

**Формы**

Формы предоставляют интерфейс, дающий возможность пользователям осуществлять взаимодействие с сайтом.

В большинстве случаев они применяются для сбора данных, либо с целью дальнейшего использования.

Формы имеют широкий диапазон сфер применения, от таких простых, как поиск, подписка на почтовую рассылку, гостевые книги и обзоры, до сложных, таких, как системы электронной коммерции.

Формы принимают входные данные при помощи элементов управления, таких, как кнопки, текстовые поля или прокручиваемые меню.

Элементы управления размещаются на странице. Эти элементы служат просто интерфейсом для получения информации от пользователя и не ведут реальной обработки данных.

Обработку производят серверные приложения, взаимодействующие с формами, такие, как CGI-скрипты, ASP, ASP.NET, ColdFusion, PHP или Java-сервлеты.

# Элементы форм

- `form` Создание формы
- `input` Создание различных элементов управления
- `button` Обычная кнопка ввода
- `textarea` Элемент управления для ввода нескольких строк текста
- `select` Меню с несколькими пунктами или прокручиваемый список
- `option` Пункт в элементе управления `select`
- `optgroup` Определяет группу пунктов
- `label` Надпись. Связывает информацию с элементами управления
- `fieldset` Группирует связанные между собой элементы управления и надписи
- `legend` Создает заголовок для группы полей (`fieldset`)

# Базовый элемент form

**<form> ... </form>**

Элемент form используется для того, чтобы обозначить область на Web-странице, которая будет играть роль формы.

- accept=«список типов данных»
- accept-charset=«список наборов символов»
- action=«URL» (Обязательный)
- enctype=«тип данных»
- method="get|post"
- name="текст" (В XHTML - устаревший, используйте атрибут id)
- target=«имя»

- Форма может содержать любые Web-материалы (текст, изображения, таблицы и т. д.), но ее главная функция - быть контейнером для ряда элементов управления.
- Форма также содержит атрибуты, необходимые для взаимодействия с программой, обрабатывающей данные.
- В одном документе может содержаться несколько форм, но они не могут быть вложенными, так что следите, чтобы формы не перекрывались.
- Когда пользователь заполнит форму и нажмет кнопку «Submit» (Отправить), браузер принимает эту информацию, организует ее в пары **имя/значение**, кодирует ее для передачи, а затем отправляет на сервер.

# Атрибут action

- Атрибут action элемента form указывает URL программы, используемой для обработки формы.
- Пример:
- **`<form action="/cgi-bin/guestbook.pl" method="get">`**

# Атрибут `method`

- В атрибуте `method` указывается один из двух методов отправки данных формы на сервер – **`get`** или **`post`**.
- Информация формы, как правило, передается в виде серии переменных и соответствующих значений, разделенных символом амперсанда **`variable1=content1&variable2=content2&variable3=content3`**
- Имена переменных определяются атрибутами **`name`** элементов управления формы. Значениями переменных являются данные, которые вводит пользователь.



При использовании метода **get** браузер передает данные формы за один раз, в виде части **URL** (прикрепляя их в конец **URL** и отделяя от него знаком вопроса).

При использовании данного метода информация в нашем примере будет передаваться следующим образом:

```
get http:// www.domainname.com/cgi-bin/  
guestbook.pl?name=Joseph&nickname=Josie
```

Метод **post** передает введенную в форму информацию отдельно от URL, по существу, в сообщении, состоящем из двух частей. Первая часть сообщения - это просто специальный заголовок, отправляемый браузером с каждым запросом.

Этот заголовок содержит **URL**, взятый из элемента **form**, в сочетании с объявлением, что данный запрос является post-запросом, а также несколько заголовков, которые мы здесь не рассматриваем. Затем идут реальные данные формы. Когда сервер обнаруживает слово **post** в начале сообщения, он настраивается на прием данных. При использовании метода **post** информация в нашем примере будет передаваться следующим образом:

```
post  
http://www.domainname.com/cgi-bin/guestbook.pl HTTP1.0 . . . [дополнительные заголовки] name=Joseph&nickname=Josie
```

Выбор используемого метода зависит от требований сервера. Как правило, если форма короткая и имеет несколько полей небольшого размера, лучше использовать метод **get**.

