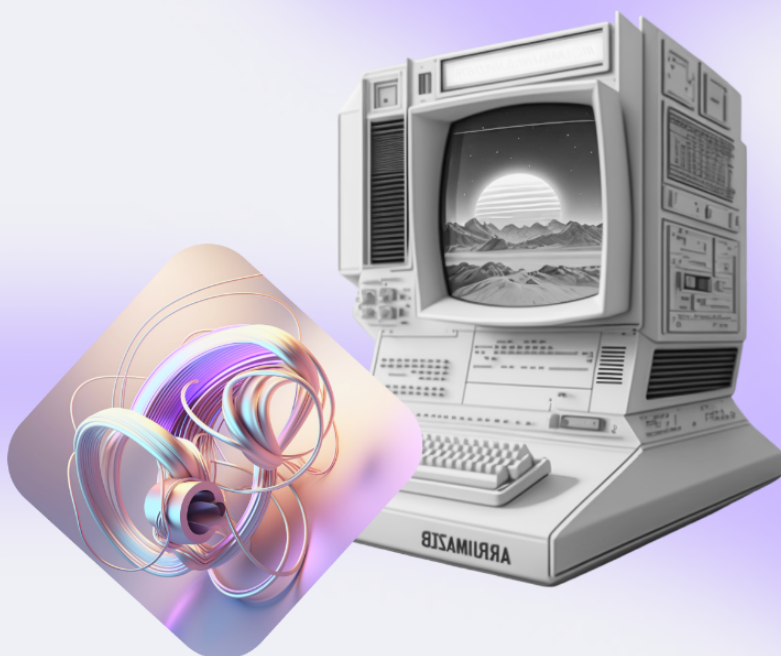


# Введение в PHP

## Основы PHP



# Оглавление

Введение	3
Словарь терминов	4
Области применения языка PHP	6
Инфраструктура для работы веб-приложения	10
PHP в командной строке	10
PHP в браузере	12
Начинаем программировать	19
Заключение	26
Домашнее задание	27
Что можно почитать еще?	27

# Введение

Этой лекцией мы начинаем наше знакомство с языком PHP – мощным языком программирования, который широко применяется для создания различных веб-приложений.

Вы уже успели ознакомиться со многими фундаментальными понятиями в сфере разработки, которые помогут в освоении нового этапа обучения.

PHP был создан в 1994 году, но основной свой виток развития получил в 1998, когда получил глубокую переработку своего движка и принял основные черты того языка, с которым программисты по всему миру работают до сих пор. В настоящее время PHP стал одним из самых используемых языков программирования в мире. Он широко применяется в крупных проектах, таких как Facebook, Wikipedia, Yahoo и многих других.

Одной из главных причин популярности PHP является его простота в освоении и широкие возможности. PHP позволяет создавать динамические веб-страницы, обрабатывать пользовательские данные и работать с различными типами хранилищ.

Кроме того, PHP имеет огромную базу пользовательских библиотек и фреймворков, которые позволяют значительно ускорить и упростить разработку веб-приложений.

PHP – это серверный язык, то есть он выполняется на стороне централизованного сервера и отвечает за генерацию ответа, который отправляется клиенту. Чаще всего это HTML-код, который браузер при получении формирует для отображения в виде страницы.

Что будет на уроке сегодня:

- Знакомство с языком;
- Инфраструктура для работы веб-приложения;
- Отличия cli от web;
- Собираем контейнеры для web и для cli;
- Напишем простой скрипт с применением различных переменных;
- Познакомимся с типами и их приведением.

PHP является одним из самых популярных языков программирования для создания веб-приложений.

Поэтому давайте поскорее начнем наше знакомство с языком.

## Словарь терминов

**PHP** – это серверный язык, то есть он выполняется на стороне централизованного сервера и отвечает за генерацию ответа, который отправляется клиенту. Чаще всего это HTML-код, который браузер при получении формирует для отображения в виде страницы.

**PHP** – интерпретируемый язык программирования.

**CMS (Content Management System)** – это программа, которая позволяет управлять созданием, редактированием и публикацией содержимого на веб-сайте без необходимости знания программирования или HTML.

**Хостинг (hosting)** – это услуга предоставления ресурсов сервера для размещения веб-сайтов и обеспечения их доступности в интернете.

**HTTP-запрос** – набор текстовой информации, содержащий данные о том, какую страницу нужно получить.

**<?php** – это открывающий дескриптор, начало блока кода на языке PHP. Эта строка говорит интерпретатору PHP, что далее идет PHP-код. Поскольку PHP код может быть встроен в HTML, серверу нужно понимать, что из инструкций должно обрабатываться именно интерпретатором.

**<echo** – это ключевое слово на языке PHP, которое обозначает оператор, используемый для вывода текста на экран или веб-страницу. Оно может выводить любой текст, переданный ему в качестве аргумента.

**<;** – это символ точки с запятой, который означает конец оператора на языке PHP. В языке PHP каждый оператор должен быть завершен символом «;».

**Nginx** – web-сервер наиболее популярный и актуальный на данный момент, он принимает HTTP-запросы, определяет, динамические они или статические, а затем – передает в FPM динамические запросы.

