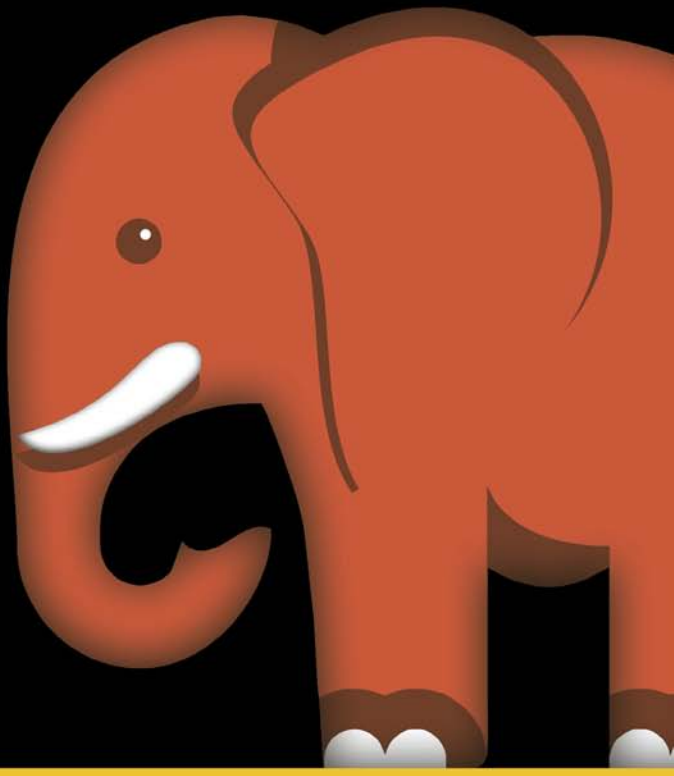
самоучитель

Максим Кузнецов, Игорь Симдянов

PHP 7



Новинки PHP 7

Шаблоны проектирования, итераторы и генераторы

Приемы работы с СУБД PostgreSQL

Взаимодействие с базами данных NoSQL (Redis и подобными)

100 заданий

Материалы на www.bhv.ru

bhv®

Максим Кузнецов
Игорь Симдянов

самоучитель

PHP 7

Санкт-Петербург
«БХВ-Петербург»
2018

УДК 004.438 PHP
ББК 32.973.26-018.1
К89

Кузнецов, М. В.

К89 Самоучитель PHP 7 / М. В. Кузнецов, И. В. Симдянов. — СПб.: БХВ-Петербург, 2018. — 448 с.: ил. — (Самоучитель)

ISBN 978-5-9775-3817-6

Книга опытных разработчиков описывает последнюю версию языка разработки серверных сценариев PHP 7. Рассмотрены все нововведения языка и связанные с ними изменения в разработке современных Web-сайтов. Изложение ведется с упором на объектно-ориентированное программирование, шаблоны проектирования, итераторы, генераторы, а также взаимодействие с современными базами данных (PostgreSQL и Redis).

В конце глав приведены более 100 заданий для закрепления материала и освоения не вошедших в книгу разделов языка. Электронный архив с исходными кодами доступен на сайтах издательства и GitHub.

Для Web-разработчиков

УДК 004.438 PHP
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-3817-6

Оглавление

Предисловие

Вы держите в руках четвертое, полностью переработанное издание популярной книги. Предыдущее издание было посвящено PHP 5 и вышло 8 лет назад. С тех пор язык обогатился большим количеством нововведений, все их мы осветим на страницах книги.

В связи с этим книгу пришлось полностью переписать, лишь 10 глав из 32-х основаны на материале предыдущей книги, хотя и их пришлось подвергнуть глубокой переработке. Остальные главы написаны с чистого листа.

Объектно-ориентированный подход

До версии PHP 5 поддержка объектно-ориентированного программирования (ООП) в языке была довольно скудна. В PHP 5 объектно-ориентированный подход долго оставался альтернативой для традиционного процедурного подхода. PHP 7 практически полностью предназначен для объектно-ориентированной разработки.

Без ООП невозможна разработка современных PHP-приложений: все больше расширений предполагают объектно-ориентированный интерфейс, компоненты оформляются в виде классов, PSR-стандарты и современные фреймворки диктуют разработку, полностью ориентированную на объектно-ориентированный подход.

Именно поэтому знакомиться с объектно-ориентированным подходом мы начинаем сразу с первых глав книги. Рассматриваем шаблоны проектирования и там, где это возможно, ориентируемся на новые расширения с объектно-ориентированным интерфейсом.

PostgreSQL и Redis

Традиционно PHP-приложения работают совместно с СУБД MySQL. В редкой книге по PHP не уделяется внимание этой популярной базе данных. Однако мы отступим от традиции.

С одной стороны, перепродажа компании AB MySQL корпорации Sun, которая в свою очередь была поглощена в 2009 г. Oracle, привела к тому, что наиболее по-

пулярная свободная СУБД оказалась в руках крупнейшего в мире производителя коммерческих баз данных. Развитие MySQL если и не находится в стагнации, то значительно уступает конкурирующим СУБД, например PostgreSQL, которая долгое время оставалась на вторых ролях.

С другой стороны, развитие объектно-ориентированного подхода и увеличение количества памяти на серверах привели к всплеску интереса к нереляционным базам данных, которые оформились в виде движения NoSQL. PostgreSQL — не совсем традиционная СУБД. Задуманная как объектно-ориентированная база данных, PostgreSQL за несколько лет предвосхитила NoSQL-движение. Поэтому уже сегодня можно наблюдать, как большинство современных Web-приложений переходят на PostgreSQL.

Кроме этого, вместо традиционного memcached мы рассмотрим NoSQL базу данных Redis. Как правило, современные Web-приложения не обходятся без одной или нескольких NoSQL баз данных, поэтому обойти их вниманием не представляется возможным. Redis выделяется среди них высокой производительностью (100 000 RPS — request per seconds), богатыми возможностями (коллекции, кластеризация, механизм Pub/Sub).

Задания

Мы не рассматриваем детальную установку полноценного окружения, включающего Web-сервер, настройку его связи с PHP, обеспечение безопасного доступа к удаленному серверу. Вместо этого используется встроенный PHP-сервер, сразу же доступный при установке PHP.

В силу ограничений по объему книги пришлось отказаться от описания многих особенностей PHP, например: детального рассмотрения сокетов, расширения curl, математических функций, преобразования изображений, управления буфером вывода, регулярных выражений и даже файловых функций. Все эти возможности выходят за рамки книги.

Вместо этого каждая глава снабжается заданиями, которые побуждают исследовать документацию, знакомиться с не вошедшими в книгу функциями и расширениями, исследовать альтернативные базы данных, читать статьи, искать решение. Всего 100 заданий.

Исходные коды

В начале каждой главы (за исключением 1-й и 29-й) указан каталог, в котором можно обнаружить примеры данной главы, названия файлов приведены в заголовках к листингам.

Исходные коды к книге можно найти на GitHub-аккаунте по адресу:

https://github.com/igorsimdyanov/phpworkshop

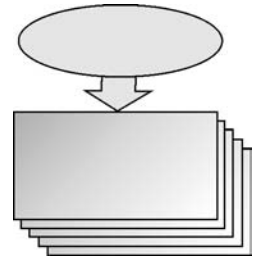
По ссылке **https://github.com/igorsimdyanov/phpworkshop/issues** можно адресовать вопросы авторам.

