**Создание серверных сценариев с использованием технологии PHP**

Разработать скрипт − программу для HTML документа, реализующую следующие функции:

* создание кнопки и связ ее с функцией;
* создание выпадающего окна;
* разработать скрипт, реагирующий на события click и onclick;
* разработать скрипт, загружающий новую страницу;
* разработать скрипт, создающий новое окно;
* разработать скрипт, создающий строку состояния;
* разработать скрипт, реализующий функции таймера;
* разработать скрипт, реализующий функции выделения элемента.

**Практическое задание № 1 Создание серверных сценариев с использованием технологии php.**

**Цель работы:** изучить работу расширения сервера Web PHP и язык интерпретатора PHP. Научиться создавать сценарии стороны сервера, встроенные в документы HTML.

Для работы сервера необходимо установить ПО «XAMMP».

Инструкции PHP внутри документа html располагаются внутри специальных скобок, состоящих из последовательностей символов "<?" и "?>" или "<?php" и "?>". Можно также для включения сценариев PHP использовать элемент SCRIPT с атрибутом language, имеющим значение "php". Текст, находящийся вне специальных скобок и называемый статической частью документа, проигрывателем сценариев PHP не анализируется и передаётся клиенту без изменений. Фактически, статическая часть является шаблоном документа HTML.

Синтаксис языка сценариев PHP похож на синтаксис языков программирования C, C++, Java. Сценарий PHP представляет собой оператор PHP или последовательность операторов PHP, разделённых символами ";".

Основными объектами языка сценариев PHP являются переменные, выражения, операторы и функции. Переменные в PHP не имеют типа. Тип переменной определяется хранимым в данной переменной значением. Тип переменной динамически изменяется в зависимости от контекста. В PHP различаются следующие типы:

* целый − integer;
* действительный − double;
* строковый − string;
* массив − array;
* объект − object.

Преобразование типов осуществляется неявно, например, при подстановке переменных в качестве параметров функций, или явно, при указании типа, к которому нужно преобразовать значение переменной, перед именем.

Переменные различаются по именам. Имя переменной в PHP начинается с символа "$". Для определения переменной нужно присвоить ей значение при помощи операции присваивания. Способы создания нового значения переменной отличаются для различных типов.

Переменной скалярного типа (integer, double, string) можно присвоить в качестве значения соответствующую константу.

Массивы в PHP являются не набором переменных, а переменными некоторого определённого типа array. В языке сценариев PHP массивы ассоциативны. Это означает, что индексом массива может являться не только число, но и строка. Индекс ассоциативного массива называется ключом. Новый массив может быть создан либо при создании элемента массива, либо при помощи вызова array. Для создания нового элемента массива нужно при присваивании значения переменной после имени указать ключ в квадратных скобках. При этом имя переменной будет именем созданного массива, а значение в квадратных скобках− ключом для получения доступа к значению созданного элемента массива. В PHP массивы являются динамическими. При присваивании значения элементу массива проверяется, существует ли уже в массиве элемент с заданным ключом. Если существует, то его значение заменяется на новое. В противном случае создаётся новый элемент массива. Все элементы массива объединены в двусвязный список. В массиве имеется внутренний указатель текущего элемента. Имеется возможность просмотреть текущее значение указателя, а также переместить указатель на предыдущий или последующий элемент в списке, и на начальный или конечный элементы списка. Таким образом, можно осуществить последовательный доступ к каждому элементу массива. Для элементов массива с целочисленными ключами ведётся счётчик индексов. Значением счётчика является увеличенное на единицу максимальное в массиве значение целочисленного ключа (индекса). Если массив не содержит элементов с целочисленными ключами, то счётчик индексов имеет нулевое значение. Если при присваивании значения элементу массива не указывать ключа в квадратных скобках, то ключом для нового элемента будет значение счётчика индексов, который после создания нового элемента увеличится на единицу. Данная возможность позволяет добавлять в конец массива элементы с последовательными индексами.

