# Лекция 5. Формы и обработка данных

*При первоначальном знакомстве с лекцией можно прочитать только пункты 5.1 – 5.7, 5.11-5.12.*

## Оглавление



## *5.1. Простой пример формы*

Формы в HTML представляют собой интерфейс с пользователем и основной способ для ввода и отправки данных. Все поля формы помещаются между тегами `<form>` и `</form>`. Например, создадим простейшую форму с текстовым полем ввода и кнопкой:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Формы в HTML5</title>
    </head>
    <body>
```

```
        <form method="post"
            action="http://localhost:8080/login.php">
            <input name="login"/>
            <input type="submit" value="Войти" />
        </form>
    </body>
</html>
```

Для настройки форм у элемента `form` определены следующие атрибуты:

- `method`: устанавливает метод отправки данных на сервер. Допустимы два значения: `post` и `get`. Значение `post` позволяет передать данные на web-сервер через заголовки протокола HTTP. А значение `get` позволяет передать данные через URL-адрес запроса.
- `action`: устанавливает адрес, на который передаются данные формы
- `enctype`: устанавливает тип передаваемых данных. Он свою очередь может принимать следующие значения:
    - `application/x-www-form-urlencoded`: кодировка отправляемых данных по умолчанию
    - `multipart/form-data`: эта кодировка применяется при отправке файлов
    - `text/plain`: эта кодировка применяется при отправке простой текстовой информации

В выше использованном примере у формы установлен метод "post", то есть все значения формы отправляются в теле запроса, а адресом служит строка *http://localhost:8080/login.php*.

Адрес здесь указан случайным образом. В реальности по указанному адресу работает web-сервер, который, используя одну из технологий серверной стороны (PHP, NodeJS, ASP.NET и т.д.), может получать запросы и возвращать ответ на них в виде разметки HTML или другого содержимого. В данном же случае мы не будем акцентировать внимание на технологиях серверной стороны, сосредоточимся лишь на тех средствах HTML, которые позволяют отправлять данные на сервер.

Часто web-браузеры запоминают вводимые данные, и при вводе браузеры могут выдавать список подсказок из ранее введенных слов:

Это может быть не всегда удобно, и с помощью атрибута `autocomplete` можно отключить **автодополнение**:

```
<form method="post" autocomplete="off"
            action="http://localhost:8080/login.php">
    <input name="login" />
    <input name="password" />
    <input type="submit" value="Войти" />
</form>
```

Если нам надо включить автодополнение только для каких-то определенных полей, то мы можем применить к ним атрибут `autocomplete="on"`:

```
<form method="post" autocomplete="off"
action="http://localhost:8080/login.php">
```

