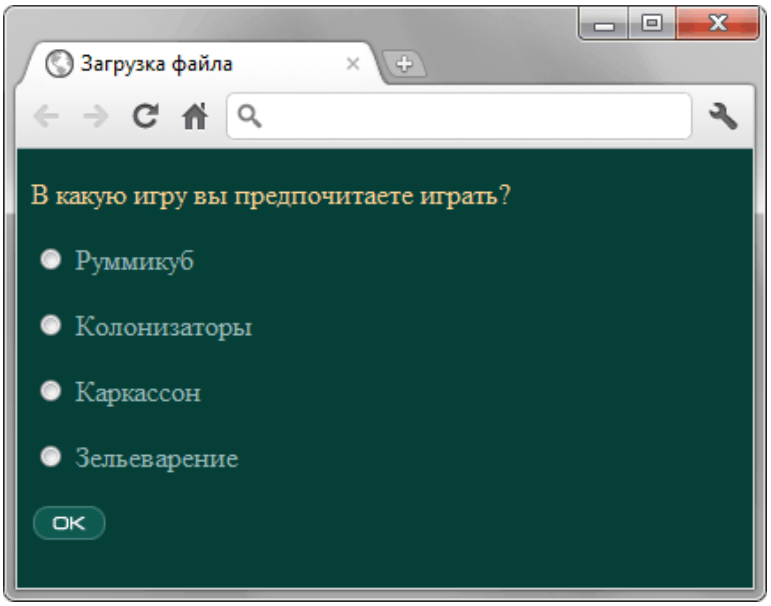


Рис. 1. Форма с графической кнопкой

В отличие от кнопки Submit на сервер также отправляются координаты точки щелчка по изображению в виде x и y. Отсчёт координат ведётся от левого верхнего угла картинки и указывается в пикселах. 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.

9.12. Загрузка файлов

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...» (рис. 1). В Safari и Chrome доступна только кнопка «Выберите файл» (рис. 2).

Рис. 1. Вид поля для загрузки файла в Firefox

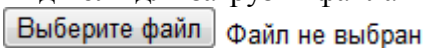


Рис. 2. Загрузка файлов в Chrome

При нажатии на кнопку открывается окно для выбора файла, где можно указать, какой файл пользователь желает использовать.

Синтаксис поля для отправки файла следующий.

```
<input type="file" атрибуты>
```

Атрибуты перечислены в табл. 1.

Табл. 1. Атрибуты поля для отправки файла

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.

Форма для загрузки файла продемонстрирована в примере 1.

```
Пример 1. Создание 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 accept 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 табл. 2 показаны некоторые допустимые значения атрибута accept.

Табл. 2. Типы файлов

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.

Использование дополнительных атрибутов показано в примере 2.

```
Пример 2. 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.

Результат примера показан на рис. 3. Обратите внимание, что из-за наличия multiple несколько изменился вид поля.

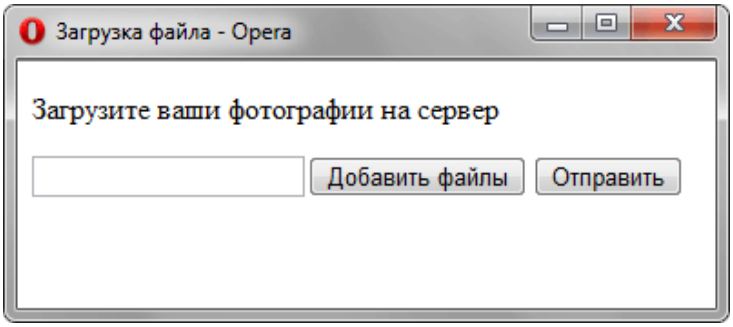


Рис. 3. Загрузка файлов в Opera

9.13. 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 по большей части совпадают с текстовым полем и приведены в табл. 1.

Табл. 1. Атрибуты поля для почтового адреса

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.

В примере 1 показано применение значения email для создания формы авторизации на сайте, где в качестве логина указывается адрес электронной почты.

```
Пример 1. Адрес 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>
```

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. Firefox поле с неверным адресом обозначает красной рамкой и сообщением об ошибке при отправке формы (рис. 1), Chrome никак сразу не выделяет поле с ошибкой, но выводит сообщение при отправке формы (рис. 2). Аналогично поступает и Opera (рис. 3). Сама форма на сервер не отправляется, пока ошибка не будет исправлена.