Для создания массива с несколькими начальными элементами используется ключевое слово array, после которого в круглых скобках перечисляются присваиваемые элементам массива значения. В качестве ключей элементов используется счётчик индексов. Такое назначение ключей по умолчанию можно изменить при помощи операции "=>". Данная операция применяется в следующем виде:

ключ => значение

Если в списке значений array встречается подобная операция, то очередному элементу массива будет присвоен указанный ключ. Данная возможность используется для создания массивов со строковыми ключами, а также для пропуска значений целочисленных ключей (индексов).

Объекты в PHP служат для поддержки объектного программирования. Для создания нового объекта используется вызов new.

Переменным любого типа можно присваивать в качестве значения другие переменные, а также значения, возвращаемые функциями и методами объектов.

В PHP можно создавать ссылки на переменные. Ссылка – это имя переменной, отличное от заданного при определении. Для создания ссылки нужно определить переменную, значением которой будет имя ссылки. Если присвоить значение переменной, в которой хранится имя ссылки, указав перед именем двойной символ "$$", то будет создана ссылка, к которой можно обращаться по имени, как к обычной переменной. При одновременной работе со ссылками и массивами возникает неопределённость, связанная с приоритетами символов "$" и квадратных скобок. Для разрешения данной неопределённости можно использовать фигурные скобки как символы явного определения приоритетов.

При выполнении сценария интерпретатор определяет ряд переменных, связанных с параметрами сервера, протокола HTTP, самого интерпретатора. Каждый орган управления формы, содержащийся в документе HTML, имеет атрибуты name и value. При разборе полученного от клиента запроса интерпретатор PHP создаёт переменные, именами которых являются значения атрибутов name, а значениями − значения атрибутов value соответствующих органов управления.

Важным понятием языка PHP является понятие выражения. Выражение – это константа, переменная или составленная из них при помощи операций конструкция. Выражение может быть составной частью более сложного выражения. Каждое выражение имеет значение, которое называется значением выражения. Значением операции является её результат. Тип значения выражения зависит от типов входящих в выражение переменных и констант, а также от операций.

Помимо типов, которые могут иметь переменные, значения выражений могут быть булевого типа. Выражения булевого типа могут принимать только значения FALSE или TRUE. Значения булевого типа не могут быть присвоены переменным, но иногда подставляются в функции и операторы. При подстановке переменных вместо булевых выражений происходит преобразование типа переменной к булевому. Для целых чисел значение FALSE имеет только 0, для строк − строки "" и "0". Операции языка интерпретатора PHP приведены в табл. 2.1.

Таблица 2.1

Операции PHP

|  |  |
| --- | --- |
| Знак | Операция |
| *Арифметические* | |
| + | Сложение |
| – | Вычитание |
| \* | Умножение |
| / | Деление |
| % | Остаток от деления |
| *Строковые* | |
| . | Конкатенация строк |
| *Присваивание* | |
| = | Обычное присваивание |
| += | Присваивание со сложением |
| –= | Присваивание с вычитанием |
| \*= | Присваивание с умножением |
| /= | Присваивание с делением |
| %= | Присваивание с получением остатка |
| .= | Присваивание с конкатенацией |
| Побитовые | |
| & | Побитовое умножение (И) |
| | | Побитовое сложение (ИЛИ) |
| ~ | Побитовое отрицание (НЕ) |
| *Логические* | |
| and | Логическое И |
| or | Логическое ИЛИ |
| xor | Логическое исключающее ИЛИ |
| ! | Логическое отрицание |
| && | Логическое И |
| || | Логическое ИЛИ |
| *Сравнение* | |
| == | Равно |
| != | Не равно |
| < | Меньше |
| > | Больше |
| <= | Меньше или равно (не больше) |
| >= | Больше или равно (не меньше) |

Сценарий PHP состоит из операторов. Помимо оператора вычисления выражения существует ряд операторов, характерных для структурных языков. Некоторые операторы PHP имеют два синтаксиса: синтаксис языка C и модифицированный синтаксис для вставки HTML−кода внутрь операторов. В таблице 2.2 приведены основные операторы PHP. Операторы записаны с использованием синтаксиса языка C.