```
    <input name="login" />
    <input name="password" autocomplete="on" />
    <input type="submit" value="Войти" />
</form>
```
Теперь для всей формы, кроме второго поля, будет отключено автодополнение.

### *5.2. Элементы форм*

Формы состоят из определённого количества элементов ввода. Все элементы ввода помещаются между тегами `<form>` и `</form>`
Наиболее распространенным элементом ввода является элемент `input`. Однако реальное действие этого элемента зависит от того, какое значение установлено у его атрибута `type`. А он может принимать следующие значения:

- `text`: обычное текстовое поле;
- `password`: тоже текстовое поле, только вместо вводимых символов отображаются звездочки, поэтому в основном используется для ввода пароля;
- `radio`: радиокнопка или переключатель. Из группы радиокнопок с одинаковым атрибутом `name` можно выбрать только одну;
- `checkbox`: элемент флажок, который может находиться в отмеченном или неотмеченном состоянии;
- `hidden`: скрытое поле;
- `submit`: кнопка отправки формы;
- `color`: поле для ввода цвета;
- `date`: поле для ввода даты;
- `datetime`: поле для ввода даты и времени с учетом часового пояса;
- `datetime-local`: поле для ввода даты и времени без учета часового пояса;
- `email`: поле для ввода адреса электронной почты;
- `month`: поле для ввода года и месяца;
- `number`: поле для ввода чисел;
- `range`: ползунок для выбора числа из некоторого диапазона;
- `tel`: поле для ввода телефона;
- `time`: поле для ввода времени;
- `week`: поле для ввода года и недели;
- `url`: поле для ввода адреса url;
- `file`: поле для выбора отправляемого файла;
- `image`: создает кнопку в виде картинки.

Кроме элемента `input` в различных модификациях есть еще небольшой набор элементов, которые также можно использовать на форме:

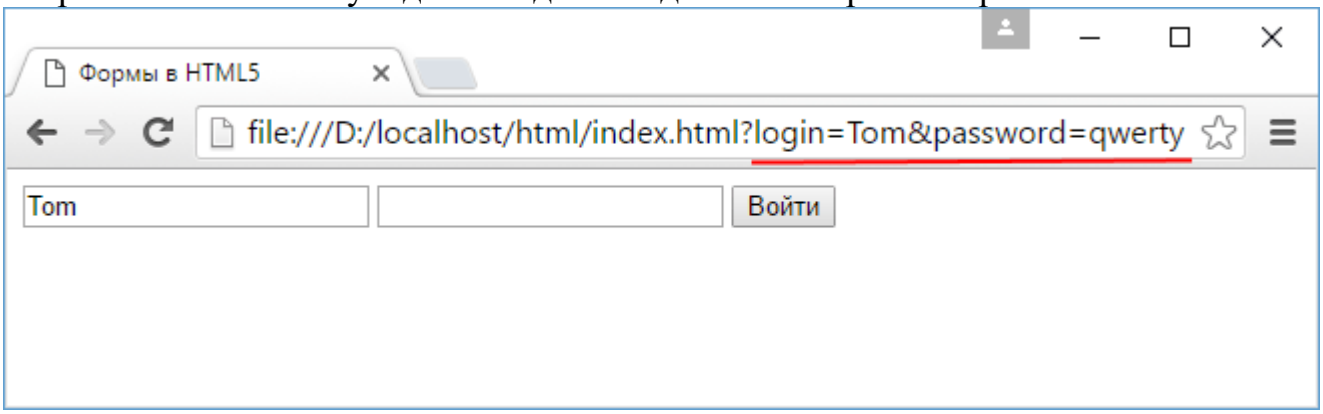
- `button`: создает кнопку;
- `select`: выпадающий список;
- `label`: создает метку, которая отображается рядом с полем ввода;
- `textarea`: многострочное текстовое поле.

У всех элементов ввода можно установить **атрибуты `name` и `value`**. Эти атрибуты имеют важное значение. По атрибуту `name` мы можем идентифицировать поле ввода, а атрибут `value` позволяет установить значение поля ввода. Например:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Формы в HTML5</title>
    </head>
    <body>
        <form method="get" action="index.html">
            <input type="text" name="login" value="Tom"/>
            <input type="password" name="password"/>
            <input type="submit" value="Войти" />
        </form>
    </body>
</html>
```

Здесь текстовое поле имеет значение "Tom" (как указано в атрибуте `value`), поэтому при загрузке web-страницы в этом поле мы увидим данный текст.

Поскольку методом отправки данных формы является метод "get", то данные будут отправляться через строку запроса. Так как нам в данном случае не важно, как данные будут приниматься, не важен сервер, который получает данные, поэтому в качестве адреса мы установили ту же самую страницу, то есть, файл `index.html`. И при отправке мы сможем увидеть введенные данные в строке запроса:




В строке запроса нас интересует следующий кусочек:

```
login=Tom&password=qwerty
```

При отправке формы браузер соединяет все данные в набор пар "ключ-значение". В нашем случае две таких пары: `login=Tom` и `password=qwerty`. Ключом в этих парах выступает название поля ввода, которое определяется атрибутом `name`, а значением - собственно то значение, которое введено в поле ввода (или значение атрибута `value`).

Получив эти данные, сервер легко может узнать, какие значения в какие поля ввода были введены пользователем.

### *5.3. Кнопки*

Кнопки представлены элементом button. Они обладают широкими возможностями по конфигурации. Так, в зависимости от значения атрибута type мы можем создать различные типы кнопок:

- submit: кнопка, используемая для отправки формы;
- reset: кнопка сброса значений формы;
- button: кнопка без какого-либо специального назначения, обычно её нажатие обрабатывается программно кодом на JavaScript;

Если кнопка используется для отправки формы, то есть у нее установлен атрибут type="submit", то мы можем задать у нее ряд дополнительных атрибутов:

- form: определяет форму, за которой закреплена кнопка отправки;
- formaction: устанавливает адрес, на который отправляется форма. Если у элемента form задан атрибут action, то он переопределяется;
- formenctype: устанавливает формат отправки данных. Если у элемента form установлен атрибут enctype, то он переопределяется;
- formmethod: устанавливает метод отправки формы (post или get). Если у элемента form установлен атрибут method, то он переопределяется

Например, определим на форме кнопку отправки и кнопку сброса:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Формы в HTML5</title>
    </head>
    <body>
        <form>
            <p><input type="text" name="login"/></p>
            <p><input type="password" name="password"/></p>
            <p>
                <button type="submit" formmethod="get"
                 formaction="index.html">Отправить</button>
                <button type="reset">Отмена</button>
            </p>
        </form>
    </body>
</html>
```

Кроме элемента button для создания кнопок можно использовать элемент input, у которого атрибут равен submit или reset. Например:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
```

```
        <meta charset="utf-8">
        <title>Формы в HTML5</title>
    </head>
    <body>
        <form>
            <p><input type="text" name="login"/></p>
            <p><input type="password" name="password"/></p>
            <p>
                <input type="submit" value="Отправить" />
                <input type="reset" value="Отмена" />
            </p>
        </form>
    </body>
</html>
```

