Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Е. А. РУЖИЦКАЯ, Е. П. КЕЧКО

ОСНОВЫ WEB-ТЕХНОЛОГИЙ: РАБОТА С ДАННЫМИ ФОРМЫ В PHP

Практическое пособие

для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий»,
1-40 04 01 «Информатика и технологии программирования»

Гомель
ГГУ им. Ф. Скорины
2022

УДК 004.774(076)
ББК 32.973.434
Р837

Рецензенты:

кандидат физико-математических наук Е. В. Зубей,
кандидат физико-математических наук Т. И. Васильева

Рекомендовано к изданию научно-методическим советом учреждения образования «Гомельский государственный университет имени Франциска Скорины»

Ружицкая, Е. А.

Р837 Основы web-технологий : работа с данными формы в PHP : практическое пособие / Е. А. Ружицкая, Е. П. Кечко ; Гомельский гос. ун-т им. Ф. Скорины. – Гомель : ГГУ им. Ф. Скорины, 2022. – 47 с.

ISBN 978-985-577-883-8

Практическое пособие предназначено для оказания помощи студентам при выполнении заданий по теме «Работа с данными формы в PHP», которая изучается в рамках дисциплины «Основы web-технологий» на 2 курсе во 2 семестре. В нем излагается теоретический материал, даны практические задания и примеры их выполнения.

Адресовано студентам 2 курса специальностей 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 04 01 «Информатика и технологии программирования».

УДК 004.774(076)
ББК 32.973.434

ISBN 978-985-577-883-8

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Формы – это элементы HTML страницы, которые используются для ввода и передачи данных, например, для регистрации пользователей, заполнения анкет, оформления заказа в интернет-магазине и т. д. Через формы можно отправлять как простую текстовую информацию, так и файлы. Эти данные могут быть получены в PHP и обработаны нужным образом.

HTML описывает то, из каких элементов состоит форма, и как она выглядит. Затем данные с формы отправляются на сервер, который принимает эти данные и обрабатывает, например записывает в файлы или базу данных и формирует ответ.

PHP содержит множество средств для работы с формами. Это позволяет очень просто решать типичные задачи, которые часто возникают в web-программировании:

– регистрация и аутентификация пользователя;

– отправка комментариев на форумах и социальных сетях;

– оформление заказов.

Практически любой современный сайт содержит, как минимум, несколько разных HTML-форм.

Практическое пособие адресовано студентам 2 курса специальностей 1-40 01 01 «Программное обеспечение информационных технологий», 1-40 04 01 «Информатика и технологии программирования».

ТЕМА 1. ФОРМА И ЕЁ ЭЛЕМЕНТЫ HTML5

1.1 Формат задания формы

Форма (англ. *form*) – раздел документа, позволяющий пользователю вводить информацию для последующей обработки системой. Формы являются основным средством для ввода и передачи данных интерактивным элементам сайта, например, сценариям.

Формат задания формы:

```
<form name="имя_формы"
      action="URL"
      method="метод_передачи_данных"
      enctype="тип_кодировки"
      target="значение">
   Содержание формы
</form>
```

В качестве параметра атрибута `action` в кавычках указывается строка вызова скрипта, который использует данная форма (при нажатии кнопки `submit`), например, `"http://www.myserver.by/имя_сценария.php"`. Если в качестве значения атрибута `action` указать обращение к электронной почте, например, `action=mailto:ivanov@gsu.unibel.by` браузер автоматически отправит результаты, введенные в форму, по указанному адресу. Для корректной интерпретации данных используется параметр `enctype="text/plain"`.

Если атрибут `action` отсутствует, то в качестве его значения подставляется URL самого документа, т. е. после отправки формы текущая страница перезагружается, возвращая все элементы формы к их значениям по умолчанию.

Значение атрибута `method` устанавливает метод передачи данных из формы на сервер: `"GET"` или `"POST"`.

При задании значения `"GET"` передача данных происходит в один этап адресной строкой. Пары «имя=значение» присоединяются в этом случае к адресу после вопросительного знака и разделяются между собой амперсандом (&). Метод `"GET"` используется для передачи небольших текстовых данных, поиска по сайту.

При задании значения `"POST"` передача данных происходит, как минимум, в два этапа: браузер устанавливает связь с сервером, указанным в

атрибуте action; затем отдельной передачей происходит посылка дополнительных данных. Переданные данные не отображаются в адресной строке. Метод "POST" используется для передачи данных форм, имеющих много длинных полей, пересылки файлов (фотографий, архивов, программ и др.), отправки комментариев, добавления и редактирования сообщений на форуме или блоге.

Атрибут "enctype" устанавливает тип для данных, отправляемых вместе с формой. Обычно не требуется определять значения параметра enctype, однако если используется поле для отправки файла (INPUT type=file), то следует задать параметр enctype ="multipart/form-data".

Атрибут "target" определяет окно, в которое будет загружаться итоговая web-страница:

target=_blank – загрузка страницы в новое окно браузера;

target=_self – загрузка страницы в текущее окно (используется по умолчанию).

1.2 Основные элементы формы

Содержание формы описывается тегом <input>, который имеет следующий формат:

```
<input type="тип элемента"
       name="имя элемента"
       id="уникальный идентификатор"
       value="строка"
       size="целое число"
       checked
       maxlength="целое число"
       align="значение"
       src="URL"
       tabindex="значение"
       autocomplete="on | off"
       autofocus
       disabled
       readonly
       list="идентификатор"
       max="максимальное число"
       min="минимальное число"
       step="число"
```

```
multiple
pattern="выражение"
placeholder="текст-заместитель"
required>
```

Атрибуты тега `input`:

`type` задает тип элемента формы;

`name` задает уникальное имя для каждого элемента формы;

`value` указывает первоначальное значение текущего поля;

`size` определяет размер текстового поля в символах;

`checked` устанавливает выделенный объект из нескольких в случае, если значением атрибута `type` является `radio` или `checkbox`;

`maxlength` определяет максимально возможную длину текстового поля в символах для полей ввода текста;

`align` определяет положение элементов формы на web-странице;

`src` используется совместно с атрибутом `type=image` и задает `URL` нужного изображения;

`tabindex` позволяет установить порядок перемещения фокуса по элементам формы при нажатии клавиши табуляции;

`autocomplete` – по умолчанию большинство браузеров помогают пользователю вводить данные с применением функции автозаполнения значений в полях формы;

`autocomplete=on` включает автозаполнение текста;

`autocomplete=off` отключает автозаполнение (используется для отмены сохранения в браузере паролей, номеров банковских карт, а также редко вводимых или уникальных данных (капча).

Автозаполнение можно отключить также для всей формы (но не для набора полей), воспользовавшись в объявлении формы атрибутом `autocomplete`:

```
<form method="POST" autocomplete="off">
```

`autofocus` автоматически устанавливает фокус в поле формы (по умолчанию атрибут `autofocus` не установлен). Этот атрибут должен использоваться в форме только один раз;

`disabled` блокирует доступ и изменение поля формы (оно отображается серым и недоступным для активации пользователем и не может получить фокус);

`readonly` устанавливает, что поле не может изменяться пользователем;

`list="идентификатор"` указывает на список вариантов, созданный с помощью тега `<datalist>`, который можно выбирать при наборе текста

(изначально этот список скрыт и становится доступным при получении полем фокуса);

`max` – максимальное значение для ввода числа или даты;

`min` – минимальное значение для ввода числа или даты;

`step` – шаг приращения для числовых полей;

`multiple` позволяет указывать одновременно несколько файлов в поле для загрузки файлов, а также реализовать список со множественным выбором;

`pattern` устанавливает шаблон ввода – регулярное выражение, согласно которому требуется вводить и проверять данные в поле формы (если присутствует атрибут `pattern`, то форма не будет отправляться, пока поле не будет заполнено правильно). Примеры регулярных выражений:

```
Введите имя по формату: Ivanov Ivan
<input type=text
       name="name"
       pattern="([a-zA-Z]{3,30}\s*)+[a-zA-Z]{3,30}"
       placeholder="Ivanov Ivan">
Введите имя по формату: Ivanov I.I.
<input type=text
       name="name"
       pattern="([a-zA-Z]{3,30}\s*)+[a-zA-Z]\.[a-zA-Z]\."
       placeholder="Ivanov I.I.">
```

`placeholder` выводит подсказывающий текст – текст внутри поля формы, который исчезает при получении фокуса. Если данные в поле не введены, а фокус переведен в другое место, текст-заместитель останется в поле. Придать стиль тексту атрибута `placeholder` можно с помощью псевдокласса `:placeholder-shown`, например:

```
input:placeholder-shown {
/* стилевые свойства */
}
```

`required` устанавливает поле формы обязательным для заполнения перед отправкой формы на сервер. Выделить обязательные для заполнения поля можно с помощью псевдокласса `:required`, например:

```
input:required {
/* стилевые свойства */
}
```

Задать стилевые свойства обязательного для заполнения поля только при получении им фокуса можно с использованием псевдоклассов `:focus` и `:required`, например:

```
input:focus:required {
/* стилевые свойства */
}
```

На рисунке 1.1 представлены элементы формы.

