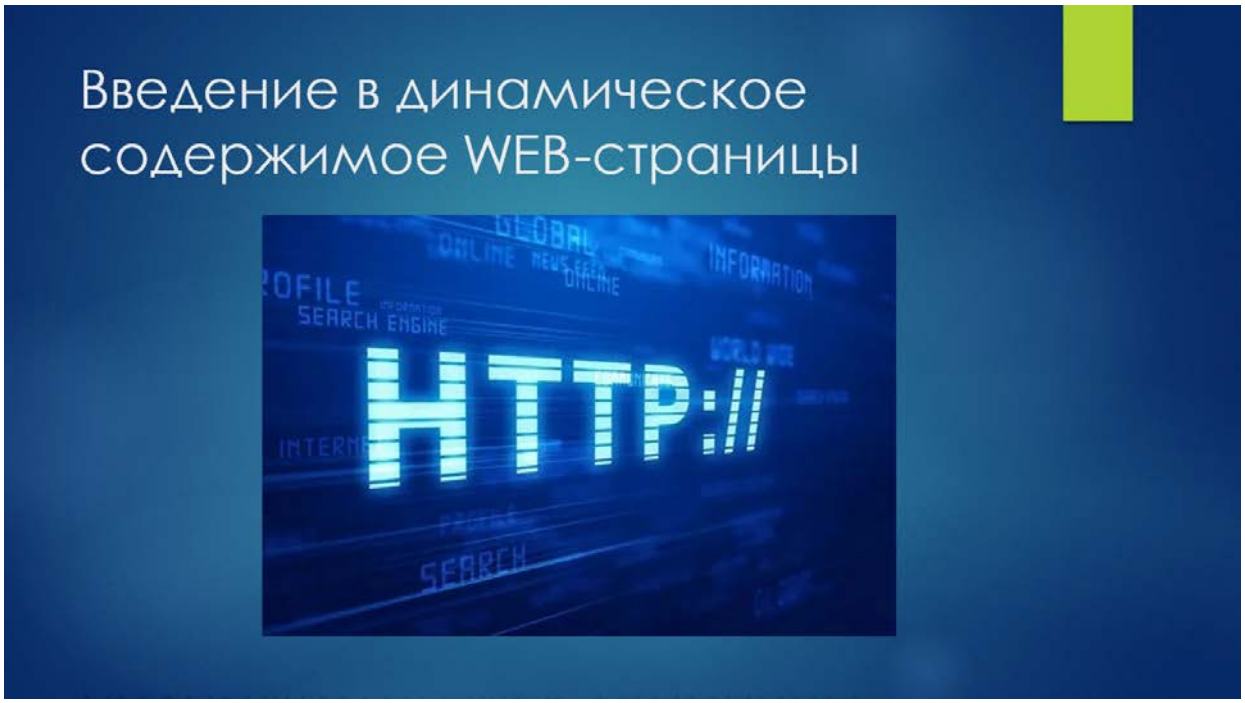
Проектирование и разработка WEB приложений