Таблица 2.2

Операторы PHP

|  |  |
| --- | --- |
| Синтаксис | Оператор |
| if (cond1) {...}  elseif (cond2) {...}  ...  else {...} | Ветвление |
| while (cond) {...} | Цикл с предусловием (цикл ПОКА) |
| do {...} while (cond) | Цикл с постусловием (цикл ДО) |
| for (init; cond; expr) {...} | Цикл с предусловием |
| break | Безусловный переход на конец операторов while, do while, for и switch |
| continue | Безусловный переход на следующий цикл |
| switch (var) {  case val1: ...  case val2: ...  ...  default: ...  } | Выбор |
| function name(parameters)  {...} | Определение функции |
| return val | Возврат значения |

Синтаксис определения функции в интерпретаторе PHP отличается от синтаксиса языка C. Функция – это набор операторов или часть кода, выполняемая при разборе специальной операции, называемой вызовом функции. Для вызова функции указывается её имя, за которым в круглых скобках следует набор параметров, передаваемых функции.

Пользовательские функции определяются при помощи оператора function. При определении функции указывается её имя и список параметров. Параметры перечисляются через запятую. Каждый параметр считается локальной переменной функции. Внутри блока операторов функции могут определяться новые переменные. Каждая переменная, определённая на внутреннем уровне является локальной. Все обычные локальные переменные создаются в стеке при каждом вызове функции и уничтожаются после завершения.

Функции не имеют доступа к глобальным переменным. Для получения доступа к глобальным переменным последняя должна быть локально переобъявлена. Локальное переобъявление выполняется указанием ключевого слова global перед именем глобальной переменной, к которой необходимо получить доступ.

Помимо обычных (автоматических) локальных переменных в функции можно определить статические переменные, которые создаются при определении функции и существуют всё время, пока выполняется программа. Для определения статической переменной перед её именем нужно указать ключевое слово static. Статические переменные сохраняют своё значение после выхода из функции.

Функции могут возвращать значения. Для этого используется оператор return. Тип возвращаемого значения зависит от типа параметра оператора return.

Стандартно параметры передаются в функцию по значению. Это означает, что при вызове функции для каждого параметра создаётся локальная переменная, куда копируется значение передаваемого параметра. В PHP имеется возможность организовать передачу параметров функции по ссылке. Для этого в списке параметров при определении функции перед именем параметра необходимо указать символ "&". При передаче по ссылке при вызове функции для передаваемого параметра создаётся ещё одно имя, которое является именем параметра или, что то же самое, именем локальной переменной функции. При передаче по ссылке у функции имеется возможность изменять значение передаваемой переменной. Если необходимо при некоторых вызовах одной и той же функции передавать параметр по значению, а при других по ссылке, то функция должна быть определена обычным образом. При этом при вызове функции с передачей параметра по ссылке следует в списке параметров перед соответствующим параметром указать символ "&".

Параметры функции могут иметь значение по умолчанию. Для указания значения параметра по умолчанию в списке параметров после имени параметра ставится символ "=", за которым следует необходимое значение. Если какой−либо параметр имеет значение по умолчанию, то все последующие параметры в списке также должны иметь значения по умолчанию. При вызове функций со значениями параметров по умолчанию соответствующие параметры вызова можно опустить.

В интерпретаторе PHP имеются встроенные функции. Встроенные функции определены в самом интерпретаторе или в загружаемых модулях. В табл. 2.3 приведены основные встроенные функции PHP.