Электронный архив с исходными кодами к книге также можно скачать по ссылке **ftp://ftp.bhv.ru/9785977538176.zip**. Эта ссылка доступна и со страницы книги на сайте **www.bhv.ru**.

Благодарности

С Максимом Кузнецовым мы написали множество книг, и первое издание этой книги было стартом нашей совместной работы. Пять лет, как его с нами нет: жил быстро, ярко, помогал многим, казалось бы, в безвыходных ситуациях. Это издание книги посвящено ему.

ГЛАВА 1



Что представляет собой PHP?

Язык программирования PHP — серверный язык, при помощи которого можно создавать Web-сайты, причем как небольшие лендинги, состоящие из одной страницы, так и гигантские системы, использующие сотни и тысячи серверов. Электронная энциклопедия Wikipedia, социальные сети Facebook, "ВКонтакте", электронная площадка объявлений Avito созданы с использованием PHP.

Будучи одним из самых первых языков программирования, ориентированных на Web-разработку, PHP прошел длительный путь практически с самого начала зарождения Web. Поэтому в мире он остается одним из самых популярных и востребованных языков.

1.1. Достоинства и недостатки

В основе популярности PHP лежат следующие достоинства.

- Ориентация на Web-разработку — PHP создавался, развивался и поддерживается как язык для создания Web-сайтов. Многие конструкции и решения в нем созданы для удобства работы в Web-среде.
- Кроссплатформенность — PHP перенесен на все основные операционные системы: можно разрабатывать сайт в Windows, Mac OS X, а эксплуатировать на Linux-сервере. Сложности переноса будут минимальны и нивелироваться языком.
- Бесплатность — PHP является разработкой из мира свободного программного обеспечения, не потребуется платить ни за сам язык, ни за большинство сопутствующих программ (редакторы, Web-серверы, базы данных). Вдобавок большинство программных продуктов, с которыми придется иметь дело, будут иметь доступный для изучения и модификации исходный код. Вложения могут потребоваться при аренде доменного имени и сервера для публикации сайта в Интернете. Однако изучать PHP можно, не вкладывая ни копейки.
- Низкий порог входа — изучить PHP и начать создавать на нем готовые приложения много проще, чем с использованием конкурирующих технологий (.NET,

Python, Ruby, Go). Изучение PHP не закрывает для разработчика другие технологии, в Web сам язык — значительная, но меньшая часть используемых технологий. Знания, приемы работы, сопутствующие технологии (Web-серверы, базы данных, библиотеки, вспомогательные языки) пригодятся и в любой другой экосистеме, отличной от PHP. При создании собственного бизнеса собрать команду PHP-разработчиков зачастую проще и дешевле всего.

По закону сохранения, любая вещь, обладающая хоть каким-либо достоинством, имеет недостатки. Ими обладает и PHP.

- Отсутствие лидера — многие технологии и языки имеют лидера, архитектора, который определяет облик технологии, задает вектор развития, принимает решение о том, что должно быть обязательно, а чего не будет никогда (Linux, Python, Ruby и т. п). В PHP лидера нет, многие решения и конструкции — это компромисс заинтересованных групп и исторически сложившихся реалий.
- Непоследовательный синтаксис — при изучении языка PHP, особенно старой части, основанной на функциях, можно заметить, что часть функций имеет префиксы array_, str_, часть не имеет. Параметры функций могут быть расположены не совсем логично и не так, как в другой функции этой же группы.
- PHP — уже довольно долго живущий язык. Когда язык только появляется, он довольно элегантный и внутренне согласованный. По мере жизненного цикла язык обрастает дополнительными ключевыми словами, артефактами, устаревшими конструкциями, которые вроде есть, работают, но которыми не рекомендуется пользоваться. У PHP была довольно бурная молодость, в ходе которой была отменена масса директив и приемов, которые на первый взгляд должны были облегчать разработку, а на практике оборачивались серьезными проблемами безопасности. Сам PHP, стартовавший как необъектно-ориентированный язык, в настоящий момент стал полноценным объектно-ориентированным языком. Однако в нем полно старых процедурных артефактов, которыми придется пользоваться.
- Сообщество PHP-разработчиков велико и разъединено, т. к. PHP — это одна из первых технологий для разработки Web-проектов, половина Интернета создана с его участием. В PHP-разработку одновременно было вовлечено огромное количество программистов по всему миру. Все это породило большое число самых разных подходов, фреймворков и не совместимых друг с другом экосистем. Более того, благодаря усилиям мощных и влиятельных социальных сетей (в первую очередь Facebook, "ВКонтакте") появились альтернативные реализации PHP. Это плохо, т. к. многие экосистемы внутри PHP не совместимы, а сообщество раздроблено и тратит силы на создание одних и тех же библиотек в рамках разных групп. Ситуация исправляется и при помощи PSR-стандартов. Разработчики договариваются о единых правилах и интерфейсах, обеспечивающих совместимость фреймворков, но этот процесс еще в начале пути, в то время как конкурирующие технологии (.NET, Ruby) уже имеют единую платформу для всех фреймворков.

1.2. Структура PHP

Язык PHP имеет ядро и расширения языка. Между ядром и расширениями довольно трудно провести границу, т. к. многие расширения давно включены в состав ядра или распространяются в виде скомпилированных бинарных библиотек и легко устанавливаются.

Есть и другая часть — код, созданный на PHP, который условно можно поделить на следующие типы:

- компоненты — библиотеки на PHP, которые собираются при помощи менеджера пакетов Composer *(см. главу 26)*;
- фреймворки — готовые сборки, зачастую из компонентов, при помощи которых можно создавать сайты любой степени сложности. В книге, к сожалению, мы их не касаемся, однако если вы выберите PHP в качестве основного языка разработки, то не пройдете мимо них. На следующие PHP-фреймворки стоит обратить внимание: Symfony, Laravel, Zend, Yii. Всего их сотни, если не тысячи;
- готовые приложения — готовые к использованию разработки на PHP. Это системы управления контентом (Wordpress, Drupal), форумы (phpBB), Web-интерфейсы управления базами данных (phpMyAdmin, pgAdmin).

Книга, которую вы держите в руках, познакомит вас с языком; описанные выше системы созданы с использованием языка PHP, но требуют отдельного изучения, и, к сожалению, их обсуждение выходит за рамки нашей книги.

1.3. Сопутствующие технологии

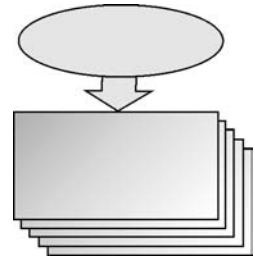
При помощи PHP можно быстро разрабатывать Web-сайты, однако современные реализации PHP — это не самая быстрая и эффективная часть сайта. Поэтому для запуска сайта потребуются дополнительное программное обеспечение и технологии.

- Web-сервер — программа, которая обеспечивает взаимодействие клиента и вашего приложения посредством протокола HTTP. На протяжении всей книги мы используем встроенный PHP-сервер *(см. главу 3)*, хотя для эксплуатации настоящего сайта потребуется Web-сервер nginx или Apache.
- Сервер базы данных — данные нужно где-то хранить. В книге довольно подробно рассматриваются две базы данных: PostgreSQL *(см. главы 27–28)* и Redis *(см. главы 29–30)*. Однако это далеко не все базы данных, которые вам встретятся на практике, да и каждая из затронутых заслуживает отдельной книги.
- Система контроля версий Git, которая служит для хранения истории разработки, резервного копирования, доставки кода на сервер, организации командной работы. Работаете ли вы в коллективе или в одиночку — Git в настоящий момент превратился в основной инструмент современного программиста, какой бы язык программирования не был выбран в качестве базового.

