# Верстка web-страниц

## Формы

Титова Ольга Ивановна
Минск, 2017

# Содержание



# Формы

**Форма** (англ. form) – важный объект интерфейса, **позволяющий** пользователям вводить данные, осуществлять выбор, обмениваться информацией, менять поведение приложения.

Форма состоит из **элементов управления**, которые позволяют пользователю вводить\выбирать необходимую информацию и отправлять ее на сервер для последующей обработки.

# Формы. Что нового?

На данный момент создано множество библиотек и примеров кода, предназначенных **для обработки форм**.

**HTML5 стандартизирует** эти возможности, предоставляя к использованию **новые атрибуты, элементы и целые API-интерфейсы**.

Теперь функциональность, предназначенная **для обработки данных в формах в режиме реального времени**, **встраивается прямо в браузер и полностью описывается стандартами**.

# Формы

Применяя CSS, можно создать стили для элементов веб-форм. Отдельных css-свойств для форм нет, но к ним можно применять практически все из описанных выше по курсу.

Однако следует помнить о том, что **браузеры могут несколько по-разному отображать результат применения свойства к элементам управления.**

# Формы

Также, **применяя стилистические решения к формам, следует помнить** о том, что не стоит слишком изменять привычные и узнаваемые элементы интерфейса.

В противном случае у пользователей могут возникнуть недопонимания во взаимодействии с веб-страницей.

Всегда проверяйте тот факт, что пользователи распознают форму для ввода данных и понимание принципа взаимодействия с элементами данной формы будет простым и логичным.

# Формы

Чтобы поместить форму на страницу, используется тег **<form>…</form>**.

Информация из всех управляющих элементов, находящихся внутри этого тега, **будет отправляться единым блоком**.

# Формы

index.html

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="utf-8">
5       <title>Forms</title>
6       <link rel="stylesheet" href="reset.css">
7       <link rel="stylesheet" href="style.css">
8   </head>
9   <body>
10      <section>
11          <form name="myForm" method="get" action="file.php">
12          </form>
13      </section>
14  </body>
15  </html>
```

lengtl Ln:15 Col:8 Sel:0|0 Dos\Windows UTF-8 INS

# Формы

Элемент **form** **определяет саму форму и ее компоненты**.

**Элементы формы** (**элементы управления**) располагаются между открывающим и закрывающим тегом.

**Для тега form** **необходимо указать ряд атрибутов**, определяющих, как введенная информация будет обрабатываться и передаваться

# Атрибуты формы

**name** – имя формы; данный атрибут может быть у любого элемента html, но особенно полезен для элементов, расположенных внутри объекта form

**method** – определяет метод, используемый для отправки формы на сервер; имеет два возможных значения – **get** и **post**;

**метод get** используется для передачи ограниченной по объему общедоступной информации (данные передаются через url и обычно не превышают по объему 256 байт);

**метод post** используется для передачи приватной информации произвольного объема (данные при этом не видны пользователю)

# Атрибуты формы

**action** – задает url файла на сервере, который будет обрабатывать информацию, собранную и отправленную формой

**target** – определяет, где будет показан ответ сервера; возможные значения:

**_blank** (новое окно)

**_self** (тот же фрейм) **– по умолчанию**

**_parent** (родительский фрейм)

**_top** (окно, содержащее фрейм)

# Атрибуты формы

**enctype** – определяет способ кодирования данных, посланных формой

Возможны три значения:

**application/x-www-form-urlencoded** (символы кодируются) – **по умолчанию**

**multipart/form-data** (символы не кодируются)

**text/plain** (кодируются только пробелы)

# Атрибуты формы

**accept-charset –** определяет тип кодировки, применяемый к форме; чаще всего используются значения **utf-8** и **ISO-8859-1**

Значение атрибута по умолчанию устанавливается тегами meta

# Управляющие элементы

**Каждый управляющий элемент формы задается отдельным тегом.**

Управляющие теги могут быть как **одинарными** так **и парными**.