Таблица 2.3

Встроенные функции PHP

|  |  |
| --- | --- |
| Функция | Описание |
| 1 | 2 |
| *Математические функции* | |
| mixed abs(mixed x) | Абсолютное значение аргумента x |
| int ceil(double x) | Наименьшее целое, большее x |
| int floor(double number) | Наибольшее целое, меньшее x |
| double round(double x) | Округление |
| int rand([int min,  int max]) | Получить случайное число |
| void srand(int seed) | Инициализировать генератор случайных чисел |
| int mt\_rand([int min,  int max]) | Получить случайное число |
| void mt\_srand(int seed) | Инициализировать генератор случайных чисел |
| double sqrt(double x) | Квадратный корень |
| double exp(double x) | Экспонента |
| double log(double x) | Натуральный логарифм |
| double log10(double x) | Десятичный логарифм |
| double pi() | Число Пи |
| double cos(double x) | Косинус |
| double sin(double x) | Синус |
| double tan(double x) | Тангенс |
| double acos(double x) | Арккосинус |
| double asin(double x) | Арксинус |
| double atan(double x) | Арктангенс |
| double atan2(double y,  double x) | Арктангенс y/x |
| *Обработка строк* | |
| void print(string str) | Вывести строку в стандартный поток вывода |
| int printf(string format,  mixed arg, ...) | Форматированный вывод |
| string sprintf(  string format,  mixed arg, ...) | Форматировать строку |
| string addslashes(  string str) | Экранирование символов "$", "\", "'" и 0 в строке string |
| string stripslashes(  string str) | Удалить экранирующие символы из строки |
| string chr(int ascii) | Получить символ с заданным кодом |
| int ord(string ch) | Код символа |
| string convert\_cyr\_string(  string str, string from,  string to) | Перевод строки из одной русскоязычной кодировки в  Продолжение табл 2.3 |
| void parse\_str(string str) | Разбить строку запроса и создать соответствующие переменные |
| *Обработка массивов* | |
| int count(mixed arr) | Число элементов массива |
| mixed current(array arr) | Текущий элемент массива |
| mixed reset(array arr) | Установить внутренний указатель массива на начальный элемент |
| mixed prev(array arr) | Переместить указатель массива на предыдущий элемент |
| mixed next(array arr) | Переместить указатель массива на следующий элемент |
| mixed end(array arr) | Установить внутренний указатель массива на последний элемент |
| void sort(array arr) | Сортировка массива |
| void rsort(array arr) | Сортировка массива в обратном порядке |
| void asort(array arr) | Сортировка ассоциативного массива array |
| void arsort(array arr) | Сортировка ассоциативного массива arr в обратном порядке |
| int ksort(array arr) | Сортировка ассоциативного массива по ключам |
| *Процессы* | |
| string system(  string cmd [, int ret]) | Выполнить команду и возвратить результат |
| int chdir(string dir) | Изменить текущий рабочий каталог |
| void putenv(string str) | Установить значение переменной окружения |
| string getenv(string var) | Получить значение переменной окружения |
| void sleep(int sec) | Задержка выполнения текущего процесса |
| void usleep(int us) | Задержка выполнения текущего процесса |
| int sem\_get(int key  [, int max [, int perm]]) | Получить идентификатор семафора |
| int sem\_acquire(int sem) | Уменьшение счётчика семафора |
| int sem\_release(int sem) | Увеличение счётчика семафора |
| *Файлы и каталоги* | |
| int copy(string src,  string dest) | Копировать файл |
| int rename(string old,  string new) | Переименовать файл |
| int unlink(  string filename) | Удалить файл |
| int readfile(  strinf filename) | Прочитать текстовый файл в стандартное у  Продолжение табл 2.3 |
| array file(string filename) | Прочитать текстовый файл в массив строк |
| int filesize(  string filename) | Размер файла |
| int file\_exists(  string filename) | TRUE, если файл существует |
| int fopen(string filename,  string mode) | Открыть файл |
| string fgetc(int fd) | Прочитать символ из файла |
| string fgets(int fd,  int maxlen) | Прочитать из файла строку максимальной длиной maxlen |
| int fputs(int fd, string str  [, int length]) | Записать строку в файл |
| int ftell(int fd) | Текущая позиция указателя |
| int fseek(int fd, int offset) | Переместить внутренний указатель файла |
| int rewind(int fd) | Переместить указатель файла на начало |
| int feof(int fd) | TRUE, если дескриптор fd указывает на конец файла |
| int fclose(int fd) | Закрыть файл |
| int pclose(int fd) | Закрыть поток, открытый popen |
| int mkdir(string dir,  int mode) | Создать директорию |
| int rmdir(string dir) | Удалить директорию |
| int opendir(string dir) | Открыть директорию |
| string readdir(int fd) | Получить имя очередного файла в каталоге |
| void rewinddir(int fd) | Переместить указатель каталога на первый файл |
| void closedir(int fd) | Закрыть директорию |
| void clearstatcache() | Очистить файловый кэш |
| flush() | Очистка буфера стандартного вывода |
| int fsockopen(string host,  int port) | Открыть сокет |
| *Дата и время* | |
| int time() | Получить текущее время в формате UNIX |
| string date(string format,  int timestamp) | Строковое представление локальной даты и времени в указанном формате |
| string gmdate(  string format,  int timestamp) | Преобразовать дату время GMT в формате UNIX в форматированную строку |
| array getdate(  Окончание табл 2.3 | Преобразовать дату время в ассоциативный массив |
| int mktime(int hour,  int minute, int second,  int month, int day,  int year) | Дата и время в формате UNIX |
| int gmmktime(int hour,  int minute, int second,  int month, int day,  int year) | Дата и время GMT в формате UNIX |
| *TCP/IP, HTTP, CGI, PHP* | |
| string gethostbyname(  string host) | Получить IP−адрес хоста |
| string gethostbyaddr(  string addr) | Получить имя хоста по IP−адресу |
| int header(string hdr) | Отправить заголовок HTTP |
| string htmlspecialchars(  string str) | Преобразовать специальные символы HTML в строке str в escape−последовательности |
| int isset() | TRUE, если переменная определена |
| void eval(string code) | Обработка code в качестве вложенного сценария |
| void exit() | Завершить текущий сценарий |
| int phpinfo() | Вывод информации о текущей реализации PHP |
| string phpversion() | Вывод информации о текущей версии PHP |