Этот список можно продолжать и продолжать. Однако на самом деле можно начать даже без этого — опираясь просто на язык PHP. Если вам интересно, что вас ждет после изучения PHP, загляните в заключение.

Задание

Установите систему Git и загрузите исходные коды к книге с GitHub: **https://github.com/igorsimdyanov/phpworkshop**.



ГЛАВА 2

Установка PHP

Листинги данной главы можно найти
в подкаталоге buildin.

В текущей главе будет рассмотрен порядок установки PHP в операционных системах Windows, Mac OS X и Linux (дистрибутив Ubuntu). Наличие PHP-интерпретатора позволит запускать PHP-программы, однако для запуска Web-приложений потребуется также Web-сервер. В ранних версиях PHP для этих целей использовался промышленный Web-сервер: Apache или nginx. Начиная с версии 5.4, дистрибутив PHP снабжается встроенным сервером, возможностей которого вполне хватает для локальной разработки и тестирования Web-приложений.

2.1. Установка в Windows

Для загрузки Windows-дистрибутива следует посетить раздел загрузки бинарных файлов официального сайта PHP: **http://windows.php.net/download**. Каждый релиз снабжается четырьмя вариантами:

- **x86 Non Thread Safe** — 32-битный CGI-вариант дистрибутива;
- **x86 Thread Safe** — 32-битный вариант для установки в качестве модуля Web-сервера;
- **x64 Non Thread Safe** — 64-битный CGI-вариант дистрибутива;
- **x64 Thread Safe** — 64-битный вариант для установки в качестве модуля Web-сервера.

Вариант **Thread Safe** предназначен для безопасного выполнения PHP в параллельных потоках в рамках одного системного процесса, например, если PHP устанавливается в качестве модуля Web-сервера Apache. Так как мы собираемся использовать встроенный сервер, не имеет значения, какой дистрибутив будет выбран, лучше всего воспользоваться вариантом **Non Thread Safe**. Последний вариант так же применяется при подключении PHP в качестве внешнего FastCGI-приложения, которое запускается на каждый внешний запрос.

Перед названием дистрибутива может быть помещена одна из аббревиатур VC11, VC14, означающих версии Visual Studio (2012 и 2015, соответственно), при помощи которой был скомпилирован дистрибутив. Для того чтобы успешно запустить проект, следует загрузить соответствующий распространяемый пакет Visual C++ для Visual Studio, который содержит необходимые динамические библиотеки:

- вариант VC11

  http://www.microsoft.com/en-us/download/details.aspx?id=30679;

- вариант VC14

  http://www.microsoft.com/en-us/download/details.aspx?id=48145.

> ***Внимание!***
> Необходимы библиотеки именно от английского варианта Visual Studio, русский вариант пакета не подойдет.

После загрузки zip-архива его следует распаковать в какую-нибудь папку, например C:\php.

Убедиться в том, что PHP доступен, можно, запустив командную строку, а затем перейти в папку C:\php при помощи команды[1]

```
> cd C:\php
```

Выполнив в командной строке команду `php` с параметром `-v`, можно узнать текущую версию PHP:

```
> php -v
PHP 7.0.0 (cli) (built: Dec  3 2015 09:31:54) ( NTS )
Copyright (c) 1997-2015 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2015 Zend Technologies
```

Для того чтобы команда PHP была доступна в любой точке файловой системы, путь к PHP-интерпретатору следует прописать в переменной окружения `PATH`.

Для доступа к переменным окружения нужно открыть Панель управления, перейти к разделу **Система**. Самый быстрый способ добраться до этого пункта — это щелкнуть правой кнопкой мыши по кнопке **Пуск** и выбрать пункт **Система** из контекстного меню. В операционных системах, предшествующих Windows 8, следует выбрать в меню **Пуск** пункт **Компьютер** и в контекстном меню выбрать пункт **Свойства**. В открывшемся окне Панели управления с активным разделом **Система** слева щелкнуть по ссылке **Дополнительные параметры системы**. Затем в окне **Свойства системы** на вкладке **Дополнительно** необходимо нажать кнопку **Переменные среды**. В открывшемся диалоговом окне в разделе **Системные переменные** следует отыскать переменную окружения `PATH` и дополнить ее путем к каталогу C:\php. Отдельные пути в значении переменной `PATH` разделяются точкой с запятой (в конце всей строки точка с запятой не требуется). После этого команда `php` будет доступна в любой папке компьютера.

---

[1] Полужирным шрифтом будем выделять команды или текст, вводимый пользователем.

2.2. Установка в Mac OS X

Прежде чем устанавливать PHP, следует установить Command Line Tools for XCode из магазина AppStore. XCode — это интегрированная среда разработки приложений для Mac OS X и iOS. Полная загрузка XCode не обязательна, достаточно установить инструменты командной строки и компилятор. Убедиться в том, установлен ли XCode, можно при помощи команды

```
$ xcode-select -p
/Applications/Xcode.app/Contents/Developer
```

Если вместо указанного выше пути выводится предложение установить Command Line Tools, следует установить этот пакет, выполнив команду

```
$ xcode-select --install
```

Теперь можно приступать к установке PHP, для чего лучше всего воспользоваться менеджером пакетов Homebrew. На момент написания книги установить Homebrew можно было при помощи команды

```
$ ruby -e "$(curl -fsSL
↳ https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Впрочем, точную команду всегда можно выяснить на официальном сайте **http://brew.sh**. После установки в командной строке будет доступна команда `brew`, при помощи которой можно загружать, удалять и обновлять пакеты с программным обеспечением.

Сразу после установки будет не лишним установить дополнительные библиотеки, которые могут потребоваться расширениям PHP:

```
$ brew install freetype jpeg libpng gd zlib
```

Для того чтобы получить доступ к репозиториям с PHP-дистрибутивами, следует выполнить серию команд:

```
$ brew tap homebrew/dupes
$ brew tap homebrew/versions
$ brew tap homebrew/homebrew-php
```

По умолчанию команда `brew tap` предполагает, что ей передаются названия GitHub-репозиториев. Таким образом, предыдущие команды прописывают доступ к следующим репозиториям:

https://github.com/Homebrew/homebrew-dupes

https://github.com/Homebrew/homebrew-versions

https://github.com/Homebrew/homebrew-php

Можно самостоятельно найти на GitHub дистрибутив или альтернативный проект и добавить его в менеджер пакетов Homebrew. Если пакет больше не нужен, его можно исключить при помощи команды `brew untap`. Разумеется, по возможности лучше использовать официальные пакеты Homebrew.

После выполнения приведенных выше команд надо убедиться в том, что пакеты успешно добавлены. Для этого следует выполнить команду `brew tap` без параметров. В результате будет выведен список добавленных репозиториев.

```
$ brew tap
homebrew/dupeshomebrew/php
homebrew/versions
```

Теперь можно установить PHP, запустив команду установки:

```
$ brew install php71
```

После установки PHP готов к работе.