Данные больших и сложных форм лучше пересылать с помощью метода **post**. Если имеют значение вопросы безопасности (например, при использовании элемента `input type="password"`), используйте метод `post`, поскольку он позволяет зашифровать данные, а не пересылать их открытым текстом, прикрепленными к URL.

# Элемент input

- **Атрибуты**
- on focus, onblur, onselect, onchange
- alt=«текст»
- accept="MIM-тип"
- accesskey="символ"
- checked="checked"
- disabled="disabled"
- maxlength="число"
- name="текст" (обязателен для всех типов, кроме submit и reset)
- readonly="readonly"
- size="число"
- src="URL"
- tabindex="число"
- type="text|password|checkbox|radio|submit|reset|file|hidden| image|button"
- value="текст"

# Поле для ввода текста

- Самой простой тип элемента управления в форме - это поле для ввода текста (`type="text"` **type = «password»** ). Это тип элемента `input`, принятый по умолчанию.
- **<input type = «text»>**
- **Атрибуты**
- `disabled="disabled"`
- `maxlength="число"`
- `name="текст"` (Обязательный)
- `readonly="readonly"`
- `size="число"`
- `value="текст"`

# Скрытое поле (type=«hidden»)

- Скрытое поле (type="hidden") представляет собой элемент управления, не отображаемый в браузере, но передаваемый программе обработки при отправке данных.

**input type = «hidden»**

## Атрибуты

accesskey="символ"

tabindex="число"

name="текст" (Обязательный)

value="текст" (Обязательный)

Скрытые элементы управления полезны для отправки информации, которую следует обрабатывать вместе с пользовательскими данными, например названий, которые скрипт использует для сортировки форм. Пользователи не могут видеть или изменять скрытые элементы управления. Для правильного функционирования некоторых скриптов бывает необходимо добавить на форму специфические скрытые поля.

```
<p>This is a hidden element</p>
```

```
<input type="hidden" name="extra_info" value="important" />
```

# Флажок (type=«checkbox»)

- Флажки (type="checkbox") напоминают выключатели, которые могут включаться и выключаться пользователем.
- При отправке данных формы, на сервер отправляются только значения установленных (включенных) флажков
- **Атрибуты**
- *Базовые* (id, class, style, title), *Интернационализация*, *События*
- Фокус(accesskey, tabindex, onfocus, onblur)
- align="left|right|top|texttop|middle|absmiddle|baseline|bottom| absbottom"
- checked="checked"
- disabled="disabled"
- name="текст" (Обязательный)
- readonly="readonly"
- value="текст" (Обязательный)
- Флажки можно использовать для передачи конкретных пар имя/значение на сервер, если флажок установлен. По умолчанию флажок не установлен; чтобы сделать его установленным при загрузке страницы, просто добавьте атрибут checked к соответствующему элементу input.
- Когда флажок установлен, соответствующее значение передается вместе с данными формы программе обработки на сервере. Значения для неустановленных флажков не пересылаются.

# Переключатель

- Переключатели (радиокнопки) (type="radio") - это еще один вид кнопок, который пользователь может включать и выключать. В отличие от флажков в группе переключателей, имеющих **одно имя**, включенным может быть только один переключатель, а остальные при этом отключаются. Такой вид элементов управления применяется для взаимоисключающих вариантов выбора.
- *Базовые* (id, class, style, title), *Интернационализация*, *События*
- Фокус(accesskey, tabindex, onfocus, onblur)
- checked="checked"
- disabled^"disabled"
- name="текст" (Обязательный)
- readonly="readonly"
- value="текст" (Обязательный)
- **<p>Какую из приведенных операционных систем вы используете?</p>**
- **<input type="radio" name="os" value="WinXP"> Windows XP**
- **<input type="radio" name="os" value="Linux"> Linux**
- **<input type="radio" name="os" value="OSX" checked="checked" /> Macintosh OSX**
- **<input type="radio" name="os" value="DOS"> DOS**