```
<form>
    <управляющий элемент></>
    <управляющий элемент></>
    ...
    <управляющий элемент></>
</form>
```

# Управляющие элементы

Каждому управляющему элементу необходимо придумать **имя** и задать его через параметр **name**

**Данные** всех действующих элементов формы отправляются на обработку **в виде пар имя-значение**

# Элемент input

Один из самых важных элементов формы **input**

Он **задает поле**, **куда пользователь может ввести данные**.

**Характеристики элемента** и **тип данных**, которые можно ввести, **зависят от значения атрибута type**.

Этот атрибут определяет, данные какого типа ожидают от пользователя

# Элемент input

**Type может принимать следующие значения**:

**text** – создает **поле для обычного теста**;

**hidden** – создает **скрытое поле**, которое обычно используется для передачи дополнительной информации;

**password** – создает **поле ввода для пароля**, в котором символы скрыты от просмотра; символы, введенные пользователем, обычно показываются на экране в виде точек или звездочек в зависимости от браузера

# Элемент input

**checkbox** – создает **переключатель**; данный тип ввода требует еще одного **атрибута value**, указывающего, какое значение будет послано на сервер; значение посылается только при установленном переключателе (**если значение не было указано, то посылается on**);

**radio** – создает **переключатель для выбора одного варианта из нескольких возможных**; используя то же значение для **атрибута name**, можно группировать несколько кнопок (вариантов выбора) вместе; выбранное значение **атрибута value** посылается на сервер, когда отправляется сама форма; **атрибут checked** позволяет задать значение, выбираемое по умолчанию

# Элемент input

**file** – создает **поле для выбора файла на компьютере пользователя**;

**button** – создает **кнопку, которая сама по себе не совершает никаких действий** и не отправляет форму на сервер; это многофункциональная кнопка, управляемая JavaScript (обычно через действие click); чтобы указать текст на самой кнопке, используется **атрибут value**;

**reset** – создает **кнопку для очистки формы**;

# Элемент input

**submit** – создает **кнопку, которая отправляет форму на сервер**; использует **атрибут value,** чтобы указать текст на самой кнопке; форму можно отправить и с помощью **JavaScript, используя метод submit()**;

**image** – загружает **изображение**, которое используется вместо **кнопки submit**; для работы элемент должен содержать **атрибут src**, указывающий путь к изображению

# Пример

```
<section id="form">
    <form name="myForm" method="get" action="file.php">
    <br>
    <label for="myname">Text: </label>
    <input type="text" name="myname" id="myname">
    <br>
    <input type="radio" name="myoption" value="1" checked> 1
    <input type="radio" name="myoption" value="2"> 2
    <input type="radio" name="myoption" value="3"> 3
    <br>
    <label for="mypassword">Password: </label>
    <input type="password" name="mypassword" id="mypassword">
    <br>
    <label for="mycheckbox">Checkbox: </label>
    <input type="checkbox" name="mycheckbox" id="mycheckbox" value="123">
    <input type="hidden" value="secret key">
    <input type="submit" value="Send">
    </form>
</section>
```

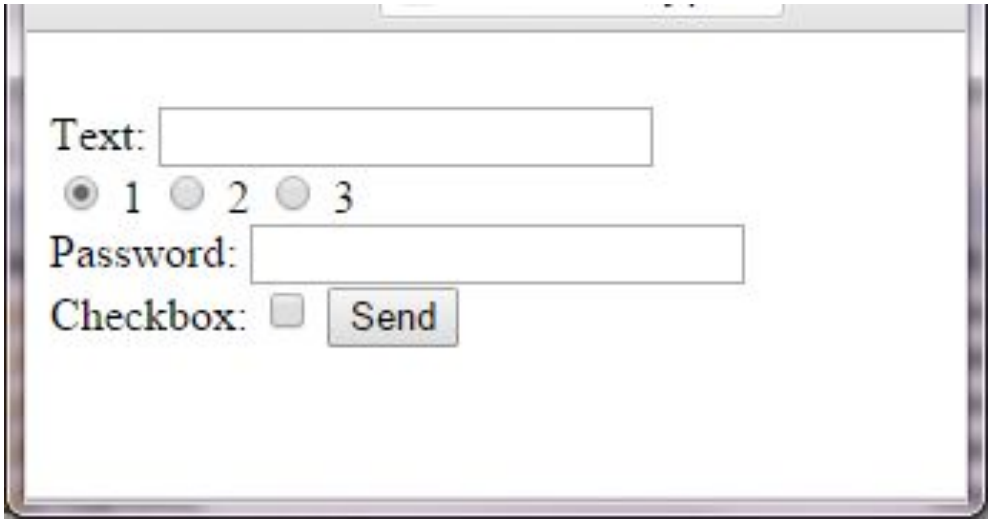
# Пример

**ВАЖНО:**

**Элемент <label>** задает метку для элемента формы.

**Атрибут for** используется в этом элементе, чтобы **связать метку и соответствующий ей элемент** (**значение атрибута for должно быть равно значению атрибута id элемента, для которого предназначена метка**)

# Пример




**ЗАДАНИЕ:**

Создайте код, приведенный в данном примере, сохраните, откройте в браузере;

Обратите внимание на то, что при клике на текстовой надписи автоматически выбирается соответствующий ей элемент управления

# Другие элементы форм

Кроме элемента <input> есть и другие распространенные элементы формы:

**<textarea>** - создает **поле для ввода нескольких строк текста**; размер поля можно задать целыми числами с помощью **атрибутов rows и cols**;

**<select>** - создает **раскрывающийся список**, варианты выбора указаны в элементах <option>;

**<option>** - задает каждый **вариант выбора** для элемента <select>

# Textarea

**Атрибуты для textarea**

**cols** - ширина поля в символах

**disabled** - блокирует доступ и изменение элемента.

**maxlength** - максимальное число введенных символов.

**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

**tabindex** - порядок перехода между элементами при нажатии на клавишу Tab.

# Select

**тег <select>**

Позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором, как показано далее.

**Ширина** списка определяется самым широким текстом, указанным в теге <**option**>, а также может изменяться с помощью стилей.

# Select

## Атрибуты для select

**disabled** - блокирует доступ и изменение элемента.

**multiple** - позволяет одновременно выбирать сразу несколько элементов списка.

**name** - имя элемента для отправки на сервер или обращения через скрипты.

**size** - количество отображаемых строк списка.

**tabindex** - определяет последовательность перехода между элементами при нажатии на клавишу Tab

# Атрибуты

**атрибут size**

Ширина текстового поля, которое определяется числом символов моноширинного шрифта (которые должны поместиться в видимой части поля). Иными словами, ширина задается количеством близстоящих букв одинаковой ширины по горизонтали.

**атрибут tabindex**

Атрибут tabindex определяет последовательность перехода между элементами при нажатии на клавишу Tab.

# Формы

**атрибут readonly**

Когда к тегу <input> добавляется атрибут readonly, текстовое поле не может изменяться пользователем, в том числе вводиться новый текст или модифицироваться существующий.

Тем не менее, состояние и содержимое поля можно менять с помощью скриптов.

