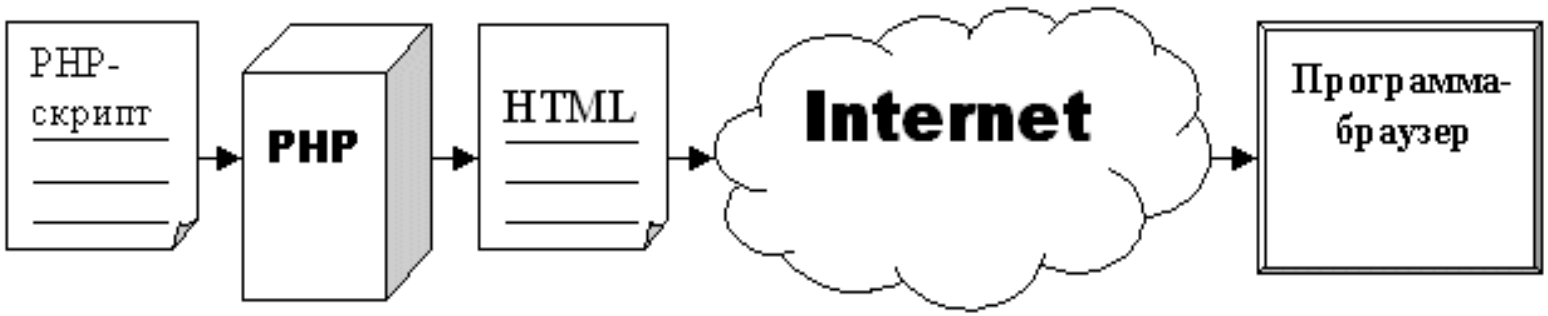
Знакомство с PHP

Что такое PHP?

PHP — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

- Personal Home Page (личная домашняя страница)
- HyperText Preprocessor



История PHP

- 1995 г. — Расмус Лердорф (Rasmus Lerdorf)

Первая версия PHP представляла собой набор макросов (сценарий Perl/CGI) для подсчета количества посетителей сайта, прочитавших его онлайн-резюме.

Сценарий решал 2 задачи:

1. Регистрация данных посетителя
2. Вывод количества посетителей на web-странице

- 1997 г. — вышла вторая версия обработчика, написанного на C, добавлена программа обработки форм: PHP/FI 2.0 (Personal Home Page — Form Interpreter)

- 1998 г. — Энди Гутманс и Зив Зураски (Andi Gutmans, Zeev Surasky) полностью переписали код интерпретатора, добавили возможность расширения ядра дополнительными модулями: PHP 3.0 («PHP: HyperText Preprocessor»)

- 2000 г. — переработано ядро PHP, создан новый движок Zend Engine для увеличения производительности сложных приложений и улучшение модульности базиса кода PHP. Версия PHP 4.0 (поддержка сессий, буферизация вывода, более безопасные способы обработки вводимой пользователем информации, несколько новых языковых конструкций)

- 2004 г. — PHP 5.0: обновление ядра Zend (Zend Engine 2), введена поддержка языка разметки XML. Полностью переработаны функции ООП (в частности, введён деструктор, открытые, закрытые и защищённые члены и методы, интерфейсы и клонирование объектов)

- 2006-2010 г. — PHP 6: было сделано множество нововведений, должна была появиться поддержка Юникода, но в марте 2010 года разработка PHP 6 была признана бесперспективной, исходный код PHP 6 перемещён на ветвь, а основной линией разработки стала версия 5.4.

- 2015 г. — PHP 7.0.0: Новая версия основывается на phpng (англ. PHP Next Generation), упор на увеличение производительности и уменьшение потребления памяти. Добавлена возможность указывать тип возвращаемых из функции данных, добавлен контроль передаваемых типов для данных, а также новые операторы.

- 2019 г. — PHP 7.4. В ядро были добавлены типизированные свойства и стрелочные функции, а также ограниченная ковариация возвращаемого типа и контравариантность типа аргумента.