# Кнопки Submit и Reset

- Кнопки Submit (Отправка), которые используются для отправки данных формы агенту, выполняющему обработку, добавляются при помощи элемента input типа submit.
- Кнопка Reset (Сброс) восстанавливает исходные значения для всех элементов управления формы, и ее можно разместить при помощи элемента input типа reset.
- **input type = «submit»**
- Создает кнопку отправки данных. При нажатии этой кнопки данные формы немедленно отправляются на сервер для обработки.
- **Атрибуты**
- disabled="disabled"
- name="текст"
- value="текст"
- **input type = «reset»**
- Создается кнопка сброса, которая очищает содержимое элементов формы (или задает для них значения по умолчанию)
- **Атрибуты**
- disabled="disabled"
- value="текст"

# Пользовательская кнопка

- Авторы могут создать настраиваемую кнопку для управления клиентскими скриптами (JavaScript), для чего нужно задать для элемента `input` тип `button`.
- **`input type = «button»`**
- **Атрибуты**
- `align = "left|right|top|texttop| middle|absmiddle|baseline|bottom| absbottom"`
- `disabled="disabled"`
- `name="текст"`
- `value="текст"`
- Эта кнопка (`type = "button"`) не имеет заранее заданной функции, и она является инструментом общего применения, который можно настраивать при помощи языка скриптов.

# Графическая кнопка

Если вы хотите использовать для кнопки отправки свое изображение, применяйте элемент **input** типа **image**.

## Атрибуты

`align="top|middle|bottom"`

`alt=«текст»`

`disabled="disabled"`

`name="текст"` (Обязательный)

`src="URL"`

Вы можете заменить кнопку отправки выбранным графическим изображением. Щелчок мышью по изображению отправляет данные формы на сервер, добавляя к ним координаты курсора мыши.

Для графических кнопок рекомендуется также использовать альтернативный текст (при помощи атрибута `alt`).

```
<input type="image" src="graphics/sendme.gif" alt="Send me">
```

# Выбор файла

Элемент **input** типа **file** позволяет пользователям отправлять внешние файлы при отправке формы. Этот элемент включает в себя текстовое поле и кнопку Browse (Обзор), которая позволяет выбирать файлы на локальном компьютере.

## Атрибуты

accept="MIME-тип"

disabled="disabled"

maxlength="число"

name="текст" (Обязательный)

readonly="readonly"

size=«число»

value="текст"

При использовании элемента **input** типа **file** вы должны указать атрибут **enctype="multipart/form-data"** в элементе **form**.

```
<form enctype="multipart/form-data">
```

```
<p>Send this file with my form information:</p>
```

```
<input type="file" size="28">
```

```
</form>
```

# Многострочные текстовые поля

Элемент `textarea` создает многострочное поле для ввода текста с возможностью прокрутки, которое позволяет пользователю вводить большие текстовые записи.

```
<textarea> ... </textarea>
```

## Атрибуты

`cols="число"` (Обязательный)

`disabled="disabled"`

`name="текст"` (Обязательный)

`readonly="readonly"`

`rows="число"` (Обязательный)

```
<p>What did you dream last night?</p>
```

```
<textarea name="dream" rows="4" cols="45">Tell us your  
dream in 100 words or less</textarea>
```

Укажите количество строк текста, которое должно отображаться в элементе при помощи атрибута **rows** . В атрибуте **cols** указывается ширина элемента (в символах).

Полосы прокрутки появляются, если пользователь вводит больше текста, чем помещается в выделенном месте.

# Элемент `select`

Элемент **`select`** создает меню, которое более компактно, чем группа флажков или переключателей.

Меню отображается либо как раскрывающееся меню, либо как прокручиваемый список вариантов. Зависит это от метода указания размера.

Элемент `select` служит **контейнером** для любого числа элементов **`option`**. Он также может содержать один или несколько элементов **`optgroup`**, которые используются для создания логической группы элементов **`option`**.

## **`select`**

```
<select> ... </select>
```

### **Атрибуты**

»

```
multiple="multiple"
```

```
name="текст" (Обязательный)
```

```
size="число"
```

```
tabindex="число"
```

## **`option`**

```
<option> ... </option>
```

### **Атрибуты**

```
disabled="disabled"
```

```
label="текст"
```

```
selected="selected"
```

```
value="Текст"
```

## **`optgroup`**

```
<optgroup> ... </optgroup>
```

### **Атрибуты**

```
disabled="disabled"
```

```
label="текст" (Обязательный)
```

# Раскрывающиеся меню

Элемент `select` отображается в виде раскрывающегося меню, если размер не указан (по умолчанию) или если используется атрибут `size=«1»`.