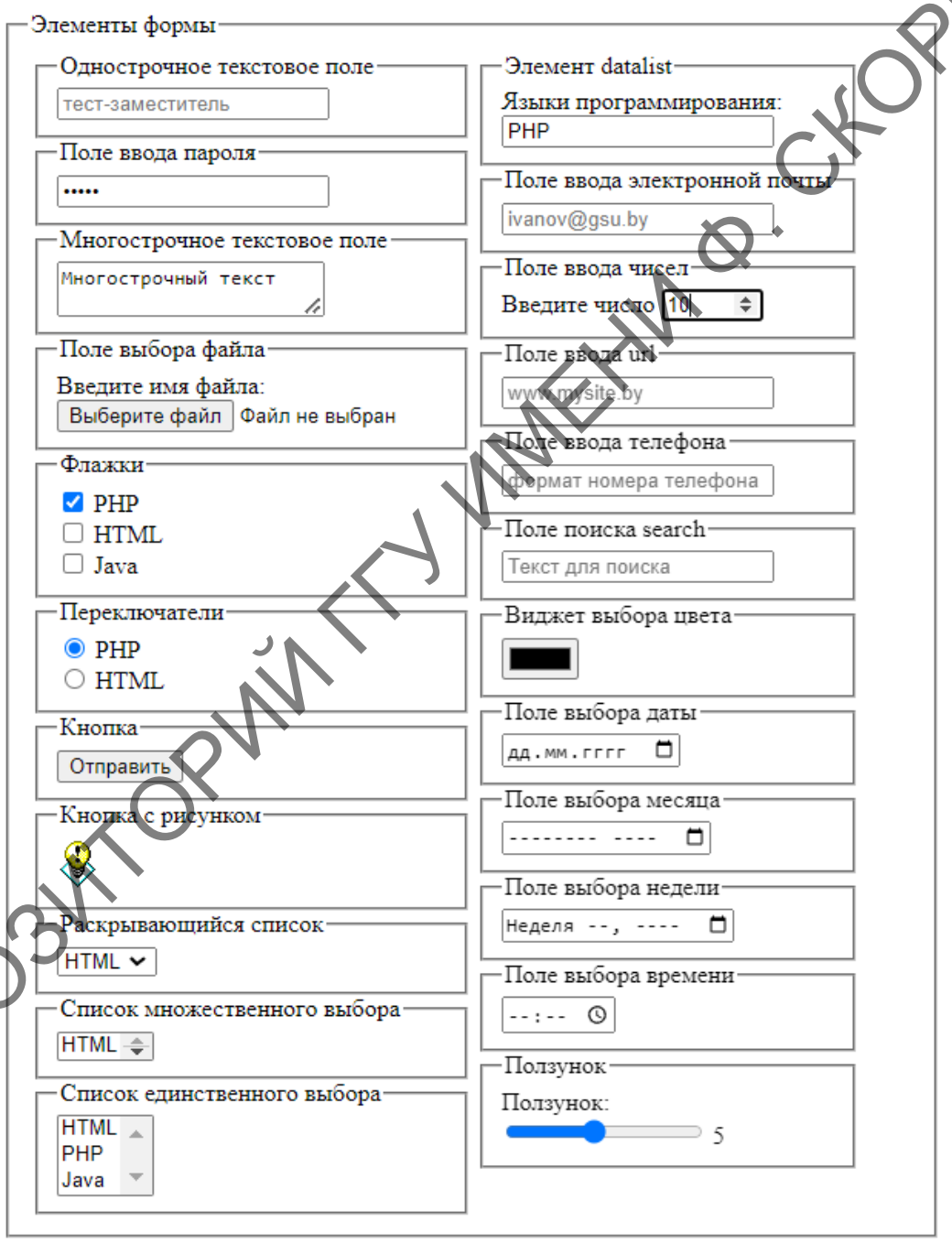


Рисунок 1.1 – Элементы формы

Однострочное текстовое поле. Оно называется *полем ввода* или *полем редактирования* и предназначено для ввода пользователем строки текста. Формат записи:

```
<input type="text"
       name="имя элемента"
       value="начальный текст, содержащийся в поле"
       placeholder="текст-заместитель"
       size="ширина поля"
       maxlength="максимальное количество вводимых символов"
       required>
```

Пример:

```
<input type=text
       size=40 name=user_name
       placeholder="Введите имя">
```

Поле ввода пароля. Это обычное текстовое поле, вводимый текст в котором отображается звездочками. Формат задания:

```
<input type=password
       name="имя элемента"
       value="начальный текст, содержащийся в поле"
       size="ширина поля"
       maxlength="максимальное количество вводимых символов">
```

Скрытые поля. Это поля, которые не отображаются на странице и встраиваются в HTML-файл, когда необходимо передать серверу техническую информацию. Скрытые поля служат доступной альтернативой файлам cookies – специальным файлам, в которых сохраняются индивидуальные настройки пользователя, и позволяющим, например, восстановить последнее состояние формы при повторном посещении пользователем содержащей эту форму страницы. Формат задания:

```
<input type=hidden
       name="имя элемента"
       value="текст, содержащийся в поле">
```

Пример:

```
<input type=hidden name="form_main" value="c3576-236-21">
```

Многострочное текстовое поле. Используется для передачи текста большого размера. Формат задания:

```
<textarea name="имя элемента"
          rows="целое число"
          cols="целое число"
          wrap=значение
          disabled
          readonly>
  Текст, выводимый в текстовом поле по умолчанию
</textarea>
```

Атрибуты тега:

`rows` и `cols` указывают соответственно максимально допустимое количество строк вводимого текста и символов в строке. В случае, если набираемый пользователем текст не умещается в видимую часть текстового контейнера, по краям поля появляются вертикальные и горизонтальные полосы прокрутки;

`wrap` управляет переносом слов и имеет следующие значения:

`wrap=off` – запрет автоматического переноса, при этом сохраняются переносы, определенные пользователем;

`wrap=virtual` – перенос слов при отображении браузером, а серверу введенные данные передаются одной строкой;

`wrap=physical` – сохраняется перенос слов как при отображении браузером, так и при передаче серверу;

`disabled` блокирует доступ и изменение текстового поля (поле *отображается серым цветом*, недоступно для активации пользователем и не может получить фокус);

`readonly` – текстовое поле недоступно для изменения пользователем, в него не допускается вводить новый текст или модифицировать существующий, оно не может получить фокус, однако поле отображается обычным цветом.

Пример:

```
<textarea name="message"
          rows=25
          cols=40>
  Текст сообщения
</textarea>
```

Поле выбора файлов. Генерирует на экране кнопку, при нажатии на которую на экране появляется Проводник Windows, позволяющий присоединить к отсылаемым на сервер данным любой файл с локального компьютера пользователя. Рядом с кнопкой отображается небольшое текстовое поле, куда автоматически заносится имя отсылаемого файла и путь к нему на локальном диске. Поле выбора файлов работает корректно лишь при методе пересылки `POST` и формате кодировки `multipart/form-data`. Формат задания:

```
<input type=file
       name="имя элемента"
       size="ширина поля"
       maxlength="максимальная длина текста">
```

Пример:

```
<form enctype="multipart/form-data"
      action="URL"
      method=POST>
  Введите имя файла:<BR>
  <input type=file name=myfile> <br>
  <input type=submit value="Отправить">
</form
```

Флажки. Флажки используют, когда необходимо выбрать два или более варианта из предложенного списка. Элемент представляет собой простую форму выбора, принимающую одно из двух состояний: «отмечено» – «не отмечено». Несколько флажков могут объединяться в группу, которая будет отвечать набору параметров выбора. Данный элемент оперирует *с булевыми переменными*, то есть переменными, каждая из которых может принимать значение `true` или `false`. Формат задания:

```
<input type=checkbox
       name="имя флажка или группы флажков"
       value="значение установленного флажка"
       checked
       title="всплывающая подсказка">
```

По умолчанию начальное положение флажка считается не установленным. Чтобы задать начальное положение установленного флажка, надо дополнить его атрибутом `checked`.

Значением установленного флажка является строка, заданная атрибутом `value`.

