**Лабораторная работа №8 формы**

**Формы**

Формы — это часть языка HTML. Формы нужны для передачи данных от клиента на сервер. Чаще всего формы используются для регистрации пользователей, заполнения анкет, оформления заказа в интернет магазине, и так далее.  
Через формы можно отправлять как простую текстовую информацию, так и файлы.  
Большую часть времени программирования на PHP вы будете так или иначе работать с формами и данными из них.

HTML описывает то, из каких элементов состоит форма, и как она выглядит. Но без принимающей стороны, то есть сервера, который принимает эти данные и обрабатывает их нужным образом, создавать формы нет никакого смысла.

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

**Отправка формы**

Рассмотрим один типичный пример — форма обратной связи. Для связи пользователей с авторами сайта, как правило, используются формы обратной связи, где человек указывает имя, почту для обратной связи и текст своего сообщения.  
Такая форма в HTML может выглядеть следующим образом:

1. **<form** **name=**"feedback" **method=**"POST" **action=**"form.php"**>**
2. **<label>**Ваше имя: **<input** **type=**"text" **name=**"name"**></label>**
3. **<label>**Ваш email: **<input** **type=**"text" **name=**"email"**></label>**
4. **<label>**Сообщение: **<textarea** **name=**"message"**></textarea></label>**
5. **<input** **type=**"submit" **name=**"send" **value=**"Отправить"**>**
6. **</form>**

Это очень простая форма, состоящая из трёх полей и одной кнопки отправки.

Почти весь приведённый код описывает внешний вид и содержание формы, но следует обратить внимание на два атрибута тега <form>, которые нужны для указания на способ обработки данных:

* method — этот атрибут используется для определения метода HTTP, который будет использован для передачи данных на сервер. Вы уже знакомы с HTTP-методом GET, предписывающим серверу просто вернуть определённый документ.  
  Метод POST сообщает о намерении передать на сервер некоторую информацию, что, впрочем, не отменяет последующее получение контента.
* action — содержит адрес PHP-скрипта, который должен обработать эту форму.

После нажатия на кнопку «отправить», браузер выполняет POST запрос со введёнными данными на адрес, указанный в атрибуте action.

**Обработка формы**

После отправки формы управление передаётся PHP-скрипту, который должен получить переданные данные, выполнить с ними какие-либо действия (например, сохранить в базе данных) и показать результат.  
Результатом может быть какое-нибудь сообщение об успешном завершении операции, например, *«ваши данные успешно отправлены»*.

Поэтому требуется в первую очередь научиться получать данные из формы в сценарии .  
В PHP это делается легко — **все данные из формы находятся в глобальном ассоциативном массиве $\_POST**.  
Этот массив всегда будет неявно присутствовать в сценарии, если он был загружен по методу POST.  
Каждое поле из формы будет находиться в массиве, где ключом будет значение атрибута name, а значением содержимое поля.  
Например, чтобы вывести из формы всю информацию на экран, можно написать такой сценарий:

1. **<?php**
2. **if** (**isset**($\_POST)) {
3. **echo** "Имя: " . $\_POST['name'];
4. **echo** "<br>Email: " . $\_POST['email'];
5. **echo** "<br>Сообщение: " . $\_POST['message'];
6. }

Функция isset служит для определения, существует ли переданная ей переменная. Так мы проверяем, что сценарий загружен методом POST, то есть была отправлена форма.

Как правило, после обработки формы в PHP, сценарий должен переадресовать пользователя на другую страницу. Это связано с тем, что если форма была отправлена через метод POST, то после обновления страницы данные будут отправлены ещё раз, а это, в большинстве случаев, нежелательное поведение.

**Отправка файлов**

Кроме текстовой информации, существует возможность отправлять на сервер файлы любых типов.  
Пример формы для загрузки файла:

1. **<form** **name=**"file\_upload" **method=**"POST" **action=**"form.php" **enctype=**"multipart/form-data"**>**
2. **<label>**Ваш аватар: **<input** **type=**"file" **name=**"avatar"**></label>**
3. **<input** **type=**"submit" **name=**"send" **value=**"Отправить файл"**>**
4. **</form>**

Тут есть два важных отличия от первого примера:

* Добавился новый атрибут enctype, который всегда должен иметь значениеmultipart/form-data. Если его не будет, то файл не отправится.
* Сам файл загружается при помощи поля с типом «file».