Ещё один элемент `input` с атрибутом `type="image"` позволяет использовать в качестве кнопки изображение:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Форма ввода в HTML5</title>
    </head>
    <body>
<form>
 <p>
  <input type="text" name="search" />
  <input type="image" src="search.png" name="submit" />
 </p>
</form>
</body></html>
```

Кроме наличия изображения в остальном эта кнопка будет аналогична стандартной кнопке отправки `input type="submit"` или `button type="submit"`.

### *5.4. Текстовые поля*

Однострочное текстовое поле создается с помощью элемента `input`, когда его атрибут `type` имеет значение `text`:

```
<input type="text" name="login" />
```

С помощью ряда дополнительных атрибутов можно настроить текстовое поле:

- `dirname`: устанавливает направление ввода текста;
- `maxlength`: максимально допустимое количество символов в текстовом поле;
- `pattern`: определяет шаблон, которому должен соответствовать вводимый текст;
- `placeholder`: устанавливает текст, который по умолчанию отображается в текстовом поле;

- `readonly`: делает текстовом поле доступным только для чтения;
- `required`: указывает, что текстовое поле обязательно должно иметь значение;
- `size`: устанавливает ширину текстового поля в видимых символах;
- `value`: устанавливает значение по умолчанию в текстовом поле

Применим некоторые атрибуты:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Текстовые поля в HTML5</title>
    </head>
    <body>
        <form>
            <p><input type="text" name="userName"
             placeholder="Введите имя" size="18" /></p>
            <p><input type="text" name="userPhone"
                 placeholder="Введите номер телефона"
                size="18" maxlength="11" />
            </p>
            <p>
                <button type="submit">Отправить</button>
                <button type="reset">Отмена</button>
            </p>
        </form>
    </body>
</html>
```

В этом примере во втором текстовом поле сразу устанавливаются атрибуты `maxlength` и `size`. При этом `size` - **количество символов, которые помещаются в видимое пространство** поля, больше, чем **допустимое количество символов**. Однако все равно ввести символов больше, чем `maxlength`, мы не сможем.

В данном случае также важно различать атрибуты `value` и `placeholder`, хотя оба устанавливают видимый текст в поле. Однако `placeholder` устанавливает своего рода **подсказку** или приглашение к вводу, поэтому он обычно отмечается серым цветом. В то время как значение `value` представляет введенный в поле текст по умолчанию:

```
<p><input type="text" name="userName" value="Том" /></p>
<p><input type="text" name="userPhone" placeholder="Номер
телефона" /></p>
```

Атрибуты `readonly` и `disabled` **делают текстовое поле недоступным**, однако сопровождаются разным визуальным эффектом. В случае с `disabled` текстовое поле затеняется:

```
<p><input type="text" name="userName" value="Том" readonly
/></p>
```

```
<p><input type="text" name="userPhone" value="+12345678901"
disabled /></p>
```

Среди атрибутов текстового поля также следует отметить такой атрибут как `list`. Он содержит ссылку на элемент `datalist`, который определяет набор значений, появляющихся в виде подсказки при вводе в текстовое поле. Например:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Текстовые поля в HTML5</title>
    </head>
    <body>
        <form>
            <p><input list="phonesList" type="text" name="model"
               placeholder="Введите модель" /></p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
        <datalist id="phonesList">
            <option value="iPhone 6S" label="54000"/>
            <option value="Lumia 950">35000</option>
            <option value="Nexus 5X"/>
        </datalist>
    </body>
</html>
```

Атрибут `list` текстового поля указывает на `id` элемента `datalist`. Сам элемент `datalist` с помощью вложенных элементов `option` определяет элементы списка. И при вводе в текстовое поле этот список отображается в виде подсказки.

Для создания **полей поиска** предназначен элемент `input` с атрибутом `type="search"`. Формально он представляет собой простое текстовое поле:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Поиск в HTML5</title>
    </head>
    <body>
    <form>
        <input type="search" name="term" />
        <input type="submit" value="Поиск" />
    </form>
    </body>
</html>
```

Для ввода пароля используется элемент input с атрибутом type="password". Его отличительной чертой является то, что вводимые символы маскируются точками:

```
<form>
    <p><input type="text" name="login" /></p>
    <p><input type="password" name="password" /></p>
    <input type="submit" value="Авторизация" />
</form>
```

### *5.5. Метки и автофокус*

Вместе с полями ввода нередко используются **метки**, которые представлены элементом label. Метки создают аннотацию или заголовок к полю ввода, указывают, для чего это поле предназначено.