Пример задания группы флажков:

```
<input type=checkbox
       name=lang_prog
       value="html"
       checked> HTML<br>
<input type=checkbox
       name=lang_prog
       value="php"> PHP<br>
<input type=checkbox
       name=lang_prog
       value="java"> Java<br>
```

В рассмотренном примере надписи рядом с флажками созданы как простой текст. Для того чтобы флажок устанавливался не только щелчком непосредственно по квадратику флажка, но и щелчком по надписи, необходимо связать надпись с флажком с помощью тега `label`, в котором содержится ссылка на связанный элемент управления с помощью атрибута `for`. Этому атрибуту ставится в соответствие идентификатор `id`.

Пример связывания надписи с флажком:

```
<input type=checkbox
       id=id_html
       name=lang_prog
       value="html"
       checked>
<label for=id_html> HTML</label>
<br>
<input type=checkbox
       id=id_php
       name=lang_prog
       value="php">
<label for=id_php> PHP</label>
<br>
<input type=checkbox
       id=id_java
       name=lang_prog
       value="java">
<label for=id_java> Java</label>
<br>
```

Переключатели (кнопки выбора или радиокнопки) применяются в случае, когда какая-либо логическая переменная может принимать только одно значение из множества возможных. Формат задания:

```
<input type=radio
       name="имя переключателя или группы"
       value="значение установленного переключателя"
       checked
       title="всплывающая подсказка">
```

Все элементы `radio` одной группы обозначаются одним и тем же значением атрибута `name`. Использование радиокнопок требует явного указания значений атрибута `value`, одна из кнопок должна быть выделена атрибутом `checked`. Если атрибут `checked` не присвоен ни одному из переключателей группы, браузер при загрузке установит по умолчанию первый переключатель. При обработке формы на сервере будет отправлено значение установленного переключателя.

Пример использования:

```
<input type=radio
       name=auto
       value="bmw"
       checked> BMW<br>
<input type=radio
       name=auto
       value="hyunday"
       checked> Hyundai Creta<br>
```

Как и в случае флажков, надписи можно связать с соответствующими переключателями так, чтобы каждый переключатель устанавливался при щелчке по надписи. Для связывания, каждому переключателю должен быть присвоен уникальный идентификатор, а все переключатели должны образовывать группу с определенным именем.

Пример связывания надписи с переключателем:

```
<input type=radio
       id=id_bmw
       name=auto
       value="bmw"
       checked>
<label for=id_bmw> BMW</label><br>
<input type=radio
```

```
        id=id_hyunday
        name=auto
        value="hyunday"
        checked>
<label for=id_hyunday> Hyundai Creta</label><br>
```

Кнопки. Это элементы управления, которые используются для представления формы (`submit`), сброса данных формы (кнопка `reset`), создания эффектов для кнопки (`button`). Кнопку можно создать двумя способами.

1. Использование тега `input`. Формат задания:

```
<input type=button
        name="имя элемента"
        value="надпись на кнопке">
```

2. Использование тега `button`. На таких кнопках можно размещать любые элементы HTML, в том числе изображения и таблицы, и изменять вид кнопки. Формат задания:

```
<button>
   надпись или изображение
</button>
```

В HTML предусмотрены два типа кнопок, которые создаются без использования значения `button`. Это кнопки специального назначения: Подача запроса (`submit`) и Сброс (`reset`).

Кнопка `submit` предназначена для запуска процедуры передачи формы на сервер. Формат задания:

```
<input type=submit
        name="имя элемента"
        value="надпись на кнопке">
```

или

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

Если атрибут `value` отсутствует, то кнопка по умолчанию в браузере Chrome имеет надпись «Отправить». В форме можно применять несколько кнопок `submit`.

Кнопка `reset` предназначена для приведения формы в начальное положение (сброс всех введенных данных). Формат задания:

```
<input type=reset
       name="имя элемента"
       value="надпись на кнопке">
```

или

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

Если атрибут `value` отсутствует, то кнопка по умолчанию в браузере Chrome имеет надпись «Сбросить».

Кнопка с изображением. Это кнопка отсылки, аналогичная `submit`, но с использованием графического изображения. Обычно применяется в случаях, когда стандартная серая прямоугольная кнопка «не вписывается» в дизайн сайта. Формат записи:

```
<input type=image
       name="имя элемента"
       src="URL изображения"
       value="надпись на кнопке">
```

В этом случае тег `input` может содержать все атрибуты тега `img`.

Пример задания кнопок:

1. Обычная кнопка

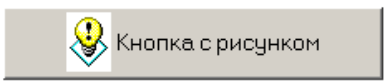
```
<input type=button
       name=press
       value="Нажми меня!">
```

2. Обычная кнопка

```
<button>
   Кнопка с текстом
</button>
```

3. Кнопка с рисунком

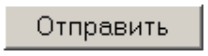
```
<button>
<img src="tips.gif"
```



```
     align=absmiddle>
Кнопка с рисунком
</button>
```

4. Кнопка `submit`

```
<input type=submit
       value="Отправить">
```



5. Кнопка `reset`

```
<input type=reset
       value="Очистить">
```

6. Кнопка с изображением

```
<input type=image
       src="tips.gif">
```

Списки. Эти элементы предоставляют пользователю список вариантов для выбора. Существует три типа списков:

1) *раскрывающийся список*, представляющий собой однострочное поле с треугольной стрелкой, которая раскрывает список;

2) *поле-список*, в котором на экран выводится заданное число строк; для просмотра всех строк список может быть снабжен полосой прокрутки;

3) *список со множественным выбором*, позволяющий, благодаря полосе прокрутки, просматривать все позиции списка и выбирать одновременно несколько позиций.

Формат записи:

```
<select name="имя списка"
        size="целое число"
        multiple>
   <option value="значение"
           selected>
      Пункт 1
   </option>
</select>
```

Атрибуты тега `select`:

`multiple` включает режим выбора нескольких элементов из списка, т. е. определяет список со множественным выбором;

`size` устанавливает высоту списка. Если значение `size=1`, то список становится раскрывающимся. При добавлении параметра `multiple` список отображается как «крутилка»;

`selected` делает текущий элемент списка выделенным.

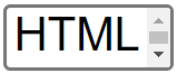
Примеры задания списка:

1. Раскрывающийся список

```
<select name=lang_prog>
  <option value=html>HTML</option>
  <option value=PHP>PHP</option>
  <option value=Java>Java</option>
</select>
```

2. Список множественного выбора

```
<select multiple size=1>
  <option value=html>HTML</option>
  <option value=PHP>PHP</option>
  <option value=Java>Java</option>
</select>
```



3. Список единственного выбора

```
<select size=3>
  <option value=html>HTML</option>
  <option value=PHP>PHP</option>
  <option value=Java>Java</option>
</select>
```

Атрибут `list` и связанный с ним элемент `datalist` позволяют ряду вариантов выбора быть представленными пользователю сразу же, как только он начнет вводить значение в поле.

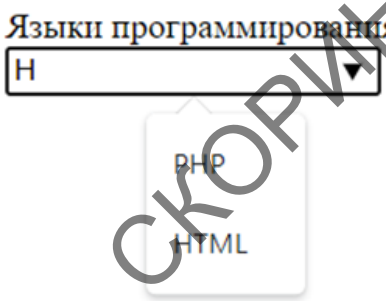
Значение, указываемое для атрибута `list`, является ссылкой на идентификатор элемента `datalist`. Таким способом `datalist` привязывается к полю ввода. Хотя это поле ввода ничем не отличается от обычного поля, предназначенного для ввода текста, при наборе в нем данных под ним появляется окно выбора с наиболее подходящими результатами из перечня, указанного в `datalist`.

Атрибут `list` и элемент `datalist` не мешают пользователю ввести в поле какой-нибудь другой текст, но они предоставляют еще один полезный способ добавления общих функциональных возможностей и улучшения

условий работы пользователя, применяя для этого исключительно разметку HTML5.

Пример использования:

```
<div>
  <label for="lang_prog">
    Языки программирования:
  </label>
  <br>
  <input id="lang_prog"
         name="program"
         type="text"
         list="select_progr">
  <datalist id="select_progr">
    <select>
      <option value=""></option>
      <option value="PHP"></option>
      <option value="HTML"></option>
      <option value="Java"></option>
    </select>
  </datalist>
</div>
```



В HTML5 добавлен ряд дополнительных типов информации, которые позволяют наложить ограничения на данные, вводимые пользователями, не прибегая при этом к применению внешнего кода `JavaScript`. Если браузер не поддерживает этот тип, то поля превращаются в обычные текстовые поля ввода.