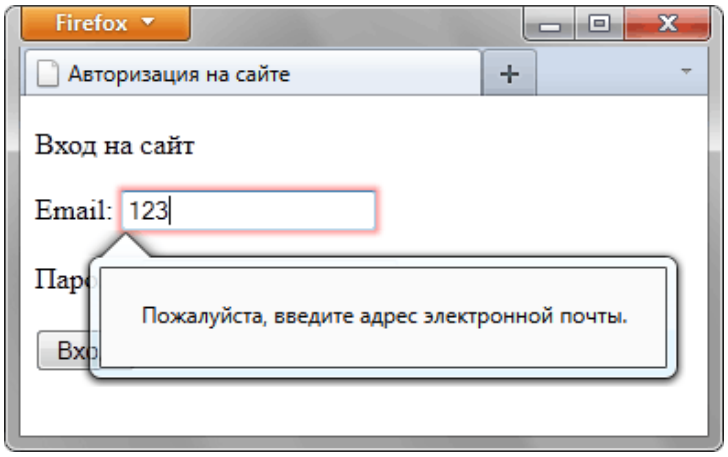


Рис. 1. Отправка формы в Firefox

9.14. 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 Opera 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">).

В примере 1 показано использование поля для веб-адреса, которое должно быть заполнено перед отправкой формы.

```
Пример 1. Ввод адреса сайта
<!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, Firefox выводит предупреждение как показано на рис. 1, а Chrome — как показано на рис. 2.

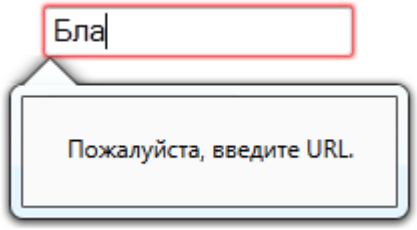


Рис. 1. Вид сообщения в Firefox

Рис. 2. Вид сообщения в Chrome

Opera не выводит предупреждений при вводе неверного веб-адреса, исправляя его автоматически.

По своему виду поле для веб-адреса не отличается от текстового, за исключением браузера Opera, который несколько увеличивает ширину поля для веб-адреса.

9.15. 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.

В примере 1 показано, как создать поле для указания желаемого цвета.

```
Пример 1. Выбор 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>
```

На рис. 1 показан поле, как оно исходно отображается в браузере Opera, на рис. 2 это же поле после щелчка по нему.

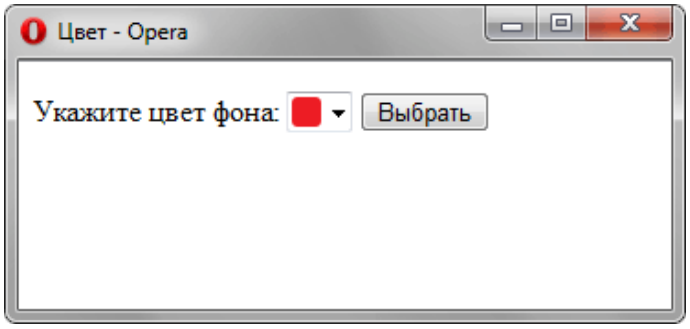


Рис. 1. Поле для выбора цвета

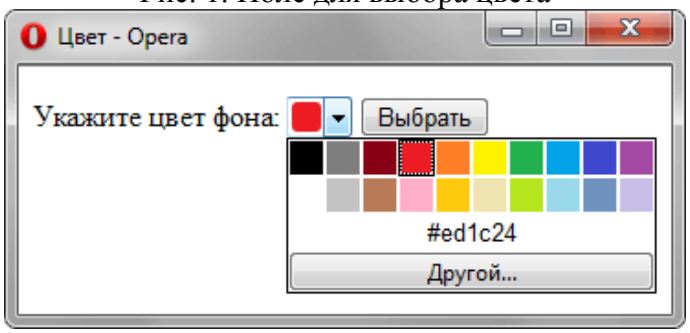


Рис. 2. Отображение палитры

9.16. 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 (рис. 1).

Рис. 1. Вид поля для ввода чисел

Синтаксис создания поля следующий:

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

Допустимые атрибуты перечислены в табл. 1.

Табл. 1. Атрибуты поля с числом

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 (пример 1). 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 1. Ограничение 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, как показано в примере 2.

```
Пример 2. 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>
```

Результат примера продемонстрирован на рис. 2.

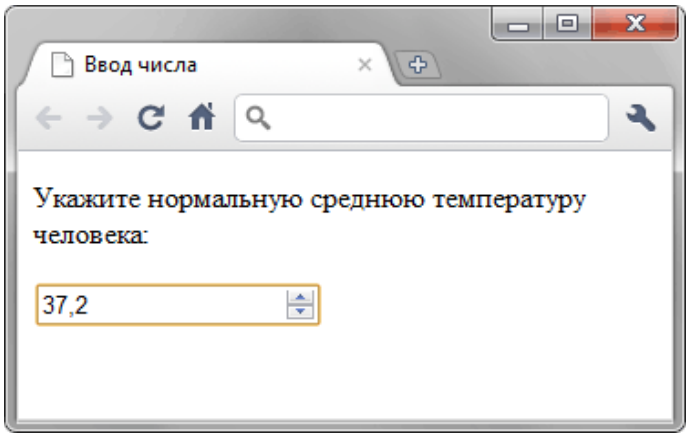


Рис. 2. Ввод дробных чисел в поле

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

9.17. 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 рис. 1 продемонстрирован вид ползунка в разных браузерах.

Рис. 1. Вид ползунка в 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. В примере 1 с помощью ползунка изменяется размер изображения, такая возможность часто используется в различных фотогалереях.