Из раскрывающегося меню можно выбрать только один пункт.

Добавление атрибута `multiple` превращает меню в прокручиваемый список.

```
<p>What is your favorite ice cream flavor?</p>
```

```
<select name="ice-cream">
```

```
<option>Rocky Road</option>
```

```
<option>Mint Chocolate Chip</option>
```

```
<option>Pistachio</option>
```

```
<option selected="selected">Vanilla</option>
```

```
<option>Chocolate</option>
```

```
<option value="swirl">Fudge Ripple</option>
```

```
<option label="Praline Pecan">Super-duper Praline Pecan Smashup</option>
```

```
<option>Bubblegum</option>
```

```
</select>
```

По умолчанию при загрузке формы отображается первый элемент **option**. Чтобы сделать другой элемент **option** элементом по умолчанию, используйте атрибут **selected** этого элемента.

**Текст** внутри каждого элемента **option** представляет собой значение, которое посылается на сервер. Если вы хотите посылать значение, отличное от отображаемого, укажите его в атрибуте **value** элемента **option**.

Если указывается атрибут **label**, его значение отображается вместо содержимого элемента **option**.



# Прокручиваемые меню

Чтобы меню отображалось в виде прокручиваемого списка, просто укажите количество строк текста, которое вы хотите видеть отображаемым в списке, при помощи атрибута `size` или добавьте к элементу `select` атрибут `multiple`.

Атрибут `multiple` позволяет пользователям выбирать из списка сразу несколько вариантов.

```
<p>What are your favorite ice cream flavors?</p>
<select name="ice_cream" size="6" multiple="multiple">
<option>Rocky Road</option>
<option>Mint Chocolate Chip</option>
<option>Pistachio</option>
<option selected="selected">Vanilla</option>
<option selected="selected">Chocolate</option>
<option value="swirl">Fudge Ripple</option>
<option>Super-duper Praline Pecan Smashup</option>
<option>Bubbligum</option>
</select>
```

В данном примере также используется атрибут **selected** для предварительного выбора вариантов, и атрибут **value** - для указания значения, отличного от отображаемого текста.

# Группы вариантов

Логические группы вариантов можно организовать при помощи элемента **optgroup**. Этот элемент браузеры могут использовать для отображения иерархических каскадных меню. Значение обязательного атрибута **label** отображается в качестве заголовка для последующих вариантов.

Содержимым элемента `optgroup` является один или несколько элементов `option`.

Элемент `optgroup` не может содержать других элементов `optgroup`.

```
<p>What are your favorite ice cream flavors?</p>
<select name="ice_cream" size="6" multiple="multiple">
  <optgroup label="traditional">
    <option>Vanilla</option>
    <option>Chocolate</option>
    <option>Mint Chocolate Chip</option>
    <option>Pistachio</option>
    <option>Fudge Ripple</option>
  </optgroup>
  <optgroup label="specialty">
    <option>Inside-out Rocky Road</option>
    <option>Super-duper Praline Pecan Smashup</option>
    <option>Bubblemum</option>
  </optgroup>
</select>
```

Элемент `button` определяет пользовательскую «кнопку», которая работает примерно так же, как тег `input`. Элемент `button` может содержать изображения (но не графические карты ссылок), а также любые другие материалы, за исключением элементов `a`, `form` и элементов управления формы.

Кнопки могут отображаться в «трехмерном» виде с тенями, имитирующими движение вверх-вниз (как кнопки отправки и сброса), в отличие от элемента `input` типа `image`, который отображается в виде простого плоского изображения.