Поле ввода адреса электронной почты `email`. Поддерживающие это свойство браузеры будут ожидать пользовательского ввода, соответствующего синтаксису электронного адреса. Формат задания:

```
<input type="email"
       name="имя элемента"
       placeholder="адрес электронной почты"
       required>
```

Пример:

```
<label for="my_email">
  Введите адрес электронной почты:
</label>
```

```
<input type="email"
       name="email"
       id="my_email"
       placeholder="ivanov@gsu.by"
       required>
```

Введите адрес электронной почты: ivanov@gsu.by

Поле ввода чисел `number`. Браузеры, поддерживающие это свойство, ожидают ввода числа. А браузеры, поддерживающие поле ввода чисел с возможностью прокрутки их последовательности, предоставляют еще и это средство выбора. Речь идет о небольшом фрагменте пользовательского интерфейса, позволяющем пользователям для изменения вводимого значения просто нажимать клавиши со стрелками на клавиатуре или щелкать указателем мыши на стрелках вверх и вниз.

Формат задания:

```
<input type="number"
       name="имя элемента"
       min="минимальное_значение"
       max="максимальное_значение"
       step="шаг"
       required>
```

Пример:

```
<label for="my_number">
  Введите число
</label>
<input type="number"
       id="my_number"
       name="enter_number"
       min="5"
       max="30"
       step=5
       required>
```

Введите число 15

Поле ввода web-адреса `url`. Тип вводимых данных `url` предназначен для ввода значений URL-адресов. Формат задания:

```
<input type="url"
       name="имя элемента"
```

```
        placeholder="URL-адрес"
        required>
```

Пример:

```
<label for="web">
  Введите URL-адрес:
</label>
<br>
<input type="url"
       id="web"
       name="web"
       placeholder="www.mysite.by">
```

Введите URL-адрес:
www.mysite.by

Поле ввода номера телефона `tel`. Формат задания:

```
<input type="tel"
       name="имя элемента"
       placeholder="формат номера телефона"
       autocomplete="off | on"
       required>
```

Пример:

```
<label for="tel">
  Телефон:
</label>
<br>
<input type="tel"
       id="tel"
       name="tel"
       placeholder="+375-29-111-11-11"
       autocomplete="off"
       required>
```

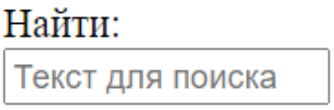
Телефон:
+375-29-111-11-11

Поле поиска `search`. Поля с этим типом ведут себя так же, как стандартные поля ввода текста. Формат задания:

```
<input type="search"
       name="имя элемента"
       placeholder="Текст для поиска"
       required>
```

Пример:

```
<label for="my_search">
  Найти:
</label>
<br>
<input type="search"
       id="my_search"
       name="search"
       placeholder="Текст для поиска">
```

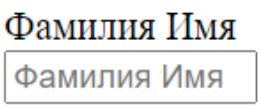


Использование атрибута `pattern`. Поле ввода можно настроить на ожидание ввода конкретного шаблона. Атрибут `pattern` позволяет с использованием регулярного выражения указать синтаксис данных, разрешенных для ввода в данном поле. Формат задания:

```
<input pattern="регулярное выражение"
       name="имя элемента"
       placeholder="Пример задания"
       required>
```

Пример:

```
<label for="name">
  Фамилия Имя
</label>
<br>
<input id="name"
       name="name"
       pattern="([a-zA-Z]{3,30}\s*)+
                [a-zA- Z]{3,30}"
       placeholder="Фамилия Имя"
       required>
```

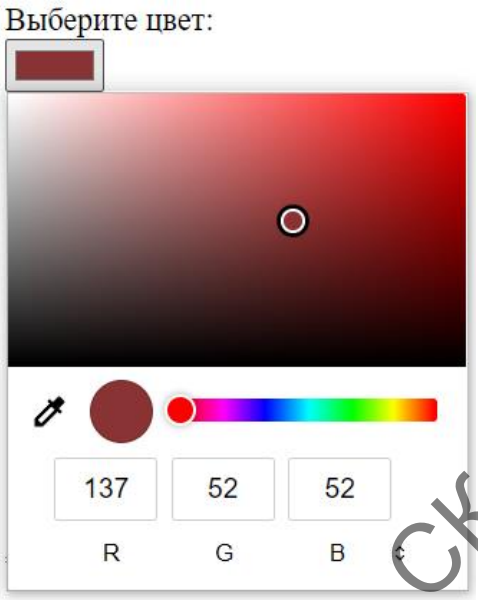


Виджет выбора цвета `color`. Тип ввода `color` инициирует в поддерживающих его браузерах появление панели выбора цвета, позволяя пользователям выбирать значения цвета в виде шестнадцатеричного числа. Формат задания:

```
<input type="color"
       name="имя элемента">
```

Пример:

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

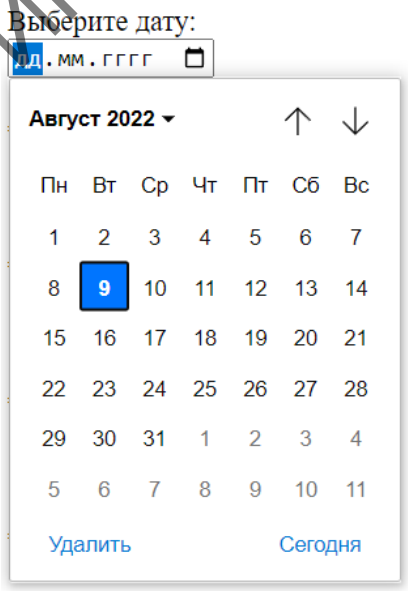


Поле выбора календарной даты `date`. Формат задания:

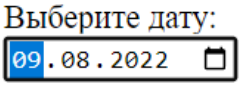
```
<input type="date"
       name="имя_переменной">
```

Пример:

```
<label for="my_date">
  Выберите дату:
</label><br>
<input type="date"
       id="my_date"
       name="my_date">
```



Результат выбора:

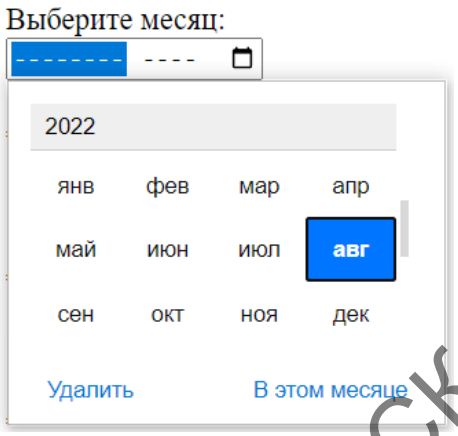


Поле выбора месяца `month`. Интерфейс позволяет выбрать месяц и предоставляет ввод в виде года и месяца, например 2022-08. Формат задания:

```
<input type="month"
       name="имя_переменной">
```

Пример:

```
<label for="my_month">
  Выберите месяц:
</label><br>
<input type="month"
       id="my_month"
       name="my_month">
```



Результат выбора:

Выберите месяц:

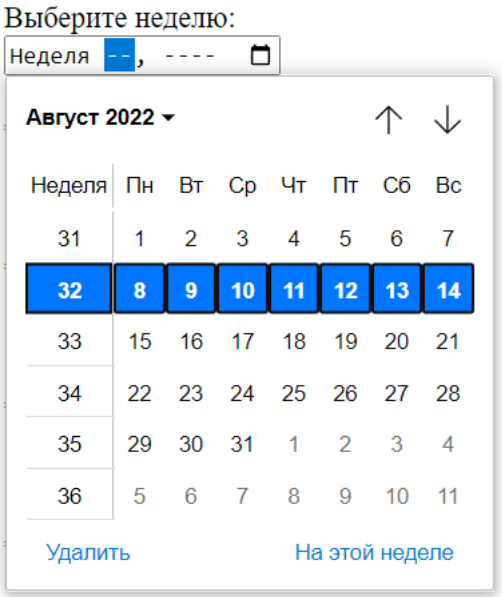
Август 2022

Поле выбора недели `week`. Панель позволяет выбрать неделю года и предоставляет ввод в формате 2022-W47. Формат задания:

```
<input type="week"
       name="имя_переменной">
```

Пример:

```
<label for="my_week">
  Выберите неделю:
</label><br>
<input type="week"
       id="my_week"
       name="my_week">
```



Результат выбора:

Выберите неделю:

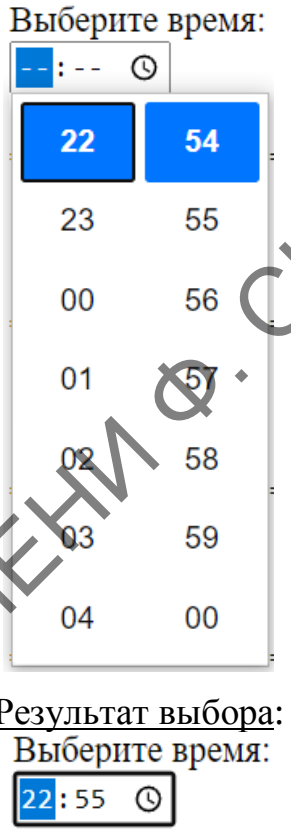
Неделя 32, 2022

Поле выбора времени `time`. Поле позволяет вводить значение в 24-часовом формате, например 23:50. Формат задания:

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

Пример:

```
<label for="my_time">
  Выберите время:
</label>
<br>
<input type="time"
       id="my_time"
       name="my_time">
```



Результат выбора:

Ползунок для выбора чисел в указанном диапазоне `range`. По умолчанию устанавливается диапазон от 0 до 100. Величины `min`, `max` и `step` задают соответственно минимальное, максимальное значения и шаг изменения. Формат задания:

```
<input type="range"
       name="имя элемента"
       min="минимальное значение"
       max="максимальное значение"
       step="шаг"
       value="начальное значение">
```

Однако при перемещении ползунка пользователь не видит текущего значения. Для отображения текущего значения можно использовать небольшой фрагмент кода JavaScript:

```
<script>
  function showValue(newValue) {
    document.getElementById("range").innerHTML=newValue;
  }
</script>
```

Пример:

```
<label for="my_range">
  Ползунок:
</label><br>
<input type="range"
       id="my_range"
       name="my_range"
       min="1"
       max="10"
       value="5"
       onchange="showValue(this.value)">
<span id="range">5</span>
```

Группирование элементов управления. Чтобы страница имела законченный вид, элементы формы должны быть распределены по группам. Формат задания группы:

```
<fieldset>
   <legend> Легенда группы элементов  </legend>
…
</fieldset>
```

Теги <legend> выводят надпись, которая помещается в разрыв рамки, обрамляющей группу.

Пример:

```
<fieldset>
  <legend>Флажки</legend>
  <input type=checkbox checked>
    HTML
  <br>
  <input type=checkbox>
    PHP <br>
  <input type=checkbox>
    Java <br>
</fieldset>
```

Практическое задание

Для заданной предметной области разработать три формы, содержащие все основные элементы. Поля формы должны быть сгруппированы. При открытии формы предусмотреть установку фокуса на первое поле. Форма должна содержать обязательные и необязательные поля, отключение автозаполнения, текст-заместитель. Обязательные поля выделить с использованием стилевых свойств. При вводе элементов формы предусмотреть проверку заполнения обязательных полей.

Предметные области:

1. ГАИ.
2. Кинотеатр.
3. Библиотека.
4. Компьютеры и комплектующие.
5. Сотрудники предприятия.
6. Студенты университета.
7. Поликлиника.
8. Детский садик.
9. Магазин.
10. Школа.
11. Аптека.
12. Продажа авиабилетов.
13. Расписание поездов.
14. Автовокзал.
15. Приемная комиссия.
16. Гаражный кооператив.
17. Дачный кооператив.
18. Почта.
19. Парикмахерская.
20. Банк.
21. Кафе.
22. Продажа автомобилей.
23. Продажа недвижимости.
24. Фабрика «Спартак».
25. Пиццерия.

ТЕМА 2. ПЕРЕДАЧА ДАННЫХ ФОРМЫ НА СЕРВЕР

2.1 Трансляция полей формы в переменные

Одним из способов передачи данных с локального компьютера на серверный является форма. Рассмотрим передачу данных с простейшей формы, например, для продажи автомобиля (рисунок 2.1).

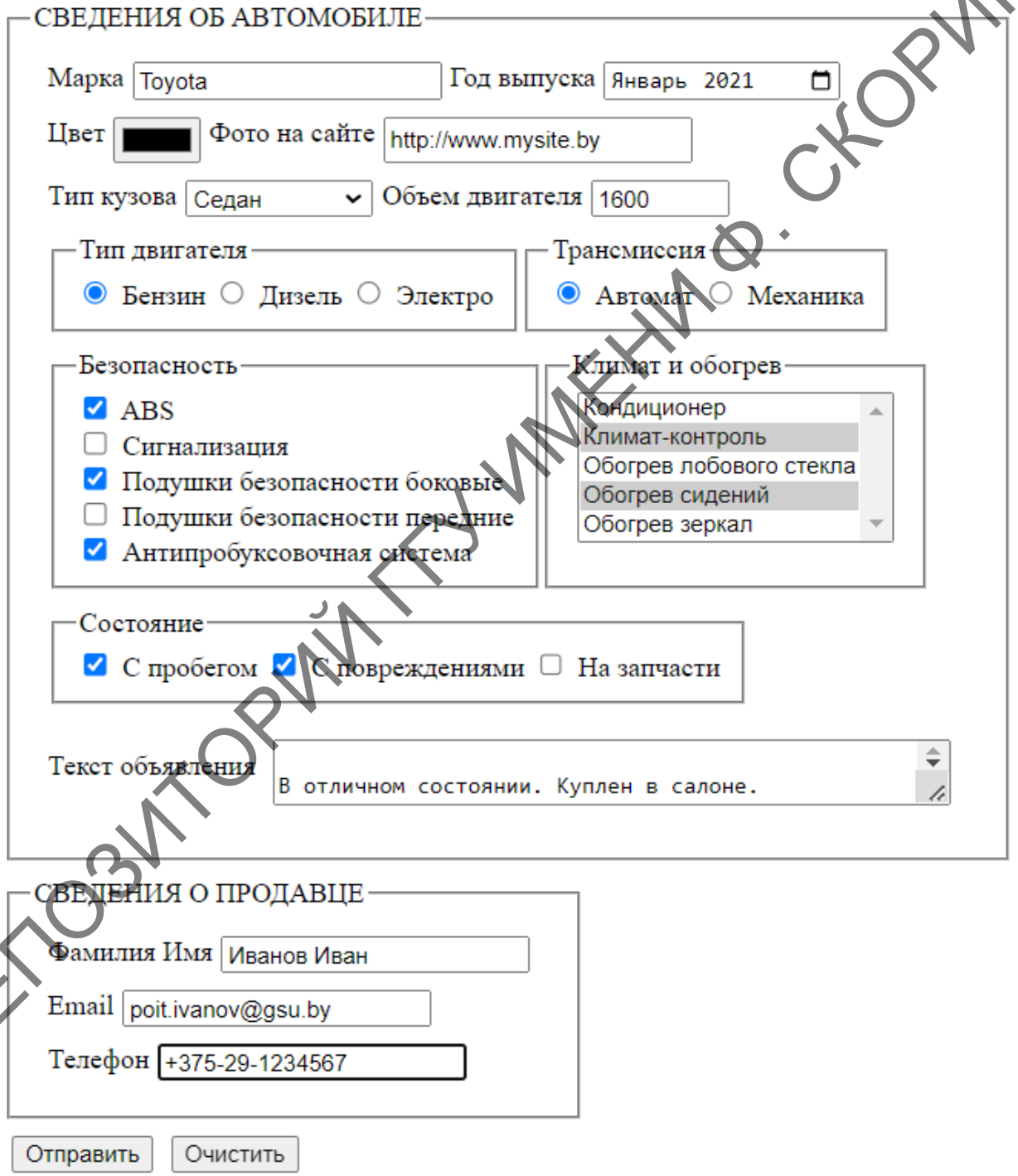


Рисунок 2.1 – Форма продажи автомобиля

Для того чтобы пользователь мог передавать данные, форма должна содержать следующие элементы:

1. В теге форм должен быть атрибут `action`, в котором указывается имя скрипта, загружаемого при отправке данных. Если имя не указывается, то происходит перезагрузка текущего документа.

2. Метод передачи данных: `GET` или `POST`. По умолчанию используется `GET`.

3. Все элементы форм, введенные значения которых передаются на сервер, должны иметь атрибут `name`. В нем указывается имя переменной, которая создается при отправке данных. В этой переменной будут содержаться введенные пользователем данные.

Пример задания формы: часть формы для продажи автомобиля, содержащая сведения о продавце:

```
<form action="answer.php" method="POST">
  Фамилия Имя
  <input type=text
         name="seller_fio"
         pattern="([А-Я][а-я]{3,30}\s*)+[А-Я][а-я]{3,}"
         placeholder="Иванов Иван"
         required>
  <input type=submit value="Отправить">
</form>
```

После отправки формы браузер автоматически обратится к сценарию `answer.php` и передаст все атрибуты, расположенные внутри тега `<input>`. Так как в атрибуте `action` тега `<form>` задан относительный путь, то сценарий `answer.php` будет искаться браузером в том же самом каталоге, что и файл с исходной формой.