```
<input type="text" readonly>
<input type="password" readonly>
```

# Формы

**атрибут disabled**

Блокирует доступ и изменение поля формы.

Оно в таком случае отображается серым и недоступным для активации пользователем. Кроме того, такое поле не может получить фокус путем нажатия на клавишу Tab, мышью или другим способом. Тем не менее, такое состояние поля можно менять с помощью скриптов. Заблокированное в поле значение **не** передается на сервер.

<input type="..." disabled>

# Формы

**атрибут checked**

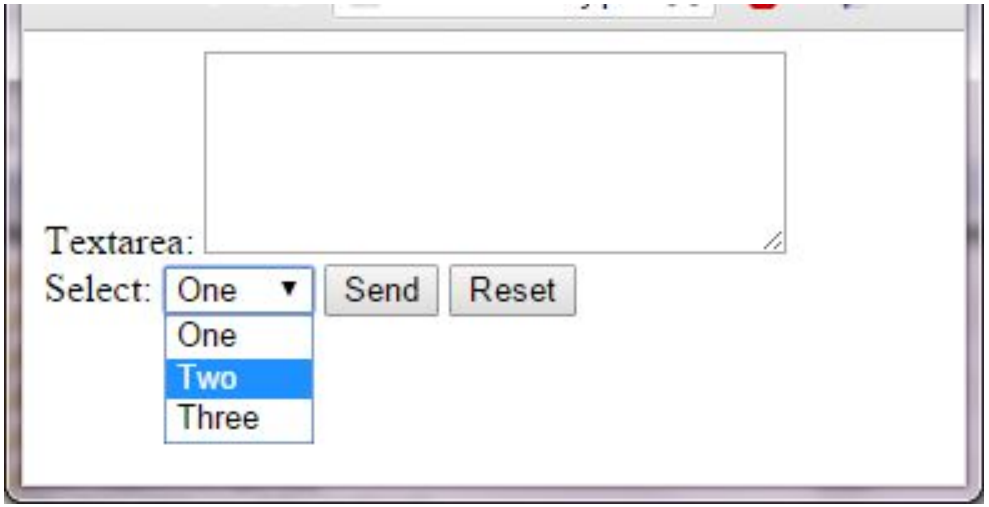
Этот атрибут определяет, помечен ли заранее такой элемент формы, как флажок или переключатель. В случае использования переключателей (radio), может быть отмечен только один элемент группы, для флажков (checkbox) допустимо пометить хоть все элементы.

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

# Пример