- 2020 г. — PHP 8.0: поддержка union-типов, JIT-компиляция и атрибуты (также известны как аннотации).

- 9 июля 2020 г. Дэйл Хирт, менеджер проекта PHP в Microsoft, сообщил о том, что после выпуска версии PHP 8.0 Microsoft прекратит поддержку разработки этого языка программирования для Windows. В сообществе разработчиков PHP сообщили, что примут все необходимые меры, чтобы найти в ближайшее время альтернативный вариант для организации поддержки PHP 8.0 и выше для Windows, например, своими силами.

Почему следует предпочесть PHP?

- PHP — это открытый продукт
- Скорость разработки
- Переносимость
- Платформы, серверы и базы данных

Основные конструкции PHP

Обрамление блока PHP-команд

Вид тегов	Открывающий тег	Закрывающий тег
Стандартные	<?php	?>
Короткие	<?	?>
ASP	<%	%>
Программные	<SCRIPT LANGUAGE="php">	</SCRIPT>

Из перечисленных в таблице тегов только стандартные и программные гарантированно работают в любой конфигурации PHP. Использование коротких тегов и тегов ASP должно быть явно разрешено в файле php.ini.

Рассмотрим конструкции языка PHP на примере простейшей PHP-программы:

<? print "Hello Web!"; ?>

Функция print ()

```
<? print "Hello Web!"; ?>
```

Функция print () предназначена для вывода данных в окно браузера. Поскольку в приведенном выше примере аргументом функции print() является строка символов, то она обязательно должна быть заключена в кавычки — двойные или одинарные.

В общем случае после имени функции должны находиться скобки, независимо от того, передаются ей какие-то аргументы или нет. Функция print() — это исключение из правила, и вы не обязаны заключать в скобки строку, которую вы хотите вывести в окно браузера. Поэтому мы будем опускать скобки при вызове функции print().

Точка с запятой обязательно должна стоять в конце каждой команды. Исключением из этого может быть оператор, завершающий блок команд. Но в большинстве случаев пропуск точки с запятой сбивает интерпретатор с толку и приводит к ошибке.

Взаимодействие HTML и PHP

Можно создать смешанный документ, добавив теги HTML перед открывающим и после закрывающего тегов PHP.

Документ, содержащий PHP-команды и HTML-текст

```
<html>
<head>
<title> Документ, содержащий PHP-команды и HTML-текст </title>
</head>
<body>
<p><i>Проверка</i>
<?php
    print "<div style=\"color: blue\">PHP работает! </div>";
?>
</body>
</html>
```

Комментарии в PHP-программе

Отдельная строка комментария начинается двумя символами косой черты // или одним символом #. Любой текст от этих знаков до конца строки или до закрывающего тега PHP игнорируется.

```
// Это комментарий.
# Это тоже комментарий.
```

Несколько строк комментариев начинаются парой символов /* и заканчиваются парой */.

```
/* Это комментарии.
Все эти строки будут проигнорированы интерпретатором.
*/
```

Многострочные комментарии особенно удобны для записи сводной информации обо всей программе или ее части.

Переменные

Переменная — это область памяти для хранения данных определенного типа. Каждая переменная имеет имя, начинающееся со знака доллара, $. Имя переменной может состоять из букв, цифр и знака подчеркивания, при этом регистр символов учитывается. В имени не могут встречаться пробелы и какие-нибудь символы, отличные от букв и цифр.

В следующем примере приведены правильные имена переменных:

```
$a;
$a_long_variable_name;
$_2453;
```

Как правило, создание переменной и присваивание ей значения выполняется в одной и той же команде:

```
$num1 = 8; $num2 = 23;
```

Строковое присваивание

Строка — последовательность символов, которая рассматривается как единое целое. Строки делятся на две категории в зависимости от типа ограничителя — пары кавычек (" ") или апострофов (' '). Между этими категориями существуют два принципиальных отличия.