Все перекодирования и преобразования, которые нужны для URL-кодирования данных, осуществляются браузером автоматически. В частности, буквы кириллицы превратятся в %xx, где xx – шестнадцатеричное число, обозначающее код символа.

Все данные из полей формы интерпретатор преобразует в глобальные *одноименные* переменные. Значение поля `name` после начала работы программы будет храниться в переменной `$name`.

После передачи данных из формы, в `answer.php` введенные данные будут доступны через глобальные массивы `$_GET`, `$_POST`, `$_REQUEST`. В `$_GET` будут находиться данные, отправленные методом `GET`, в `$_POST` будут лежать данные, отправленные методом `POST`, а в `$_REQUEST` – данные,

отправленные и тем, и другим методом одновременно. Название, указанное в атрибуте name, будет доступно в качестве ключа глобального массива в зависимости от используемого метода передачи данных. Так как данные нашей формы передаются методом POST, введенное значение будет находиться одновременно в двух массивах $_POST и $_REQUEST по ключу seller_fio: $_POST['seller_fio'] или $_REQUEST['seller_fio']. В answer.php в окне браузера отобразим введенное имя.

Листинг answer.php (вывод переменной seller_fio):

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta http-equiv="content-type"
          content="charset=utf-8" />
  </head>
  <body>
    <?php
      echo "<hr><b>СВЕДЕНИЯ О ПРОДАВЦЕ</b><br><hr>";
      echo "<b>Фамилия: </b>".$_POST['seller_fio']."<br>";
    ?>
  </body>
</html>
```

Теперь изменим код таким образом, чтобы при запуске без параметров сценарий выдавал документ с формой, а при нажатии кнопки – выводил нужный текст. Для того чтобы определить, был ли сценарий запущен без параметров, нужно проверить, существует ли переменная с именем, совпадающим с именем кнопки отправки. Если такая переменная существует, то пользователь запустил программу, нажав на кнопку.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta http-equiv="content-type"
          content="charset=utf-8" />
  </head>
  <body>
    <?php
    if(!@$doGo){
    ?>
        <form action="<?=$SCRIPT_NAME?>" method="POST">
```

```
            Фамилия Имя
            <input type=text
                   name="seller_fio"
             pattern="([А-Я][а-я]{3,30}\s*)+[А-Я][а-я]{3,}"
                   placeholder="Иванов Иван"
                   required>
            <input type=submit
                   name="doGo"
                   value="Отправить">
          </form>
    <?php
      } else {
      echo "<hr><b>СВЕДЕНИЯ О ПРОДАВЦЕ</b><br><hr>";
      echo "<b>Фамилия: </b>".$_POST['seller_fio']."<br>";
            }
    ?>
  </body>
</html>
```

Конструкция `<?=выражение?>` является более коротким обозначением для `<?echo(выражение)?>`, и предназначена для того, чтобы вставлять величины прямо в HTML-страницу.

В параметре `action` тега `<form>` явно не задано имя файла сценария, оно извлекается из переменной `$SCRIPT_NAME` (которая устанавливается автоматически перед запуском сценария). Это позволило не «привязываться» к имени файла, т. е. теперь можно его в любой момент переименовать без потери функциональности.

Если PHP установлен не как модуль Apache, а как отдельный обработчик, то переменная `$SCRIPT_NAME` будет содержать не то значение, на которое мы рассчитываем.

К тому же, теперь исчезла необходимость и в промежуточном файле, содержащем форму, его код встроен в сценарий.

Для отключения выдачи ошибок в окно браузера используется оператор @(отключение ошибок). Если разместить этот оператор перед любым выражением, то сообщения об ошибках в этом выражении будут подавлены и в окне браузера не отображены.

2.2 Трансляция списков

Механизм трансляции полей формы в PHP работает приемлемо, когда среди них нет полей с одинаковыми именами. Если же таковые встречаются,

то в переменную записываются только данные последнего встретившегося поля. Это неудобно при работе с элементами checkbox и списком множественного выбора:

```
<fieldset style="display:inline">
  <legend>Безопасность</legend>
  <input type="checkbox"
         name="car_security"
         value="ABS"
         checked> ABS<br>
  <input type="checkbox"
         name="car_security"
         value="Сигнализация"> Сигнализация<br>
  <input type="checkbox"
         name="car_security"
         value="Подушки безопасности боковые"
         checked> Подушки безопасности боковые<br>
  <input type="checkbox"
         name="car_security"
         value="Подушки безопасности передние">
         Подушки безопасности передние<br>
  <input type="checkbox"
         name="car_security"
         value="Антипробуксовочная система"
         checked> Антипробуксовочная система<br>
</fieldset>
<fieldset style="display:inline">
  <legend>Климат и обогрев</legend>
  <select size=5
          multiple
          name="car_climate">
    <option value="Кондиционер">
      Кондиционер </option>
    <option value="Климат-контроль" selected>
      Климат-контроль
    </option>
    <option value="Обогрев лобового стекла">
      Обогрев лобового стекла
    </option>
    <option value="Обогрев сидений"
            selected>
      Обогрев сидений
    </option>
```

```
    <option value="Обогрев зеркал">
      Обогрев зеркал
    </option>
  </select>
</fieldset>
```

В элементах форм checkbox и список множественного выбора можно выбрать (подсветить) не один, а сразу несколько значений. Пусть в элементе checkbox выбраны значения "ABS", "Подушки безопасности боковые", "Антипробуксовочная система", а в списке со множенственным выбором значения – "Климат-контроль" и "Обогрев сидений". Тогда после отправки формы сценарию придет строка параметров

```
car_security=ABS&car_security=Подушки безопасности
боковые&car_security=Антипробуксовочная система
&car_climate=Климат-контроль&car_climate=Обогрев сидений
```

и в переменной car_security окажется только "Антипробуксовочная система", а в переменной car_climate только значение "Обогрев сидений". Для решения подобных проблем в PHP предусмотрена возможность давать имена полям формы в виде индексных массивов, например, car_security[] и car_climate[]:

```
<fieldset style="display:inline">
  <legend>Безопасность</legend>
  <input type="checkbox"
         name="car_security[]"
         value="ABS"
         checked> ABS<br>
  <input type="checkbox"
         name="car_security[]"
         value="Сигнализация">
    Сигнализация<br>
  <input type="checkbox"
         name="car_security[]"
         value="Подушки безопасности боковые"
         checked>
    Подушки безопасности боковые<br>
  <input type="checkbox"
         name="car_security[]"
         value="Подушки безопасности передние">
```

```
    Подушки безопасности передние<br>
  <input type="checkbox"
         name="car_security[]"
         value="Антипробуксовочная система"
         checked>
    Антипробуксовочная система<br>
</fieldset>
<fieldset style="display:inline">
  <legend>Климат и обогрев</legend>
  <select size=5
          multiple
          name="car_climate[]">
    <option value="Кондиционер">
      Кондиционер
    </option>
    <option value="Климат-контроль"
            selected>
      Климат-контроль
    </option>
    <option value="Обогрев лобового стекла">
      Обогрев лобового стекла
    </option>
    <option value="Обогрев сидений"
            selected>
       Обогрев сидений
    </option>
    <option value="Обогрев зеркал">
      Обогрев зеркал
    </option>
  </select>
</fieldset>
```

Теперь сценарию придет строка

```
car_security=ABS&car_security=Подушки безопасности
боковые&car_security=Антипробуксовочная система
&car_climate=Климат-контроль&car_climate=Обогрев сидений
```

и интерпретатор обнаружит, что необходимо создать «автомассив» (то есть массив, который не содержит пропусков и у которого индексация начинается с нуля), и, действительно, создаст переменные `$car_security` и `$car_climate` типа массив, содержащие следующие элементы:

```
$car_security = array (
    0=>"ABS",
    1=>"Подушки безопасности боковые",
    2=>"Антипробуксовочная система"),
$car_climate = array (
    0=>"Климат-контроль",
    1=>"Обогрев сидений").
```

Теперь эти массивы можно обрабатывать с помощью сценария.

Выводить элементы автомассива можно разными способами. Рассмотрим два их них:

1. Использование функции `implode()`.

Функция `implode()` используется для слияния нескольких строк в одну большую результирующую строку, вставляя между ними указанный разделитель:

```
string implode (string $glue, list $List)
```

Функция берет ассоциативный массив (обычно это список) `$List`, заданный во втором параметре, и «склеивает» его значения при помощи «строки-клея» `$glue` в первом параметре.