**PHP-FPM (PHP FastCGI Process Manager)** – это компонент PHP, который предоставляет реализацию FastCGI для PHP-скриптов. Он является отдельным процессом, выполняющим PHP-скрипты и взаимодействующим с веб-сервером через протокол FastCGI. PHP-FPM предлагает ряд опций настройки и управления процессами PHP, таких как управление пулами процессов PHP, регулирование количества рабочих процессов PHP и управление ресурсами.

**FastCGI (Fast Common Gateway Interface)** – это протокол взаимодействия между веб-сервером и веб-приложениями, который позволяет серверу обращаться к внешним процессам, выполняющим скрипты на сервере, и получать от них ответы. FastCGI используется для улучшения производительности веб-серверов, таких как Nginx, при обработке динамических контентных страниц, таких как PHP, Python, Ruby и других.

**hosts** – это локальный файл на компьютере, который используется для сопоставления IP-адресов и доменных имен.

**code** – это папка для volume, который будет пробрасываться в контейнеры.

**fpm** – это папка с Dockerfile для PHP-FPM.

**nginx** – это папка для Dockerfile и конфигурации nginx.

**Переменная** – это контейнер в памяти сервера.

«\$» – переменная в PHP объявляется знаком доллара.

«=» – присваивание значения переменной происходит оператором присваивания.

**title** – переменная, которая объявляется явно.

**today** – переменная, которая формируется при помощи встроенной в PHP функции date.

**Тип null** – пустота. Это тип, который имеет переменная, если её создать, но не присвоить ей значение.

**Тип bool** – булевский тип, принимающий true или false.

**Тип int** – целочисленный тип, ограниченный разрядностью вашего процессора.

**Тип float** – число с плавающей точкой (дробь).

**Тип string** – строка, которую мы уже использовали выше.

**Тип array** – массив.

**Тип object** – объект.

**Тип resource** – ресурс, который применяется для хранения ссылок на подключение к базам данных, файлам и внешним сетевым ресурсам.

**Тип never** – это не тип переменной, но его можно будет указать как возвращаемый у функции, о чём мы поговорим позже.

**Тип void** – это не тип переменной, но его можно будет указать как возвращаемый у функции, о чём мы поговорим позже.

**Приведение типов в PHP** – это процесс изменения типа переменной на другой тип. PHP автоматически приводит типы при выполнении операций, но иногда требуется явное приведение типов.

**PSR (PHP Standards Recommendation)** – это набор рекомендаций, разработанных PHP-сообществом, для стандартизации разработки на языке программирования PHP. PSR представляет собой набор соглашений и рекомендаций, которые помогают разработчикам создавать читаемый, поддерживаемый и совместимый код на PHP.

**PHP-FIG** – это группа разработчиков из разных проектов и фреймворков на PHP, которые сотрудничают для создания стандартов и рекомендаций, чтобы облегчить совместную работу между различными проектами и повысить качество PHP-кода в целом.

**Алиас (Alias)** – псевдоним или сокращение для другого объекта или команды.

## Области применения языка PHP

Как мы уже выяснили выше, наиболее подходящей областью для языка является создание web-сайтов. Неспроста статистика показывает, что на языке PHP работает около **80 %** сайтов во всей сети Интернет.



Как правило, самый простой способ создания таких сайтов – это применение CMS (Content Management System).

Это программа, которая позволяет управлять созданием, редактированием и публикацией содержимого на веб-сайте без необходимости знания программирования или HTML. Однако отсутствие необходимости знания программирования вовсе не означает, что сама система обходится без кода. Такие системы часто пишутся на языке PHP.


Существует множество CMS на PHP, но вот некоторые из наиболее популярных:

- **WordPress:** одна из самых популярных и распространенных CMS на PHP. WordPress предоставляет широкий спектр возможностей, от простых блогов до сложных корпоративных веб-сайтов.
- **Joomla:** еще одна популярная CMS на PHP, предлагающая множество расширений и шаблонов для создания различных типов веб-сайтов.


- **Drupal:** это мощная CMS на PHP, которая используется для создания сложных и высокопроизводительных веб-сайтов. Drupal предлагает широкие возможности для настройки и расширения функциональности.
- **Magento:** CMS, специализирующаяся на создании электронной коммерции. Magento предоставляет функции для создания онлайн-магазинов с возможностью управления товарами, заказами, платежами и другими функциями электронной коммерции.
- **Битрикс:** это система, которая предоставляет различные инструменты и функциональность для создания и управления различными типами веб-сайтов, включая корпоративные сайты, интернет-магазины, порталы и другие онлайн-проекты.

Это только некоторые примеры CMS на PHP. Существует и другие CMS на PHP, которые могут соответствовать конкретным требованиям и потребностям разработчиков и владельцев веб-сайтов.

Поскольку развернуть сайт на CMS достаточно просто, многие хостинги предоставляют возможность получить такой сайт вообще без каких-либо действий, кроме кликов мышки.

 Хостинг (hosting) – это услуга предоставления ресурсов сервера для размещения веб-сайтов и обеспечения их доступности в интернете. Хостинг-провайдеры предлагают серверное пространство и другие необходимые ресурсы для хранения файлов веб-сайта и обработки запросов от пользователей.