Для связи с полем ввода метка имеет атрибут for, который указывает на идентификатор id поля ввода:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Label в HTML5</title>
    </head>
    <body>
        <form>
            <p>
                <label for="login">Логин: </label>
                <input type="text" id="login" name="login" />
            </p>
            <p>
                <label for="password">Пароль: </label>
                <input type="password" id="password"
                 name="password" />
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
</html>
```

Так, текстовое поле здесь имеет атрибут id="login". Поэтому у связанной с ним метки устанавливается атрибут for="login". Нажатие на эту метку позволяет перевести фокус на текстовое поле для ввода логина.

Особенно необходим тег label с атрибутом for при создании "кликабельных" подписей к радиокнопкам и чекбоксам (см. п. 5.7).

Также мы можем установить **автофокус** по умолчанию на какое-либо поле ввода. Для этого применяется атрибут `autofocus`:

```
<form>
    <p>
        <label for="login">Логин: </label>
        <input type="text" autofocus id="login" name="login" />
    </p>
    <p>
        <label for="password">Пароль: </label>
        <input type="password" id="password" name="password" />
    </p>
    <p>
        <button type="submit">Отправить</button>
    </p>
</form>
```

Здесь при запуске страницы фокус сразу же переходит на текстовое поле.

### *5.6. Элементы для ввода чисел*

Для ввода чисел используется элемент `input` с атрибутом `type="number"`. Он создает числовое поле, которое мы можем настроить с помощью следующих атрибутов:

- `min`: минимально допустимое значение
- `max`: максимально допустимое значение
- `readonly`: доступно только для чтения
- `required`: указывает, что данное поле обязательно должно иметь значение
- `step`: значение, на которое будет увеличиваться число в поле
- `value`: значение по умолчанию

Используем числовое поле:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Числовое поле в HTML5</title>
    </head>
    <body>
<form>
 <p>
  <label for="age">Возраст: </label>
  <input type="number" step="1" min="1" max="100" value="10"
       id="age" name="age"/>
 </p>
 <p>
  <button type="submit">Отправить</button>
 </p>
</form>
```

```
</body></html>
```

Здесь числовое поле по умолчанию имеет значение 10 (`value="10"`), минимально допустимое значение, которое мы можем ввести, - `1`, а максимальное допустимое значение - `100`. И атрибут `step="1"` устанавливает, что значение будет увеличиваться на единицу.
В зависимости от браузера визуализация этого поля может отличаться. Но как правило, у большинства современных браузеров, кроме IE 11 и Microsoft Edge, справа в поле ввода имеются стрелки для увеличения/уменьшения значения на величину, указанную в атрибуте `step`.
Как и в случае с текстовым полем мы можем здесь прикрепить список `datalist` с диапазоном возможных значений:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Числовое поле в HTML5</title>
    </head>
    <body>
        <form>
            <p>
                <label for="price">Цена: </label>
                <input type="number" list="priceList"
                    step="1000" min="3000" max="100000"
                    value="10000" id="price" name="price"/>
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
        <datalist id="priceList">
            <option value="15000" />
            <option value="20000" />
            <option value="25000" />
        </datalist>
    </body>
</html>
```

**Ползунок** представляет собой шкалу, на которой мы можем выбрать одно из значений. Для создания ползунка применяется элемент `input` с атрибутом `type="range"`. Во многом ползунок похож на простое поле для ввода чисел. Он также имеет атрибуты `min`, `max`, `step` и `value`:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
```

```
        <meta charset="utf-8">
        <title>Ползунок в HTML5</title>
    </head>
    <body>
        <form>
            <p>
                <label for="price">Цена:</label>
                1<input type="range" step="1" min="0" max="100"
                  value="10" id="price" name="price"/>100
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
</html>
```

### *5.7. Флажки и переключатели*

**Флажок** (чекбокс) представляет собой элемент, который может находиться в двух состояниях: отмеченном и неотмеченном. Флажок создается с помощью элемента `input` с атрибутом `type="checkbox"`:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Чекбокс в HTML5</title>
    </head>
    <body>
        <h2>Изучаемые технологии</h2>
        <form>
            <p>
                <input type="checkbox" checked
                   name="html5"/>HTML5
            </p>
            <p>
                <input type="checkbox" name="dotnet"/>.NET
            </p>
            <p>
                <input type="checkbox" name="java"/>Java
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
```