```
<section id="form">
    <form name="myForm" method="post" action="file.php">
    <label for="mytext">Textarea: </label>
    <textarea name="mytext" id="mytext" rows="5" cols="30"></textarea>
    <br>
    <label for="mylist">Select: </label>
    <select name="mylist" id="mylist">
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
    </select>
    <input type="submit" value="Send">
    <input type="reset" value="Reset">
    </form>
</section>
```

# Пример




**ЗАДАНИЕ:**

Создайте код, приведенный в данном примере, сохраните, откройте в браузере; протестируйте;

наберите произвольный текстовый фрагмент, а затем нажмите на кнопке Reset

# Отправка формы

**Когда форма отправляется на сервер**, данные для каждого элемента посылаются в виде пары **«имя» - «значение»**, где

**имя** – это значение атрибута name

**значение** – то, что ввел пользователь

**Пары «имя/значение» посылаются через URL или скрытно** – в зависимости от метода, указанного в атрибуте формы method

Если указан метод get, то данные будут видны в url после того, как форма будет отправлена

# Отправка формы

**К примеру,**

**www.koe-chto-for-primer.com?myname=Robert**

– данные передаются через url, и это элемент управления с name=myname и введенным значением Robert

**Чтобы прочитать эту информацию на сервере**, **нужно использовать метод, соответствующий способу отправки**

# Отправка формы

**К примеру,**