Когда вы создаете веб-сайт, все его файлы, включая HTML-страницы, изображения, видео, скрипты и другие файлы, должны быть размещены на сервере, который всегда включен и подключен к интернету.

 Хостинг-провайдеры предоставляют пространство, где вы можете разместить свои файлы и обеспечить доступ к ним через интернет.

Основные типы хостинга включают:

- **Общий хостинг (shared hosting):** ваш сайт размещается на сервере, который разделяется с другими клиентами хостинг-провайдера. Вы платите только за

свою долю ресурсов сервера. Общий хостинг является наиболее доступным вариантом и хорошим выбором для небольших и средних веб-сайтов с низкой нагрузкой.

- **Виртуальный выделенный хостинг (virtual private server, VPS):** ваш сайт размещается на виртуальном сервере, который имеет отдельные выделенные ресурсы. VPS-хостинг предоставляет большую гибкость и масштабируемость по сравнению с общим хостингом.
- **Выделенный хостинг (dedicated hosting):** вы полностью арендуете физический сервер и имеете полный контроль над всеми его ресурсами. Выделенный хостинг предоставляет высокую производительность и подходит для больших и высоконагруженных веб-проектов.
- **Облачный хостинг (cloud hosting):** ваш сайт размещается на нескольких серверах, работающих вместе в кластере. Облачный хостинг обеспечивает высокую доступность, масштабируемость и отказоустойчивость.

Но в целом для получения сайта на базе CMS Wordpress вам нужно сделать ряд достаточно [простых шагов](#).


- **Шаг 1.** Выберите хостинг. В зависимости от типа хостинга будет меняться стоимость и условия предоставления услуг. Самый простой вариант – это **shared hosting**. Как только вы создадите аккаунт и оплатите услуги, вам станут доступны ресурсы сервера.

Скорее всего, вам пришлют доступ к сайту по протоколу SSH или FTP. Поэтому стоит озаботиться программой для подключения. На базе Windows это может быть WinSCP, на Linux – Ásbrú Connection Manager, а на Mac – Termius.

- **Шаг 2.** Скачайте Wordpress. Это можно сделать с [официального сайта](#). Полученный архив нужно распаковать и положить в корневую директорию сайта вашего хостинга, подключившись к нему через установленный на шаге 1 клиент. На разных хостингах они настроены по-разному. Но, как правило, формируются из указания доменного имени.
- **Шаг 3.** Купите доменное имя. Это можно сделать как на базе хостинга (что проще всего), так и у сторонних регистраторов (что сложнее). Доменное имя нужно будет делегировать на сервера хостинга. Как правило, если вы покупаете имя прямо на хостинге, ничего дополнительно с ним делать не требуется.



- **Шаг 4.** Создайте базу данных. Это будет хранилище вашего сайта. На хостинге можно это сделать прямо в интерфейсе, указав имя базы, логин и пароль для пользователя. Каждый хостинг здесь также предоставляет свою собственную документацию. Эти данные надо будет указать в настройках Wordpress при установке. А вообще, они будут храниться в файле **wp-config.php**. При копировании из архива, этот файл будет иметь имя **wp-config-sample.php**. Полученные настройки для подключения будут вписаны в поля
  - DB\_NAME – имя базы.
  - DB\_USER – имя пользователя.
  - DB\_PASSWORD – пароль.
  - DB\_HOST – адрес базы.
- **Шаг 5.** Установите Wordpress. Если вы все сделали корректно, то при вызове вашего домена появится страница менеджера установки, который проверит наличие необходимых модулей, запросит данные для подключения к БД и произведет установку сайта.

 Но CMS – это самый простой сценарий применения. На PHP можно писать код и самостоятельно, создавая сложные приложения, такие как интернет-магазины, банковские приложения, доски объявлений и многое другое.

Помимо этого, сайт может потребовать не только ответа на браузерные запросы, но и фоновых задач.

Например, нужно автоматически рассылать поздравления пользователям и скидки в их дни рождения, отправлять сформированные и оплаченные заказы на склад, проверять статусы финансовых транзакций.

За всё это будет отвечать уже консольный вызов PHP, который организуется без привлечения пользователей, но работает в автоматическом режиме, например, по расписанию. Создание таких скриптов мы также обязательно рассмотрим на курсе.

# Инфраструктура для работы веб-приложения

Для того чтобы начать работу с PHP, нам необходимо подготовить соответствующую инфраструктуру.

Для удобства разработки и отладки PHP-скриптов рекомендуется использовать специальные среды разработки, такие как:

- PhpStorm;
- Visual Studio Code,

которые обеспечивают удобный интерфейс для написания, отладки и тестирования PHP-кода.

## PHP в командной строке

Самый простой вариант работы PHP – это **CLI (Command Line Interface)**.

В таком режиме мы не сможем обслуживать запросы от браузера, но уже можем писать некую логику. Скрипты такого типа понадобятся нам для решения рутинных задач, не требующих участия пользователя. Например, если захочется раз в сутки отправлять в чат Telegram-канала какое-то конструируемое сообщение.