```
Пример 1. Использование 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><img src="images/figure.jpg" alt="" id="pic"></p>
 </body>
</html>
```

В данном примере 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 картинки при желании уменьшается или наоборот, увеличивается. Результат примера при крайнем значении ползунка в браузере Chrome показан на рис. 2.

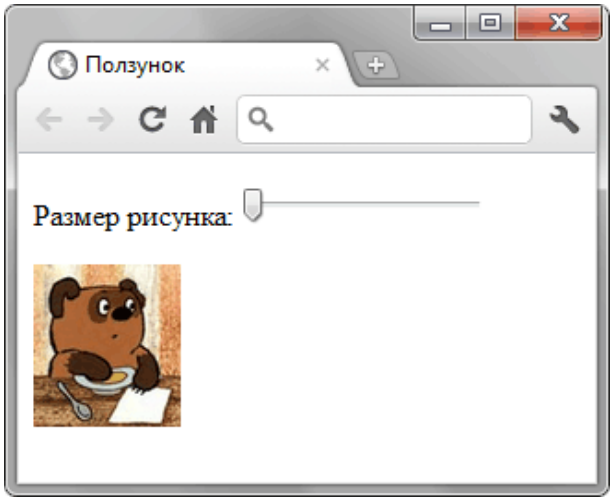


Рис. 2. Управление шириной картинки с помощью ползунка

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

9.18. 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 Opera, выводя виджет для выбора любой даты (рис. 1).

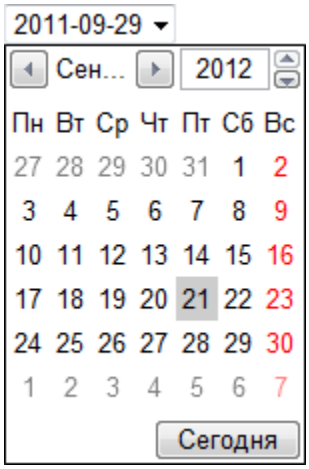


Рис. 1. Календарь в браузере Opera

Браузер Chrome также поддерживает календарь, но делает это весьма скупо (рис. 2). По сути вы только можете прокручивать дату или вводить её как текст.

Рис. 2. Календарь в Chrome

В примере 1 показано создание календаря для выбора произвольной даты.

```
Пример 1. Календарь
<!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 вам требуется сузить диапазон ввода до ±3 дней от даты 01.06.2012, то код запишется, как показано в примере 2.

```
Пример 2. Ограничение даты
<!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 (рис. 3).

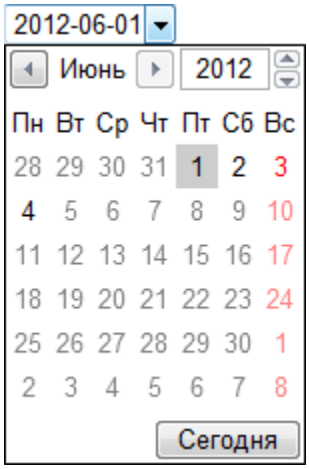


Рис. 3. Календарь с диапазоном ввода

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 (рис. 4).

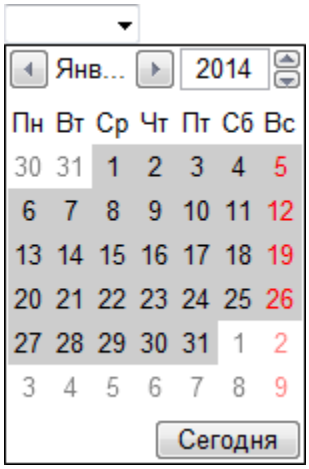


Рис. 4. Выбор месяца в Opera

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

Похожим образом выглядит и виджет для выбора недели (рис. 5), 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.

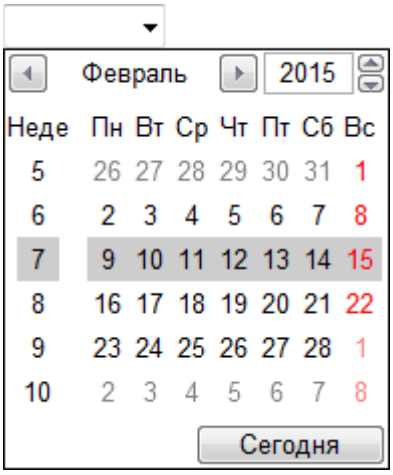


Рис. 5. Выбор недели в Opera

В примере 3 показано создание поля для ввода месяца.