```
<?php
    print(‘Your name is: ’ .$_GET[‘myname’]);
?>
```

– описанные выше скрипт может быть помещен в файл, имя которого указано в атрибуте action для формы; этот файл читается на сервере после отправки формы, там же на сервере код выполняется и результат возвращается браузеру;

В нашем случае мы использовали глобальную переменную $_GET – что соответствует выбранному методу (в противном случае нужна переменная $_POST)

# Новые типы полей

Html5 усовершенствовал **поле <input>**, **увеличив количество возможных значений атрибута type**.

Теперь **типы говорят** не только о том, **какой ввод ожидается**, но и о том, **что делать с введенной информацией**

**Браузер обработает** введенные данные в соответствии со значением атрибута type **и сделает вывод** – верны они или нет.

**Этот атрибут работает совместно с дополнительными атрибутами**, необходимыми для ограничения и контроля данных, вводимых пользователем в реальном времени

# Новые типы полей

**Тип email**

**поле для ввода адреса электронной почты**

Ранее с этой целью использовался тип text, однако правильность ввода требовалось проверять с помощью JavaScript. Теперь браузер сам проверить правильность ввода e-mail.

**Задание:** найдите самостоятельно примеры кода на JS, проверяющие корректность заполнения поля, отведенного под e-mail

# Новые типы полей

**Тип email**

**поле для ввода адреса электронной почты**

**<input type="email" name="myemail">**

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип search**

**поле для организации поиска**

Данный тип никак не контролирует ввод данных – это всего лишь способ сообщить браузеру о предназначении поля

**<input type="search" name="mysearch">**

Некоторые браузеры меняют представление по умолчанию для этого элемента, давая пользователю подсказку, для чего используется это поле

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип url**

**поле для сетевых адресов**

Данный тип работает аналогично типу email, считает допустимыми только абсолютные url-адреса и возвращает ошибку, если значение не соответствует этому формату

**<input type="url" name="myurl">**

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип tel**

**поле для ввода телефонных номеров**

Данный тип не требует какого-то определенного синтаксиса; служит лишь для напоминания браузеру о его особенности

**<input type="tel" name="mytel">**

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип number**

**поле для ввода только числовых данных**

С этим типом можно использовать несколько полезных новых атрибутов

**min –** определяет минимальное допустимое значение для данного поля

**max** – определяет максимальное допустимое значение

**step** – определяет шаг увеличения или уменьшения значения данного поля

# Новые типы полей

**Тип number**

**<input type="number" name="mynumber" min="0" max="20" step="5">**

В данное поле вы сможете ввести только числа из 0; 5; 10; 15; 20 – в соответствии с установленными атрибутами ограничениями

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип range**

Этот тип ввода заставляет браузер создавать на странице новый тип элемента управления

**Он позволяет пользователю выбрать значение из некоего числового диапазона**

Обычно на экране этот элемент принимает вид ползунка или стрелочек, увеличивающих или уменьшающих значение (общего стандартного дизайна пока нет).

**Используются атрибуты min, max, step**

# Новые типы полей

**Тип date**

**поле для ввода дат**

В браузерах этот элемент управления выглядит как поле, при каждом щелчке на котором открывается календарь;

интерфейс данного элемента в спецификации не описывается;

чаще всего используется формат «год-месяц-день»

**Задание:** добавьте данный объект к предыдущему примеру

# Новые типы полей

**Тип week**

**поле для выбора недели**

Аналогичен типу date; чаще всего используется синтаксис 2011-W50 – год-номер недели;

**Тип month**

**поле для выбора месяца**

Аналогичен предыдущему типу; чаще всего используется формат год-месяц;

**Задание:** добавьте данные объекты к предыдущему примеру

# Новые типы полей

**Тип time**

**поле для обработки времени**

Принимает формат часов и минут; обычно ожидается значение «часы:минуты:секунды»

**Тип datetime**

**поле для ввода полной даты, включая время и часовой пояс**

**Тип datetime-local**

**поле для ввода полной даты, включая время, безчасового пояса**

**Задание:** добавьте данные объекты к предыдущему примеру

# Новые типы полей

**Тип color**

**Поле, предоставляющее стандартный интерфейс для выбора цвета**

Обычно в таком поле ожидается значение в шестнадцатеричном формате

**Задание:** добавьте данный объект к предыдущему примеру

# Новые атрибуты

**Атрибут autocomplete**

Этот старый атрибут теперь описан в стандарте; может принимать **два значения – on и off**; **значение по умолчанию on**;

Когда значение атрибута равно **off**, то составляющие форму элементы input **не заполняются автоматически**, т.е. для них не отображаются списки ранее введенных значений.

Данный атрибут **можно использовать и для всей формы** – тогда он будет влиять на каждый элемент формы

<input type="search" name="mysearch" **autocomplete="off"**>

# Новые атрибуты

**Атрибуты novalidate и formnovalidate**

Одна из особенностей форм html5 – **встроенная возможность проверки**. Корректность содержимого формы проверяется по умолчанию, автоматически.

**Чтобы запретить такое поведение нужно добавить атрибут novalidate.**

Это **булев атрибут** используется **в теге form**

Когда он присутствует, форма посылается на сервер без проверки

# Новые атрибуты

**Атрибуты novalidate и formnovalidate**

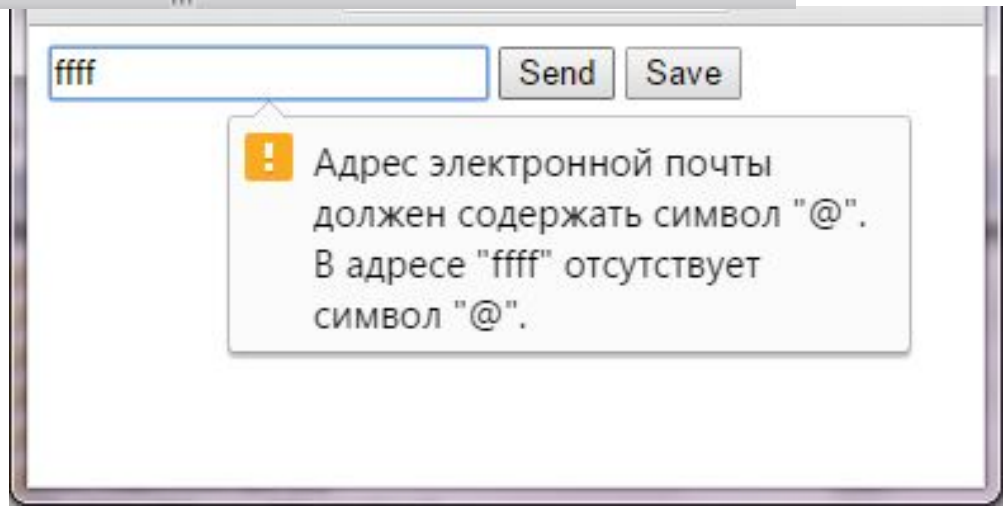
**Иногда проверка нужна только при определенных обстоятельствах** (допустим, информацию нужно сохранить, чтобы вернуться к ней позднее)

Тогда атрибут formnovalidate нужно прописать к определенным тегам input

# Новые атрибуты

При нажатии Save – проверки не будет, как при использовании кнопки Send