Для старта в нашем редакторе кода создадим новый файл, который назовём `start.php` следующего содержания:

```
<?php  
  
echo "Привет, GeekBrains!";
```

Немного остановимся на синтаксисе и основах работы PHP.



PHP – интерпретируемый язык программирования.

Это значит, что в отличие от, например, C или Java нам не нужно предварительно трансформировать файл с кодом в исполняемый бинарный файл.

PHP построчно читает непосредственно код и работает с ним:

- «<?php» – это открывающий дескриптор, начало блока кода на языке PHP. Эта строка говорит интерпретатору PHP, что далее идет PHP-код. Поскольку PHP код может быть встроен в HTML, серверу нужно понимать, что из инструкций должно обрабатываться именно интерпретатором.
- «echo» – это ключевое слово на языке PHP, которое обозначает оператор, используемый для вывода текста на экран или веб-страницу. Оно может выводить любой текст, переданный ему в качестве аргумента.
- "Привет, GeekBrains!" – это аргумент, передаваемый в «echo». Это строка, которая будет выведена на экран или веб-страницу.
- «;» – это символ точки с запятой, который означает конец оператора на языке PHP. В языке PHP каждый оператор должен быть завершён символом «;». Он указывает интерпретатору PHP, что оператор закончился и нужно перейти к следующему.

Таким образом, этот простой код на языке PHP позволяет вывести текст на экран или веб-страницу и познакомиться с основами синтаксиса PHP. Теперь нам надо его запустить.

В директории нашего проекта создадим папку php-cli, где разместим код скрипта start.php. Запустим код в контейнере следующей командой

```
docker run --rm -v ${pwd}/php-cli/:/cli php:8.2-cli php /cli/start.php
```

Обратите внимание, что скрипт в данном примере запускается из корневой директории проекта, к которой мы обращаемся через **\${pwd}**.

Фигурные скобки здесь **{}** – это обращение к команде pwd в PowerShell, которая возвращает нам адрес текущей директории.



Если вы запускаете команду через Linux, то замените фигурные скобки на круглые.

В итоге после сборки и запуска контейнера мы увидим в консоли ответ:

Привет, GeekBrains!



После выполнения скрипта контейнер остановится и удалится.

## PHP в браузере

Мы научились запускать простой скрипт, но как обеспечить работу целого сайта?

Здесь cli недостаточно. Давайте вспомним, как работает сайт с применением протокола HTTP.

HTTP – запрос от браузера к серверу происходит следующим образом:

1. Пользователь вводит адрес сайта в адресной строке браузера, затем браузер создает HTTP-запрос.
2. HTTP – запрос представляет собой набор текстовой информации, содержащий данные о том, какую страницу нужно получить.

Этот файл состоит из трех основных частей

- a. Строка запроса содержит метод запроса (GET, POST, PUT и т.д.), URI (Uniform Resource Identifier) и версию протокола HTTP.
  - b. Заголовки содержат дополнительную информацию о запросе, такую как тип и версия браузера, тип контента и т.д.
  - c. Тело запроса содержит дополнительную информацию, такую как параметры формы при использовании метода POST.
3. Браузер отправляет HTTP-запрос на сервер, указанный в URI.
  4. Сервер получает HTTP-запрос и начинает его обработку. Сервер может отвечать на запросы как статическими, так и динамическими страницами.
  5. Если запрошенная страница является статической (простой HTML-код, например), сервер отправляет ее обратно в браузер в виде HTTP-ответа, который содержит статусный код, заголовки и тело ответа.
  6. Если запрошенная страница является динамической (а это как раз случай с PHP), сервер обрабатывает запрос, передаёт генерацию данных интерпретатору и ждет результат. Затем сервер отправляет эту страницу обратно в браузер в виде HTTP-ответа.
  7. Браузер получает HTTP-ответ и обрабатывает его. Если ответ содержит данные в виде HTML, CSS и JavaScript, браузер отображает страницу на экране.



Таким образом, помимо PHP нам нужен некий web-сервер, который будет принимать соединения на входе.

Наиболее популярным и актуальным на данный момент является web-сервер **Nginx** – он будет принимать HTTP-запросы, определять, динамические они или статические, а затем – передавать в FPM динамические запросы.

Дело в том, что Nginx сам по себе не способен понимать код на PHP или любом другом языке – у него иная задача.

В роли интерпретатора будет выступать уже не cli-контейнер, а специальный обработчик, который взаимодействует с Nginx посредством протокола FastCGI. Это будет PHP-FPM – менеджер процессов FastCGI для PHP. Он позволяет запускать и управлять несколькими процессами PHP-FPM в качестве фоновых процессов, обрабатывающих запросы веб-сервера.

FastCGI (Fast Common Gateway Interface) – это протокол взаимодействия между веб-сервером и веб-приложениями, который позволяет серверу обращаться к внешним процессам, выполняющим скрипты на сервере, и получать от них ответы. FastCGI используется для улучшения производительности веб-серверов, таких как Nginx, при обработке динамических контентных страниц, таких как PHP, Python, Ruby и других.