```
$ php -v
PHP 7.1.2 (cli) (built: Feb 17 2017 10:51:21) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2017 Zend Technologies
```

Если PHP недоступен, следует убедиться, что путь к sbin-каталогу Homebrew прописан в переменной окружения `PATH` в файле ~/.bash_profile:

```
# Homebrew sbin path
PATH=/usr/local/sbin:$PATH
```

2.3. Установка в Linux (Ubuntu)

Для того чтобы установить PHP в Ubuntu, следует воспользоваться штатным менеджером пакетов `apt-get`. Предварительно нужно обновить сведения о репозиториях и текущие пакеты при помощи команд:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Затем можно приступить к установке:

```
$ sudo apt-get install php7.0
```

После успешной установки команда `php` должна быть доступна в любой точке компьютера:

```
$ php -v
PHP 7.0.15-0ubuntu0.16.04.4 (cli) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
    with Zend OPcache v7.0.15-0ubuntu0.16.04.4, Copyright (c) 1999-2017,
by Zend Technologies
```

2.4. Встроенный сервер

Для проверки работоспособности интерпретатора PHP создадим скрипт index.php, который выводит традиционную для книг по программированию фразу "Hello, world!" (листинг 2.1).

Листинг 2.1. Проверочный скрипт Hello World. Файл index.php

```
<?php
echo 'Hello, world!';
?>
```

Если у текущего пользователя отсутствуют права администратора, тогда в папке со скриптом index.php следует выполнить команду

```
php -S localhost:4000
```

Команда запустит на 4000-м порту Web-сервер. Обратившись в браузере по адресу **http://localhost:4000/**, можно увидеть фразу "Hello, world!". Если работа ведется из-под учетной записи системного администратора (Windows) или задействована команда `sudo` (Mac OS X или Linux), встроенный сервер можно запустить на стандартном 80-м порту:

```
php -S localhost:80
```

В этом случае порт в адресе можно не указывать, браузер автоматически будет обращаться по 80-порту, закрепленному за протоколом HTTP: **http://localhost/**. По умолчанию в качестве корневого каталога выступает текущая папка, именно в ней будет произведен поиск индексного файла index.php. Однако при помощи параметра `-t` можно указать произвольную папку:

```
php -S localhost:4000 -t buildin
```

Журнальные записи или, как еще говорят, логи сервера выводятся непосредственно в консоль, в которой он был запущен. Остановить сервер можно, нажав комбинацию клавиш <Ctrl>+<C>.

2.5. Файл hosts

В предыдущем разделе в качестве домена использовался localhost, который является псевдонимом для IP-адреса 127.0.0.1. На локальном хосте соответствие псевдонима IP-адресу прописывается в файле hosts, который в UNIX-подобной операционной системе можно обнаружить по пути /etc/hosts, а в Windows по пути C:\Windows\system32\drivers\etc\hosts.

Как правило, в файле присутствуют как минимум две записи, сопоставляющие домену localhost локальный IP-адрес 127.0.0.1 в IPv4- и IPv6-форматах:

```
127.0.0.1       localhost
::1             localhost
```

Именно наличие этих записей позволило нам запустить встроенный сервер с использованием в качестве домена localhost. Для того чтобы настроить альтернативные псевдонимы, можно добавить дополнительные записи:

```
127.0.0.1       site.dev
127.0.0.1       www.site.dev
```

```
127.0.0.2       project.dev
127.0.0.2       www.project.dev
```

IP-адреса, начинающиеся со 127, предназначены для локального использования, поэтому для тестирования собственных проектов вы можете назначать любые адреса из этого диапазона.

После добавления новых псевдонимов в файл hosts их можно использовать совместно со встроенным сервером, например **http://site.dev:4000/**.

2.6. Вещание вовне

При использовании локальных IP-адресов из диапазона 127.*X.X.X* можно быть уверенным, что никто извне не сможет обратиться к серверу. Если же наоборот требуется продемонстрировать результат работы вашего приложения, в качестве хоста можно указать IP-адрес 0.0.0.0:

```
php -S 0.0.0.0:4000
```

В этом случае можно получить доступ к Web-серверу, обратившись по IP-адресу хоста, где запущен сервер, например **http://192.168.0.1:4000**. Для того чтобы можно было обратиться к хосту по псевдониму, либо его придется прописать в hosts-файле каждого компьютера, с которого идет обращение к серверу, либо зарегистрировать доменное имя и связать его с IP-адресом компьютера, на котором сервер запущен. Разумеется, в этом случае при запуске в качестве хоста потребуется указать это доменное имя.

```
php -S example.com:80
```

Впрочем, встроенный сервер предназначен лишь для разработки, для обеспечения работы полноценного сайта лучше воспользоваться промышленными серверами, такими как Apache или nginx.

2.7. Настройка PHP

PHP имеет огромное количество разнообразных настроек, которые сосредоточены в файле php.ini. Сразу после установки вместо файла php.ini можно обнаружить лишь два файла:

- php.ini-production — рекомендованный набор параметров для рабочего сервера;
- php.ini-development — рекомендованный набор параметров для рабочей станции разработчика.

Для локальной разработки файл php.ini-development следует переименовать в php.ini. По умолчанию интерпретатор PHP последовательно ищет конфигурационный файл в следующих местах:

- по пути, указанному в переменной окружения `PHPRC`;
- в текущем каталоге (если скрипт выполняется под управлением Web-сервера);

- в каталоге C:\Windows\ в случае Windows, в каталоге /etc/ или /etc/php7/ в Linux, в каталоге /usr/local/etc/php7.0 в Mac OS X или в каталоге компиляции в случае любой другой операционной системы.

Если имеется несколько файлов php.ini и нет уверенности в том, какой из них используется в настоящий момент, выяснить путь к используемому файлу php.ini можно из отчета информационной функции `phpinfo()` (листинг 2.2).

Листинг 2.2. Информация о PHP. Файл phpinfo.php

```
<?php
phpinfo();
?>
```

Функция выводит подробную информацию обо всех параметрах PHP. Путь к файлу php.ini, который сейчас используется PHP, указывается в параметре `Loaded Configuration File`.

Можно явно указать PHP местоположение файла php.ini при помощи параметра `-c`. Для Windows команда может выглядеть следующим образом:

```
php -S 127.0.0.1:4000 -c C:\php\php.ini
```

Для UNIX-подобной операционной системы:

```
php -S 127.0.0.1:4000 -c /etc/php.ini
```

Конфигурационный файл PHP является обычным текстовым файлом, который можно открыть при помощи любого редактора. Содержимое файла php.ini состоит из секций и директив. Секции заключаются в квадратные скобки, например `[PHP]`, после которых следуют директивы, имеющие такой формат:

directive = *value*

Здесь *directive* — это название директивы, а *value* — ее значение. Все строки, в начале которых располагается точка с запятой (`;`), считаются комментариями и игнорируются.

Допускается не указывать значение *value*. В этом случае директива инициализируется пустой строкой. Этого же результата можно добиться, присвоив ей значение `none`.

Сразу после установки PHP следует отредактировать как минимум одну директиву — текущий часовой пояс. Для того чтобы выставить московский часовой пояс, необходимо найти директиву `date.timezone` и привести ее к виду:

```
date.timezone = 'Europe/Moscow'
```

Если встроенный сервер при этом оставался запущенным, потребуется его перезапуск для того, чтобы изменения в php.ini вступили в силу.

2.8. Расширения

Интерпретатор PHP строится по модульному принципу. Язык состоит из ядра, к которому при необходимости могут подключаться расширения — библиотеки, дополняющие язык новыми возможностями. Например, для обработки изображений можно подключить расширение GDLib, для связи с базами данных — расширение PDO, для поддержки многобайтовых строк (UTF-8) — mbstring. Часть расширений получили настолько широкую популярность, что их включили в состав ядра. Например, это строковые функции из расширения calendar или поддержка сессий из расширения session.

> ***Внимание!***
> Материал текущего раздела можно пропустить при первом чтении.

Каждое расширение увеличивает размер оперативной памяти, которую занимает интерпретатор PHP. Чем больше памяти потребляет интерпретатор, тем меньше процессов PHP можно запустить на сервере, тем меньше соединений сервер может обслужить в одну секунду. Поэтому расширения подключаются и отключаются по мере необходимости. Для того чтобы уменьшить размер исполняемого файла PHP, а следовательно, и объем потребляемой оперативной памяти, большинство расширений не включаются в состав ядра PHP. Вместо этого они компилируются в виде внешних динамических библиотек: *.dll для Windows и *.so для UNIX-подобных операционных систем.

Самый простой способ выяснить, подключено ли расширение, — это выполнить команду `php -m`, которая выводит список доступных расширений. Можно так же воспользоваться отчетом функции `phpinfo()` (см. листинг 2.2). В разделе `Configuration` указываются подключенные расширения, в таблицах, которые сопровождают каждое из расширений, приводится список параметров данного расширения. Большинство из этих параметров могут быть скорректированы в конфигурационном файле php.ini.

В дистрибутивах для операционной системы Windows расширения, оформленные в виде динамических библиотек, скомпилированы, но не подключены. Обнаружить их можно в подкаталоге ext основной папки PHP-дистрибутива:

```
php_bz2.dll
php_com_dotnet.dll
php_curl.dll
...
php_tidy.dll
php_xmlrpc.dll
php_xsl.dll
```

Для того чтобы подключить одно из таких внешних расширений, необходимо отредактировать конфигурационный файл php.ini. Путь к нему можно обнаружить в отчете функции `phpinfo()`. В конфигурационном файле php.ini следует найти директиву `extension_dir` и указать в ней путь к папке с расширениями:

```
extension_dir = "ext"
```

После того как путь к папке с расширением указан, можно активировать сами расширения, воспользовавшись директивой `extension`. В конфигурационном файле php.ini, как правило, уже добавлены закомментированные директивы для всех расширений из папки ext. Нужное расширение необходимо активировать, убрав точку с запятой из начала строки:

```
extension=php_mbstring.dll
```

В случае UNIX-подобных операционных систем установка расширений осуществляется еще проще. Как правило, они оформлены в виде отдельных пакетов.

Так в Mac OS X, в основном, используется менеджер пакетов Homebrew. В *разд. 2.2* PHP устанавливался при помощи команды

```
$ brew install php71
```

Точно так же устанавливается расширение, только название пакета `php71` дополняется именем расширения, например, для CURL команда может выглядеть следующим образом:

```
$ brew install php71-curl
```

В современных Linux-дистрибутивах установка расширений также осуществляется при помощи менеджера пакетов, в случае Ubuntu это `apt-get`. Напомним, что установка PHP осуществляется командой

```
$ sudo apt-get install php7.0
```

Точно так же устанавливаются расширения. Так, CURL можно установить при помощи команды

```
$ sudo apt-get install php7.0-curl
```

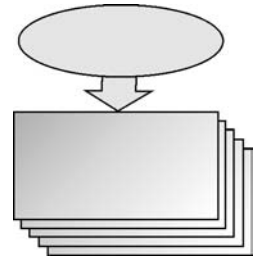
2.9. Документация

Главным источником информации о PHP для всех разработчиков является официальный сайт **http://php.net**. На его страницах можно обнаружить раздел документации, в том числе на русском языке. К сожалению, перевод на русский язык иногда не полон и отстает от английской версии. Мы не сможем осветить на страницах книги все возможности языка, нам просто не хватит для этого места, поэтому вам придется обращаться к документации PHP для того, чтобы воспользоваться всеми возможностями PHP.

Задания

1. При помощи `php -h` выведите справочную информацию о параметрах интерпретатора PHP. Найдите способ запуска интерактивного режима и выполните в нем фразу "Hello world".

2. Найдите в сети редактор Sublime Text и установите его в своей операционной системе. Создайте в нем проект Hello World с программой из листинга 2.1.
3. На сайте **http://php.net** воспользуйтесь поиском, найдите информацию о конструкции `echo` и функции `phpinfo()`.
4. Установите VirtualBox и запустите в нем альтернативную операционную систему: Ubuntu, если вы работаете в Windows или Mac OS X, или Windows, если ваша рабочая система — Ubuntu.
5. Установите Web-сервер nginx и запустите php-fpm, настройте их на совместную работу таким образом, чтобы они поддерживали обработку PHP-скриптов.



ГЛАВА 3

Быстрый старт

Листинги данной главы
можно найти в подкаталоге start.

Несмотря на то, что PHP является универсальным языком программирования и может применяться для разработки практически любого программного обеспечения, основная его специализация — Web-разработка.

В текущей главе будут показаны наиболее типичные приемы работы с PHP. Часть конструкций (`if`, `include`) будут освещены вскользь. В последующих главах мы остановимся на них более детально.

3.1. Скрипты

Язык программирования PHP считается скриптовым языком, поэтому программы, написанные на нем, называют *скриптами*. Главное отличие традиционных программ от скриптов заключается в том, что скрипты работают только в определенной среде и используют ресурсы данной среды.

Например, скриптовый язык программирования JavaScript работает преимущественно в Web-браузерах, Visual Basic for Applications — только в среде Microsoft Office. С использованием этих языков программирования невозможно создать программу, работающую без соответствующей среды.

В случае PHP в качестве такой среды выступает Web-окружение (Web-сервер, сервер базы данных, почтовый сервер и т. п.). Именно он принимает запросы от клиента, обеспечивает их параллельное выполнение и отправку данных. PHP-скрипт получает всю информацию о запросе и выполняет запрос, а затем отправляет данные обратно серверу.

ЗАМЕЧАНИЕ

Впрочем, язык программирования PHP допускает создание программ, работающих независимо от Web-сервера, однако в такой форме он не получил сколько бы то ни было широкого распространения.

Одной из главных особенностей языка программирования PHP является тот факт, что его код может располагаться вперемешку с HTML-кодом. Для того чтобы интерпретатор PHP различал HTML- и PHP-коды, последний заключается в специальные теги `<?php` и `?>`, между которыми располагаются конструкции и операторы языка программирования PHP.

В листинге 3.1 приводится классический пример, выводящий в окно браузера при помощи конструкции `echo` фразу "Hello, world!" ("Привет, мир!"). Содержимое листинга 3.1 следует поместить в файл с расширением php, например в файл index.php.

> ***Замечание***
>
> Первоначальная версия PHP разрабатывалась как шаблонизатор — система, встраиваемая в HTML-код для выполнения операций, которые не поддерживаются статическим HTML. По мере того, как PHP трансформировался в полноценный язык, стала приобретать популярность обратная тенденция — отделение PHP и HTML-кода.

Листинг 3.1. Простейший PHP-скрипт. Файл index.php

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <title>Простейший PHP-скрипт</title>
    <meta charset='utf-8'>
  </head>
  <body>
    <?php
      echo "Hello, world!";
    ?>
  </body>
</html>
```

Конструкция `echo` выводит одну или несколько строк в стандартный вывод. В результате работы скрипта в окно браузера будет выведена фраза "Hello, world!".