С.С. Науменко

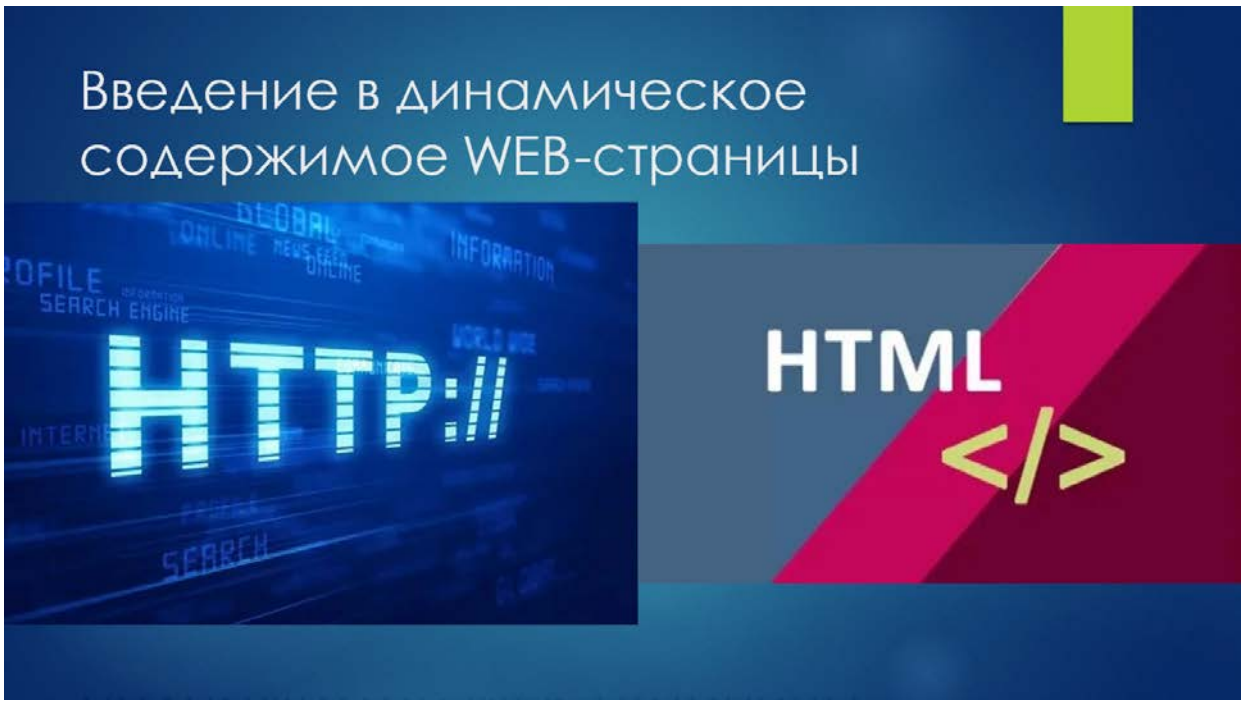
Лекция 2

1. Введение в динамическое содержимое Веб страницы

Всемирная паутина — это непрерывно развивающаяся сеть, ушедшая далеко вперед от своей концепции ранних 1990-х, когда ее создание было обусловлено решением конкретных задач. Высокотехнологичные эксперименты в ЦЕРНе (Европейском центре физики высоких энергий, известном в наши дни в качестве обладателя Большого адронного коллайдера) выдавали невероятно большой объем данных, который был слишком велик для распространения среди участвующих в экспериментах ученых, разбросанных по всему миру.



К тому времени Интернет уже существовал и к нему было подключено несколько сотен тысяч компьютеров, поэтому Тим Бернерс-Ли (специалист ЦЕРНа) придумал способ навигации между ними с использованием среды гиперссылок — так называемого протокола передачи гиперссылок (Hyper Text Transfer Protocol (HTTP)).



Он также создал специальный язык разметки, названный языком гипертекстовой разметки (Hyper Text Markup Language (HTML)). Для того чтобы собрать все это воедино, он создал первые браузер и веб-сервер, которые теперь воспринимаются нами как должное.