```
</html>
```
Атрибут `checked` позволяет установить флажок в отмеченное состояние.

**Переключатели** или **радиокнопки** похожи на флажки, они также могут находиться в отмеченном или неотмеченном состоянии. Только для переключателей можно создать одну группу, в которой одновременно можно выбрать только один переключатель. Например:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Радиокнопки в HTML5</title>
    </head>
    <body>
        <form>
            <h2>Укажите пол</h2>
            <p>
                <input type="radio" value="man" checked
                  name="gender"/>мужской
            </p>
            <p>
                <input type="radio" value="woman"
                  name="gender"/>женский
            </p>
            <h2>Выберите технологию</h2>
            <p>
                <input type="radio" value="html5" checked
                   name="tech"/>HTML5
            </p>
            <p>
                <input type="radio" value="net"
                   name="tech"/>.NET
            </p>
            <p>
                <input type="radio" value="java"
                  name="tech"/>Java
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
</html>
```

Для создания радиокнопки надо указать атрибут `type="radio"`. И теперь другой атрибут `name` указывает не на имя элемента, а на имя группы, к которой принадлежит элемент-радиокнопка. В данном случае у нас две группы радиокнопок: `gender` и

tech. Из каждой группы мы можем выбрать только один переключатель. Опять же чтобы отметить радиокнопку, у нее устанавливается атрибут checked
Важное значение играет атрибут value, который при отправке формы позволяет серверу определить, какой именно переключатель был отмечен.

### *5.8. Элементы для ввода цвета, url, email, телефона*

За **установку цвета** в HTML5 отвечает специальный элемент input с типом color:

```
<label for="favcolor">Выберите цвет</label>
<input type="color" id="favcolor" name="favcolor" />
```

Элемент отображает выбранный цвет. А при нажатии на него появляется системное диалоговое окно для установки цвета.
Значением этого элемента будет числовой шестнадцатеричный код выбранного цвета.
С помощью элемента datalist мы можем задать набор цветов, из который пользователь может выбрать нужный:

```
<label for="favcolor">Выберите цвет</label>
<input type="color" list="colors" id="favcolor"
 name="favcolor" />
<datalist id="colors">
    <option value="#0000FF" label="blue">
    <option value="#008000" label="green">
    <option value="#ff0000" label="red">
</datalist>
```

Каждый элемент option в datalist должен в качестве значения принимать шестнадцатеричный код цвета, например, "#0000FF". После выбора цвета данный числовой код устанавливается в качестве значения в элементе input.

Ряд полей input предназначены для ввода таких данных, как url-адрес, адрес электронной почты и телефонного номера. Они однотипны и во многом отличаются только тем, что для атрибута type принимают соответственно значения email, tel и url.
Для их настройки мы можем использовать те же атрибуты, что и для обычного текстового поля:

- maxlength: максимально допустимое количество символов в поле;
- pattern: определяет шаблон, которому должен соответствовать вводимый текст;
- placeholder: устанавливает текст, который по умолчанию отображается в поле;
- readonly: делает текстовом поле доступным только для чтения;
- required: указывает, что текстовое поле обязательно должно иметь значение;
- size: устанавливает ширину поля в видимых символах;
- value: устанавливает значение по умолчанию для поля;
- list: устанавливает привязку к элементу datalist со списком возможных значений

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Форма ввода в HTML5</title>
    </head>
    <body>
  <form>
   <p>
    <label for="email">Email: </label>
    <input type="email" placeholder="user@gmail.com"
           id="email" name="email"/>
  </p>
  <p>
   <label for="url">URL: </label>
   <input type="url" id="url" name="url"/>
  </p>
 <p>
  <label for="phone">Телефон: </label>
  <input type="tel" placeholder="(XXX)-XXX-XXXX"
         id="phone" name="phone"/>
 </p>
 <p>
  <button type="submit">Отправить</button>
 </p>
</form>
</body></html>
```

Основное преимущество подобных полей ввода перед обычными текстовыми полями состоит в том, что поля ввода для email, url, телефона для проверки ввода используют соответствующий шаблон. Например, если мы введем в какое-либо поле некорректное значение и попробуем отправить форму, то браузер может отобразить нам сообщение о некорректном вводе, а форма не будет отправлена.

### *5.9. Элементы для ввода даты и времени*

Для работы с датами и временем в HTML5 предназначено несколько типов элементов `input`:

- `datetime-local`: устанавливает дату и время;
- `date`: устанавливает дату;
- `month`: устанавливает текущий месяц и год;
- `time`: устанавливает время;
- `week`: устанавливает текущую неделю

Например, используем поле для установки даты:

```
<!DOCTYPE html>
```

```
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Форма ввода в HTML5</title>
    </head>
    <body>
        <form>
            <p>
                <label for="firstname">Имя: </label>
                <input type="text" id="firstname"
                 name="firstname"/>
            </p>
            <p>
                <label for="date">Дата рождения: </label>
                <input type="date" id="date" name="date"/>
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
</html>
```

И при вводе в поле для даты будет открываться календарик. Действие этого элемента зависит от браузера.

Применение остальных элементов:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Форма ввода в HTML5</title>
    </head>
    <body>
        <form>
            <p>
                <label for="week">Неделя: </label>
                <input type="week" name="week" id="week" />
            </p>
            <p>
                <label for="localdate">Дата и время: </label>
                <input type="datetime-local" id="localdate"
                 name="date"/>
            </p>
            <p>
                <label for="month">Месяц: </label>
                <input type="month" id="month" name="month"/>
            </p>
```

```
            <p>
                <label for="time">Время: </label>
                <input type="time" id="time" name="time"/>
            </p>
            <p>
                <button type="submit">Отправить</button>
            </p>
        </form>
    </body>
</html>
```