```
Пример 3. Выбор 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>
```

9.19. 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>
```

Как и в случае с календарём пока поддержка этого поля имеется только в браузерах Opera (рис. 1) и Chrome (рис. 2). Атрибуты те же, что и для календаря.

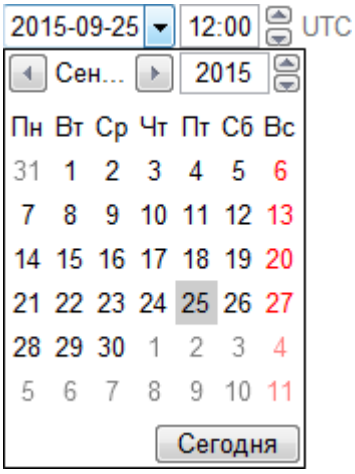


Рис. 1. Вид поля для выбора даты и времени в Opera

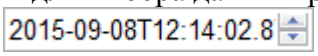


Рис. 2. Ввод даты и времени в Chrome

Opera предоставляет удобный виджет сочетающий календарь и ввод времени, Chrome же ограничивается только полем, в котором можно прокручивать секунды или вводить текст вручную. Данные на сервер по умолчанию пересылаются в виде ГГГГ-ММ-ДДТчч: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. Более того, при их наличии Chrome выводит сообщение «Недопустимые данные» и не отправляет введённый текст.

В примере 1 показано создание поля для ввода даты и времени.

```
Пример 1. Дата и время
<!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">, это поле также работает только в Opera и Chrome.

В примере 2 показано создание поля для ввода времени в указанном диапазоне.

```
Пример 2. Ввод времени
<!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.

9.20. 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. На платформах, где поисковая форма имеет свой собственный уникальный вид, легко придать полю аналогичное оформлении. В большинстве браузеров поле для поиска по своему виду никак не отличается от текстового поля, за исключением Chrome, Safari и IE10, которые добавляют небольшой крестик в правой части для быстрой очистки введённого текста (рис. 1).

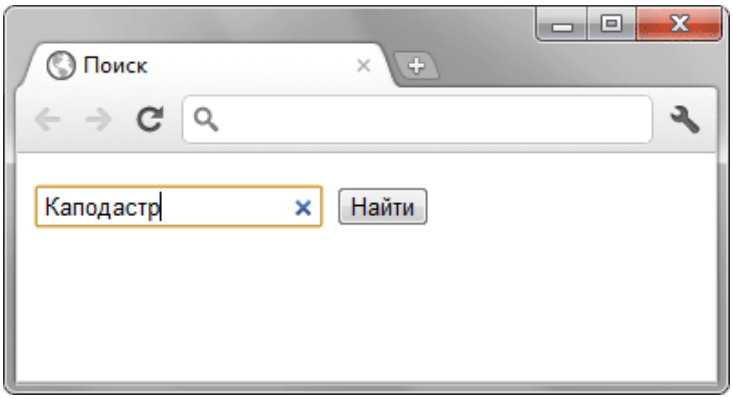


Рис. 1. Вид поля для поиска в Chrome

В примере 1 показано создание формы с полем для поиска.

```
Пример 1. Поле 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>
```

9.21. 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.

Создание поля для телефона показано в примере 1. Требуется ввести телефон в указанном формате, иначе браузер выведет сообщение об ошибке и не станет отправлять форму, пока поле не будет заполнено корректно.

```
Пример 1. Поле 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>
```

Результат примера при вводе неправильного телефона показан на рис. 1.

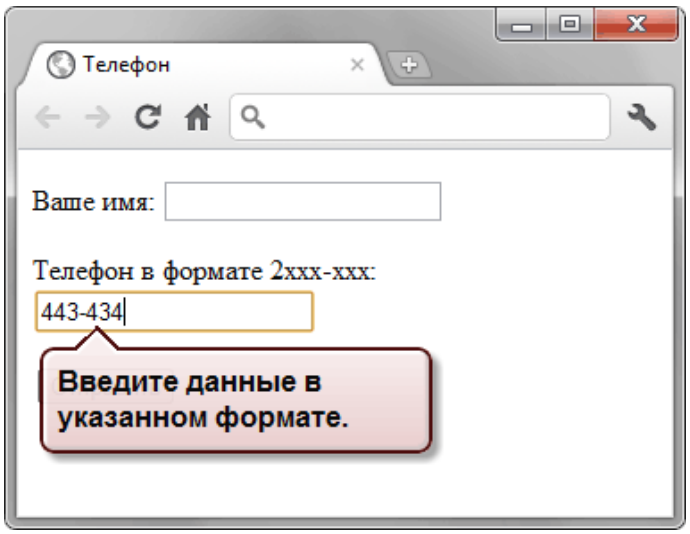


Рис. 1. Ввод телефона в браузере Chrome

Браузеры IE и Safari не поддерживают этот вид поля и соответственно не делают проверку на шаблон телефона.