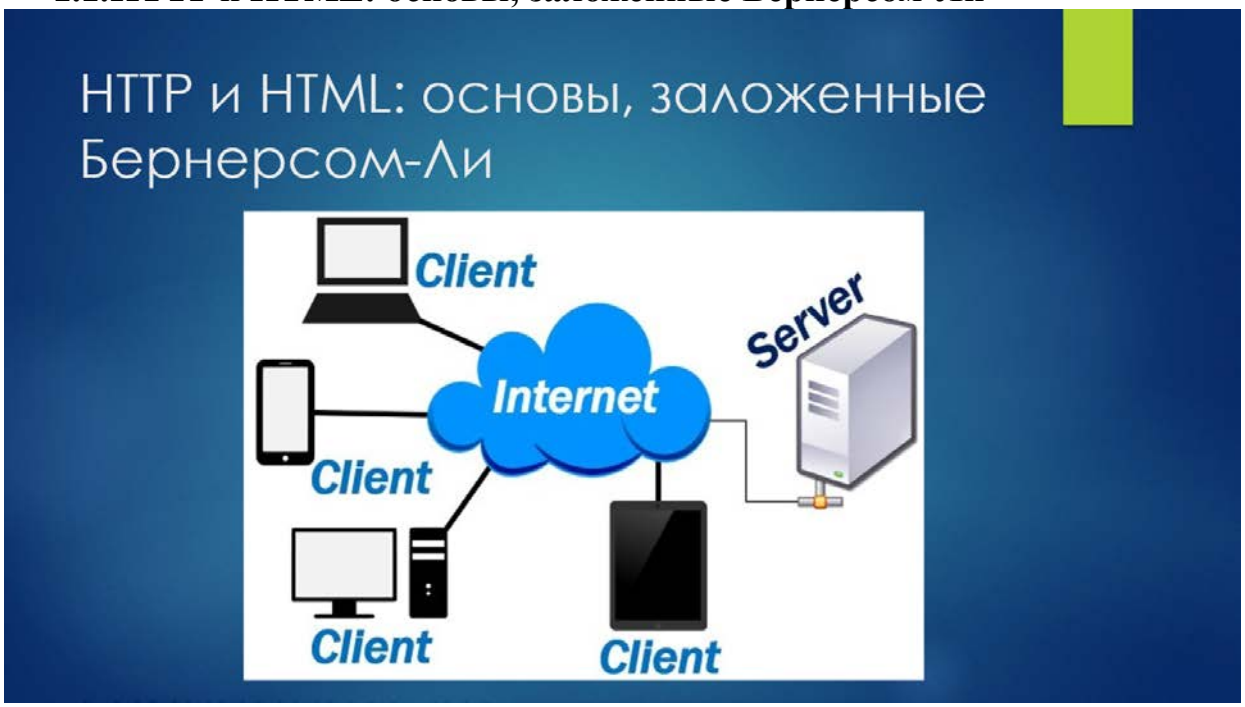
Но в то время эта концепция носила революционный характер. До этого основной объем соединений приходился на пользователей домашних модемов, дозванивавшихся и подключавшихся к электронным доскам объявлений, которые базировались на отдельном компьютере и позволяли общаться и обмениваться данными только с другими пользователями данной службы. Следовательно, для эффективного электронного общения с коллегами и друзьями нужно было становиться участником многих электронных досок объявлений.

Но Бернерс-Ли изменил все это одним махом, и к середине 1990-х годов уже существовали три основных конкурирующих друг с другом графических браузера, пользовавшихся вниманием 5 млн посетителей. Однако вскоре стало очевидно, что кое-что было упущено. Конечно, текстовые и графические страницы, имеющие гиперссылки для перехода на другие страницы, были блестящей концепцией, но результаты не отражали текущий потенциал компьютеров и Интернета по удовлетворению насущных потребностей пользователей в динамическом изменении контекста. Всемирная паутина оставляла весьма невыразительное впечатление, даже при наличии прокрутки текста и анимированных GIF-картинок.

Корзины покупателей, поисковые машины и социальные сети внесли существенные коррективы в порядок использования Всемирной паутины. В этой лекции будет дан краткий обзор различных компонентов, формирующих

ее облик, и программного обеспечения, способствующего обогащению и оживлению наших впечатлений от ее использования.

