Основы PHP. Базовый синтаксис PHP

У каждого языка есть свои правила и конструкции, следуя которым, мы выражаем свои мысли и делаем их понятными для другого человека. В программировании веб-сайтов всё точно также. Но вместо человеческого языка мы используем язык программирования PHP, а в роли нашего собеседника выступает PHP-интерпретатор. Поэтому, чтобы выразить свою мысль, мы должны сделать её понятной для интерпретатора.

Теги PHP

Когда PHP обрабатывает файл, он ищет открывающие и закрывающие теги, такие как <?php и ?>, которые указывают PHP, когда начинать и заканчивать обработку кода между ними. Подобный способ обработки позволяет PHP внедряться во все виды различных документов, так как всё, что находится вне пары открывающих и закрывающих тегов, будет проигнорировано парсером PHP.

PHP включает в себя короткий echo-тег <?=, который является сокращением для более многословного <?php echo.

Пример: Открывающие и закрывающие теги PHP:

1. <?php echo 'если вы хотите хранить код PHP в документах XHTML или XML, то используйте эти теги'; ?>
2. Вы можете использовать короткий 'echo'-тег чтобы <?= 'напечатать эту строку' ?>. Этот тег эквивалентен такому коду <?php echo 'напечатать эту строку' ?>.
3. <? echo 'этот код с короткими тегами, но он будет работать только если '. включена опция "short_open_tag"; ?>

Короткие теги (третий пример) доступны по умолчанию, но их можно отключить с помощью директивы short_open_tag в конфигурационном файле *php.ini* или отключены по умолчанию, если PHP был скомпилирован с опцией --disable-short-tags.

Если файл содержит только код PHP, предпочтительно опустить закрывающий тег в конце файла. Это помогает избежать добавления случайных символов пробела или перевода строки после закрывающего тега PHP, которые могут послужить причиной нежелательных эффектов, так как PHP начинает выводить данные в буфер при отсутствии намерения у программиста выводить какие-либо данные в этой точке скрипта.

```
<?php
echo "Hello world";

// ... ещё код

echo "Последнее выражение";

// Скрипт заканчивается тут без закрывающего тега PHP
```

Изолирование от HTML

Все, что находится вне пары открывающегося и закрывающегося тегов, игнорируется интерпретатором PHP, у которого есть возможность обрабатывать файлы со смешанным содержимым. Это позволяет PHP-коду быть встроенным в документы HTML, к примеру, для создания шаблонов.

```
<p>Это будет проигнорировано PHP и отображено браузером.</p>
<?php echo 'А это будет обработано.'; ?>
<p>Это тоже будет проигнорировано PHP и отображено браузером.</p>
```

Переменные — это основа любого языка программирования. Во время написания кода происходит постоянная работа с переменными. Понять концепцию переменных очень просто.

Переменная — это контейнер, в котором содержатся данные, так же как напиток содержится в чашке.

Любая информация, которая используется в коде, сначала сохраняется в переменной.

У переменной обязательно должно быть имя, поэтому переменная всегда состоит из имени и значения. Значение — это любая информация, которая хранится внутри переменной.

Например, мы можем попросить посетителя страницы указать свой возраст, а затем использовать это значение для других целей — узнать год рождения или показать возраст самой странице.

Любую переменную следует вначале объявить, то есть дать ей имя и присвоить значение.

В синтаксисе PHP имя переменной записывается латинскими символами, но первым символом всегда должен быть знак доллара $, а затем идёт имя.

Не допускается начинать имя переменной с цифры, а также использовать любые значения, кроме букв алфавита и знака подчеркивания.

Примеры допустимых имён переменных:
$age;
$favorite_color;
$name2.

Примеры *недопустимых* имён:
Age — забыт знак доллара в начале;
$42 — начинается с цифры;
$my-age — содержит дефис.

Пустая переменная не является полезной, поэтому рекомендуется в неё что-то поместим. Такое действие называется операцией присваивания.

Пример операции присваивания в новую переменную:

```
$favorite_color = "green";
```

Знак “равно” в PHP является *оператором* и всегда означает операцию присваивания.

Здесь записано слово “green” в переменную под именем *favorite_color*.

Стоит обратить внимание: было взято в кавычки слово green. Кавычки всегда строго необходимы, если речь идёт об использовании текста. Но если в переменную помещается не текст, а число, то кавычки не нужны.