9.22. Группирование элементов форм

При создании сложной формы можно группировать некоторые элементы форм между собой, такое группирование удобно для пользователя и позволяет визуально отделить один логический блок от другого. Для этой цели применяется элемент <fieldset>, который создаёт рамку в форме с заголовком или без него. Структура кода следующая.

```
<fieldset>
  <legend>Заголовок</legend>
  ...
</fieldset>
```

Элемент <legend> не обязателен, но если присутствует, должен идти сразу же после тега <fieldset>. Другие теги или текст перед <legend> недопустимы. Внутри <legend> можно использовать теги форматирования вроде <b>, <i>, <sup>, <sub> и др.

В примере 1 показано использование группирование на практике.

```
Пример 1. Создание группы
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Группа</title>
 </head>
 <body>
  <form>
   <fieldset>
    <legend>Вход на сайт</legend>
    <p>Логин: <input name="login"></p>
    <p>Пароль: <input type="password" name="pass"></p>
    <p><input type="submit" value="Вход"></p>
   </fieldset>
  </form>
 </body>
</html>
```

Результат данного примера в браузере Chrome показан на рис. 1.

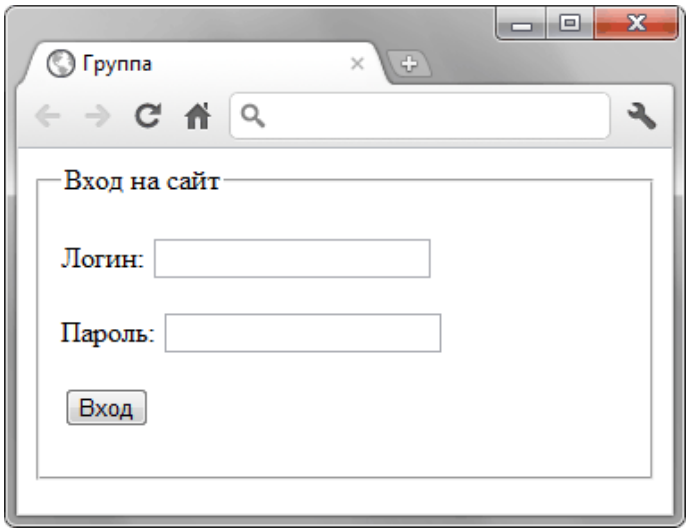


Рис. 1. Вид группы в форме

Вид рамки в браузерах в целом одинаков — заголовок располагается в левом верхнем углу, встраиваясь прямо в рамку. Однако есть и мелкие различия, к примеру, в IE рамка имеет скруглённые углы, а текст плотнее прилегает к границе (рис. 2).

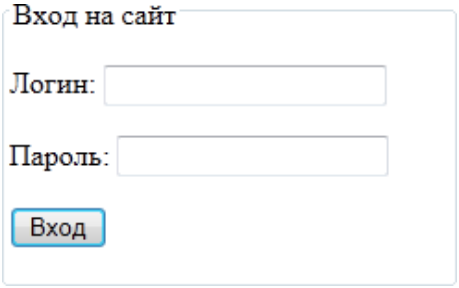


Рис. 2. Вид рамки в Internet Explorer 9

9.23. 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. В примере 1 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.

```
Пример 1. Использование 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>
```

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

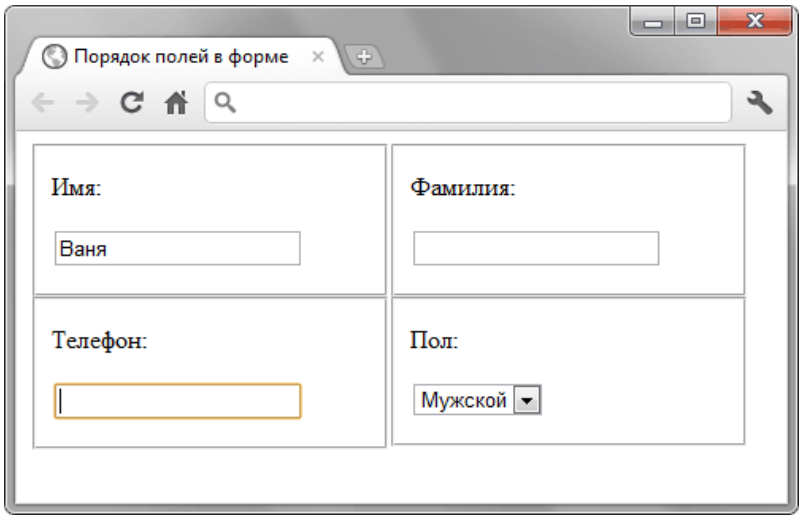


Рис. 1. Порядок ввода данных

9.24. 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).

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 рис. 1 представлены разные элементы форм в заблокированном состоянии.