## **button**

```
<button> ... </button>
```

### **Атрибуты**

```
disabled="disabled"
```

```
name="текст"
```

```
value=" текст"
```

```
type="submit|reset|button"
```

В следующем примере показано использование элементов `button` вместо кнопок отправки (**Submit**) и сброса (**Reset**). Обратите внимание, что элементы **button** включают как изображение, так и текст.

```
<button type="submit" name = "submit">  
Finished. Ready for step two.</button>
```

```
<button type="reset" name = " reset "><img src = " down . gif" alt="down  
icon"> Попробуйте снова.</button>
```

Обратите внимание, что текст, отображаемый на кнопке, (например, «Попробуйте снова»), не обязательно должен совпадать с именем переменной (`reset`).

# Средства обеспечения доступности

В Рекомендации **HTML 4.01** появился ряд элементов и атрибутов форм, которые помогают обеспечить доступность. Некоторые из них предлагают усовершенствованные методы группировки и создания надписей в структуре и материалах формы. Другие предоставляют клавиатурные альтернативы для выбора и активизации полей форм (в частности, передача фокуса).

# Надписи

Элемент label (надпись) используется для связывания с полем формы какого-либо описывающего его текста. Это дает важную подсказку пользователям, которые обращаются к форме с использованием речевых браузеров. Каждый элемент label связывается только с одним элементом управления формы.

label

```
<label> ... </label>
```

Атрибуты

Существует два способа связывания надписи с элементом управления формы.

- Один из них - вложение элемента управления и связанного с ним описания внутрь элемента label.

```
<form action="/cgi-bin/guestbook.pl" method="GET">  
<label> Учетная запись: <input type="text" name="login"></label>  
<label> Пароль: <input type="password" name="password"></label>  
<input type="submit">  
</form>
```

Другой метод - это связывание элемента label со значением id поля формы. Атрибут for указывает, для какого элемента управления предназначена данная надпись. Данный метод полезен для тех полей формы, которые находятся рядом с их описаниями, например когда они охватывают несколько разных ячеек таблицы.

```
<form action="/cgi-bin/guestbook.pl" method="GET">  
<label for="log">Login account:</label>  
<input type="text" name="log" id="log">  
<label for="pswd">Password:</label>  
<input type="password" name="pswd" id="pswd">  
<input type="submit">  
</form>
```

# Атрибуты **id** и **name** в элементах формы

Атрибуты **id** и **name**, применяемые в элементах управления формы (таких, как `input`, `select` и т. п.), имеют разные и четко отличающиеся функции.

Значение атрибута **name** пересылается программе обработки при отправке данных формы.

Атрибут **id** используется для того, чтобы присвоить элементу уникальный идентификатор, на который могут ссылаться правила таблиц стилей, скрипты или элемент `label`.

Атрибут **id** не может использоваться вместо атрибута **name**, поскольку его значение не будет передаваться в составе данных формы.

Это не относится к самому элементу **form**. Для элемента **form** атрибуты **id** и **name** выполняют сходную функцию - присвоение форме уникального имени. Использование того или иного варианта зависит от языка разметки.

В HTML для присвоения форме имени, чтобы к ней могли обращаться скрипты, используется атрибут `name`.

В XHTML может использоваться только атрибут **id**.

# Элементы `fieldset` и `legend`

Элемент **fieldset** используется для создания логической группы элементов управления формы. Элемент `fieldset` может содержать элемент **legend** - описание входящих в группу полей, которое может использоваться невидимыми браузерами.

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

## Атрибуты

`accesskey`="символ"

`align`="top | bottom | left | right" (Устаревший)

Следующая форма была разделена на группы при помощи элементов `fieldset`, к которым были добавлены элементы `legend` в качестве описания.

```
<form>
```

### **<fieldset>**

```
<legend> Информация о покупателе</legend>
```

```
<label>Полное имя <input type="text" name="name"></label>
```

```
<label>Адрес email <input type="text" name="email"></label>
```

```
<label> Штат <input type="text" name="state"></label> </fieldset>
```

### **<fieldset>**

```
<legend> Подписка на список рассылки </legend>
```

```
<label>Добавьте меня в ваш список рассылки <input type="radio" name="list" value="yes" checked="checked"></label>
```

```
<label>Нет, спасибо <input name="list" value="no"></label>
```

### **</fieldset>**

```
</form>
```

# Атрибуты **accesskey** и **tabindex**

Атрибуты перемещают фокус на элемент формы без использования традиционного метода, связанного с наведением курсора мыши и щелчком. Любой пользователь может применять эти методы для перемещения по форме.

