Содержание

[Введение 3](#_Toc161948968)

[Глава 1 Теоретическая часть 4](#_Toc161948969)

[1.1 Что такое веб-программирование 4](#_Toc161948970)

[1.1.1 Какие есть виды веб-программирования 4](#_Toc161948971)

[1.1.1.1 Frontend 4](#_Toc161948972)

[1.1.1.2 Backend 4](#_Toc161948973)

[1.1.1.3 Верстка 5](#_Toc161948974)

[1.1.2 Какие навыки нужны в веб-программировании 5](#_Toc161948975)

[1.1.3 Какие языки нужно знать программисту 6](#_Toc161948976)

[1.1.3.1 JavaScript 7](#_Toc161948977)

[1.1.3.2 PHP 7](#_Toc161948978)

[1.1.3.3 Python 8](#_Toc161948979)

[1.1.3.4 HTML и CSS 8](#_Toc161948980)

[1.2 Язык HTML 9](#_Toc161948981)

[1.2.1 Сферы применения языка html 9](#_Toc161948982)

[1.2.2 Когда не стоит применять html 9](#_Toc161948983)

[1.2.3 История языка HTML 10](#_Toc161948984)

[1.2.4 Как работает HTML 11](#_Toc161948985)

[1.2.5 Базовая структура страницы и ее элементов 12](#_Toc161948986)

[1.2.6 Зачем нужен HTML 13](#_Toc161948987)

[1.2.7 Возможности HTML 14](#_Toc161948988)

[1.2.7.1 Что такое теги HTML 15](#_Toc161948989)

[1.2.7.2 Заголовки HTML 16](#_Toc161948990)

[1.2.7.3 Абзац в HTML 17](#_Toc161948991)

[1.2.7.4 Списки в HTML 17](#_Toc161948992)

[1.2.8 Преимущества и недостатки HTML 17](#_Toc161948993)

[1.2.8.1 Плюсы HTML 17](#_Toc161948994)

[1.2.8.2 Минусы HTML 19](#_Toc161948995)

[1.2.9 Почему HTML не является языком программирования 19](#_Toc161948996)

[1.2.10 Зачем нужен HTML 20](#_Toc161948997)

[1.3 PHP и его история 20](#_Toc161948998)

[1.3.1 Основное предназначение PHP 22](#_Toc161948999)

[1.3.2 Свойства PHP 22](#_Toc161949000)

[1.3.3 Что такое интерпретируемость в PHP 24](#_Toc161949001)

[1.3.4 Преимущества интерпретируемости PHP: 25](#_Toc161949002)

[1.3.5 Слабая динамическая типизация в PHP 26](#_Toc161949003)

[1.3.6 Как работает PHP 27](#_Toc161949004)

[1.3.7 Особенности синтаксиса 28](#_Toc161949005)

[1.3.8 Где используется 29](#_Toc161949006)

[1.3.9 Перспективы PHP 30](#_Toc161949007)

[1.3.10 Почему многие ненавидят PHP 30](#_Toc161949008)

[1.3.11 Плюсы и минусы 31](#_Toc161949009)

[1.3.11.1 Плюсы в php 31](#_Toc161949010)

[1.3.11.2 Минусы в PHP 32](#_Toc161949011)

[Глава 2 Практическая часть 33](#_Toc161949012)

[Заключение 34](#_Toc161949013)

[Библиографический список 35](#_Toc161949014)

[Приложение 36](#_Toc161949015)

# Введение

Пока-что пусто

# Глава 1 Теоретическая часть

# 1.1 Что такое веб-программирование

Веб-программирование – Это разработка сайтов и сервисов, которые функционируют в интернете. Например, магазинов, информационных ресурсов, лендингов, онлайн-кинотеатров, форумов. Все в сети – от «Википедии» до «ВКонтакте», – создавали в том числе веб-программисты.

Это одно из самых популярных направлений в IT. Специалисты быстро развивают карьеру, потому что востребованы в большинстве сфер. Это связано с тем, что почти у каждой компании есть сайт. В статье рассмотрим особенности направления и навыки, которые нужны, чтобы его освоить.

Важно понимать разницу между терминами «веб-программирование» и «веб-разработка». Разработка – широкое понятие, которое включает и программирование тоже. Обычно подразумевает создание ПО с нуля: от составления технического задания до поддержки продукта после релиза. Программирование – важный этап разработки, написание кода по готовому ТЗ для решения конкретной задачи.

## 1.1.1 Какие есть виды веб-программирования

### 1.1.1.1 Frontend

Frontend – разработка видимой части ПО, с которой взаимодействует пользователь. Это дизайн страницы, тип и расположение кнопок, меню, разделов, визуальных компонентов. Frontend-разработчик делает интерфейс удобным и понятным для посетителя, адаптирует продукт под разные устройства: ноутбуки, смартфоны, планшеты.

### 1.1.1.2 Backend

Backend – программирование серверной части сайта или сервиса. Пользователи ее не видят, но именно она отвечает за корректную работу ПО. Backend-разработчики создают нужный функционал – например, форму заказа или авторизации, – и обеспечивают безопасность данных.

### 1.1.1.3 Верстка

Верстка – это часть frontend. Ей занимается отдельный специалист – HTML-верстальщик. Frontend-разработчик отвечает за то, как будет выглядеть интерфейс, его логику. А верстальщик с помощью языков HTML и CSS описывает все, что должно быть на странице, и воплощает идеи дизайнера. Например, работает со шрифтами, оттенками, отступами. От верстки зависит скорость загрузки сайта, его ранжирование в поисковых системах и то, как он будет отображаться на разных устройствах.