FastCGI обеспечивает структурированный обмен данными и командами между веб-сервером и внешним процессом в формате записей (records), которые могут содержать различные типы данных, такие как параметры запроса, переменные окружения, данные запроса и т. д. Протокол также предоставляет механизмы управления состоянием процесса FastCGI, такие, как управление пулами процессов, перезагрузка процессов и т. д.

PHP-FPM (PHP FastCGI Process Manager) – это компонент PHP, который предоставляет реализацию FastCGI для PHP-скриптов. Он является отдельным процессом, выполняющим PHP-скрипты и взаимодействующим с веб-сервером через протокол FastCGI. PHP-FPM предлагает ряд опций настройки и управления процессами PHP, таких как управление пулами процессов PHP, регулирование количества рабочих процессов PHP и управление ресурсами.

С помощью PHP-FPM можно управлять и настраивать работу процессов PHP, задавая такие параметры, как:

- количество процессов,

- время жизни процессов,
- максимальный размер памяти и т.д.

Это позволяет улучшить производительность и стабильность работы веб-сервера при больших нагрузках.

PHP-FPM также поддерживает использование пулов соединений, что уменьшает количество процессов, необходимых для обработки запросов и позволяет лучше управлять памятью и ресурсами сервера.

Основываясь на понимании контейнеров, нам потребуется уже два контейнера, которые будут работать постоянно. А значит, мы будем применять docker-compose.

💡 Внимательно смотрите на поясняющие комментарии в файлах конфигурации. Они помогут вам разобраться в составляющих частях настроек

```
# версия синтаксиса
version: '3'

# в этом блоке мы описываем контейнеры, которые будут запускаться
services:

  #контейнер с Nginx
  nginx:
    build:
      context: ./nginx
      dockerfile: Dockerfile
    image: myapp/nginx
    container_name: webserver

    # проброс портов
    ports:
      - "80:80"
```

```

volumes:
  - ./code:/data/mysite.local

networks:
  - app-network

#Контейнер с PHP-FPM, назовём его app
app:
  # Если нет секции build, то система будет искать образ в репозиториях

  build:
    context: ./fpm
    dockerfile: Dockerfile

  image: myapp/php # имя будущего образа
  container_name: app # имя контейнера после запуска


  volumes:
    - ./code:/data/mysite.local

  # мы можем создать для контейнеров внутреннюю сеть

  networks:
    - app-network

#Docker Networks
networks:
  app-network:
    driver: bridge

```

 Мы будем собирать сайт, отвечающий на имя mysite.local.

Разумеется, такого домена в глобальной сети не существует, поэтому нам надо добавить соответствующую запись в файл `hosts` – это локальный файл на компьютере, который используется для сопоставления IP-адресов и доменных имен. Расположение файла `hosts` может отличаться в разных операционных системах.

Вот расположение файла hosts в некоторых популярных операционных системах:

- В операционной системе Windows файл hosts обычно находится по пути C:\Windows\System32\drivers\etc\hosts.
- В операционной системе MacOS файл hosts находится в папке /private/etc/hosts.
- В операционной системе Linux файл hosts находится в папке /etc/hosts.

🔥 Заметьте, что в некоторых операционных системах, таких как MacOS и Linux, для доступа к файлу hosts требуются права суперпользователя. Если вы не имеете соответствующих прав, вам нужно будет запустить текстовый редактор с правами суперпользователя, например, с помощью команды `sudo`.

Добавим в конец этого файла запись:

```
127.0.0.1    mysite.local
```

Если внимательно посмотреть на docker-compose, то мы увидим, что нам потребуется создать три дополнительных папки:

- code – это папка для volume, который будет пробрасываться в контейнеры.
- fpm – это папка с Dockerfile для PHP-FPM.
- nginx – это папка для Dockerfile и конфигурации nginx.

Внимательно посмотрим на конфигурацию Nginx, лежащую в mysite.local.conf.

Он содержит инструкции, которые говорят, как веб-сервер должен обработать тот или иной URL, по которому пришёл запрос.

Мы указываем имя нашего домена:

```
server_name mysite.local;
```

Также обозначаем, где внутри контейнера лежат файлы HTML, CSS, JS и PHP:



```
root /data/mysite.local;
```

Инструкция, которая реагирует на все адреса, обращающиеся к статическим файлам, и отвечает за то, чтобы отдавать их максимально быстро:

```
location ~* \.(jpg|jpeg|gif|css|png|js|ico|html)$
```

Далее мы говорим, что все остальные адреса будут обращаться к нашему коду на PHP:

```
location / {  
    try_files $uri $uri/ /index.php?$query_string;  
}
```

Мы еще разберем этот подход чуть позже в курсе.

И наконец данный блок инструкций:

```
location ~* \.php$ {  
    try_files $uri = 404;  
    fastcgi_split_path_info ^(.+\.php) (/.+)$;  
    fastcgi_pass app:9000;  
    #fastcgi_pass unix:/var/run/php-fpm.sock;  
    fastcgi_index index.php;  
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
    include fastcgi_params;  
}
```

Отвечает как раз за то, чтобы при всех обращениях к файлам с расширением PHP в браузере Nginx через FastCGI передавал выполнение вызова в PHP-FPM и ждал от него ответа.