1.1.HTTP и HTML: основы, заложенные Бернерсом-Ли



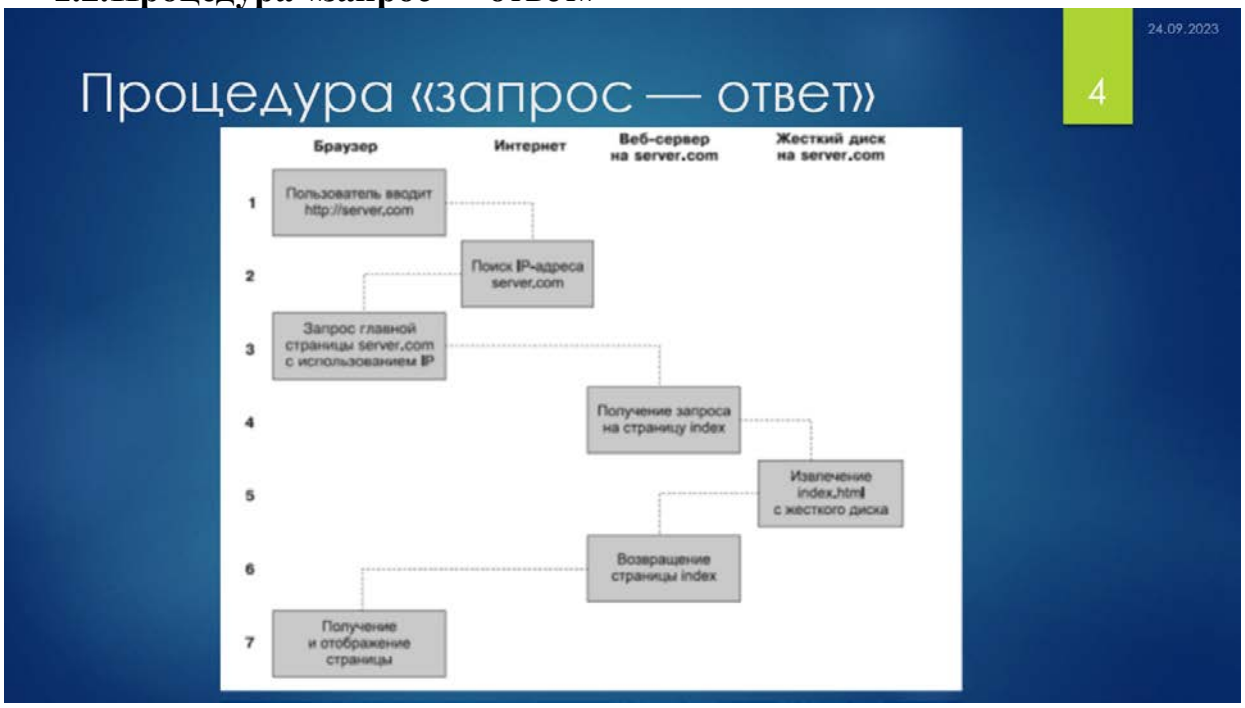
HTTP представляет собой стандарт взаимодействия, регулирующий порядок направления запросов и получения ответов — процесса, происходящего между браузером, запущенным на компьютере конечного пользователя, и веб-сервером.

Задача сервера состоит в том, чтобы принять запрос от клиента и попытаться дать на него содержательный ответ, обычно передавая ему запрошенную веб-страницу. Именно поэтому и используется термин «сервер» («обслуживающий»). Партнером, взаимодействующим с сервером, является клиент, поэтому данное понятие применяется как к браузеру, так и к компьютеру, на котором он работает.

Между клиентом и сервером может располагаться ряд других устройств, например маршрутизаторы, модули доступа, шлюзы и т. д. Они выполняют различные задачи по обеспечению безошибочного перемещения запросов и ответов между клиентом и сервером. Как правило, для отправки этой информации используется Интернет.

Обычно веб-сервер может обрабатывать сразу несколько подключений, а при отсутствии связи с клиентом он находится в режиме ожидания входящего подключения. При поступлении запроса на подключение сервер подтверждает его получение отправкой ответа.

1.2.Процедура «запрос — ответ»



В наиболее общем виде процесс «запрос — ответ» состоит из просьбы браузера к веб-серверу отправить ему веб-страницу и выполнения браузером данной просьбы. После этого браузер занимается отображением страницы

При этом соблюдается такая последовательность действий.