## 1.1.2 Какие навыки нужны в веб-программировании

Чтобы начать писать код для сайтов и сервисов, необходимо:

1. Знать один или несколько языков программирования. Выбор зависит от направления – frontend или backend. Для frontend-разработки нужно знать JavaScript, для backend выбор больше – PHP, Go или Java.
2. Уметь верстать страницы с помощью HTML и CSS, адаптировать ПО под разные устройства и браузеры.
3. Работать с библиотеками и фреймворками – они облегчают написание кода.
4. Понимать основы безопасности данных (БД).
5. Знать, как устроены протоколы HTTP – они отвечают за обмен информацией в интернете, отображение страниц, позволяют работать с базами данных.

Этих навыков достаточно, чтобы войти в профессию. Но чтобы развивать карьеру, постепенно нужно расширять знания и изучать новые инструменты для выбранного направления.

Для успешной работы специалисту также важно обладать определенными личностными качествами. Среди них:

1. Готовность выполнять рутинные, монотонные задачи.
2. Умение планировать работу, соблюдать дедлайны.
3. Внимательность к деталям.
4. Навык быстрого переключения между заданиями.
5. Самостоятельность, умение организовать работу.
6. Развитое логическое и критическое мышление.
7. Способность взаимодействовать с другими членами команды.
8. Желание совершенствоваться в профессии.

## 1.1.3 Какие языки нужно знать программисту

Чтобы писать код для сайтов и веб-приложений, необходимо освоить несколько языков: чем их больше, тем разнообразнее задачи сможет брать программист. Рассмотрим популярные варианты.

### 1.1.3.1 JavaScript

Один из основных в веб-программировании. Чаще всего его используют для frontend-разработки. JavaScript поддерживает большинство браузеров, он подходит для создания мобильных и десктопных ПО.

Особенности языка:

* относительно простой синтаксис;
* много обучающей литературы, инструкций;
* легкая интеграция с другими языками;
* универсальность – подходит и для frontend, и для backend.

Также JavaScript позволяет делать страницы интерактивными: подсвечивать кнопки, добавлять динамичные изображения и движущийся текст.

На базе JavaScript создана React – библиотека для работы с сайтами. Инструмент чаще всего применяют для проектирования пользовательских интерфейсов. React помогает объединять визуальную часть страницы с ее внутренним содержанием.

### 1.1.3.2 PHP

Создан специально для веб-программирования. Код на PHP легко встроить в HTML. Но чтобы использовать все возможности PHP, нужно знать основы работы с базами данных и освоить язык запросов SQL, систему управления базами данных MySQL.

Особенности PHP:

* простой синтаксис;
* много библиотек и фреймворков;
* возможность писать код в любом редакторе;
* динамическая типизация – одной переменной можно присваивать несколько значений.

При этом функционал PHP меньше, чем у JavaScript. Поэтому разработчики редко пишут на нем код с нуля, но используют для конкретных задач.

1.1.3.3 Python

Язык со множеством функций и логичным, структурированным кодом. У него простой и понятный синтаксис, похожий на английский, поэтому Python часто учат новички.

Другие преимущества:

* поддерживается всеми операционными системами;
* не требует длинных команд для написания кода;
* подходит для решения любых задач;
* у него много библиотек и дополнительных модулей.

Сайты крупных компаний по всему миру написаны на этом языке, в том числе Google. Также с помощью Python создают алгоритмы сбора данных из разных источников.

## 1.1.3.4 HTML и CSS

Для создания верстки и грамотного соединения всех элементов сайта, включая текст, видео, изображения, кнопки, графики, используют языки программирования HTML и CSS.

HTML — язык разметки, считываемый браузерами. Благодаря ему пользователи видят определенную структуру наполнения страницы. HTML состоит из тегов: <table> — таблица, <img> — изображение, <strong> — жирный шрифт и так далее.

CSS — язык стилей, который делает приложения и сайты визуально красивыми. Он позволяет работать с видами и размером шрифтов, подбирать подходящую цветовую гамму, регулировать ширину межстрочного интервала и отступов, располагать мультимедийные элементы в конкретных местах.

# 1.2 Язык HTML

HTML - это аббревиатура, происходящая от английских слов HyperText Markup Language, или “язык гипертекстовой разметки”. Формально html - не язык программирования. Он не использует функции, операторы и прочие атрибуты для создания полноценных программ. Гипертекст работает через теги. С их помощью он сообщает интернет-браузеру, какой элемент нужно показать на экране и где его можно найти. Но разберемся во всем по порядку.

1.2.1 Сферы применения языка html

Гипертекстовая разметка предназначена для разработки сайтов. Через нее на веб-страницы добавляются тексты, таблицы, изображения, видеоролики и другие объекты. Кроме того, на html можно создать простейшие формы для сбора данных - например, поля для ввода текста или таблицы с выбором одной из предложенных опций.

Рассматриваемый язык (будем все же говорить так, несмотря на формальности) - профессиональная область знаний для вебмастеров, seo-специалистов, интернет-маркетологов. При этом для всех владельцев сайтов - в том числе директоров и менеджеров коммерческих компаний - будет не лишним понимание основ html.