**Пример выполнения лабораторной работы:**

<HTML>

<HEAD>

<META http−equiv="Content−Type" content="text/html;

charset=koi8−r">

<TITLE>Пример выполнения </TITLE>

</HEAD>

<BODY>

<H1>Создание документов HTML с использованием технологии PHP</H1>

<HR>

<?

echo "Версия PHP: ";

echo phpversion();

echo "<BR>";

?>

<HR>

<FORM method="GET">

Введите имя папки:<BR>

<INPUT type="text" name="dirname" size="50"

<?

//Проверка наличия значения переменной dirname

if (isset($dirname)) {

//Вывод имени папки

echo stripslashes("value=\"$dirname\"");

}

?>

><BR>

<?

//Проверка наличия значения переменной dirname

if (isset($dirname)) {

//Открыть директорию для получения списка файлов

opendir($dirname);

echo "Выберите файл в данной папке:<BR>";

echo "<SELECT name=\"filename\" size=\"5\">";

//Получение очередного имени файла из директории

while($file = readdir()) {

echo "<OPTION value=\"$file\">$file";

}

echo "</SELECT><BR>";

//Закрытие директории после получения всего списка

closedir();

}

//Проверка наличия значения переменной filename

if (isset($filename)) {

//Вывести имя файла

echo "Файл ";

echo stripslashes($dirname);

echo "\\";

echo stripslashes($filename);

echo "<BR>";