При работе с серверными языками программирования, такими как PHP, следует помнить, что скрипты, расположенные между тегами `<?php` и `?>`, выполняются на сервере. Клиенту приходит лишь результат работы PHP-кода, в чем можно легко убедиться, просмотрев исходный код HTML-страницы.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <title>Простейший PHP-скрипт</title>
    <meta charset='utf-8'>
  </head>
  <body>
    Hello, world!  </body>
</html>
```

3.2. Начальные и конечные теги

Как было указано в предыдущем разделе, PHP-скрипт должен быть размещен между начальным тегом <?php и конечным тегом ?> для того, чтобы интерпретатор мог разделить HTML- и PHP-коды. Даже если HTML-код не используется, указание PHP-тегов является обязательным, в противном случае PHP-код будет выведен в окно браузера как есть, без интерпретации. Помимо тегов <?php и ?>, PHP поддерживает специальный тип тегов <?= ... ?> для вывода результата одиночного PHP-выражения. Например, скрипт из листинга 3.1 можно было бы переписать так, как это продемонстрировано в листинге 3.2.

Листинг 3.2. Альтернативные теги. Файл shortags.php

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <title>Альтернативные теги</title>
    <meta charset='utf-8'>
  </head>
  <body>
    <?= "Hello, world!"; ?>
  </body>
</html>
```

Как видно из примера выше, для вывода строки "Hello, world!" не требуется использовать конструкцию echo, тег <?= автоматически выводит результат в стандартный поток.

Следует отметить, что HTML-страница может содержать более чем одну PHP-вставку. В листинге 3.3 приводится пример, который содержит две вставки: одна задает название страницы (в HTML-теге <title>), а вторая определяет содержимое страницы (в HTML-теге <body>).

Листинг 3.3. Допускается несколько PHP-вставок в HTML-код. Файл few.php

```
<html>
  <head>
    <title><?php echo "Вывод текущей даты" ?></title>
  </head>
  <body>
    <?php
      echo "Текущая дата:<br />";
      echo date(DATE_RSS);
    ?>
  </body>
</html>
```

Если после завершающего тега ?> нет никакого вывода, его можно опустить (листинг 3.4).

Листинг 3.4. Завершающий тег ?> можно опускать. Файл missing.php

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

Более того, стандарт кодирования PSR-2, который определяет форматирование PHP-кода для распространяемых компонентов, требует не указывать завершающий тег ?> во всех случаях, где это возможно.

Встретив символ, например, пробел, интерпретатор PHP считает, что начинается вывод HTTP-документа и формирование предшествующего ему HTTP-заголовка завершено. Поэтому более поздние попытки отправить HTTP-заголовки будут завершаться ошибкой. Если же завершающий тег не используется, такие ошибки исключаются как класс.

Стандарты PSR определяют правила разработки компонентов PHP, их цель — унифицировать фреймворки и компоненты, распространяемые через менеджер Composer *(см. главу 2б)*, чтобы компоненты одного фреймворка могли использоваться в другом.

На момент написания книги было принято 8 стандартов и 9 находилось в процессе подготовки. Следить за процессом принятия стандартов можно на странице **http://www.php-fig.org/psr/**. Стандарты PSR-1 и PSR-2 определяют форматирование и стилевое оформление PHP-кода. Мы будем придерживаться этих стандартов и сообщать о правилах форматирования по мере изучения языка.

3.3. Использование точки с запятой

Совокупность конструкций языка программирования, завершающуюся точкой с запятой, будем называть *выражением*.

Как видно из листинга 3.3, после строки "Вывод текущей даты" не указывается точка с запятой. Выражение одно, и надобность отделять его от других выражений отсутствует. Однако, как можно видеть во второй вставке, в конце каждой из конструкций echo имеется точка с запятой. Если забыть указать этот разделитель, интерпретатор языка программирования PHP посчитает выражение на новой строке продолжением предыдущего и не сможет корректно разобрать скрипт. В результате будет сгенерировано сообщение об ошибке "Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ',' or ';'" ("Ошибка разбора: синтаксическая ошибка, неожиданно встречена конструкция echo, ожидается либо запятая ',', либо точка с запятой ';'").

Последнее выражение перед завершающим тегом ?> можно не снабжать точкой с запятой. Например, в листинге 3.3 после выражения echo date(DATE_RSS) точку с запятой можно не указывать. Однако настоятельно рекомендуется не пользовать-

ся этой особенностью и помещать точки с запятой после каждого выражения, т. к. добавление новых операторов может привести к появлению трудноулавливаемых ошибок.

Переводы строк никак не влияют на интерпретацию скрипта, выражение может быть разбито на несколько строк — интерпретатор PHP будет считать, что выражение закончено лишь после того, как обнаружит точку с запятой или завершающий тег ?>. В листингах 3.5 и 3.6 представлены два скрипта, аналогичные по своей функциональности.

Листинг 3.5. Использование точки с запятой. Файл semicolumn.php

```
<?php
echo 5 + 5;
echo 5 - 2;
echo "Hello, world!";
```

Листинг 3.6. Альтернативная запись скрипта из листинга 3.5. Файл mech.php

```
<?php
echo 5
     +
     5; echo 5 -
     2; echo"Hello, world!"
     ;
```

Следует избегать конструкций, подобной той, которая приведена в листинге 3.6. Чем более понятно и ожидаемо написан код, тем проще и быстрее его отлаживать.

3.4. Составные выражения. Фигурные скобки

Фигурные скобки позволяют объединить несколько выражений в группу, которую обычно называют *составным выражением* (листинг 3.7).

Листинг 3.7. Составное выражение. Файл curly.php

```
<?php
{
    echo 5 + 5;
    echo 5 - 2;
    echo "Hello, world!";
}
```

Как видно из примера выше, выражения внутри фигурных скобок располагаются с отступом. Такой отступ не обязателен, однако он повышает читаемость программы. Стандарт кодирования PSR-2 требует, чтобы отступ оформлялся 4 пробелами.

Если вы привыкли к использованию символа табуляции, следует настроить свой редактор на замену символа табуляции пробелами.

Само по себе составное выражение практически никогда не используется, основное его предназначение — совместная работа с условными операторами, операторами цикла и т. п., которые мы рассмотрим в последующих главах.

Составное выражение может быть расположено в нескольких PHP-вставках. В листинге 3.8 приводится пример двух составных выражений, которые разбиты несколькими PHP-вставками. Задача скрипта сводится к случайному выводу в окно браузера либо зеленого слова "Истина", либо красного слова "Ложь". Без использования фигурных скобок оператор `if` распространял бы свое действие только на одно выражение, использование составного выражения позволяет распространить его действие на несколько простых выражений.

Замечание

Условный оператор `if` рассматривается в *главе 8*.