## 1.2.2 Когда не стоит применять html

Для полноценного форматирования страниц сайтов - смены шрифтов, обозначения отступов и редактирования прочих параметров обычно используют не html код, a CSS. Это отдельные файлы, в которых на специальном языке прописываются правила - как и во что браузер должен “покрасить” каждый элемент заданной через html структуры.

CSS значительно экономит время, затрачиваемое на верстку страниц. Вебмастеру не приходится каждый раз по новой указывать параметры форматирования для однотипных элементов. Вместо этого стили прописываются один раз в отдельном файле.

Активные формы для сбора обрабатываемых данных, как и пользовательские скрипты, обычно пишутся на JavaScript. Этот язык программирования обладает значительно большим количеством функций, чем html, и с его помощью проще обеспечить безопасность передаваемой информации.

## 1.2.3 История языка HTML

Гипертекст появился на рубеже 80-х и 90-х годов как стандартизированное средство для быстрой верстки web страниц, и изначально включал в себя лишь 18 базовых элементов. В первые годы существования язык разметки активно развивался. Уже к 1999 году появилась версия html 4.01, содержащая множество новых возможностей в сравнении с первыми спецификациями.

Актуальная версия html - 5.3, действует с конца 2018 года.Она поддерживается большинством браузеров. Сайты, использующие устаревший синтаксис, могут отображаться некорректно в современных программах.

Отдельно стоит упомянуть XHTML - гипертекстовый язык, основанный на синтаксисе XML. Он схож со стандартным html, но предъявляет более строгие правила к написанию кода. Например, все спецсимволы в нем (&, <> / и другие) прописываются в виде сущностей - например, “&”.

## 1.2.4 Как работает HTML

Когда пользователь вводит адрес сайта в соответствующую строку, браузер посылает запрос к серверу. В качестве ответа сервер отправляет html-документ, который расшифровывается браузером. Передаваемый файл содержит как отформатированный через теги текст, так и гиперссылки на дополнительные документы - изображения, видео, стили и скрипты. Браузер последовательно выгружает всю полученную информацию. Пользователь видит страницу уже в готовом виде. Открыть исходный код можно кликнув правой кнопкой мыши в области просмотра сайта и выбрав в меню пункт “Просмотр кода страницы”.

Гипертекст выступает “каркасом” сайта, его общей схематической картой.

Основная единица html - тег. С помощью тегов задаются конкретные элементы страницы. Так <p> обозначает текстовый абзац, а <img> - изображение.

Тэги бывают одиночными и парными. После открывающего тега <p> всегда должен идти закрывающий - </p>. То же самое относится к контейнерам (div), заголовкам (h1-h6), таблицам (table) и другим объектам. Но, например, для разрыва строки - <br>, закрывающая метка не требуется.

Часто теги сопровождаются атрибутами, через которые элементам на странице присваиваются дополнительные параметры - цвет, размер, расположение на экране. Так, чтобы выровнять текст по правому краю, к метке p нужно добавить атрибут align со значением right.

<p align = “right”>Текст абзаца</p>

Похожим образом на страницу добавляются ссылки.

<a href = “https://example.com”>Текст ссылки</a>

Тег img также сопровождается атрибутами.

<img src="picture.jpg" width="250" height="500" alt="альтернативное описание" >

В примере задаются ширина и высота изображения, а также текст, который появится на экране, если браузер не сможет загрузить картинку.

Кавычки и знак равно - стандартный синтаксис для указания значений атрибутов.

Опечатки в html коде приводят ошибкам в отображении страницы. Если тег будет прописан неправильно, браузер просто не сможет определить, какой фрагмент нужно отобразить на экране. Но регистр написания тегов значения не имеет. Записи <img> и <IMG> воспринимаются браузером одинаково.

## 1.2.5 Базовая структура страницы и ее элементов

HTML документ состоит из заголовка и основного тела. Заголовок обозначается парным тегом <head>. Внутри него размещаются все служебные метки и элементы. Вот основные из них:

title - название страницы;

description - краткое описание контента;

кодировка страницы;

ссылки на файлы со стилями;

java-скрипты, например, счетчики аналитики.

Непосредственно демонстрируемый пользователю контент размещается внутри тега <body>. Все элементы - абзацы, таблицы, изображения и пр. последовательно прописываются в теле страницы.

HTML документ может также содержать неотображаемые комментарии. С их помощью веб разработчик может пояснить назначение конкретных элементов в коде или указать другую важную информацию. Синтаксис комментария выглядит так:

<!-- Поясняющий текст -->

В начале документа всегда указывается его тип - <!DOCTYPE html> и открывающий тег <html>.

Базовая страница с заголовком и текстом будет выглядеть так:

<!DOCTYPE html>

<html>

<head>

<title>Информативный заголовок</title>

<meta name="description” content="Описание страницы”>

</head>

<body>

<h1>Главный заголовок</h1>

<p>Hello world!</p>

</body>

</html>

Этот код можно скопировать в блокнот и, сохранив документ в формате.

Когда вы открываете любой сайт в интернете, браузер подгружает с сервера HTML-файл — текстовый файл с информацией о структуре сайта. Благодаря этому файлу браузер понимает, какой у страницы дизайн, в каком порядке показывать заголовки и текст, откуда загружать изображения, видео и скрипты.

## 1.2.6 Зачем нужен HTML

Основная цель HTML — структурировать и оформлять контент на сайте.

HTML создает иерархическую структуру веб-страницы, используя заголовки, абзацы, списки и таблицы. Такая структура помогает пользователю легче ориентироваться на сайте.