Атрибут **accesskey** указывает символ, который можно использовать как клавиатурный «ярлык» элемента. На практике может потребоваться ввести комбинацию клавиш, например Alt-клавиша (Windows) или Command-клавиша (Macintosh).

Атрибут **accesskey** можно применять с элементами управления **button**, **input**, **label**, **legend** и **textarea**.

```
<b>A</b>ddress<input type="text" name="address" accesskey="l" >
```

Еще один метод перемещения фокуса по полям формы - это нажатие клавиши **Tab**, которое переводит фокус от одного поля к другому. По умолчанию браузеры, которые поддерживают данную функциональность, будут переводить фокус в том порядке, в каком поля находятся в документе. Если вы хотите изменить порядок перевода фокуса, не изменяя порядка расположения элементов разметке, используйте атрибут **tabindex**. Этот атрибут может использоваться с элементами **button**, **input**, **select** и **textarea**. Элементы с нулевым значением **tabindex** получают фокус после элементов с положительными значениями. Элементы с отрицательными значениями **tabindex** не получают фокус при нажатии клавиши **Tab**. Отключенные элементы также не могут получать фокус.



Address `<input type="text"  
name="address" tabindex="1" />`

Zip code `<input type="text" name="zip"  
tabindex="3" />`

Phone number `<input type="text"  
name="phone" tabindex="2" />`

Хотя атрибут **tabindex** изначально создавался как метод обеспечения доступности, многие эксперты по доступности не рекомендуют его использовать. В большинстве случаев порядок расположения элементов управления формы вполне логичен, и его должно быть достаточно.

## Атрибут title

Еще один атрибут, повышающий доступность полей формы (а также ссылок, изображений и других ресурсов) - это атрибут **title**. Используйте его для создания описаний полей или для указания специальных инструкций. Речевые браузеры могут произносить содержимое данного атрибута, когда поле получает фокус. Визуальные браузеры могут отображать эти данные в виде всплывающей подсказки, когда курсор мыши наводится на поле.

## Атрибуты disabled и readonly

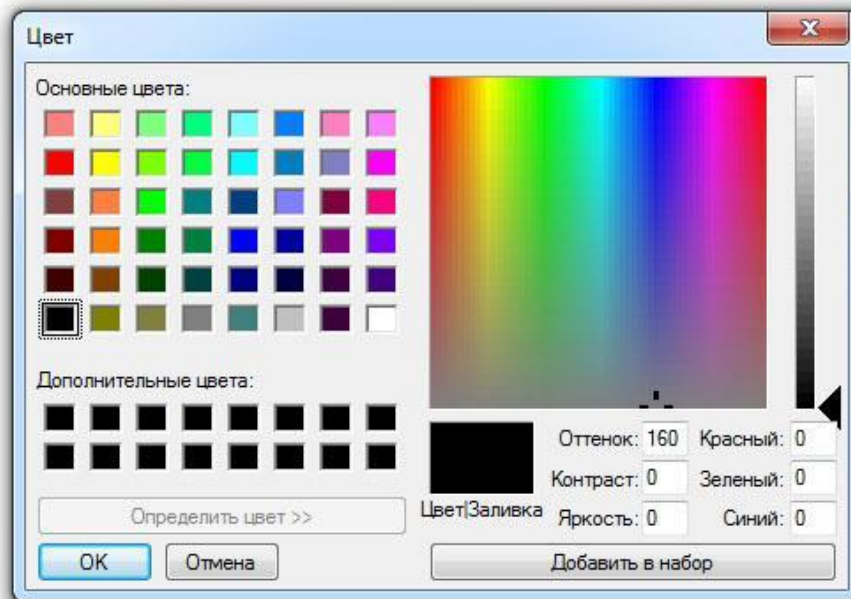
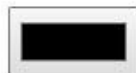
Атрибуты **disabled** и **readonly** блокируют для пользователя возможность выбрать или изменить значение в поле формы.