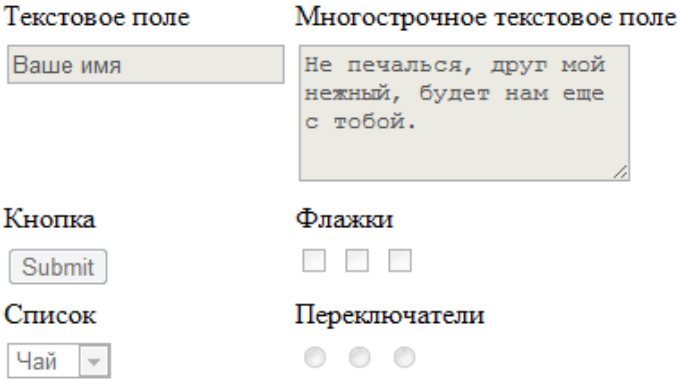


Рис. 1. Вид заблокированных элементов в Chrome

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. В примере 1 показано применение скриптов для изменения блокировки кнопки.

```
Пример 1. Блокирования поля
<!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>
```

Результат данного примера в Safari показан ниже (рис. 2).

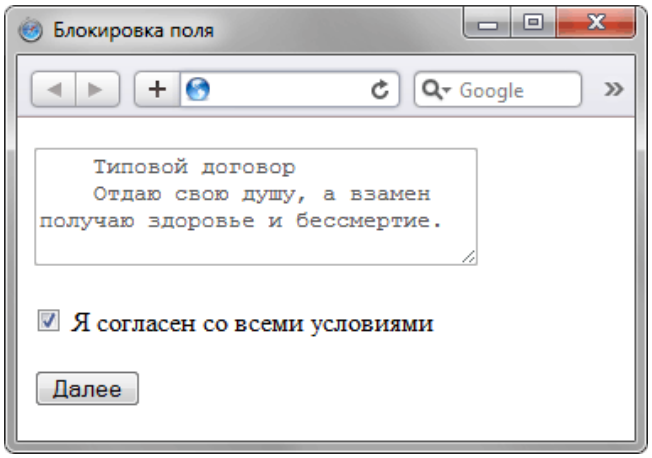


Рис. 2. Заблокированное текстовое поле

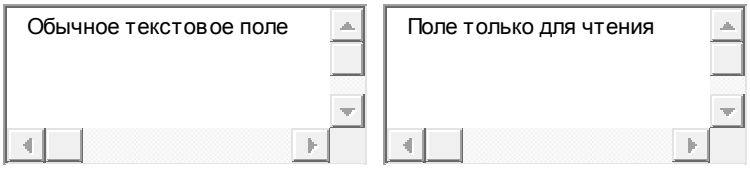
В данном примере применяется блокирование кнопки, но оно снимается, как только пользователь поставит флажок возле текста «Я согласен со всеми условиями».

Поле 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».



В примере 2 показано создание поля для чтения.

```
Пример 2. Использование readonly
<!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>
```

9.25. 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.ru вы можете 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.

В примере 1 показано создание формы авторизации с автофокусом.

```
Пример 1. Использование автофокуса
<!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>
    <p><input type="password" name="pass"></p>
    <p><input type="submit" value="Вход"></p>
   </fieldset>
  </form>
 </body>
</html>
```

Результат примера показан на рис. 1. Браузер Chrome фокус полей формы выделяет с помощью оранжевой рамки.

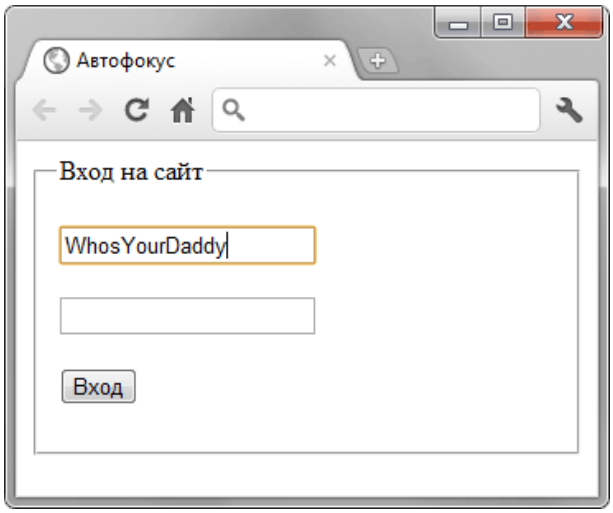


Рис. 1. Автофокус в Chrome

9.26. 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.

В примере 1 показано добавление подсказки к полям формы для создания авторизации.

```
Пример 1. Подсказывающий текст
<!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>
```

Результат примера в браузере Chrome показан на рис. 1.