Листинг 3.8. Составное выражение в нескольких PHP-вставках. Файл fews.php

```
<?php
if(mt_rand(0, 1)) {
    ?>
    <div style='color:green'><?= "Истина"; ?></div>
    <?php
} else {
    ?>
    <div style='color:red'><?= "Ложь" ?></div>
    <?php
}
```

Как видно из листинга 3.8, составное выражение в любой момент может быть прервано тегами `<?php` и `?>`, а затем продолжено. Впрочем, существуют исключения, например, составное выражение, применяемое для формирования класса нельзя разбивать тегами `<?php` и `?>`.

3.5. Комментарии

Код современных языков программирования является достаточно удобным для восприятия человеком по сравнению с машинными кодами, ассемблером или первыми языками программирования высокого уровня. Тем не менее, конструкции языка продиктованы архитектурой компьютера, и, создавая программы, разработчик волей-неволей использует компьютерную, а не человеческую логику. Это зачастую приводит к созданию достаточно сложных построений, которые нуждаются в объяснении на обычном языке. Для вставки таких пояснений в код предназначены *комментарии*.

PHP предоставляет несколько способов для вставки комментариев, варианты которых представлены в табл. 3.1.

Таблица 3.1. *Комментарии PHP*

Комментарий	Описание
`// ...`	Комментарий в стиле языка C++, начинающийся с символа двух слешей `//` и заканчивающийся переводом строки
`# ...`	Комментарий в стиле скриптовых языков UNIX, начинающийся с символа диеза `#` и заканчивающийся переводом строки
`/* ... */`	Если два предыдущих комментария ограничены лишь одной строкой, то комментарий в стиле языка C `/* ... */` является многострочным

В листинге 3.9 демонстрируется использование всех трех видов комментариев из табл. 3.1.

Листинг 3.9. Комментарии. Файл comments.php

```
<?php
/*
  Демонстрация разных типов комментариев
  в языке программирования PHP
*/
echo 'Hello'; // это комментарий
echo 'Hello'; # и это комментарий
```

Естественно, что комментарии PHP действуют только внутри тегов-ограничителей `<?php ... ?>`. То есть, если символы комментариев будут находиться вне тегов-ограничителей, то они, как и любой текст, будут отображены браузером (листинг 3.10).

Листинг 3.10. Комментарии действуют только внутри `<?php` и `?>`. Файл into.php

```
<?php
echo "Hello<br />"; // комментарий PHP
?>
// этот текст отобразится браузером.
<!-- Этот текст не будет отображен браузером, поскольку заключен между
символами, являющимися комментариями HTML. Однако он может быть просмотрен
в исходном коде HTML-страницы -->
```

Комментарии можно вставлять не только после точки с запятой, но и в середине выражения (листинг 3.11).

Листинг 3.11. Комментарий в списке аргументов функции. Файл position.php

```
<?php
echo strstr( // эту функцию мы рассмотрим позднее
    "Hello, world", "H");
```

3.6. Включение PHP-файла

До этого момента мы имели дело лишь с одним PHP-скриптом. Однако PHP-скрипты можно подключать к другим PHP-скриптам при помощи двух конструкций: `include` и `require`. Обе принимают единственный аргумент — путь к включаемому файлу, и результатом их действия является подстановка содержимого файла в место их вызова в исходном скрипте. Если в качестве включаемого скрипта выступает PHP-скрипт, то сначала происходит его подстановка в исходный скрипт, а затем интерпретация результирующего скрипта (листинг 3.12).

Листинг 3.12. Использование инструкции `include`. Файл include.php

```
<?php
echo 'Основной скрипт<br />';
include 'included.php';
echo 'Основной скрипт<br />';
```

Пусть файл included.php содержит код, представленный в листинге 3.13.

Листинг 3.13. Файл included.php

```
<?php
echo 'Включаемый файл<br />';
?>
<h3>Текст не обязательно должен выводиться оператором echo</h3>
```

В результате запуска скрипта из листинга 3.12 в браузер будут выведены следующие строки

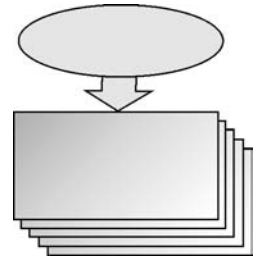
```
Основной скрипт
Включаемый файл
Текст не обязательно должен выводиться оператором echo
Основной скрипт
```

Различие `include` и `require` заключается в их реакции на отсутствие включаемого файла. В случае `include` выводится предупреждение, весь последующий код продолжает выполняться; в случае `require`, если нельзя найти файл, работа скрипта останавливается.

Задания

1. Найдите в документации на сайте **http://php.net** функцию `date()`, которая занимается форматированием даты, изучите приемы работы с функцией.
2. Найдите в документации функции для работы с датой и временем `time()` и `mktime()`, ознакомьтесь с возможностями, которые предоставляют эти функции. С использованием календарных функций выведите текущие дату и время в следующем формате: `'10.07.2018 14:10'`.
3. Найдите в документации функцию генерации случайных чисел `mt_rand()`, ознакомьтесь с ее возможностями. Создайте скрипт, который генерировал бы случайное число от 0 до 1000.

ГЛАВА 4



Переменные и типы данных

Листинги данной главы
можно найти в подкаталоге variables.

Ключевым объектом практически любого языка программирования является переменная. Под *переменной* в общем случае понимается именованная область памяти. В этой области может храниться либо строка, либо число, либо сложный объект. Манипулировать этим значением можно при помощи имени переменной (его мы далее для простоты будем называть просто переменной). То, что хранится в области памяти, будем называть *значением переменной*.

4.1. Объявление переменной. Оператор =

В PHP переменные начинаются со знака доллара (`$`), за которым может следовать любое количество буквенно-цифровых символов и символов подчеркивания, но первый символ не может быть цифрой. Таким образом, допустимы следующие имена переменных: `$n`, `$n1`, `$user_func_5` и т. д. В отличие от ключевых слов, имена переменных в PHP *чувствительны к регистру*, т. е. переменные `$user`, `$User` и `$USER` являются различными (листинг 4.1).

Листинг 4.1. Зависимость переменных от регистра. Файл case_sensitive.php

```
<?php
$user = "Владимир";
$User = "Дмитрий";
$USER = "Юрий";
echo $user; // Владимир
echo $User; // Дмитрий
echo $USER; // Юрий
```

Как видно из листинга 4.1, для присвоения значения переменной необходимо воспользоваться оператором присваивания `=`, который позволяет инициализировать переменную.

При объявлении числовых значений в качестве разделителя целого значения и дробной части выступает точка (листинг 4.2).

Листинг 4.2. Объявление чисел. Файл number_set.php

```
<?php
$number = 1;
$var = 3.14;
```

Допускается инициализация одним значением сразу нескольких переменных за счет того, что оператор = возвращает результат присвоения. В листинге 4.3 переменным `$num`, `$number` и `$var` присваивается значение 1 в одну строку за счет использования оператора = в цепочке.

Листинг 4.3. Инициализация переменных одним значением. Файл multi_set.php

```
<?php
$num = $number = $var = 1;
```

4.2. Типы данных

Язык PHP является слаботипизированным, в большинстве случаев переменные языка не требуют строгого задания типа при их объявлении, а в ходе выполнения программы тип переменной может быть практически всегда изменен неявным образом без специальных преобразований.

Например, переменная, объявленная строкой, может использоваться далее в арифметических операциях, выступать как логическая переменная, а в конце ей в качестве значения может быть присвоен объект. Все это позволяет разработчику практически не задумываться о типах данных.