1) Имена переменных в строках, заключенных в кавычки, заменяются соответствующими значениями, а строки в апострофах интерпретируются буквально, даже если в них присутствуют имена переменных. Два следующих присваивания дают одинаковый результат:

```
$user = "Ник";  $user = 'Ник';
```

Однако результаты следующих присваиваний сильно различаются:

```
$var1 = "Мой друг $user";  $var2 = 'Мой друг $user';
```

- Переменной $var1 присваивается строка Мой друг Ник (переменная $user автоматически интерпретируется)
- Переменной $var2 присваивается строка Мой друг $user.

2) Второе принципиальное различие между строками, заключенными в апострофы и в кавычки, связано с обработкой служебных символов. В PHP, как и в других языках программирования, строки могут содержать служебные символы (например, символы новой строки или табуляции), перечисленные в таблице.

Служебные символы в строках

Последовательность	**Описание**
`\n`	Новая строка
`\r`	Перевод строки
`\t`	Горизонтальная табуляция
`\\`	Обратная косая черта как символ
`\$`	Знак доллара как символ
`\"`	Кавычка как символ
`\'`	Апостроф как символ

Служебные символы \n, \r и \t используются лишь для создания удобочитаемого HTML-файла, на вывод текста в браузере они никак не влияют.

В строках, заключенных в кавычки, распознаются все существующие служебные символы, а в строках, заключенных в апострофы — только служебные символы \\ и \'. Следующий пример наглядно демонстрирует это различие:

```
print "первая строка \r вторая строка";
print 'первая строка \r вторая строка’;
```

Если вывести обе строки в браузере, окажется, что в строке в кавычках, в HTML-файле, будет выполнен перевод строки, хотя на экране все будет в одной строке: «первая строка вторая строка». А в строке в апострофах последовательность \r выведется на экран как обычные символы.

Динамические переменные

В некоторых ситуациях бывает удобно использовать переменные, содержимое которых может динамически интерпретироваться как имя другой переменной. Таким образом, выражения присваивания

```
$client = "user"; $$client = "Nic";
```

эквивалентны следующей записи

```
$user = "Nic";
```

При обращении к динамической переменной важную роль играет использование или не использование кавычек:

```
$user = "Nic";  print $user;
```

Это эквивалентно следующему:

```
$user = "Nic"; $client = "user"; print $$client;
```

Однако, для того чтобы вывести имя переменной, нужно обратиться к ней по-другому. Например, следующий фрагмент не выводит в окно браузера строку Nic, как можно было бы предположить, поскольку в операторе print() переменная стоит в кавычках:

```
$user = "Nic"; $client = "user"; print "$$client";
```

Вместо этого выводится знак $, а потом строка user, образуя строку $user. Когда вы обрамляете переменную кавычками, PHP подставляет вместо нее соответствующее значение. В данном случае PHP подставляет вместо переменной $client ее значение user.

Программа, в которой с помощью строки, хранящейся в переменой, создается и инициализируется новая переменная $user.

Создание динамической переменной и обращение к ней

```
<html> <head>
<title> Создание динамической переменной и обращение к ней
</title> </head> <body>
<?php
$client = "user";  $$client = "Nic"; // <=> $user = "Nic";
print "1) $client<br>";      //выводится user
print '2) $client<br>';      //выводится $client
print "3) $user<br>";        //выводится Nic
print "4) $$client<br>";     //выводится $user
print "5) "; print $$client; //выводится Nic
?>
</body> </html>
```

Ссылки на переменные

Если присвоить значение переменной $var1 другой переменной, $var2, то копия значения первой переменной будет записана во вторую. В дальнейшем никакие изменения значения первой переменной никак не отразятся на значении второй. Но в PHP можно сделать по-другому, заставив переменную $var2 постоянно иметь то же самое значение, что и у переменной $var1.