1. Вы вводите в адресную строку браузера http://server.com.
2. Ваш браузер ищет IP-адрес, соответствующий доменному имени server.com.
3. Браузер посылает запрос на главную страницу server.com.
4. Запрос проходит по Интернету и поступает на веб-сервер server.com.
5. Веб-сервер, получивший запрос, ищет веб-страницу на своем жестком диске.
6. Сервер извлекает веб-страницу и отправляет ее по обратному маршруту в адрес браузера.
7. Браузер отображает веб-страницу.

При передаче статической веб-страницы этот процесс осуществляется для каждого имеющегося на ней объекта: элемента графики, встроенного видео- или Flash-ролика и даже шаблона CSS.

Обратите внимание на то, что на шаге 2 браузер ищет IP-адрес, принадлежащий доменному имени server.com. У каждой машины, подключенной к Интернету, включая и ваш компьютер, есть свой IP-адрес. Но, как правило, доступ к веб-серверам осуществляется по именам, таким как google.com. Вам, должно быть, известно, что браузер обращается к вспомогательной интернет-службе, так называемой службе доменных имен (Domain Name Service (DNS)), для того чтобы найти связанный с сервером IP-адрес, а затем воспользоваться им для связи с компьютером.

При передаче динамических веб-страниц процедура состоит из большего количества действий, поскольку к ней могут привлекаться как PHP, так и MySQL

1. Вы вводите в адресную строку браузера http://server.com.

2. Ваш браузер ищет IP-адрес, соответствующий доменному имени server.com.

3. Браузер посылает запрос на главную страницу server.com.

4. Запрос проходит по Сети и поступает на веб-сервер server.com.

5. Веб-сервер, получивший запрос, ищет веб-страницу на своем жестком диске.

6. Теперь, когда главная страница размещена в его памяти, веб-сервер замечает, что она представлена файлом, включающим в себя PHP-сценарии, и передает страницу интерпретатору PHP.

7. Интерпретатор PHP выполняет PHP-код.

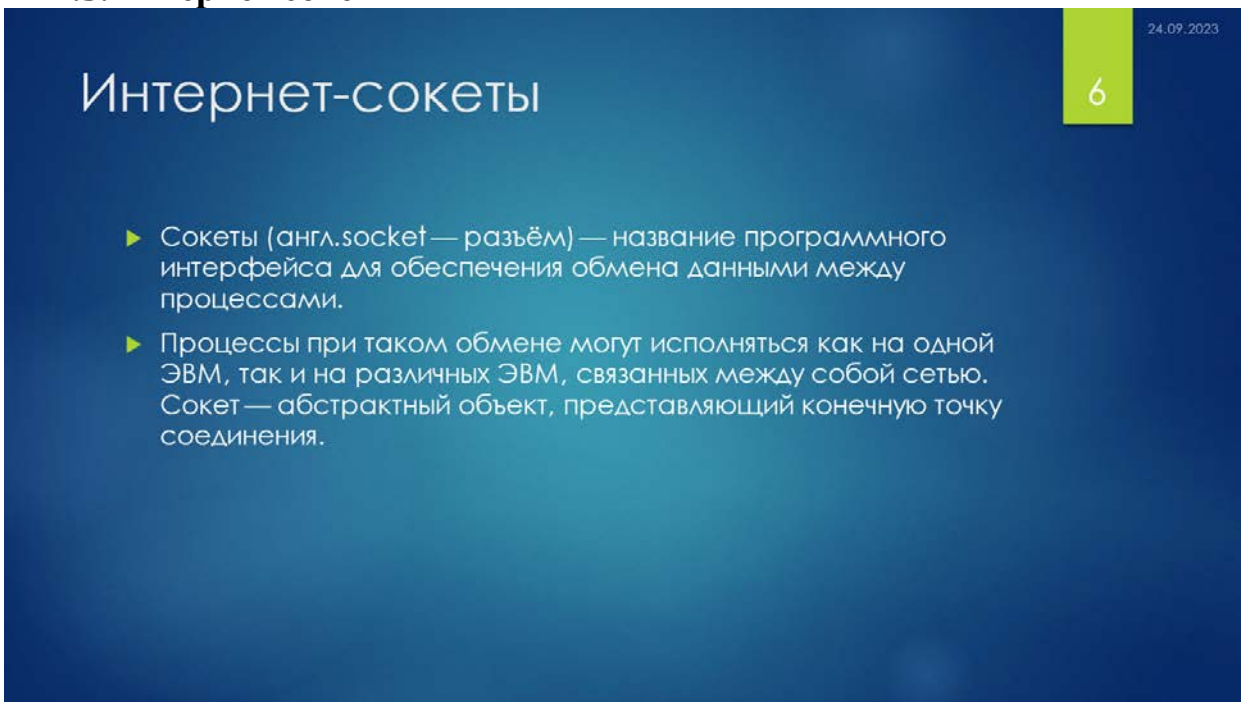
8. Кое-какие фрагменты кода PHP содержат MySQL-инструкции, которые интерпретатор PHP, в свою очередь, передает процессору базы данных MySQL.