Теперь мы можем запустить наши контейнеры командой:

```
docker-compose up
```

Если всё сделано правильно, то в браузере по адресу <http://mysite.local> мы увидим содержимое нашего сайта, которое будет выводить полную информацию о том, какую версию PHP мы используем.

Привет, GeekBrains!  
2023-03-29 11:41:23

Что-то еще

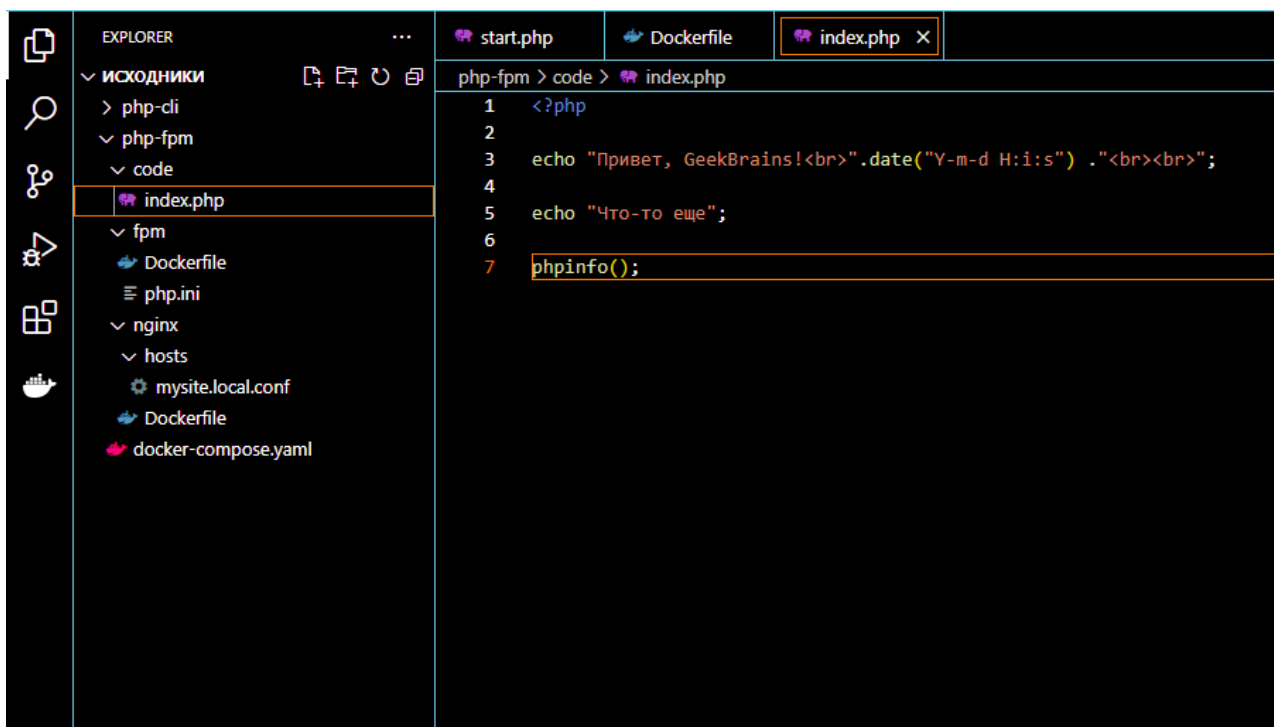
PHP Version 8.2.4	
	
System	Linux 899e7c149de7 5.15.90.1-microsoft-standard-WSL2 #1 SMP Fri Jan 27 02:56:13 UTC 2023 x86_64
Build Date	Mar 27 2023 22:41:28
Build System	Linux cc00826d1f83 5.10.0-13-cloud-amd64 #1 SMP Debian 5.10.106-1 (2022-03-17) x86_64 GNU/Linux
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-fpm.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini, /usr/local/etc/php/conf.d/php-custom.ini
PHP API	20220829
PHP Extension	20220829

Мы уже использовали несколько простых инструкций. Вызов функции `phpinfo()` будет вам полезен в работе, если вы хотите посмотреть на то, какие модули доступны в вашей сборке.

В консольной сборке ту же информацию можно получить, выполнив команду:

```
php -i
```

Поскольку у нас проброшен volume, код мы можем редактировать, не пересобирая контейнер – мы можем править логику прямо в директории code!



Теперь у нас готовы обе сборки, которые понадобятся нам для эффективного обучения. В дальнейшем мы будем их улучшать и модифицировать, но начинаем с достаточно простой конфигурации, чтобы научиться всё поддерживать самостоятельно.

## Начинаем программировать

Мы уже написали пару несложных скриптов, которые умеют выводить что-то на экран как в консоли, так и в браузере.

Давайте же теперь попробуем написать что-то более осознанное, чтобы разобраться с переменными и типами в PHP!

Попробуем встроиться в HTML код при помощи PHP, чтобы сделать наш сайт динамическим.

Возьмём понравившийся шаблон сайта (пример кода вы сможете найти в материалах к этому уроку) и разместим его HTML-код, а также CSS и JS файлы в пробрасываемой директории.