Вместо списка во втором аргументе можно передавать любой ассоциативный массив – в этом случае будут рассматриваться только его значения.

2. Просмотреть элементы автомассива можно с помощью цикла `for`. Для определения количества элементов используется функция `count()`.

Листинг `answer.php` (вывод переменных `$car_security` и `$car_climate`):

```
<?php
  // 1-й способ работы с автомассивом
  echo "<b>Безопасность: </b>";
  $carsecurity=implode(', ',$_POST['car_security']);
  echo $carsecurity."<br>";
  // 2-й способ работы с автомассивом
  echo "<b>Климат и обогрев: </b>";
  $carclimate=$_POST['car_climate'];
  $kol=count($_POST['car_climate']);
  for($i=0; $i<$kol; $i++)
    echo $carclimate[$i]."; ";
  echo "<br>";
?>
```

2.3 Трансляция ассоциативных массивов

Использование ассоциативных массивов – еще один способ передачи данных. *Ассоциативный массив*, или его еще называют *хэш*, – это набор из нескольких элементов, каждый их которых представляет собой пару вида

```
ключ=>значение
```

(символом => обозначается соответствие определенному ключу какого-то значения). Доступ к отдельным элементам осуществляется указанием их ключа. В отличие от автомассивов, ключами могут служить не только целые числа, начиная с нуля, но и любые строки.

Рассмотрим часть формы, введенные значения которой передаются с помощью ассоциативных массивов:

```
<fieldset>
  <legend>Состояние</legend>
  <input type="checkbox"
         name="car_station['with_mileage']"
         value="С пробегом"
         checked> С пробегом
  <input type="checkbox"
         name="car_station['without_mileage']"
         value="С повреждениями"
         checked> С повреждениями
  <input type="checkbox"
         name="car_station['for_parts']"
         value="На запчасти"> На запчасти
</fieldset>
```

После передачи данных сценарию на PHP в нем будет инициализирован ассоциативный массив `$car_station` с ключами `with_mileage`, `without_mileage` и `for_parts`. То есть, имена полям формы можно давать не только простые, но и представленные в виде одномерных ассоциативных массивов.

Для просмотра элементов ассоциативного массива можно использовать три способа:

1. Косвенный перебор элементов массива.

Для перебора такого массива можно воспользоваться следующей конструкцией:

```
<?php
// Косвенный перебор элементов массива
  echo "<b>Состояние: </b>";
  $carstation =$_POST['car_station'];
  for(Reset($carstation); $k=key($carstation);
      Next($carstation)){
    echo "$carstation[$k] <br>";
  }
?>
```

Кроме того, что массивы являются направленными, в них есть еще и такое понятие, как текущий элемент. Функция `key`() возвращает ключ, который имеет текущий элемент (если он указывает на конец массива, возвращается пустая строка, что позволяет использовать вызов `key`() в контексте второго выражения `for`).

Функции `Reset`() и `Next`() возвращают следующие значения:

– функция `Reset`() возвращает значение первого элемента массива (или пустую строку, если массив пуст);

– функция `Next`() возвращает значение элемента, следующего за текущим (или пустую строку, если такого элемента нет).

Если необходим перебор элементов массива с конца, то можно воспользоваться следующей конструкцией:

```
for(End($carstation); $k=key($carstation);
    Prev($carstation)){
  echo "$carstation[$k] <br>";
}
```

Функция `End`() устанавливает позицию текущего элемента в конец массива, а `Prev`() передвигает ее на один элемент назад.

Функция `current`() возвращает не ключ, а величину текущего элемента (если он не указывает на конец массива).

Основное *достоинство* косвенного перебора – «читабельность» и ясность кода, а также то, что массив можно перебрать как в одну, так и в другую сторону.

Недостатки косвенного перебора:

– вложенные циклы. Нельзя одновременно перебирать массив в двух вложенных циклах или функциях, так как второй вложенный `for` «испортит» положение текущего элемента у первого `for'a`;

– нулевой ключ. Если в массиве встретится ключ 0, то в этом случае выражение `key($carstation)` будет равно нулю и цикл оборвется.

2. Классический прямой перебор массива. В косвенном переборе сначала вычисляется очередной ключ, а затем по нему косвенно находится значение элемента массива, при прямом переборе на каждом «витке» цикла одновременно получается и ключ, и значение текущего элемента.

Перебрать массив `$car_station` при помощи прямого перебора можно следующим образом:

```
<?php
// Классический прямой перебор массива
  $carstation=$_POST['car_station'];
  for(Reset($carstation); list($k,$v)=each($carstation);
    /*пусто*/)
    echo "$v<br>";
?>
```

В начале заголовка цикла используется функция `Reset()`. Дальше переменным `$k` и `$v` присваивается результат работы функции `each`(). Третье условие цикла просто отсутствует.

Функция `each`(), во-первых, возвращает список, нулевой элемент которого хранит величину ключа текущего элемента массива `$carstation`, а первый – значение текущего элемента. Во-вторых, она продвигает указатель текущего элемента к следующей позиции. Если следующего элемента в массиве нет, то функция возвращает не список, а `false`. Поэтому она и размещена в условии цикла `for` и не указан третий блок операторов в цикле `for`: он просто не нужен, ведь указатель на текущий элемент и так смещается функцией `each` ().

Инструкция `list()` используется для присвоения переменным `$k,$v` очередного ключа и соответствующего ему значения элемента массива `$carstation`.

3. Прямой перебор в стиле PHP. Прямой перебор массивов можно осуществить с помощью цикла `foreach`. Данный тип цикла предназначен специально для перебора всех элементов ассоциативного массива. Формат:

```
foreach (массив as $key=>$value)
  команды;
```

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара `ключ=>значение` оказывается в переменных `$key` и `$value`.

У цикла `foreach` имеется и другая форма записи, которую следует применять, когда нас не интересует значение ключа очередного элемента:

```
foreach (массив as $value)
   команды;
```

В этом случае доступно лишь *значение* очередного элемента массива, но не его ключ. Такой цикл применяется для работы с массивами-списками.

Цикл `foreach` оперирует не исходным массивом, а его *копией*. Это означает, что любые изменения, которые вносятся в массив, не могут быть «видны» из тела цикла. Что позволяет, например, в качестве массива использовать не только переменную, но и результат работы какой-нибудь функции, возвращающей массив (в этом случае функция будет вызвана всего один раз – до начала цикла, а затем работа будет производиться с копией возвращенного значения).

```
<?php
// Прямой перебор в стиле PHP
  echo "<b>Состояние: </b>";
  $carstation=$_POST['car_station'];
  foreach($carstation as $v)
    echo " $v;";
  echo "<br>";
?>
```

Этот способ перебора работает с максимально возможной скоростью – даже быстрее, чем перебор списка при помощи `for` и числового счетчика.

2.4 Код программы передачи данных формы

Файл form.php

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta http-equiv="content-type"
          content="charset=utf-8" />
     <link href="form.css"
          rel="stylesheet"
          type="text/css">
  </head>
  <body>
```

```
<form action="answer.php" method="POST">
  <div>
    <fieldset>
      <legend>СВЕДЕНИЯ ОБ АВТОМОБИЛЕ</legend>
      <div>
        <label for="car_model_id">Марка</label>
        <input id="car_model_id"
               name="car_model"
               type="text"
               list="car_model_list">
        <datalist id="car_model_list">
          <select>
            <option value="Toyota"></option>
            <option value="BMW"></option>
            <option value="Opel"></option>
            <option value="Reno"></option>
            <option value="Audi"></option>
            <option value="Hunday"></option>
          </select>
        </datalist>
        <label for="car_month_id">Год выпуска</label>
        <input id="car_month_id"
               type="month"
               name="car_month">
      </div>
      <div>
        <label for="car_color_id">Цвет</label>
        <input type="color"
               id="car_color_id"
               name="car_color">
        <label for="car_url_id">Фото на сайте</label>
        <input type="url"
               id="car_url_id"
               name="car_url"
               placeholder="http://www.mysite.by">
      </div>
      <div>
        Тип кузова
        <select name="car_type_body">
          <option value="Седан">
             Седан
          </option>
          <option value="Внедорожник">
            Внедорожник
```