При использовании этих элементов также надо учитывать, что Firefox поддерживает только элементы `date` и `time`, для остальных создаются обычные текстовые поля. А IE11 сосвсем не поддерживают эти элементы.

### *5.10. Отправка файлов*

За выбор файлов на форме отвечает элемент `input` с атрибутом `type="file"`:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Отправка файлов в HTML5</title>
    </head>
    <body>
        <form enctype="multipart/form-data" method="post"
         action="http://localhost:8080/postfile.php">
            <p>
                <input type="file" name="file" />
            </p>
            <p>
                <input type="submit" value="Отправить" />
            </p>
        </form>
    </body>
</html>
```

При нажатии на кнопку "Выберите файл" открывается диалоговое окно для выбора файла. А после выбора рядом с кнопкой отображается имя выбранного файла.

Важно отметить, что для отправки файла на сервер форма должна иметь атрибут `enctype="multipart/form-data"`. При этом форма должна передаваться методом `post` (в теле http-запроса).

С помощью ряда атрибутов мы можем дополнительно настроить элементы выбора файла:

- `accept`: устанавливает тип файл, которые допустимы для выбора;
- `multiple`: позволяет выбирать множество файлов;
- `required`: требует обязательной установки файла

Пример множественного выбора файлов:

```
<form enctype="multipart/form-data" method="post"
action="http://localhost:8080/postfile.php">
    <p>
        <input type="file" name="file" multiple />
    </p>
    <p>
        <input type="submit" value="Отправить" />
    </p>
</form>
```
При нажатии на кнопку также открывается диалоговое окно для выбора файлов, только теперь, зажав клавишу Ctrl или Shift, мы можем выбрать несколько файлов, а после выбора рядом с кнопкой отобразится количество выбранных файлов.

### *5.11. Список select*

Элемент `select` создает список. В зависимости от настроек это может быть выпадающий список для выбора одного элемента, либо раскрытый список, в котором можно выбрать сразу несколько элементов.
Создадим выпадающий список:
```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Элемент select в HTML5</title>
    </head>
    <body>
        <form method="get">
            <p>
                <label for="phone">Выберите модель:</label>
<select id="phone" name="phone">
 <option value="iphone 6s">iPhone 6S</option>
 <option value="lumia 950">Lumia 950</option>
 <option value="nexus 5x">Nexus 5X</option>
 <option value="galaxy s7">Galaxy S7</option>
</select>
            </p>
            <p>
                <input type="submit" value="Отправить" />
            </p>
        </form>
    </body>
</html>
```
Внутри элемента `select` помещаются элементы `option` - элементы списка. Каждый элемент `option` содержит атрибут `value`, который хранит значение элемента. При этом значение элемента `option` не обязательно должно совпадать с отображаемым им текстом. Например:

```
<option value="apple">iPhone 6S</option>
```
С помощью атрибута `selected` мы можем установить выбранный по умолчанию элемент - это необязательно должен быть первый элемент в списке:
```
<select id="phone" name="phone">
    <option value="iphone 6s">iPhone 6S</option>
    <option value="lumia 950">Lumia 950</option>
    <option value="nexus 5x" selected>Nexus 5X</option>
</select>
```
С помощью другого атрибута `disabled` можно запретить выбор определенного элемента. Как правило, элементы с этим атрибутом служат для создания заголовков:
```
<select id="phone" name="phone">
    <option disabled selected>Выберите модель</option>
    <option value="iphone 6s">iPhone 6S</option>
    <option value="lumia 950">Lumia 950</option>
    <option value="nexus 5x" selected>Nexus 5X</option>
</select>
```
Для создания списка с множественным выбором к элементу select надо добавить атрибут multiple:
```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Элемент select в HTML5</title>
    </head>
    <body>
        <form method="get">
            <p>
                <label for="phone">Выберите модель:</label>
<br/>

<select multiple id="phone" name="phone">
 <option value="iphone 6s">iPhone 6S</option>
 <option value="lumia 950">Lumia 950</option>
 <option value="nexus 5x">Nexus 5X</option>
 <option value="galaxy s7">Galaxy S7</option>
</select>
            </p>
            <p>
                <input type="submit" value="Отправить" />
            </p>
        </form>
    </body>
</html>
```
Зажав клавишу Ctrl, мы можем выбрать в таком списке несколько элементов.
Select также позволяет группировать элементы с помощью тега `<optgroup>`:
```
<!DOCTYPE html>
```