9. База данных MySQL возвращает результаты выполнения инструкции интерпретатору PHP.

10. Интерпретатор PHP возвращает веб-серверу результаты выполнения кода PHP, а также результаты, полученные от базы данных MySQL.

11. Веб-сервер возвращает страницу выдавшему запрос клиенту, который отображает эту страницу на экране.

1.3. Интернет сокеты



Реализация передачи данных между компьютерами обычно выполняется с использованием Интернет сокетов – специальным программных объектов, которые позволяют организовать передачу данных между выполняемыми процессами, с использованием протокола TCP/IP. Сокеты — это некоторые абстракции (объекты), с помощью которых приложение может посылать и получать данные, во многом аналогично тому, как с помощью указателя на открытый файл приложение может читать и писать данные на внешнее устройство хранения. Сокет позволяет приложению подключиться к сети и выполнять взаимодействие (обмен данными) с другими приложениями, которые с помощью своих сокетов подключаются к той же самой сети. Данные переданные сокету на одном компьютере, могут читаться другим приложением, использующим сокеты, на другом компьютере.

Сокеты обычно реализуются с помощью API библиотек, как например библиотеки «Berkeley sockets», первоначально созданной в 1983. Большинство

реализаций библиотек сокетов создано на основе данной библиотеки, например, библиотека Winsock, разработанная в 1991 году. Разработка прикладных программ, использующих такие API библиотеки, называется сетевым программированием.

1.4. Система доменных имен

Примером одной из информационных систем Интернет является

«Система доменных имен» (Domain Name System, DNS). Данная система является частью инфраструктуры сети, которая позволяет хранить и быстро находить специальные записи, связывающие IP адреса с символьными именами (доменными именами).

Доменное имя, это символьное имя, служащее для обозначения иерархической структуры подобластей сети Интернет. Каждая из таких подобластей называется доменом.

Доменные имена могут соответствовать компьютерам (серверам, предоставляющим услуги в сети, хосты), web-сайтам, почтовым (e-mail) серверам.

Доменные имена компьютеров и web-сайтов, хранимые в DNS используются в URL адресах ресурсов web-сети. Полное доменное имя состоит из имени ближайщего (самого низкого уровня) домена и далее имѐн всех доменов более высокого уровня, в которые он входит, разделѐнных точками. Например, полное имя ru.wikipedia.org обозначает домен третьего уровня ru, который входит в домен второго уровня wikipedia, который входит в домен верхнего уровня org, который входит в безымянный корневой домен. Обычно под доменным именем понимают полное доменное имя.

Система DNS поддерживается с помощью иерархически организованных DNS-серверов, взаимодействующих по определѐнному протоколу, которые предоставляют доступ к иерархически распределенной базе данных. Для повышения устойчивости системы используется множество серверов, содержащих идентичную информацию, а в протоколе работы данной системы имеются средства, позволяющие выполнять синхронизацию информации, расположенной на разных серверах. Существует 13 корневых серверов, их адреса практически не меняются.