Также с помощью HTML отображается текст, изображения, таблицы, видео и аудио. А еще благодаря коду на HTML мы можем переходить по ссылкам в интернете с одного сайта на другой.

## 1.2.7 Возможности HTML

С помощью HTML можно:

* **Делать текстовую разметку** — форматировать текст, выделять фрагменты, создавать списки, добавление сноски.
* **Встраивать медиа**. HTML позволяет размещать на сайте изображения, аудио, видео, карты.
* **Создавать ссылки и навигацию**. Гиперссылки и списки меню помогают быстрее найти информацию и сориентироваться на странице.
* **Создавать таблицы**. Нередко информацию удобно представить в табличном виде. HTML умеет работать с таблицами.
* **Создавать формы**. Формы нужны для регистрации посетителей сайта по телефону и электронной почте, оформления заказов, опросов и сбора обратной связи — отзывов, комментариев, предложений.

В HTML можно даже создавать простой дизайн: например, устанавливать цвет и шрифт текста или фоновый цвет блока. Но более сложный дизайн страницы разработчики делают с помощью CSS — языка стилей, который создали специально для работы в связке с HTML.

Возможностей HTML не хватает, чтобы «оживлять» сайты, делать их функциональными и интерактивными. Здесь приходит на помощь язык программирования JavaScript, благодаря которому мы можем взаимодействовать с формами на сайте, видеть динамические элементы и анимации.

Так выглядит HTML-код главной страницы блога Хекслета

HTML-код пишут и сохраняют в документе — текстовом файле с расширением .html или .htm. HTML-файл можно написать в любом текстовом редакторе, даже в стандартном Блокноте Windows. Чтобы просматривать HTML-файлы, интернет не нужен.

Если создать несколько HTML-файлов в одной папке и поставить внутри гиперссылки, то можно переходить по этим ссылкам из одного документа в другой.

### 1.2.7.1 Что такое теги HTML

Теги HTML — это команды, которые говорят браузеру, что и в каком порядке показывать на экране. У каждого тега есть имя, которое расположено в угловых скобках.

Самая простая HTML-страница состоит из трех тегов: <html>, <head> и <body>.

Теги <head> и <body> используют на странице только один раз.

HTML-файл всегда начинается с тега <!DOCTYPE html>, по которому браузер понимает, как правильно отобразить страницу.

В теге <head> <head/> хранится служебная информация — заголовок и кодировка страницы.

Внутри тега <body> <body/> помещается содержимое страницы, которое отображается в браузере — текст, изображения, видео.

В теге <title> <title/> — заголовок веб-страницы. Его браузер отобразит как название и сохранит в описании, если сохранить страницу в закладки.

Тег <img> помещает изображение в нужное место страницы.

Есть парные и непарные HTML-теги, первых — большинство. Парный тег состоит из открывающего и закрывающего тега. Закрывающий тег содержит дополнительный элемент / — эта косая черта называется «слэш».

Например, тег <p>, внутри которого размещается текст, парный:

<p>Слава роботам!<p/>

Тег <img>, который позволяет разместить картинку на сайте, — непарный:

<img src='robot-bender.jpg' alt='picture-of-Bender-the-robot'>

У каждого тега есть атрибуты. Они расширяют возможности тегов: задают стили, добавляют ссылки, управляют мультимедиа.

В примере выше показаны два основных атрибута непарного тега <img> — это src и alt. Атрибут src добавляет ссылку на файл изображения, атрибут alt — подпись к картинке, которую покажет браузер, если файл изображения не загрузится.

Важно помнить про закрытые теги. Если тег не закрыт или закрыт неправильно, то верстка страницы может сломаться и сайт будет отображаться неверно.

Для логичной последовательной подачи информации используют заголовки, абзацы и списки.

### 1.2.7.2 Заголовки HTML

Заголовки в HTML — это элементы, которые используются для создания заголовков и подзаголовков на веб-странице. Они определяют важность и иерархию информации на странице.

В HTML применяют шесть тегов заголовков: <h1>, <h2>, <h3>, <h4>, <h5> и <h6>. Тег <h1> обычно используется для главного заголовка страницы, а теги <h2>-<h6> используются для подзаголовков и других заголовков на странице.

Заголовки конкретной страницы помогают поисковым системам вроде Google или «Яндекса» определить ее содержание. Для поисковиков каждый следующий тег заголовка менее важен, чем предыдущий. То есть у заголовка <h3> меньшая важность, чем у заголовка <h2>.

### 1.2.7.3 Абзац в HTML

Абзац — это элемент, который используется для создания текстовых блоков на странице. Каждый абзац текста создается парным тегом <p> <p/>.

Текст, разбитый на абзацы читать и воспринимать легче, чем сплошное «полотно». Абзац делит содержимое на логически связанные блоки.

### 1.2.7.4 Списки в HTML

Списки упорядочивают информацию на странице и улучшают ее восприятие. Обычно используются нумерованные и ненумерованные списки.

Ненумерованные или маркированные списки («буллеты») добавляют на страницу тегом <ul></ul>. Такие списки применяют, когда последовательность элементов не важна.

Для создания нумерованного списка используется тег <ol><ol/>.

Отдельный элемент в списке любого типа вводится тегом <li></li>. Этот тег нужно закрывать после каждого пункта.