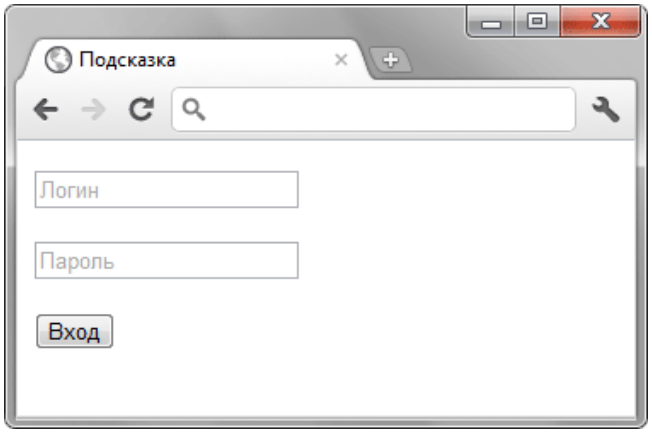


Рис. 1. Вид полей с подсказывающим текстом

9.27. 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.

Табл. 1. Регулярные выражения

Выражение	Описание
\d [0-9]	Одна цифра от 0 до 9.
\D [^0-9]	Любой символ кроме цифры.
\s	Пробел.
[A-Z]	Только заглавная латинская буква.

[A-Za-z]	Только латинская буква в любом регистре.
[А-Яа-яЁё]	Только русская буква в любом регистре.
[A-Za-zA-Яа-яЁё]	Любая буква русского и латинского алфавита.
[0-9]{3}	Три цифры.
[A-Za-z]{6,}	Не менее шести латинских букв.
[0-9]{,3}	Не более трёх цифр.
[0-9]{5,10}	От пяти до десяти цифр.
^[a-zA-Z]+$	Любое слово на латинице.
^[А-Яа-яЁё\s]+$	Любое слово на русском включая пробелы.
^[0-9]+$	Любое число.

В примере 1 показано, как сделать ввод IP-адреса

```
Пример 1. Шаблон 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. Вид и содержание этого сообщения зависит от браузера, так, в Chrome оно будет иметь вид, как показано на рис. 1.

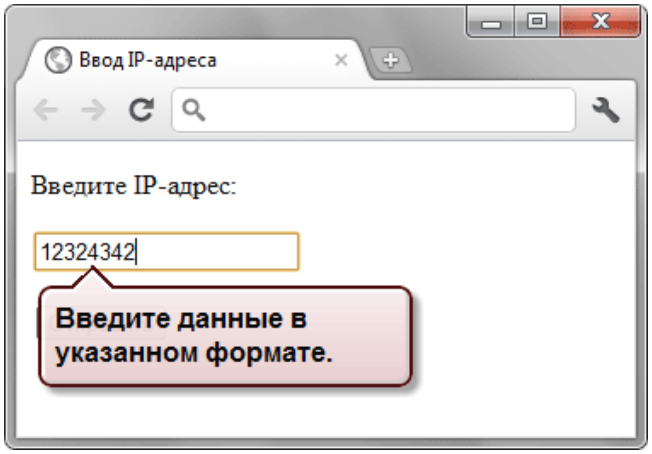


Рис. 1. Ввод неверных данных

9.28. Защита от некорректного ввода

Защитой от некорректноого ввода 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, 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.

3. Есть ряд людей, которые воспринимают инструкции как вызов и стараются поступить наоборот.

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.

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, как показано в примере 1.

```
Пример 1. Атрибут required
<!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, например Chrome выводит всплывающую подсказку, как показано на рис. 1.

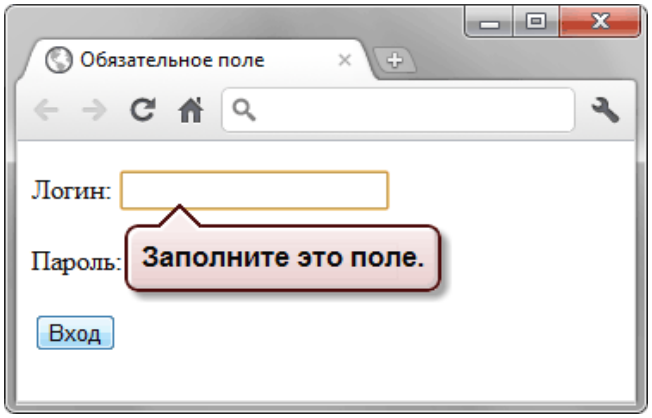


Рис. 1. Обязательное поле не заполнено

Корректность данных

Исходно имеется два поля, в котором вводимые пользователем данные проверяются автоматически. Это веб-адрес и адрес электронной почты. Браузер Chrome также проверяет на корректность поле с календарными данными, но только потому, что у него не предусмотрен интерфейс выбора календаря щелчком мыши. Для этих элементов характерны следующие правила.

- 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.

У браузеров несколько различается политика по проверке данных пользователя. К примеру, Opera подставляет протокол http:// перед введённым текстом автоматически, тогда как другие

браузеры ждут его от пользователя. Chrome и Opera требуют, чтобы в почтовом адресе была точка, для Firefox она не обязательна.