```
      </option>
      <option value="Кабриолет">
        Кабриолет
      </option>
    </select>
    <label for="engine_volume_id">
      Объем двигателя
    </label>
    <input id="engine_volume_id"
           name="car_engine_volume"
           type="number"
           min="1200"
           max="2400"
           step="100">
  </div>
  <div>
    <fieldset style="display:inline">
      <legend>Тип двигателя</legend>
      <input type="radio"
             name="car_engine"
             value="Бензин"> Бензин
      <input type="radio"
             name="car_engine"
             value="Дизель"> Дизель
      <input type="radio"
             name="car_engine"
             value="Электро"> Электро
    </fieldset>
    <fieldset style="display:inline">
      <legend>Трансмиссия</legend>
      <input type="radio"
             name="car_transmission"
             value="Автомат"> Автомат
      <input type="radio"
             name="car_transmission"
             value="Механика"> Механика
    </fieldset>
  </div>
  <div>
    <fieldset style="display:inline">
      <legend>Безопасность</legend>
      <input type="checkbox"
             name="car_security[]"
             value="ABS">
```

```
        ABS<br>
      <input type="checkbox"
             name="car_security[]"
             value="Сигнализация">
        Сигнализация<br>
      <input type="checkbox"
             name="car_security[]"
             value="Подушки безопасности боковые">
        Подушки безопасности боковые<br>
      <input type="checkbox"
             name="car_security[]"
             value="Подушки безопасности
                    передние">
        Подушки безопасности передние<br>
      <input type="checkbox"
             name="car_security[]"
             value="Антипробуксовочная система">
        Антипробуксовочная система<br>
    </fieldset>
    <fieldset style="display:inline">
      <legend>Климат и обогрев</legend>
      <select size=5
              multiple
              name="car_climate[]">
        <option value="Кондиционер">
          Кондиционер
        </option>
        <option value="Климат-контроль">
          Климат-контроль
        </option>
        <option value="Обогрев лобового стекла">
          Обогрев лобового стекла
        </option>
        <option value="Обогрев сидений">
           Обогрев сидений
        </option>
        <option value="Обогрев зеркал">
          Обогрев зеркал
        </option>
      </select>
    </fieldset>
  </div>
  <div>
    <fieldset>
```

```
          <legend>Состояние</legend>
          <input type="checkbox"
                 name="car_station['with_mileage']"
                 value="С пробегом">
            С пробегом
          <input type="checkbox"
                name="car_station['without_mileage']"
                 value="С повреждениями" >
            С повреждениями
          <input type="checkbox"
                 name="car_station['for_parts']"
                 value="На запчасти">
            На запчасти
        </fieldset>
      </div>
      <div>
        <p>Текст объявления</p>
        <div>
          <textarea cols=50 rows=2 name="car_text">
            В отличном состоянии. Куплен в салоне.
          </textarea>
        </div>
      </div>
    </fieldset>
  </div>
  <div>
    <fieldset>
      <legend>СВЕДЕНИЯ О ПРОДАВЦЕ</legend>
      <div>
        Фамилия Имя
        <input type=text
               name="seller_fio"
     pattern="([А-Я][а-я]{3,30}\s*)+[А-Я][а-я]{3,30}"
               placeholder="Иванов Иван">
      </div>
      <div>
        <label for="email">Email</label>
        <input id="email"
               name="seller_email"
               type="email"
               placeholder="poit.ivanov@gsu.by"
               required>
      </div>
      <div>
```

```
            <label for="tel">Телефон</label>
            <input id="tel"
                   name="seller_tel"
                   type="tel"
                   placeholder="+375-29-1234567"
                   pattern="^\+\d{3}[-.]?\d{2}[-.]?\d{7}\b"
                   autocomplete="off"
                   required>
          </div>
        </fieldset>
      </div>
      <div>
        <input type=submit value="Отправить">
        <input type=reset value="Очистить">
      </div>
    </form>
  </body>
</html>
```

Файл form.css

```
div {
margin: 10px;
display:flex;
justify-content: flex-start;
}
input, select{
margin-left: 5px;
margin-right: 5px;
}
```

Файл answer.php

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta http-equiv="content-type"
          content="charset=utf-8" />
  </head>
  <body>
    <?php
      echo "<b>СВЕДЕНИЯ ОБ АВТОМОБИЛЕ</b><br><hr>";
      echo "<b>Марка: </b>".$_POST['car_model']."<br>";
      echo "<b>Год выпуска: </b>".
           $_POST['car_month']."<br>";
      echo "<b>Цвет: </b>".$_POST['car_color']."<br>";
```

```
echo "<b>Фото на сайте: </b>".
     $_POST['car_url']."<br>";
echo "<b>Тип кузова: </b>".
     $_POST['car_type_body']."<br>";
echo "<b>Объем двигателя: </b>".
     $_POST['car_engine_volume']."<br>";
echo "<b>Тип двигателя: </b>".
     $_POST['car_engine']."<br>";
echo "<b>Трансмиссия: </b>".
     $_POST['car_transmission']."<br>";
// 1-ый способ работы с автомассивом
echo "<b>Безопасность: </b>";
$carsecurity=implode(', ',$_POST['car_security']);
echo $carsecurity."<br>";
// 2-ой способ работы с автомассивом
echo "<b>Климат и обогрев: </b>";
$carclimate=$_POST['car_climate'];
$kol = count ($_POST['car_climate']);
for($i=0; $i<$kol; $i++)
  echo $carclimate[$i].", ";
      echo "<br>";
// Прямой перебор в стиле PHP
echo "<b>Состояние: </b>";
$carstation=$_POST['car_station'];
foreach($carstation as $v)
  echo " $v;";
echo "<br>";
echo "<b>Текст объявления: </b>";
echo $_POST['car_text']."</br>";
echo "<hr><b>СВЕДЕНИЯ О ПРОДАВЦЕ</b><br><hr>";
echo "<b>Фамилия: </b>".
     $_POST['seller_fio']."<br>";
echo "<b>Email: </b>".
     $_POST['seller_email']."<br>";
echo "<b>Телефон: </b>".
     $_POST['seller_tel']."<br>";
?>
</body>
</html>
```

Результат работы программы представлен на рисунке 2.2.

СВЕДЕНИЯ ОБ АВТОМОБИЛЕ

Марка: Toyota
Год выпуска: 2021-01
Цвет: #000000
Фото на сайте: http://www.mysite.by
Тип кузова: Седан
Объем двигателя: 1600
Тип двигателя: Бензин
Трансмиссия: Автомат
Безопасность: ABS, Подушки безопасности боковые, Антипробуксовочная система
Климат и обогрев: Климат-контроль; Обогрев сидений;
Состояние: С пробегом; С повреждениями;
Текст объявления: В отличном состоянии. Куплен в салоне.

СВЕДЕНИЯ О ПРОДАВЦЕ

Фамилия: Иванов Иван
Email: poit.ivanov@gsu.by
Телефон: +375-29-1234567

Рисунок 2.2 – Результат передачи данных с формы на сервер

Практическое задание

Для заданной предметной области разработать три формы, содержащие все основные элементы. Каждая форма должна загружаться на отдельной странице. На каждой форме предусмотреть две кнопки: «Отправить» и «Очистить».

При нажатии кнопки «Отправить» введенные данные должны передаваться на сервер.

При нажатии кнопки «Очистить» введенные пользователем данные убираются и форма принимает первоначальный вид со всеми значениями, установленными по умолчанию.

Написать две программы. Первая содержит форму и скрипт в разных файлах, вторая – в одном. Использовать три способа передачи данных:

– через переменные,

– индексные массивы,

– ассоциативные массивы.

Переданные данные вместе с подписями из значений выводить в окно браузера.

ЛИТЕРАТУРА

1. Справочник по HTML [Электронный ресурс]. – Режим доступа: http://htmlbook.ru/html/. – Дата доступа: 15.08.2022.

2. Фрэйн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрэйн. – 2-е изд. – СПб. : Питер, 2017. – 272 с.

3. Котеров, Д. В. PHP 7 / Д. В. Котеров, И. В. Симдянов. – СПб. : БХВ-Петербург, 2016. – 1088 с.

4. Руководство по PHP [Электронный ресурс]. – Режим доступа: https://www.php.net/manual/ru/tutorial.forms.php. – Дата доступа: 01.08.2022.

Производственно-практическое издание

Ружицкая Елена Адольфовна,
Кечко Елена Петровна

ОСНОВЫ WEB-ТЕХНОЛОГИЙ: РАБОТА С ДАННЫМИ ФОРМЫ В PHP

Практическое пособие

Редактор А. А. Банчук
Корректор В. В. Калугина

Подписано в печать 11.10.2022. Формат 60x84 1/16.
Бумага офсетная. Ризография. Усл. печ. л. 2,8.
Уч-изд. л. 3,1. Тираж 10 экз. Заказ 479.

Издатель и полиграфическое исполнение:
учреждение образования
«Гомельский государственный университет
имени Франциска Скорины».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1452 от 17.04.2017.
Специальное разрешение (лицензия) № 02330 / 450 от 18.12.2013.
Ул. Советская, 104, 246028, Гомель.