## 1.2.8 Преимущества и недостатки HTML

### 1.2.8.1 Плюсы HTML

* HTML — это простой язык разметки, который легко изучить и использовать. Код на нем можно писать в любом текстовом редакторе.
* Теги в современных версиях HTML позволяют делать разные версии одного сайта для удобного просмотра на любом устройстве: от смартфона и планшета до большого монитора.
* На HTML можно создавать простые сайты без использования интерактивных изменяющихся элементов. Например, лендинг, сайт-визитку компании, портфолио, каталоги, справочники, инструкции, небольшие блоги.
* Подходит почти для всех браузеров, а код можно написать в любом текстовом редакторе.
* HTML можно использовать бесплатно.
* Чтобы освоить азы языка разметки гипертекста, достаточно пары часов. У него простой и понятный синтаксис.
* Подходит для профессиональных разработчиков сайтов и для любителей.
* Статические сайты на HTML, которым не нужно обращаться к базам данных, быстро загружаются, потому что меньше весят и в них отсутствует «мусорный код». Они дешевле в разработке, и для таких сайтов подойдёт самый дешёвый хостинг.
* HTML интегрируется с другими языками программирования.
* HTML-элементы содержат в себе семантику и работают с учётом цифровой доступности — ими уже могут пользоваться незрячие люди.
* HTML способен сам закрывать теги и пропускать синтаксические ошибки. В отличие от ошибки в JS, страница не упадёт, а всё равно загрузится, пусть и с непредсказуемыми искажениями.

### 1.2.8.2 Минусы HTML

* HTML — это язык разметки, а не язык программирования. Делать современные сайты только на HTML не получится. HTML, например, не может создавать динамические веб-страницы, которые изменяются в реальном времени без перезагрузки страницы. Для создания чего-то более сложного не обойтись без CSS и JavaScript.
* Некоторые браузеры не сразу встраивают поддержку новых функций и тегов, старые версии (Internet Explorer версии 8 и более ранней) могут неадекватно интерпретировать новые теги.
* Все повторяющиеся изменения на сайте придётся вносить вручную. Например, заменить на каждой странице пункты меню и навигации. То же самое с созданием новых страниц — даже если структура повторяется, нужно создавать каждую страницу отдельно.
* Чтобы сделать хороший сайт, который будет быстро открываться на любом устройстве, в режиме чтения или со скринридера, недостаточно везде использовать < div>. Существует более 100 HTML-элементов, нужно знать их семантику, где именно их применять. Например, не всегда то, что на макете выглядит таблицей, ей является. Для этого нужно читать спецификацию по HTML и уметь искать в ней нужную информацию.
* Если нужно, чтобы сайт выглядел красиво и реагировал на действия пользователя, придётся учить ещё CSS и JS.

## 1.2.9 Почему HTML не является языком программирования

Нет. Языки программирования необходимы для создания сложных веб-сайтов, которые выполняют задачи и взаимодействуют с пользователями.

Чтобы обрабатывать данные, введенные в формы, обмениваться этой информацией с базами данных, хранящимися на сервере, создавать анимацию используют языки программирования JavaScript, PHP и Python.

HTML не обрабатывает данные, а только их отображает. И также HTML не может выполнять вычисления, поэтому назвать его языком программирования будет неправильно.

Освоить HTML можно на курсах Хекслета — пройдите их, чтобы сделать свой первый шаг в профессии фронтенд-разработчика.

## 1.2.10 Зачем нужен HTML

С помощью языка разметки HTML браузер делает запрос по адресу, который ввёл пользователь, и получает файл в формате «.html». Браузер распознаёт код, выбирает знакомые для себя сигналы: понимает, что написать словами, где поставить заголовок и какой именно. Таким образом код из файла преобразуется в необходимые визуальные объекты.

Если вы сейчас за ноутбуком или компьютером, посмотрите, как выглядит HTML этой конкретной страницы блога.

Для этого в Safari в настройках необходимо поставить галочку напротив пункта «Показывать меню „Разработка“ в строке меню» в «Дополнениях». После нажать в браузере «Разработка» и «Показать ресурсы страницы».

Для Google Chrome щёлкните правой кнопкой мыши в любом месте и выберите “View Page Source”. В зависимости от браузера команды могут различаться.

# 1.3 PHP и его история

Аббревиатура PHP расшифровывается как Preprocessed Hypertext Page. По сути это акроним словосочетания, указывающий, что изначально язык программирования обозначали как Personal Home Page. Это обозначение подчеркивало основные функции языка — обработка гипертекста, создание веб-сайтов и веб-приложений.

Разработал этот язык веб-программирования Расмус Лердорф в начале 90-х годов XX века. Задачей программиста было создать удобный инструмент для отслеживания и сбора статистических данных онлайн-резюме. В первоначальном виде ПХП был набором скриптов, которые Расмус назвал Personal Home Page Tools — то есть средства для личной домашней страницы.

Решение Лердорфа было настолько удачным, что его разработка вышла за рамки личного использования. Вот краткая выжимка из истории ПХП:

В 1995 году была выпущена первая версия PHP (PHP/FI). Она представляла собой набор CGI-скриптов (Common Gateway Interface) для обработки данных форм на веб-страницах.

В 1997 году появилась версия PHP 3. Она имела интерпретатор, что сделало PHP полноценным языком программирования.

В 2000 году вышел PHP 4, в котором было множество улучшений. Язык программирования начал набирать широкую популярность.