Тем не менее некоторые типы, такие как `null`, `object`, `array` или `resource`, настолько специфичны, что без учета их особенностей невозможна успешная разработка.

Более того, при использовании функций и методов вы сможете явно указывать тип их параметров и возвращаемых значений, которые будут рассмотрены более подробно в *главе 11*.

В табл. 4.1 представлены типы данных, которые поддерживаются PHP.

Замечание

В документации и в нашей книге помимо типов данных, представленных в табл. 4.1, вам будут встречаться *псевдотипы* — условные обозначения одного или нескольких типов. В основном они используются совместно функциями и методами (*см. главы 11* и *16*). Например, `mixed` — любой тип, `number` — либо `integer`, либо `double`, `iterable` — итерируемый объект. Особняком стоит ключевое слово `void`, обозначающее отсутствие переменной. Иногда несколько типов могут быть разделены вертикальной чертой `|` для обозначения того факта, что переменная может принимать один из указанных типов. Например, псевдотип `number` можно было бы записать, используя следующую комбинацию базовых типов: `integer|double`.

Таблица 4.1. *Типы данных PHP*

Тип данных	Описание
`integer`	Целое число, максимальное значение которого зависит от разрядности операционной системы. В случае 32-битной операционной системы число может принимать значения от –2 147 483 648 до 2 147 483 647. Если разрядность составляет 64 бита, диапазон возможных значений — от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
`double` (или `float`)	Вещественное число, минимально возможное значение которого составляет от $\pm 2.23 \times 10^{-308}$ до $\pm 1.79 \times 10^{308}$
`boolean`	Логический тип, способный принимать лишь два значения: `true` (истина) и `false` (ложь)
`string`	Строковый тип. Может хранить строку, максимальный размер которой составляет 2 Гбайт
`array`	Массив. Это объединение нескольких переменных под одним именем. Обращаться к отдельным переменным можно при помощи индекса массива. Более подробно массивы обсуждаются в *главе 10*
`object`	Объект. Это конструкция, объединяющая несколько разнотипных переменных и методы их обработки
`resource`	Дескриптор, позволяющий оперировать тем или иным ресурсом, доступ к которому осуществляется при помощи библиотечных функций. Дескрипторы применяются при работе с файлами, базами данных, динамическими изображениями и т. д. Более подробно дескрипторы будут рассмотрены в соответствующих главах
`null`	Специальный тип, который сигнализирует о том, что переменная не была инициализирована
`callable` (или `callback`)	Некоторые функции PHP могут принимать в качестве аргументов другие функции, которые называются *функциями обратного вызова*. Переменные данного типа содержат ссылки на такие функции

В текущей главе мы изучим лишь целые, вещественные числа, логический тип, строки и тип `null`. Остальные типы будут рассмотрены в следующих главах.

4.3. Целые числа

Целые числа являются наиболее распространенными в программировании. Это связано с тем, что большинство прикладных задач носит арифметический характер, а также с тем, что это наиболее быстродействующий тип данных.

> ***Замечание***
>
> В отличие от других языков программирования переполнения (т. е. выхода значения за допустимые границы) в PHP не бывает, если полученное значение не убирается в `integer`, для него автоматически выбирается больший тип данных (`double`).

Целочисленная переменная может быть объявлена несколькими способами (листинг 4.4).

Листинг 4.4. Объявление целочисленных переменных. Файл integer.php

```
<?php
$num = 1234; // десятичное число
$num = +123; // десятичное число
$num = -123; // отрицательное число
$num = 0123; // восьмеричное число (эквивалентно 83)
$num = 0x1A; // шестнадцатеричное число (эквивалентно 26)
```

Целое положительное число объявляется, как правило, без указания плюса перед ним (впрочем, его использование не приводит к ошибке). Для объявления отрицательного числа перед ним размещается символ минуса -.

По умолчанию числа задаются в десятичной системе счисления, однако PHP позволяет объявлять переменные в восьмеричной и шестнадцатеричной системах счисления.

В восьмеричном числе основанием служит 8, а для выражения всех чисел используются цифры от 0 до 7. Число 8 в восьмеричной системе счисления играет ту же роль, что число 10 в десятичной.

Шестнадцатеричная система счисления (основанием служит число 16) использует цифры от 0 до 9 и буквы английского алфавита от A до F, означающие шестнадцатеричные "цифры" 10, 11, 12, 13, 14 и 15. Числа, объявленные в восьмеричной системе счисления, предваряются префиксом 0, в шестнадцатеричной системе — префиксом 0x.

> ***Замечание***
>
> Может показаться, что восьмеричные и шестнадцатеричные числа являются избыточным наследием языка C, однако они применяются и в Web-разработке, например, при задании прав доступа к файлам и папкам, кодировании цвета и т. д.

4.4. Вещественные числа

Вещественные числа (`float` или `double`) позволяют сохранять данные в огромном интервале, за пределы которого прикладные программы не выходят практически никогда. Различают две формы записи вещественного числа: стандартную и экспоненциальную (листинг 4.5).

> ***Замечание***
>
> При выводе под число с плавающей точкой отводится 12 символов, это значение может быть изменено при помощи директивы `precision` в конфигурационном файле php.ini.

Листинг 4.5. Объявление вещественных чисел. Файл double.php

```
<?php
// Объявление положительного вещественного числа
// Стандартная запись
$var = 1.23456;
```

```
// Объявление отрицательного вещественного числа
// Стандартная запись
$var = +1.23456;
// Объявление положительного вещественного числа
// больше единицы. Научная запись.
$var = 1.23456E2; // 123.456
// Объявление положительного вещественного числа
// больше единицы. Научная запись.
$var = 1.23456E+2; // 123.456
// Объявление положительного вещественного числа
// меньше единицы. Научная запись.
$var = 1.23456E-3; // 0.00123456
// Объявление отрицательного вещественного числа
// Научная запись
$var = -1.23456e-2; // -0.0123456
```

Экспоненциальная запись помимо мантиссы (целой и дробной части) содержит порядок, который начинается со строчной или прописной буквы E и целого положительного (или отрицательного) числа. Так запись 1.2e–3 эквивалентна произведению 1.2×10^{-3} (или 0.0012). Вещественное число 1.1e+2 (или 1.1e2) эквивалентно 1.1×10^{2} (или 110.0).

Вещественные числа преобразуются в компьютерное представление с потерями. Это связано с тем, что некоторые дроби в десятичной системе счисления невозможно представить конечным числом цифр. Так дробь 1/3 в десятичной форме принимает периодическую форму 0.33333333..., и часть цифр приходится отбрасывать. Это приводит к тому, что при операциях могут накапливаться ошибки вычисления, например, число 1.3 может принимать форму 1.29999999... Такое поведение вещественных чисел следует учитывать в программах, особенно при сравнении их друг с другом *(см. главу 7)*.

4.5. Логический тип

Переменные логического типа `boolean` принимают только два значения: `true` (истина) или `false` (ложь). В листинге 4.6 приводится объявление логической переменной, принимающей значение `true`.

> ***Замечание***
>
> Константы `true` и `false` не зависят от регистра. В коде допускается использование форм `True` и `TRUE`. Однако стандарт PSR-2 требует записи констант строчными буквами: `true` и `false`. Более подробно константы обсуждаются в *главе 6*.

Листинг 4.6. Объявление переменной логического типа. Файл boolean.php

```
<?php
$bool = true;
```