```
27  <form name="myForm" method="get" action="file.php">
28  <input type="email" name="myemail">
29  <input type="submit" value="Send">
30  <input type="submit" value="Save" formnovalidate>
31  </form>
```



# Новые атрибуты

**Атрибут placeholder**

**Представляет собой короткую подсказку** – слово или фразу, помогающие правильно ввести ожидаемые данные.

Значение данного атрибута **показывается внутри поля и исчезает, когда пользователь переводит фокус ввода** на соответствующий элемент.

# Новые атрибуты

**Атрибут required**

Это **булев** атрибут, он **не позволяет подтвердить отправку формы, если поле осталось пустым**.

**К примеру,** в поле было внесено значение, но проверка правильности ввода не подтвердила допустимость, и поле считается пустым – при добавлении атрибута required браузер будет требовать обязательности заполнения поля в дополнение к проверке формата вводимых данных

# Новые атрибуты

**Атрибут multiple**

Это **булев** атрибут, можно использовать с некоторыми типами ввода, чтобы **разрешать пользователю вводить в одном поле несколько значений**

Чтобы введенная информация прошла проверку, **значения нужно разделять запятой**

При необходимости каждое из значений будет проверяться на допустимость формата

# Новые атрибуты

**Атрибут autofocus**

**Переводит фокус веб-страницы на выбранный элемент с учетом текущей ситуации**: фокус не меняется, если пользователь уже выбрал другой элемент и работает с ним (в отличие от ранее используемого метода JS)

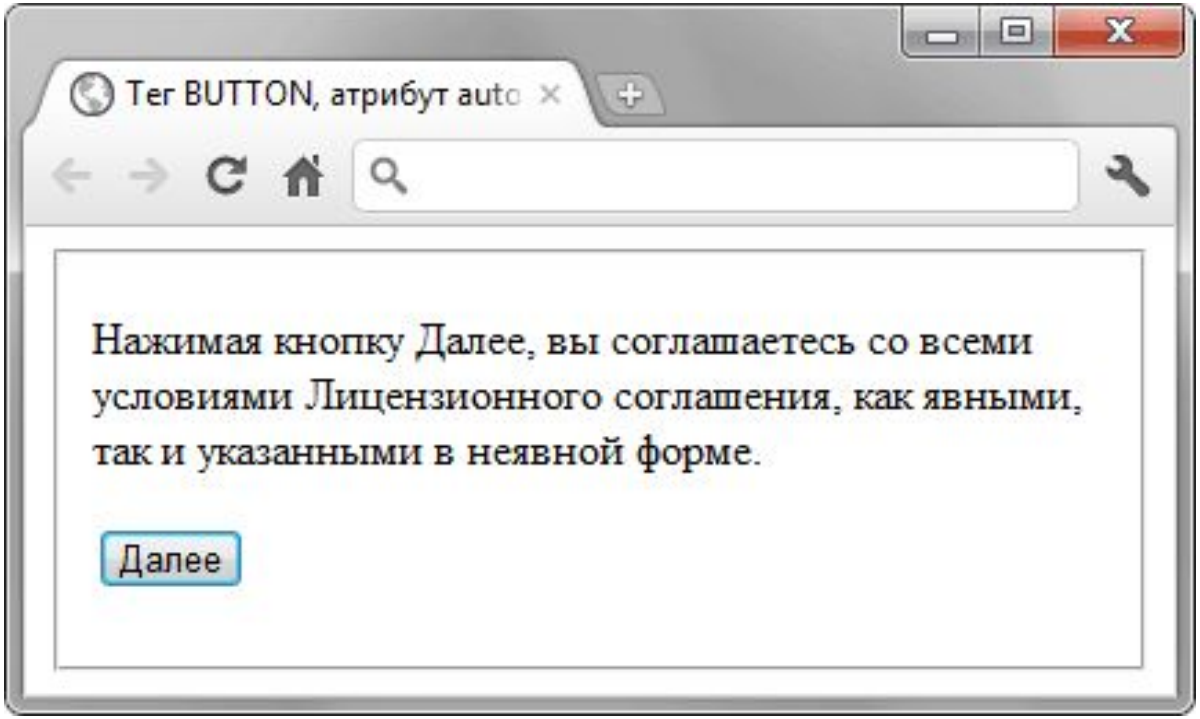
# Новые атрибуты

**Атрибут autofocus**