После старта контейнера по адресу <http://mysite.local/index.html> мы сможем увидеть стартовую страницу нашего сайта. Но она статична. Научимся добавлять к ней PHP-код.



Обратите внимание на то, что размещать сложную PHP-логику прямо в файлах с вёрсткой – дурной тон. Это мешает чтению кода. Но пока на старте мы примем такое допущение. Далее в курсе мы обязательно обсудим, как можно решать такую проблему.

Переименуем наш файл из `index.html` в `index.php`. Это не сделает сам сайт динамическим, но теперь веб-сервер Nginx поймёт, что выполнение нужно передать в PHP.

За это в файле конфигурации нашего сайта отвечает блок:

```
location ~* .php$ {  
    ...  
    fastcgi_pass app:9000;  
    ...  
}
```

Nginx разбирает переданный ему URI на базе регулярных выражений. И в каждый location в конфигурации мы размещаем правило обработки адреса по его шаблону.

Когда Nginx видит URI, он подбирает из конфигурации сверху вниз первый подходящий шаблон. Именно поэтому описание шаблонов URI должно идти от частного к общему.



В нашем случае мы говорим Nginx, что все URI, которые оканчиваются на **.php** будут направляться в контейнер с PHP-FPM на обработку.

Чуть выше по конфигурации вы можете увидеть следующий блок:

```
location / {  
    try_files $uri $uri/ /index.php?$query_string;  
}
```

Эта инструкция говорит о том, что все запросы к нашему сайту, за исключением запросов к статическим файлам (посмотрите на конструкцию в конфигурации выше), будут направляться строго на один и тот же файл – `index.php`. Это

архитектурный паттерн Front Controller. Мы ещё вернёмся к нему в рамках курса, но вы уже можете самостоятельно почитать о нём для расширения кругозора.

Теперь мы понимаем, что все наши запросы пойдут к нашему файлу `index.php`. Наберите в браузере просто <http://mysite.local> и убедитесь в том, что главная никак не изменилась.

Давайте теперь добавим в заголовок сайта информацию о названии нашего сайта и текущей дате.

В начале файла допишем следующий код:

```
<?php
$title = "<em>Geek</em>Brains";
$today = date("d.m.Y");
?>
```

Здесь мы объявили две переменные: **title** и **today** – это две строковые переменные, к которым впоследствии мы будем обращаться.

В PHP переменная объявляется знаком доллара (\$).

Затем следует имя переменной, которое может состоять из букв, цифр и символа подчеркивания, но не должно начинаться с цифры!

Присваивание значения переменной происходит оператором присваивания (=). Например:

```
$myVariable = "Hello World!";
```

Переменную **title** мы объявляем явно, задавая вручную строку, к которой будем обращаться.

Переменная **today** формируется при помощи встроенной в PHP функции [date](#). Эта функция выводит дату в заданном формате. В нашем случае это будет формат «день.месяц.год».

Вообще говоря, в версии 8.2 PHP имеет следующий набор типов:

- `null` – пустота. Это тип, который имеет переменная, если её создать, но не присвоить ей значение.

- `bool` – булевский тип, принимающий `true` или `false`.
- `int` – целочисленный тип, ограниченный разрядностью вашего процессора.
- `float` – число с плавающей точкой (дробь).
- `string` – строка, которую мы уже использовали выше.
- `array` – массив.
- `object` – объект.
- `resource` – ресурс, который применяется для хранения ссылок на подключение к базам данных, файлам и внешним сетевым ресурсам.
- `never` – это не тип переменной, но его можно будет указать как возвращаемый у функции, о чём мы поговорим позже.
- `void` – это не тип переменной, но его можно будет указать как возвращаемый у функции, о чём мы поговорим позже.

PHP имеет динамическую систему типов, что означает, что тип переменной определяется автоматически во время выполнения программы и может быть изменен во время интерпретации. Это отличается от статически типизированных языков программирования, таких как Java или C++, где тип переменной определяется во время компиляции. То есть, переменная, в которой, например, была сохранена строка, во время выполнения может разместить себе вместо неё целочисленное значение.

Чтобы типы значения трансформировались корректно, в PHP присутствует механизм приведения типов. Приведение типов в PHP – это процесс изменения типа переменной на другой тип. PHP автоматически приводит типы при выполнении операций, но иногда требуется явное приведение типов.

Существует два типа приведения типов в PHP:

- явное;
- неявное.

**Неявное приведение** типов происходит автоматически, когда PHP пытается выполнить операцию с переменными разных типов.

Например, если переменная типа **`string`**, то есть строка, на самом деле содержит число, PHP автоматически преобразует его в число для выполнения математической операции.

```
$number = 5;

$string = "10";

$sum = $number + $string; // результат - 15
```

**Явное приведение** типов происходит при использовании специальных функций для приведения переменных к нужному типу.

Например, функция **intval()** приводит переменную к целочисленному типу.

```
$float = 2.5;

$int = intval($float); // $int будет равен 2
```

```
$string = "5";

$int = (int)$string; // $int будет равен 5
```

Нужно быть готовым к автоматическому приведению типов, так как, например, при приведении **float** к **int** теряется точность числа.