В 2004 году появилась версия PHP 5 (впоследствии переименованная в PHP 5.x). Она имела более современные возможности (на тот момент), включая поддержку объектно-ориентированного программирования (ООП).

В 2015 году вышел PHP 7. Он получил множество улучшений производительности и новых функций.

PHP 7.4 — последняя стабильная версия языка программирования, вышедшая в 2019 году.

PHP 8 — последняя вышедшая версия языка веб-программирования, которая не поддерживается Windows (так решили ребята из Microsoft).

PHP написан на C и представляет собой язык с открытым исходным кодом, который разработчики могут свободно использовать и модифицировать. Именно этот факт привел к широкому распространению ПХП и росту его популярности в среде веб-разработчиков.

## 1.3.1 Основное предназначение PHP

Основное предназначение PHP состоит в создании динамических веб-страниц и веб-приложений. Этот язык позволяет:

встраивать свой код в HTML;

взаимодействовать с базами данных;

обрабатывать формы;

создавать сессии.

PHP широко используется для создания веб-сайтов, систем управления содержимым сайтов (CMS), разработки решений в электронной коммерции и для решения других задач в сфере.

## 1.3.2 Свойства PHP

ПХП относится к одному из наиболее «мощных» языков программирования. Его используют многие разработчики веб-приложений для обработки данных на сервере. Конечно, у него масса уникальных свойств, делающих его классным вариантом как для программиста-новичка, так и для эксперта.

Скриншот: Best PHP Frameworks for 2019 / apptunix.com

Рассмотрим 10 особенностей ПХП, которые объясняют широкое применение данного языка программирования:

1. Интерпретируемость. Код выполняется непосредственно на сервере без предварительной компиляции. Это облегчает разработку и обновление веб-приложений.
2. Кроссплатформенность. ПХП поддерживает различные операционные системы — Windows, Linux и macOS. Это делает его универсальным для веб-программирования на разных серверах.
3. Встраиваемость. PHP можно встраивать непосредственно в HTML-код, что упрощает разработку динамических веб-страниц. Вы можете встраивать PHP-код внутрь HTML-тегов для вывода динамического контента.
4. Большая библиотека функций. Язык программирования имеет обширную библиотеку функций, которые можно использовать для упрощения работы с базами данных, файлами, отправки электронной почты и т. д. Это значительно упрощает веб-программирование.
5. Поддержка баз данных. Язык поддерживает взаимодействие с большинством систем управления базами данных (СУБД), включая MySQL, PostgreSQL и SQLite. Это позволяет создавать мощные веб-приложения с использованием БД.
6. Обработка форм. Язык веб-программирования часто используется для обработки данных, отправленных пользователем через веб-формы. Он позволяет получать данные из любых форм, валидировать их, сохранять в базе данных.
7. Открытый исходный код. PHP — проект с открытым исходным кодом. Эта особенность способствует его активной поддержке и развитию с помощью миллионов разработчиков по всему миру.
8. Широкое распространение. ПХП очень популярен в веб-разработке и поддерживается множеством хостинг-провайдеров.
9. Многофункциональность. Язык веб-программирования подходит как для создания несложных скриптов, так и для разработки крупных веб-проектов. Он дает разработчикам возможность выбора подходящего уровня сложности для решения конкретных задач.
10. Крупное сообщество и поддержка. PHP имеет большое и активное сообщество разработчиков. Простыми словами, это открывает доступ к массе обучающих материалов, форумам.

Именно эти свойства делают ПХП невероятно популярным инструментом для создания сайтов и веб-приложений. Но 2 из них стоит рассмотреть подробнее. Речь про интерпретируемость и слабую динамическую типизацию.

## 1.3.3 Что такое интерпретируемость в PHP

Прежде всего нужно разобраться, что такое рантайм. Рантайм — это среда выполнения PHP-скриптов, в которую входят:

* PHP-интерпретатор;
* библиотеки и модули;
* окружение сервера (если PHP используется для создания веб-приложений).

Вот некоторые ключевые аспекты рантайма PHP:

1. Интерпретатор PHP. Это основная часть рантайма, которая читает, анализирует и выполняет код. Интерпретатор отвечает за выполнение инструкций, обработку данных и взаимодействие с другими компонентами рантайма.
2. Библиотеки и модули. Рантайм PHP включает библиотеки и модули, которые предоставляют разработчику различные функции и возможности. Эти библиотеки могут включать в себя инструменты для работы с базами данных, файлами, отправки электронной почты и многое другое.
3. Среда веб-сервера. Если PHP используется для создания веб-приложений, рантайм будет включать в себя среду веб-сервера (например, Apache, Nginx и т. п.), последний обрабатывает запросы от клиентов и передает их на выполнение PHP-скриптам.
4. Управление памятью. Рантайм PHP отвечает за управление памятью в процессе выполнения скриптов. Простыми словами, за выделение и освобождение памяти для обеспечения эффективного выполнения скриптов.
5. Обработка запросов. В веб-приложениях рантайм PHP обрабатывает входящие HTTP-запросы, выполняя соответствующие скрипты и генерируя HTTP-ответы, которые отправляются клиентам.

Рантайм PHP работает непосредственно на сервере и обеспечивает выполнение PHP-скриптов в реальном времени. Он предоставляет разработчикам среду для создания веб-приложений, обработки данных и выполнения различных задач, связанных с серверной стороной веб-программирования.