```
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Элемент select в HTML5</title>
    </head>
    <body>
        <form method="get">
            <p>
                <label for="phone">Выберите модель:</label>

<select id="phone" name="phone">
 <optgroup label="Apple">
  <option value="iphone 6s">iPhone 6S</option>
  <option value="iphone 6s plus">iPhone 6S Plus</option>
  <option value="iphone 5se">iPhone 5SE</option>
 </optgroup>
 <optgroup label="Microsoft">
  <option value="lumia 950">Lumia 950</option>
  <option value="lumia 950 xl">Lumia 950 XL</option>
  <option value="lumia 650">Lumia 650</option>
 </optgroup>
</select>
            </p>
            <p>
                <input type="submit" value="Отправить" />
            </p>
        </form>
    </body>
</html>
```

Использование групп элементов применимо как к выпадающему списку, так и к списку со множественным выбором.

### *5.12. Многострочное текстовое поле Textarea*

Элемент `<input type="text"/>` позволяет создавать простое однострочное текстовое поле. Однако возможностей этого элемента по вводу текста бывает недостаточно, и в этой ситуации мы можем использовать многострочное текстовое поле, представленное элементом `textarea`:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Textarea в HTML5</title>
    </head>
    <body>
<form method="get">
```

```
 <p>
  <label for="comment">Ваш комментарий:</label><br/>
  <textarea name="comment" id="comment"
    placeholder="Не более 200 символов"
    maxlength="200"></textarea>
 </p>
 <p>
  <input type="submit" value="Добавить" />
 </p>
</form>
    </body>
</html>
```

С помощью дополнительных атрибутов `cols` и `rows` можно задать соответственно количество столбцов и строк:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Textarea в HTML5</title>
    </head>
    <body>
<form method="get">
 <p>
  <label for="comment">Ваш комментарий:</label><br/>
  <textarea id="comment" name="comment"
   placeholder="Написать комментарий"
   cols="30" rows="7"></textarea>
 </p>
 <p>
  <input type="submit" value="Добавить" />
 </p>
</form>
</body>
</html>
```

Если требуется текст по умолчанию, он набирается между внутри тега `textarea`.

### *5.13. Валидация форм*

Итак, в нашем распоряжении имеются различные элементы, которые мы можем использовать в форме. Мы можем вводить в них различные значения. Однако нередко пользователи вводят не совсем корректные значения: например, ожидается ввод чисел, а пользователь вводит буквы и т.д. И для предупреждения и проверки некорректного ввода в HTML5 существует механизм валидации.

Преимущество использования валидации в HTML5 заключается в том, что пользователь после некорректного ввода может сразу получить сообщение об ошибке и внести соответствующие изменения в введенные данные.

Для создания валидации у элементов форм HTML5 используется ряд атрибутов:

- `required`: требует обязательного ввода значения. Для элементов `textarea`, `select`, `input` (с типом `text`, `password`, `checkbox`, `radio`, `file`, `datetime-local`, `date`, `month`, `time`, `week`, `number`, `email`, `url`, `search`, `tel`);
- `min` и `max`: минимально и максимально допустимые значения. Для элемента `input` с типом `datetime-local`, `date`, `month`, `time`, `week`, `number`, `range`;
- `pattern`: задает шаблон, которому должны соответствовать вводимые данные. Для элемента `input` с типом `text`, `password`, `email`, `url`, `search`, `tel`

Атрибут `required` требует **обязательного наличия** значения:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Валидация в HTML5</title>
    </head>
    <body>
<form method="get">
 <p>
  <label for="login">Логин:</label>
  <input type="text" required id="login" name="login" />
 </p>
 <p>
  <label for="password">Пароль:</label>
  <input type="password" required id="password"
    name="password" />
 </p>
 <p>
  <input type="submit" value="Отправить" />
 </p>
</form>
</body>
</html>
```

Если мы не введем в эти поля никаких данных, оставив их пустыми, и нажмем на кнопку отправки, то браузер высветит нам сообщения об ошибке, а данные не будут отправлены на сервер. В зависимости от браузера визуализация сообщения может несколько отличаться. Также границы некорректного поля ввода могут окрашиваться в красный цвет.