```
<form>
    <fieldset>
        <p>Нажимая кнопку Далее, вы соглашаетесь со всеми условиями Лицензионного соглашения, как явными, так и указанными в неявной форме.</p>
        <p>
        <button autofocus value="next">Далее</button>
        </p>
    </fieldset>
</form>
```

# Новые атрибуты

**Атрибут autofocus**



# Новые атрибуты

**Атрибут pattern**

**Позволяет изменить способ проверки, используемый по умолчанию**.

**Позволяет настраивать правила валидации добавлением регулярных выражений**

Т.е. самостоятельно прописать правило для проверки ввода, а с помощью атрибута title добавить сообщение об ошибке

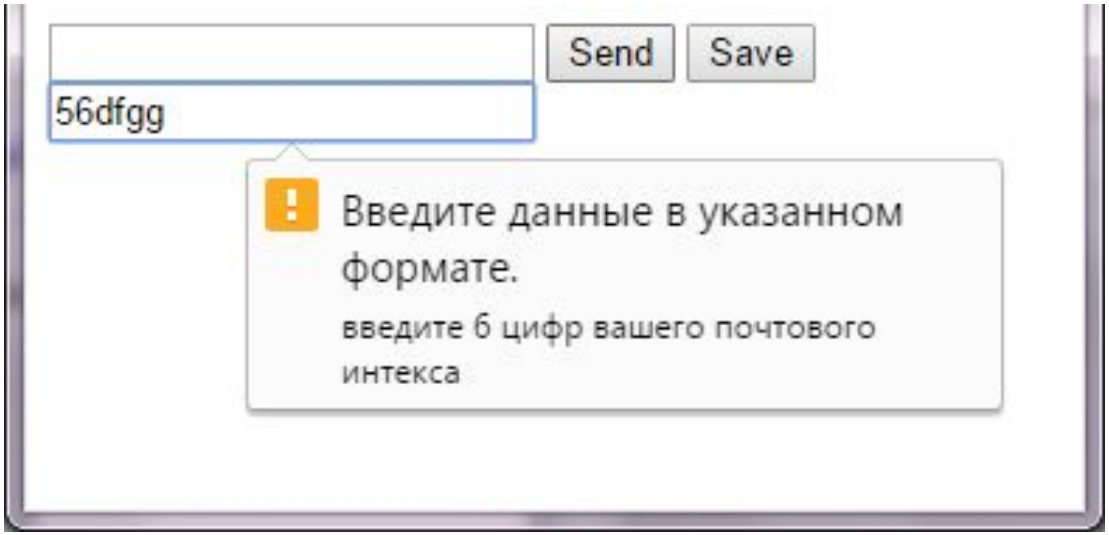
! О регулярных выражениях можно почитать больше в курсе по JavaScript

# Новые атрибуты

**Атрибут pattern**

**К примеру,**

<input pattern="[0-9]{6}" name="pcode" title="введите 6 цифр вашего почтового индекса">



# Новые элементы

**Элемент datalist**

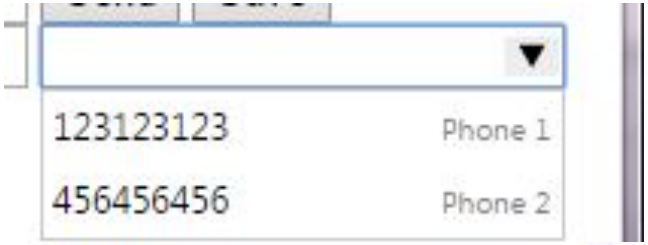
Позволяет **заранее построить список пунктов**, которые в дальнейшем будут предлагаться **в качестве вариантов заполнения полей ввода** (для этого нужно добавить **атрибут list**)