Создание ссылки на переменную

```
<html> <head>
<title> Создание ссылки на переменную </title> </head>
<body>
<?php
$var1 = 1;
$var2 = &$var1;
$var1 = 10;
print $var2;  //выводится 10
?>
</body> </html>
```

Символ & перед именем переменной $var1 говорит о том, что мы создаем ссылку на эту переменную, и теперь все изменения ее значения отразятся на значении переменной $var2. Другими словами, обе эти переменные связаны с одним и тем же значением.

Типы данных

Ниже перечислены шесть типов данных, поддерживаемых в языке PHP.

Тип	Пример	Описание
`Integer`	`5`	Целое число
`Double`	`3.234`	Число с плавающей точкой
`String`	`"hello"`	Строка символов
`Boolean`	`true`	Логический, принимающий значения `true` или `false`
`Array`	`$a [10]`	Массив
`Object`		Объект (элемент ООП)

PHP не имеет таких строгих требований по типам данных, т.е. он будет обрабатывать переменную в зависимости от того, какого типа значение в нее записано.

Проверка и изменение типа переменной

Функция **gettype():** в качестве аргумента получает имя переменной, возвращает строку, описывающую тип этой переменной.

Функция **settype():** для изменения типа переменной при вызове функции нужно указать переменную, тип которой вы хотите изменить, и новый тип данной переменной

```
<html> <head><title> Проверка и изменение типа переменной </title> </head> <body>
<?php $var = 3.14;
print gettype($var);    // double
print " - $var<br>";    // 3.14
settype($var, "string");
print gettype($var);    // string
print " - $var<br>";    // 3.14
settype($var, "integer");
print gettype($var);    // integer
print " - $var<br>";    // 3
settype($var, "double");
print gettype($var);   // double
print " - $var<br>";   // 3
settype($var, "boolean");
print gettype($var);   // boolean
print " - $var<br>"; // 1 ?>
</body> </html>
```

Преобразование типа переменной

```
<html> <head>
<title> Преобразование типа переменной </title> </head> <body>
<?php
$var = 3.14;
$var2 = (double) $var;
print gettype($var2);   // double
print " - $var2<br>";   // 3.14
$var2 = (string) $var;
print gettype($var2);   // string
print " - $var2<br> ";  // 3.14
$var2 = (integer) $var;
print gettype($var2);   // integer
print " - $var2<br>";   // 3
$var2 = (double) $var;
print gettype($var2);   // double
print " - $var2<br>";   // 3.14
$var2 = (boolean) $var;
print gettype($var2);    // boolean
print " - $var2<br>";    //   1
?>
</body> </html>
```

Операторы и выражения

Оператором называют символ или последовательность символов, с помощью которых можно из нескольких переменных получить новое значение. Те значения, к которым применяются операторы для получения новых значений, называются операндами.

Комбинация операндов и операторов, производящая некоторое значение, называется выражением. Однако не обязательно для образования выражения использовать операторы. Выражением в PHP считается все, что имеет некоторое значение. Например, константа 654, или переменная $user, или функция gettype() — все это выражения. Таким образом, выражение (4+5) состоит из двух выражений и одного оператора.

Выражение — это любая комбинация чисел, переменных и вызовов функций, объединенных операторами. Выражение можно использовать как и любое другое значение.

Оператор присваивания

Оператор присваивания записывает значение своего правого операнда в левый операнд:

```
$name = "Nic";
```

Теперь в переменной $name записана строка "Nic". Обратите внимание на то, что эта конструкция представляет собой выражение. На первый взгляд может показаться, что оператор присваивания просто записывает значение в переменную, но это не совсем так. На самом деле при выполнении оператора присваивания создается временная копия его правого операнда, и все выражение получает значение этой копии. Таким образом, следующая конструкция не только присваивает значение переменной, но и выводит в окно браузера строку "Nic".

```
print ($name = "Nic");
```