В примере 2 показана форма с обязательными полями, в которой два поля проверяется браузером.

```
Пример 2. Корректность данных
<!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>
```

Opera проверяет элемент формы только при наличии атрибута name.

Что происходит в Opera при вводе неверных данных показано на рис. 2.

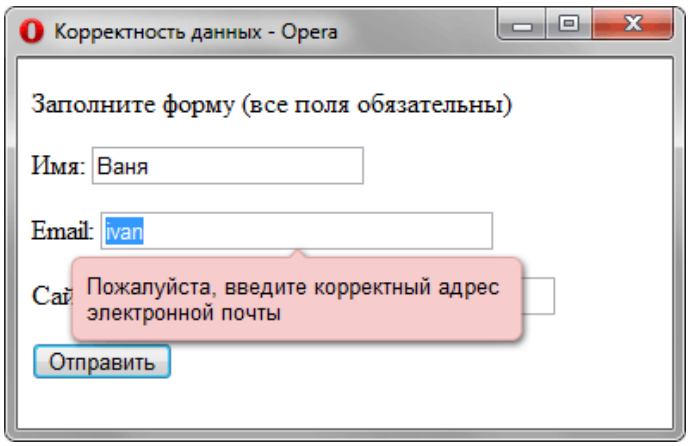


Рис. 2. Предупреждение о неправильных данных

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 России 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 табл. 1.

Табл. 1. Регулярные выражения

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

В примере 3 просят ввести шестнадцатеричное значение цвета (#ffcc00) и если оно не лежит в этом диапазоне, браузер выводит сообщение об ошибке.

```
Пример 3. Шаблон ввода
<!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>
```

На рис. 3 показано предупреждение в браузере Chrome.

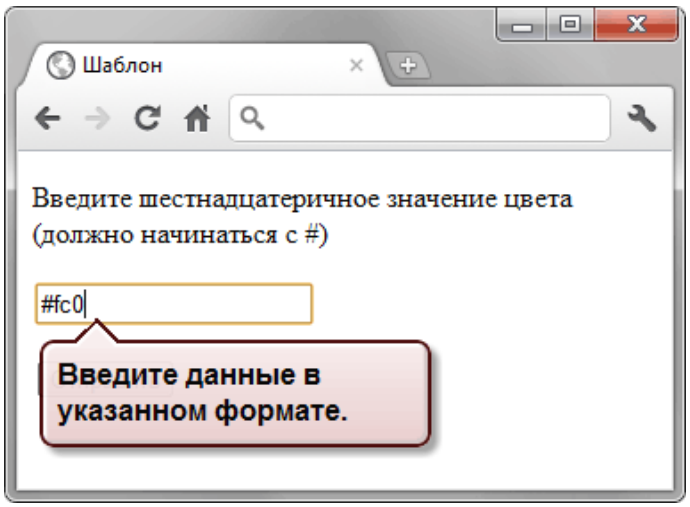


Рис. 3. Введённые данные не соответствуют шаблону

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>. В примере 4 показано использование этого атрибута.

```
Пример 4. Отмена 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 4 будет иметь следующий вид.

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

Список использованных источников

1. Муссиано Ч., Кеннеди Б. HTML и XHTML. Подробное руководство. - М.: Символ-Плюс, 2008.

2. Мейер Э. CSS. Каскадные таблицы стилей. Подробное руководство. - СПб.: Символ-Плюс, 2008.

3. Агулар Р. HTML и CSS. Основа любого сайта. - М.: Эксмо, 2010.

4. Кастро Э. HTML и CSS для создания Web-страниц. - М.: НТ Пресс, 2006.

5. Комолова Н., Яковлева Е. HTML. Самоучитель. - СПб.: Питер, 2011.

6. Тиге Д. DHTML и CSS для Internet. - М.: НТ Пресс, 2007.

7. Дакетт Д. Основы веб-программирования с использованием HTML, XHTML и CSS. -М.: Эксмо, 2010.

8. Рева О. Использование HTML, JavaScript и CSS. Руководство Web-дизайнера. - М.: Эксмо, 2008.

9. Айзекс С. Dynamic HTML. Секреты создания интерактивных Web – страниц. - СПб.: BHV, 2001.

Учебное издание
Фрикк Валерий Сергеевич
Лысанов Денис Михайлович

Печатается в авторской редакции

Подписано в печать 26.03.2015 г.
Формат 60x84/16. Бумага офсетная. Печать ризографическая
Уч.-изд.л. 3,5 Усл.-печ.л. 3,5 Тираж 50 экз.
Заказ №466
Издательско-полиграфический центр
Набережночелнинского института
Казанского (Приволжского) федерального университета

423810, г. Набережные Челны, Новый город, проспект Мира, 68/19
тел./факс (8552) 39-65-99 e-mail: ic-nchi-kpfu@mail.ru