- Если элемент формы отключен (**disabled**), его нельзя выбрать. Визуальные браузеры могут отображать такой элемент с затенением серым цветом. Состояние отключенного элемента можно изменить только при помощи скрипта.
- Атрибут **readonly** запрещает пользователю изменять значение в поле формы (хотя поле все же можно выбирать). Это позволяет разработчикам задавать значения в элементах управления на основе других данных при помощи скрипта.

# Выбор цвета

Если необходимо отобразить диалоговое окно выбора цвета, код будет выглядеть следующим образом:

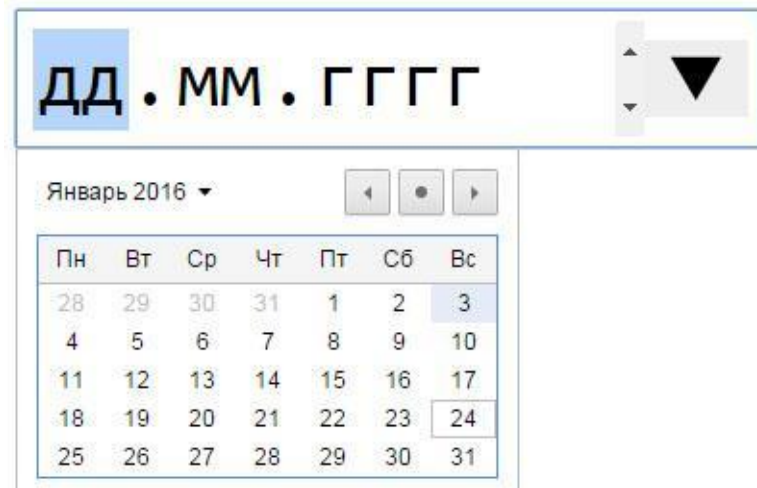
```
<input type="color" name="my_color">
```



# Календарь и время

Если нужно отобразить диалоговое окно календаря, используют значение **type="date"**.

```
<input type="date" name="new_day">
```



Если нужно отобразить календарь с локальным временем, используют

`type="datetime-local"`

```
<input type="datetime-local" name="local_time">
```



The image shows a browser's date and time picker interface. At the top, there is a text input field containing the placeholder text "01.ММ.ГГГГ -- : --" and a clear button (X). Below this is a calendar for January 2016. The calendar header shows "Январь 2016" with navigation arrows. The days of the week are listed as Пн, Вт, Ср, Чт, Пт, Сб, Вс. The dates are arranged in a grid, with the 24th of January highlighted.

Пн	Вт	Ср	Чт	Пт	Сб	Вс
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

- Если нужно задать значение времени, удобно использовать элемент `input` со значением `type="time"`

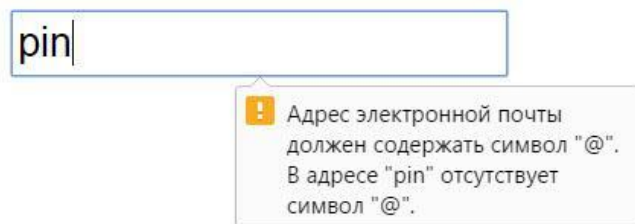
```
<input type="time" name="local_time">
```

Задайте время события:

# Поля с проверкой

Для создания поля с проверкой адреса электронной почты, можно задать тип поля "email". Атрибут **multiple** указывает, что пользователь может ввести более одного значения.

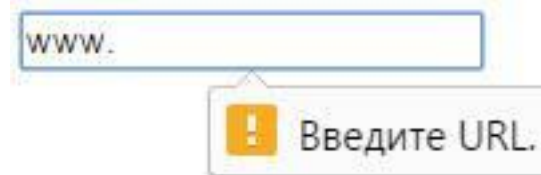
```
<input type="email" name="my_e_mail" multiple >
```



The image shows a text input field containing the text "pin". Below the field, a validation error message is displayed in a white box with a grey border. The message contains an orange warning icon and the text: "Адрес электронной почты должен содержать символ '@'. В адресе 'pin' отсутствует символ '@'."

Для проверки URL-адреса используется тип "url»:

```
<input type="url" name="site_name">
```



The image shows a text input field containing the text "www.". Below the field, a validation error message is displayed in a white box with a grey border. The message contains an orange warning icon and the text: "Введите URL."