//Вывести содержимое выбранного файла

echo "<TEXTAREA rows=\"10\" cols=\"60\">";

//Перейти в директорию, где находится выбранный файл

chdir($dirname);

//Получение содержимого файла в виде массива строк

$filecontent = file($filename);

if (count($filecontent) > 0) {

//Вывести содержимое файла в цикле по строкам

for ($i = 0; $i < count($filecontent); $i++) {

echo $filecontent[$i];

}

}

echo "</TEXTAREA><BR>";

}

?>

<INPUT type="submit"><BR>

</FORM>

<HR>

<?

//Вывести дату и время формирования документа

//Получить текущую дату и время по гринвичу в указанном формате

$gmdtstr = gmdate("D, d M Y H:i:s");

echo "<BR>";

echo $gmdtstr;

//Вывести суффикс GMT, согласно формату HTTP

echo " GMT";

echo "<br>";

?>

</BODY>

</HTML>

# **Содержание отчёта**

Отчёт по практическому заданию должен содержать следующие сведения:

* название и цель работы;
* вариант задания;
* исходные тексты разработанной программы с комментариями;
* результат работы программы.

## **Варианты заданий**

Пользователю предлагается ввести на странице имя:

1. некоторого файла в локальной файловой системе сервера. Сценарий PHP рассматривает содержимое этого файла как некоторый каталог и выводит его содержимое в документ HTML.
2. каталога в локальной файловой системе сервера. Если в данном каталоге имеется файл, имя которого начинается с "readme", то сценарий PHP выводит его содержимое.
3. каталога в локальной файловой системе сервера. Сценарий PHP выводит содержимое этого каталога в следующем формате: пиктограмма, указывающая на тип файла, имя файла, размер (или специальная пометка, если файл является каталогом), дата и время последней модификации.
4. файла в локальной файловой системе сервера, содержимое которого рассматривается как список файлов с графическими изображениями. Сценарий PHP выводит содержимое файлов из данного списка.
5. каталога в локальной файловой системе сервера. Сценарий PHP просматривает содержимое этого каталога и выводит содержимое каждого найденного им файла с графическим изображением.
6. файла в локальной файловой системе сервера. Содержимое данного файла рассматривается сценарием PHP как команда локальной операционной системы сервера. Сценарий PHP выполняет данную команду и выводит результат её выполнения.
7. некоторого файла в локальной файловой системе сервера. Сценарий PHP определяет тип данного файла: текстовый, графический или прочий, и в зависимости от этого выводит его содержимое, соответственно, в виде текста, в виде графического изображения или в виде шестнадцатеричного дампа.
8. некоторого файла в локальной файловой системе сервера. Сценарий PHP рассматривает содержимое этого файла как сценарий, который может содержать следующие команды: FILES <имя\_каталога> − вывести содержимое указанного в качестве параметра каталога; VIEW <имя\_файла> − вывести содержимое указанного в качестве параметра текстового файла; SHOW <имя\_файла> − вывести содержимое указанного в качестве параметра графического файла.

Сценарий PHP анализирует и выполняет данный сценарий.

1. некоторого файла в локальной файловой системе сервера. Сценарий PHP рассматривает файл как текстовый и выводит его содержимое на экран, причём последовательности вида "%hhhhhh%", где каждый символ "h" представляет собой шестнадцатеричную цифру, рассматриваются как команды изменения цвета дальнейшего текста на соответствующий.
2. некоторого файла в локальной файловой системе сервера. Сценарий PHP рассматривает содержимое данного файла как определение таблицы. Символы перевода строки и возврата каретки рассматриваются как определения новой строки таблицы. Пробельные символы рассматриваются как определения новой ячейки в строке таблицы. Содержимым ячейки может быть текстовая строка или графическое изображение. В последнем случае указывается имя файла, содержащего данное изображение. Сценарий PHP строит определённую в этом файле таблицу.