Пример:

```
$favorite_number = 42;
```

После того как интерпретатор PHP встречает закрывающие теги ?>, он просто начинает выводить все что найдёт (за исключением сразу следующего символа перевода строки) пока не встретит другой открывающий тег за исключением случая с

содержащимся внутри кода условным оператором, в котором интерпретатор определяет результат условия перед принятием решения что пропустить.

Пример: Продвинутое изолирование с использованием условий

```
<?php if ($expression == true): ?>
  Это будет отображено, если выражение истинно.
<?php else: ?>
  В ином случае будет отображено это.
<?php endif; ?>
```

В этом примере PHP пропускает блоки, где условие не соблюдается. Даже несмотря на то, что они находятся вне пары открывающих/закрывающих тегов, PHP пропустит их в соответствии с условием, так как интерпретатор PHP будет перепрыгивать через блоки, содержащиеся внутри условия, которое не соблюдается.

При выводе больших блоков текста выход из режима синтаксического разбора PHP обычно более эффективен, чем отправка текста с помощью функций echo или print.

Где:

echo — выводит одну или более строк (в версиях PHP 4, PHP 5, PHP 7, PHP 8);

print — выводит строку (в версиях PHP 4, PHP 5, PHP 7, PHP 8).

Разделение инструкций

Как в C или Perl, PHP требует окончания инструкций точкой запятой в конце каждой инструкции. Закрывающий тег блока PHP-кода автоматически применяет точку с запятой; т.е. нет необходимости ставить точку с запятой в конце последней строки блока с PHP-кодом. Закрывающий тег блока "поглотит" немедленно следующий за ним переход на новую строку, если таковой будет обнаружен.

Пример: Показывающий закрывающий тег, охватывающий завершающую новую строку

```
<?php echo "Какой-то текст"; ?>
Нет новой строки
<?= "А сейчас, новая строка" ?>
```

Результат выполнения данного примера:

```
Какой-то текстНет новой строки
А сейчас, новая строка
```

Примеры входа и выхода из парсера PHP:

```
<?php
   echo 'Это тест';
?>
```

```
<?php echo 'Это тест' ?>
```

```
<?php echo 'Мы опустили последний закрывающий тег';
```

Закрывающий тег PHP-блока в конце файла не является обязательным, и в некоторых случаях его опускание довольно полезно, например, при использовании include или require, так, что нежелательные пробелы не останутся в конце файла и вы всё ещё сможете добавить http-заголовки после подключения к ответу сервера. Это также удобно при использовании буферизации вывода, где также нежелательно иметь пробелы в конце частей ответа, сгенерированного подключаемыми файлами.

Где:

Выражение include включает и выполняет указанный файл, a require аналогично include, за исключением того, что в случае возникновения ошибки он также выдаст фатальную ошибку уровня E_COMPILE_ERROR. Другими словами, он остановит выполнение скрипта, тогда как include только выдал бы предупреждение E_WARNING, которое позволило бы скрипту продолжить выполнение.

Комментарии

PHP поддерживает комментарии в стиле 'C', 'C++' и оболочки Unix (стиль Perl). Пример:

```
<?php
   echo "Это тест"; // Это однострочный комментарий в стиле C++
   /* Это многострочный комментарий
      ещё одна строка комментария */
   echo "Это ещё один тест";
   echo "Последний тест"; # Это комментарий в стиле оболочки Unix
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идёт перед ними. Это означает, что HTML-код после // ... ?> или # ... ?> будет напечатан: ?> завершает режим PHP и возвращает режим HTML, а // или # не могут повлиять на это.

```
<h1>Это <?php # echo "простой";?> пример</h1>
<p>Заголовок вверху выведет 'Это пример'.</p>
```

'C'-комментарии заканчиваются при первой же обнаруженной последовательности */. Убедитесь, что вы не вкладываете друг в друга 'C'-комментарии. Очень легко допустить эту ошибку при комментировании большого блока кода.

```
<?php
 /*
   echo "Это тест"; /* Этот комментарий вызовет проблему */
 */
?>
```

Контрольные вопросы:

1. Открывающие и закрывающие теги PHP;
2. Изолирование тегов PHP от тегов HTML;
3. Понятие переменной и её значение;
4. Разделение инструкций в PHP;
5. Поддержка однострочных и многострочных комментарий в PHP.