PHP — интерпретируемый язык программирования, то есть все его скрипты выполняются интерпретатором в реальном времени, построчно, без предварительной компиляции в машинный код. Вот как это работает:

Интерпретатор анализирует и выполняет PHP-код. Когда сервер или другой запускающий механизм получает запрос на выполнение скрипта, интерпретатор берет этот скрипт, разбирает его и начинает выполнять инструкции по порядку.

Интерпретатор анализирует каждую строку ПХП-кода и выполняет соответствующие операции. Он может обращаться к базам данных, работать с файлами, генерировать HTML-страницы и многое другое, в зависимости от содержания скрипта.

PHP — язык программирования с динамической типизацией. То есть вы не обязаны объявлять типы данных переменных. Этот тип определяется автоматически в процессе выполнения, что делает PHP крайне гибким, но требует предельной аккуратности и точности при написании кода.

## 1.3.4 Преимущества интерпретируемости PHP:

1. Простота веб-программирования. Вы можете быстро изменять и тестировать ПХП-скрипты без необходимости компиляции, что упрощает процесс разработки.
2. Кроссплатформенность. Скрипты могут выполняться на разных операционных системах и серверах без изменений, так как интерпретатор PHP доступен для большинства современных платформ.
3. Динамическая адаптация ПХП легко адаптируется к изменяющимся требованиям, поскольку типы данных определяются динамически.

Нужно понимать еще один момент — интерпретируемость означает, что PHP-скрипты могут иметь некоторую потерю производительности по сравнению с языками, которые предварительно компилируются в машинный код. Впрочем, оптимизация и кэширование могут нивелировать этот недостаток.

## 1.3.5 Слабая динамическая типизация в PHP

PHP имеет слабую динамическую типизацию. Простыми словами, переменные могут автоматически изменять тип данных в зависимости от контекста операций, выполняемых над ними. Это приводит к следующим особенностям:

1. PHP позволяет выполнять операции между переменными разных типов без явного преобразования. Например, вы можете складывать строку и число, PHP попытается выполнить неявное преобразование.
2. Тип данных переменной определяется автоматически в процессе выполнения кода. Это позволяет создавать более гибкий и адаптивный код. Но также это может привести к ошибкам, если не учесть динамическое изменение типов.
3. Операторы PHP могут выполнять различные операции в зависимости от типов данных операндов. Например, оператор + может выполнять сложение чисел или конкатенацию строк.
4. При сравнении значений в PHP следует быть очень осторожным, так как операторы сравнения (== и ===) могут вести себя по-разному в зависимости от типов данных. == выполняет нестрогое сравнение с преобразованием типов, а === выполняет строгое сравнение.

Слабая динамическая типизация может сделать код более гибким, но также требует от разработчиков максимального контроля над типами данных, чтобы избежать неожиданных ошибок. Это одна из особенностей PHP, которую разработчики должны учитывать при написании кода на этом языке.

## 1.3.6 Как работает PHP

Сервер сразу отправляет клиенту файлы изображений, CSS- и НТМL-код, независимо от их содержания. Но эти файлы не содержат код, который интерпретируется на сервере, а PHP-файлы его содержат. То есть клиент получает не сам ПХП-код, а результат его исполнения. И этот результат представляет собой чистый CSS- и НТМL-код. Простыми словами, клиент не видит исходный код. Вот как работает ПХП. В плане безопасности это огромный плюс, ведь клиент не может понять логику веб-приложения или сайта.

Если сравнить ПХП с компилируемыми языками веб-программирования (они работают постоянно — от момента запуска приложения до момента его закрытия), то скрипт PHP начинает выполняться с момента обращения к серверу, а завершает выполнение в момент, когда возвращает серверу пакет сгенерированных данных.

1. Чтобы лучше понять, как работает PHP, рассмотрим самый простой вариант его взаимодействия с сервером:
2. Сервер принимает входящее соединение, проводит анализ данных. Затем они интерпретируются, формируется ответ для клиента.
3. PHP-интерпретатор генерирует НТМL-код, который распознает браузер.
4. Также в процессе работы ПХП может использоваться СУБД MySQL, которая обеспечивает обмен данными с сервером БД на языке SQL. В таком случае сервер БД и веб-сервер работают «в паре» — считывают и записывают данные, отображают веб-страницы клиентам.
5. БД отправляет ПХП-скрипту данные. Скрипт распределяет их по заданным местам на странице и формирует таким образом ответ клиенту.
6. Сформированный ответ отправляется клиенту.

Простыми словами, PHP и HTML-код можно комбинировать, что позволяет создавать динамические страницы, которые будут показывать клиенту разную информацию (в зависимости от настроек и данных, полученных от клиента).

## 1.3.7 Особенности синтаксиса

PHP написан на C, но если копнуть глубже, станет ясно, что кроме С в нем есть «частички» Perl и Java. Стандартный ПХП-код — это набор выражений, каждое из которых прописывается с новой строки и завершается точкой с запятой. При этом выражение — это инструкция, на основе которой ПХП-интерпретатор будет выполнять конкретное действие (к примеру, выполнять операцию умножения и выводить ее результат на дисплей).

Все ПХП-сценарии начинаются с <?php. Это отправляет серверу сигнал о том, что в исполняемом файле находится PHP-код.

Рассмотрим самый простой сценарий, выводящий на экран надпись «Hi!»:

<?php echo("Hi!"); ?>

ПХП и ХТМЛ можно прописывать в одном файле. Например:

<body>

<p> <?php echo("Hi!"); ?> </p>

</body>

В этом примере ?> показывает серверу, где конец ПХП-сценария. В ситуации, когда после кода в файле ничего нет, прописывать ?> не нужно.