```
<datalist id="mydata">
    <option value="123123123" label="Phone 1">
    <option value="456456456" label="Phone 2">
</datalist>
```

# Новые элементы

**Элемент datalist**

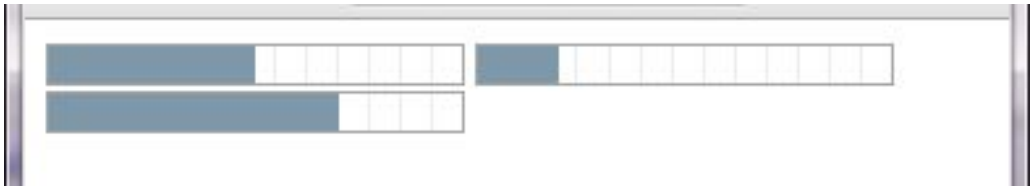
После того, как объявлен элемент datalist, **остается лишь сослаться на этот список пунктов из элемента input,** используя **атрибут list** (значение id из datalist)




```
<datalist id="mydata">
    <option value="123123123" label="Phone 1">
    <option value="456456456" label="Phone 2">
</datalist>
<input type="tel" name="myphone" list="mydata">
```

# Новые элементы

**Элемент progress**



**Иллюстрирует прогресс выполнения задачи.**

Его не обязательно использовать только с формами.

Принимает **два атрибута**, устанавливающие **его статус и лимиты**.

**Атрибут value** указывает, какая доля задачи уже выполнена, а **атрибут max** содержит значение, достижение которого соответствует завершению задачи

<progress value="0" max="100">0%</progress>

# Новые элементы

**Элемент meter**

**Используется для отображения шкалы – известного диапазона** (к примеру, пропускная способность)

**Атрибуты min и max** устанавливают границы диапазона, **value** определяет измеряемое значение, **low, high и optimum** используются для сегментирования диапазона и определения оптимальной позиции

```
<meter value="60" min="0" max="100" low="40" high="80" optimum="100">60</meter>
```

# Новые элементы

**Элемент output**

**Представляет собой особый результат вычисления**

Обычно он используется для отображения результатов обработки каких-то значений элементами формы.

Атрибут for позволяет связать элемент output с исходными элементами, участвующими в расчетах, однако чаще всего ссылки на элементы создаются и модифицируются в коде JS

**Синтаксис:** <output>значение</output>

# Элементы html-форм

**<fieldset>** - предназначен для группировки элементов, связанных друг с другом; большинство браузеров нормально отображают фоновые цвета, фоновые изображения и границы для этого тега.

**<legend>** - идет по структуре за тегом fieldset и в нем содержится название для группы – оно появится в центре верхней границы тега fieldset

```
<fieldset>
    <legend>Текст</legend>
</fieldset>
```

# Элементы html-форм

**ВАЖНО:**

При оформлении форм, элементы которых создаются через тег input, вместо задания стилевого класса можно использовать селектор атрибутов, который позволит настроить внешний вид формы, не обращаясь к классам.

**К примеру,**

```
input[type="text"] {
    background-color: blue;
}
```

# Компоновка

Все, что надо для создания формы, - добавить несколько фрагментов текстовых и других элементов на веб-страницу.

Однако, зачастую визуально это получается беспорядочно.

**Упорядочения при отображении форм** можно добиться несколькими путями:

- таблицы (очень и очень плохо);
- через css;

# Псевдоклассы

Для стилизации форм можно использовать псевдоклассы, например,

**:focus** – позволяет создать селектор, изменяющий внешний вид текстового поля при щелчке кнопкой мыши или при переходе на него с помощью клавиши табуляции.

**:checked** – работает с переключателями и флажками; предназначен для стилизации этих элементов, но обычно веб-браузеры в отношении вида этих полей ведут себя сдержанно.

# Псевдоклассы

кроме того, элементы формы можно сделать активными и неактивными (через средства JavaScript) и использовать псевдоклассы **:enabled** и **:disabled**

# Псевдоклассы

**:valid и :invalid** – обращение к элементу input с верно и неверно введенным значением

```
.text:valid {border-color:green;}
.text:invalid {border-color:red;}
```

**:optional и :required** – указывают стили для обязательных и дополнительных элементов

```
.name:optional {border-left-color: grey;}
.password:required {border-left-color: red;}
```

# Псевдоклассы

**:in-range и :out-of-range** – позволяют задавать стили для элементов с ограничением диапазона ввода (таких как number) в двух случаях: когда введенное значение лежит в заданных пределах и когда оно вне их.

```
.age:in-range {background:#eef;}
.age:out-of-range {background:#fee;}
```

# Спасибо за внимание