Для **ограничения диапазона вводимых значений** применяются атрибуты `max` и `min`:

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

```
    <head>
        <meta charset="utf-8">
        <title>Валидация в HTML5</title>
    </head>
    <body>
        <form method="get">
            <p>
                <label for="age">Возраст:</label>
                <input type="number" min="1" max="100"
                 value="18" id="age" name="age"/>
            </p>
            <p>
                <input type="submit" value="Отправить" />
            </p>
        </form>
    </body>
</html>
```

Атрибут `pattern` задает **шаблон, которому должны соответствовать данные**. Для определения шаблона используется язык так называемых регулярных выражений. Рассмотрим простейшие примеры:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Валидация в HTML5</title>
    </head>
    <body>
<form method="get">
 <p>
  <label for="phone">Телефон:</label>
  <input type="text" placeholder="+1-234-567-8901"
    pattern="\+\d-\d{3}-\d{3}-\d{4}" id="phone" name="phone" />
 </p>
 <p>
  <input type="submit" value="Отправить" />
 </p>
</form>
</body>
</html>
```

Здесь для ввода номера телефона используется регулярное выражение `\+\d-\d{3}-\d{3}-\d{4}`. Оно означает, что первым элементом в номере должен идти знак плюс `+`. Выражение `\d` представляет любую цифру от `0` до `9`. Выражение `\d{3}` означает три подряд идущих цифры, а `\d{4}` - четыре цифры подряд. То есть это выражение будет соответствовать номеру телефона в формате `"+1-234-567-8901"`.

Если мы введем данные, которые не соответствуют этому шаблону, и нажмем на отправку, то браузер отобразит ошибку.

Не всегда валидация является желаемой, иногда требуется ее **отключить**. И в этом случае мы можем использовать либо у элемента формы атрибут `novalidate`, либо у кнопки отправки атрибут `formnovalidate`:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Валидация в HTML5</title>
    </head>
    <body>
<form novalidate method="get">
 <p>
  <label for="phone">Телефон:</label>
  <input type="text" placeholder="+1-234-567-8901"
   pattern="\+\d-\d{3}-\d{3}-\d{4}" id="phone" name="phone" />
 </p>
 <p>
  <input type="submit" value="Отправить" formnovalidate />
 </p>
</form>
</body>
</html>
```

### *5.14. Элементы fieldset и legend*

Для группировки элементов формы нередко применяется элемент `fieldset`. Он создает границу вокруг вложенных элементов, как бы создавая из них группу. Вместе с ним используется элемент `legend`, который устанавливает заголовок для группы элементов:

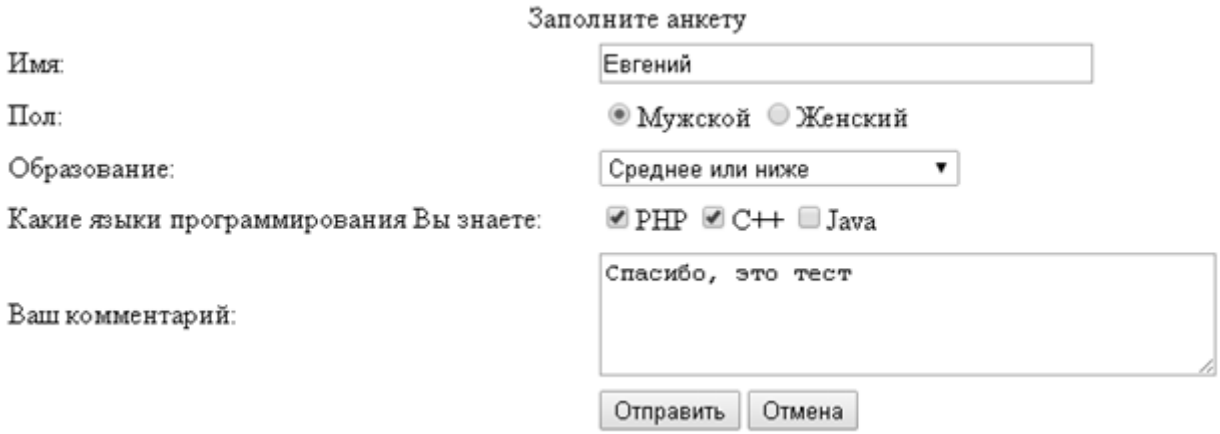
```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="utf-8">
        <title>Элементы форм в HTML5</title>
    </head>
    <body>
        <h2>Вход на сайт</h2>
<form>
 <fieldset>
  <legend>Введите данные:</legend>
  <label for="login">Логин:</label><br>
  <input type="text" name="login" id="login" /><br>
  <label for="password">Пароль:</label><br>
```

```
  <input type="password" name="password" id="password" /><br>
  <input type="submit" value="Авторизация">
 </fieldset>
</form>
</body>
</html>
```

При необходимости мы можем создать на одной форме несколько групп с помощью элементов `fieldset`.

# Задание к лабораторной работе 5:

Разработать форму HTML, предназначенную для ввода анкетных данных о студенте, например:




Добавьте к форме поля ввода данных HTML5 (например, дата рождения, E-mail). Проверьте на валидность и в работе полученный код.