## 1.3.8 Где используется

PHP — одним из самых популярных серверных языков программирования для веб-разработки. Он используется в следующих сферах:

1. Веб-программирование. Используется для создания динамических веб-сайтов и веб-приложений. С помощью PHP можно генерировать интерактивные формы, обрабатывать данные, работать с БД и создавать динамический контент.
2. Разработка фреймворков. Есть несколько PHP-фреймворков (Laravel, Symfony, Zend Framework), упрощающих разработку веб-приложений и облегчающих реализацию стандартных функций.
3. Электронная коммерция. ПХП используется для создания интернет-магазинов и систем управления контентом для них. Например, популярные WordPress, WooCommerce и Magento основаны именно на PHP.
4. Социальные сети. Многие социальные сети (например, Facebook), были созданы с использованием PHP. Также PHP применяется для интеграции с API социальных сетей.
5. Форумы и блоги. Множество популярных движков форумов и блогов работают на PHP-платформах (phpBB, WordPress).
6. API и веб-сервисы. ПХП используется для создания API и веб-сервисов, позволяющих взаимодействовать с приложениями и данными через интернет.
7. Обработка изображений и мультимедиа. Язык веб-программирования может использоваться для обработки изображений, аудио- и видеофайлов на сервере.
8. Администрирование серверов. Некоторые системные инструменты и веб-панели для управления серверами также работают на PHP.

Это лишь малая часть примеров областей применения PHP. Благодаря своей популярности и доступности, PHP остается важным инструментом для разработки и создания веб-приложений.

## 1.3.9 Перспективы PHP

Перспективы PHP как языка программирования весьма обнадеживающие, несмотря на появление и развитие других языков и технологий. Кроме того, разработчики постоянно улучшают язык. Выпуск новых версий с новыми возможностями и улучшениями поддерживает актуальность PHP на высоком уровне.

ПХП можно применять:

1. в области интернета вещей;
2. для создания серверных приложений;
3. для обработки и анализа больших объемов данных;
4. для создания API и бэкенд-части мобильных приложений.

Можно предположить, что ПХП продолжит развиваться и адаптироваться к изменяющимся потребностям разработчиков и рынка информационных технологий в целом. Но все же неизвестно, станут ли разработчики массово переходить на другие решения.

## 1.3.10 Почему многие ненавидят PHP

Из-за его простоты. Ведь именно простота языка сделала так, что появилось очень много откровенно плохого кода. Кроме того, начать программировать на PHP очень просто, этот язык устроен таким образом, что сможет «заставить работать» любой другой код, если тот написан с хоть каким-то соблюдением правил.

Простыми словами, простота программирования на PHP привела к невозможности нормально поддерживать сайт, если тот был кем-то написан на ПХП. Этот язык веб-программирования дает много свободы в плане обращения к БД, переменным, типам данным, из-за чего разобраться в чужом коде практически нереально.

Кроме того, раннее ПХП был крайне уязвимым, из-за чего практический любой пользователь мог проникнуть в БД веб-проекта и внести свои правки. Конечно, в новых версиях подобные проблемы отсутствуют, но некоторые веб-проекты остаются уязвимыми.

А еще PHP позволяет объявлять переменные в любой части документа. Это может быть удобно, но если кто-то захочет разобраться в этом коде, такая вольность в действиях сильно усложнит его работу.

PHP как язык веб-программирования не любят как минимум из-за:

1. отсутствия контроля и отладок ошибок;
2. странной реализации работы с классами и объектами;
3. малого количества фреймворков;
4. слабой реализации многопоточности.

Но все же когда вы покупаете хостинг, провайдер по умолчанию включает PHP, а другие технологии по типу Node.js или Python приходиться активировать отдельно.

## 1.3.11 Плюсы и минусы

### 1.3.11.1 Плюсы в php

Можно выделить 7 основных плюсов ПХП:

1. Высокая скорость освоения. Написать первый рабочий код на ПХП можно буквально через месяц после начала обучения. Нужно лишь освоить синтаксис и понять принцип работы.
2. Универсальность. Этот язык веб-программирования можно использовать и для маленького блога, и для крупного онлайн-магазина.
3. Огромное сообщество коллег. Веб-разработчиков, которые постоянно работают с ПХП, миллионы. Отсюда — большое количество полезной и доступной информации от практиков.
4. Свободная лицензия. PHP бесплатен — его могут использовать как коммерческие организации, так и частные лица.
5. Простота написания и редактирования. Работать с PHP-кодом можно в любом редакторе (даже обычном текстовом).
6. Расширяемость. За счет разработки новых библиотек число дополнительных опция языка постоянно растет.
7. Нетребовательность к ресурсам. Код, написанный на PHP, будет работать даже на самом дешевом арендованном сервере с устаревшим программно-аппаратным обеспечением.

### 1.3.11.2 Минусы в PHP

Из минусов — 3 наиболее серьезных:

1. ПХП не подойдет для решения задач, связанных с созданием компонентов операционной системы или приложения для десктопа.
2. PHP по сравнению с другими языками веб-программирования имеет меньшую степень защиты.
3. У ПХП есть проблемы с глобальными исключениями.

# Глава 2 Практическая часть

# Заключение

# Библиографический список

# Приложение