## Поле для ввода чисел в заданном диапазоне

Если нужно отобразить, используйте тип "number». Для ограничения значений задаваемых полем чисел используются атрибуты min и max.

```
<input type="number" name="quantity" min="1" max="5">
```

### Ползунок

Для отображения ползунка, используется тип "range".

Кроме задания диапазона значений с помощью атрибутов min и max, можно также

задать шаг атрибутом "step":

```
<input type="range" min="0" max="10" step="2" value="6">
```



# Поле для ввода номера телефона

Тип `tel` был задуман как тип для ввода в форму номеров телефонов:

```
<input type="tel" name="my_number">
```

## Атрибут

**required** - указывает, что поле ввода необходимо заполнить перед отправкой формы. Если пользователь попытается отправить форму, не введя в поле никакого значения, то на экране отобразится предупреждающее сообщение. Это сообщение будет отличаться по содержанию и стилизации в зависимости от браузера и типа поля ввода.

Атрибут `required` работает со следующими значениями атрибута `type`: `text`, `search`, `url`, `tel`, `email`, `password`, `date`, `number`, `checkbox`, `radio` и `file`.

# Поле поиска

Тип поля `search` предназначен для полей поиска. Несмотря на то, что в браузерах отображение поля с типом `search` очень напоминает `text`, в полях для организации строки поиска рекомендуется использовать именно `search`.

Пример:

**<form>**

Мы ищем:

**<input type="search" value="" required placeholder="Введите строку поиска">**

**<input type="Submit">**

**</form>**

Мы ищем:

# Список predetermined вариантов <datalist> </datalist>

HTML тег <datalist> определяет список predetermined вариантов элемента <input>, которые можно выбирать при наборе в текстовом поле.

Тег <datalist> используется обычно для "автозаполнения". Изначально этот список скрыт и становится доступным при получении полем фокуса. Атрибут list тега <input /> указывает на функциональную связь между тегами <input /> и <datalist>.

- Пример.

```
< input list="computers" />  
< datalist id="computers">  
<option value="sony" />  
<option value="toshiba" />  
<option value="asus" />  
<option value="acer" />  
< /datalist>
```



-

# Мера в пределах диапазона

Тег `<meter>` представляет собой скалярную меру в пределах известного диапазона (объем занятого дискового пространства, количество участников опроса из общего числа респондентов).

- **value** - обязательный атрибут.
- **min** и **max** - если они отсутствуют, то подразумевается диапазон от 0 до 1.
- **high** определяет диапазон, при достижении которого значение считается высоким (отображается другим цветом). В качестве значения выступает число, которое должно быть меньше, чем значение атрибута **max** и больше значений атрибутов **low** и **min**. Если атрибут **high** не установлен, то высокое значение будет равно максимальному.
- **low** низкое значение (отображается другим цветом). Если атрибут **low** не установлен, то низкое значение будет равно минимальному.
- **optimum** определяет оптимальное числовое значение, которое должно входить в пределы диапазона, определяемого атрибутами **min** и **max**.
- **value** указывает текущее числовое значение, которое должно входить в пределы диапазона, определяемого атрибутами **min** и **max**.

```
<meter value="2" min="0" max="10" high="7" low="2">2 из 10</meter><br>
```

```
<meter value="0.6">60%</meter>
```



Отправить

# Область вывода результатов вычислений

- Тег **<output>** определяет область, в которую выводится результат вычислений, которые обычно выполняются с помощью скриптов. Он может принадлежать форме, располагаясь внутри нее, в качестве дочернего элемента, или ссылаться на элемент **<form>** с помощью атрибута **form**.

<p>Введите 2 значения, чтобы посчитать их сумму.</p>

```
<form  
  oninput="x.value=parseInt(a.value)+parseInt(b.v  
  alue)">
```

```
<input type=" number " name="a" value="60">+
```

```
<input type="number" name="b" value="40">
```

```
=<output name="x" for="a b"></output>
```

```
</form>
```

+  = 100

# Индикатор выполнения

Тег `<progress>` используется для представления "индикатора выполнения", отображающего, какой процент задачи уже выполнен.

Изменения в индикаторе прогресса производятся с помощью скриптов.

```
<progress value="38" max="100"></progress>
```