В PHP загруженный файл будет доступен в другом специальном массиве — $\_FILES.

1. **<?php**
2. **if** (**isset**($\_FILES['avatar'])) {
3. $file = $\_FILES['avatar'];
4. **echo** "Загружен файл с именем " . $file['name'] . " и размером " . $file['size'] . " байт";
5. }

PHP автоматически сохраняет все загруженные файлы во временную папку на сервере. Но хранить там файлы нельзя, потому что эта директория периодически очищается, и ссылку на такой файл нельзя дать на сайте. Решение здесь только одно — переместить загруженный файл в другую папку. Перемещение файла всегда выполняют сразу после загрузки.

Для начала нужно убедиться, что в рабочей директории проекта существует папка для хранения загруженных файлов. Пусть она называется uploads.

**Перемещение загруженного файла**

Для перемещения файла нужно знать, где он находится сейчас, и адрес папки, в которую он будет переноситься.  
С текущим адресом всё крайне просто — он уже находится в массиве $\_FILES. Новый адрес файла, в свою очередь, состоит из пути к папке и имени файла. Так как папка uploads находится там же, где и текущий сценарий, получить путь к ней можно так: dirname(\_\_FILE\_\_).

Код для перемещения файла в новую папку:

1. **<?php**
2. $current\_path = $\_FILES['avatar']['tmp\_name'];
3. $filename = $\_FILES['avatar']['name'];
4. $new\_path = dirname(\_\_FILE\_\_) . '/' . $filename;
5. move\_uploaded\_file($current\_path, $new\_path);
6. **?>**

Функция move\_uploaded\_file() выполняет два действия:

1. Проверяет, что файл действительно загружен через форму.
2. Перемещает загруженный файл по новому адресу.

**Валидация формы**

Валидация формы — это проверка содержимого её полей. Задача такой проверки — убедиться, что необходимые поля заполнены, а значения в них соответствуют ожидаемому формату.  
Так, например, при регистрации пользователя на сайте, он должен заполнить поля с адресом электронной почты и придумать себе пароль. Оба поля обязательны к заполнению, но значение из поля email также должно быть корректным email-адресом.  
Помимо текстовых значений формы, можно проверять формат и размер загружаемых файлов.

**Общий подход к валидации**

При выполнении валидации любой формы порядок действий будет всегда одним:

1. Сформировать массив с именами полей, обязательных к заполнению.
2. Сформировать массив с правилами для валидации формата полей.
3. Получить значения всех полей.

**Практика**

**Часть 1**

**Задание 1**. Составьте программу на языке PHP, которая принимает три переменные: Имя, возраст и пол.

Программа должна выводить текст здравствуйте, {Значение переменной Имя}, Вы указали данные: Возраст {Значение переменной Возраст} и Возраст {Значение переменной Пол}

**Задание 2.** Составьте программу на языке PHP, которая принимает три числа. и определяет минимальное и максимальное из них.

**Задание 3.** Составьте программу, принимающую текстовую строку и заменяющую в ней все буквы «а» на буквы «о», а все буквы «к» на буквы «н» и выводит исходное и измененное слово на странице.

**Часть 2**

**Задание 1**

Напишите программу, использующую интерфейс форм, в которой нужно ввести:

1. Имя — текстовое поле
2. Группа — текстовое поле
3. Пол — радиокнопка (Мужской/Женский)
4. День рождения — выпадающий список от1 до 31
5. Месяц — выпадающий список от Января до Декабря
6. Год — выпадающий список от 2021 до 1900 (сортировка по убыванию)
7. Наличие водительского удостоверения — чекбокс

Программа должна выводить возраст и количество лет до пенсии. (Мужчины выходят на пенсию по достижению 65 лет, а женщины в 63).

**Задание 2**

Модернизируйте программу из задания 1 таким образом, чтобы при вводе пустых полей выводилось предупреждение (например: поле имя не заполнено).

**Задание 3**

Модернизируйте программу из задания 2 таким образом, чтобы заполненные поля оставались заполненными, а пустые поля выделялись. Расчет должен производится только если все поля заполнены (кроме ВУ).