Поэтому продумывайте то, какими типами будет оперировать ваш код.

В процессе отладки актуальный тип переменной можно увидеть через вызов функции **var\_dump**.

```
$float = 2.5;

var_dump($float);
```



Но обратите внимание, что это исключительно отладочная функция. И в рабочей системе ей не место.

Но вернёмся к встраиванию нашего кода в HTML, ведь мы пока только задали переменные, но никак с ними не работали.

В примере сайта, приведенном в исходниках, заменим строку

```
<a href="#"><em>Grad</em> School</a>
```

на

```
<a href="#"><?=$title; ?></a>
```



Конструкция «<?=>» отвечает за то, чтобы вывести на экран переменную.

Это же можно сделать через:

```
<a href="#"><?php echo $title; ?></a>
```

Это чуть более объёмная запись, но делает она то же самое.

Конвенции кода PSR допускают оба формата обращения к переменной. Мы будем регулярно к ним обращаться в течение курса, поэтому лучше привыкать к ним сразу же.

PSR (PHP Standards Recommendation) – это набор рекомендаций, разработанных PHP-сообществом, для стандартизации разработки на языке программирования PHP. PSR представляет собой набор соглашений и рекомендаций, которые помогают разработчикам создавать читаемый, поддерживаемый и совместимый код на PHP.

PSR были созданы группой разработчиков PHP-сообщества, известной как PHP-FIG (PHP Framework Interop Group).

PHP-FIG – это группа разработчиков из разных проектов и фреймворков на PHP, которые сотрудничают для создания стандартов и рекомендаций, чтобы облегчить совместную работу между различными проектами и повысить качество PHP-кода в целом.

PSR охватывают различные аспекты разработки на PHP, такие как:


- автозагрузка классов,
- структура каталогов,
- стандарты кодирования,



- интерфейсы HTTP-запросов и ответов,
- контейнеры зависимостей и другие.

Некоторые из наиболее известных и широко используемых PSR включают:

- PSR-4 (автозагрузка классов),
- PSR-1 (стандарты кодирования),
- PSR-12 (расширенные стандарты кодирования).

 Соблюдение PSR рекомендаций способствует созданию стандартизированного и совместимого кода на PHP, что упрощает совместную работу между различными проектами, повышает читаемость и поддерживаемость кода и способствует развитию экосистемы PHP-приложений и библиотек.


С актуальными стандартами можно ознакомиться по [ссылке](#).

Теперь выведем значение нашей сегодняшней даты. В прилагаемом примере мы сделаем это в h6 заголовке на 74 строке.

```
<h6>Сегодня <?=$today?></h6>
```

Обратите внимание, что без создания переменной ту же задачу можно решить и прямым вызовом функции:

```
<h6>Сегодня <?=date("d.m.Y") ?></h6>
```

 Каждая переменная – это контейнер в памяти сервера. Чем больше мы создаём переменных через копирование, тем больше памяти мы потребляем. Будьте внимательны к количеству создаваемых переменных.

Мы создавали переменные через копирование, но можно и ссылаться одной переменной на другую. В таком случае память на хранение копии значения расходоваться не будет, а обращаться к нему мы сможем через два алиаса.

Например,

```
<?php
$title = "<em>Geek</em>Brains";

$school = &$title; // создание ссылки

$titleCopy = $title; // создание копии

?>
```

В случае ссылки при изменении оригинального значения будет изменяться вывод при обращении к обеим переменным, чего не будет в случае копирования. Это можно увидеть в следующем примере:

```
<?php
$title = "<em>Geek</em>Brains";

$school &= $title; // создание ссылки

$titleCopy = $title; // создание копии

$title = "Новый заголовок";

echo $school; // видим, что здесь значение изменилось

echo $titleCopy; // а здесь - нет

?>
```

## Заключение

Мы познакомились с языком PHP, научившись собирать для себя окружение для дальнейшей работы. Также мы узнали о типизации в PHP и попробовали выполнить несложные действия.

Это простой, но очень важный шаг с точки зрения дальнейшего развития в рамках данного курса.

На базе полученных знаний мы будем погружаться в более сложные и мощные концепции языка.

# Домашнее задание

1. Собрать для себя окружение из Nginx + PHP-FPM и PHP CLI
2. Выполните код в контейнере PHP CLI и объясните, что выведет данный код и почему:

```
<?php

$a = 5;

$b = '05';

var_dump($a == $b);

var_dump((int)'012345');

var_dump((float)123.0 === (int)123.0);

var_dump(0 == 'hello, world');

?>
```

3. В контейнере с PHP CLI поменяйте версию PHP с 8.2 на 7.4. Изменится ли вывод?
4. Используя только две числовые переменные, поменяйте их значение местами. Например, если  $a = 1$ ,  $b = 2$ , надо, чтобы получилось:  $b = 1$ ,  $a = 2$ . Дополнительные переменные, функции и конструкции типа `list()` использовать нельзя.

## Что можно почитать еще?

1. <https://phptherightway.com/>
2. <https://php.net>
3. Книга. Котеров Д. В. PHP 8