Доменное имя и IP-адрес не являются тождественными – один IP- адрес может иметь много доменных имѐн, что позволяет поддерживать на одном компьютере набор web-сайтов (так называемый виртуальный хостинг). Справедливо также и обратное – одному доменному имени может быть сопоставлено множество IP-адресов, что позволяет поддерживать работу одного web-сайта несколькими серверами (выполнять балансировку нагрузки на web-сайт).

1.5.Как устроен сайт

Что значит «зайти на сайт»

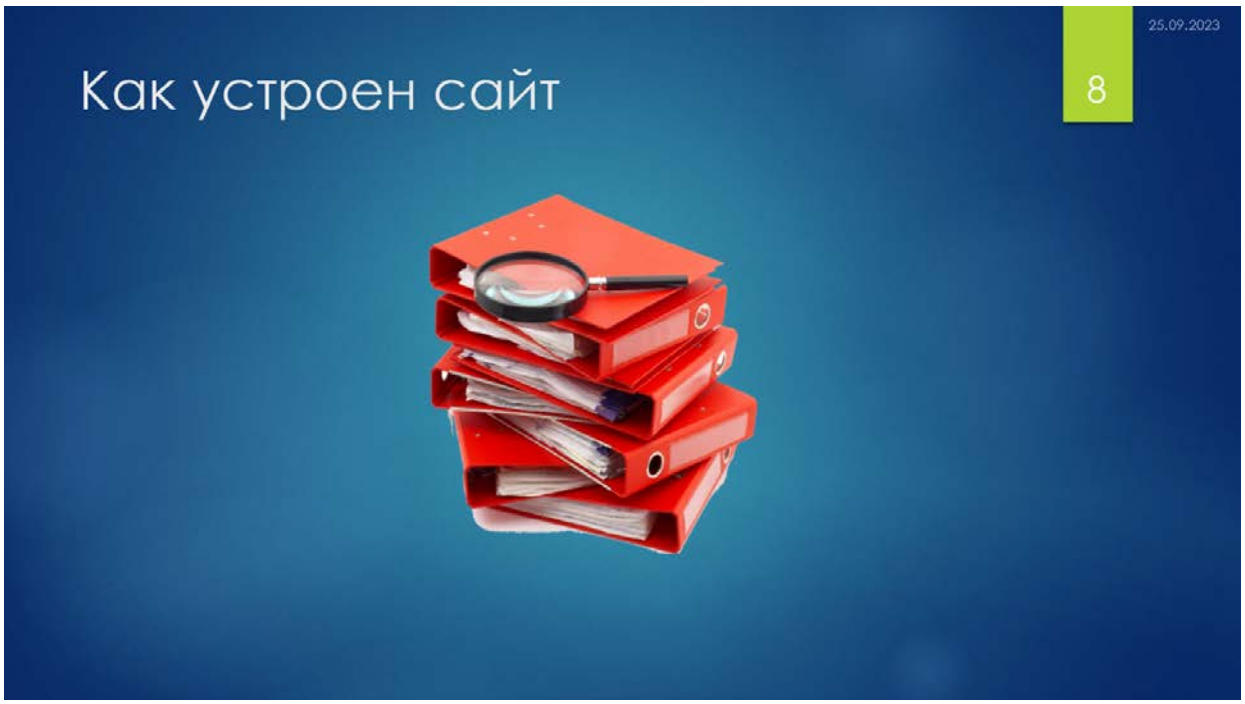
Когда вы заходите на сайт, на самом деле вы никуда не заходите. Вы делаете запрос на некий удалённый сервер, а он вам присылает в ответ документ. Документ прилетает в ваш браузер, браузер его анализирует и рисует вам веб-страничку. Вам кажется, что вы пришли в интернет, но на самом деле это кусочек интернета пришёл к вам.

Когда вы кликаете по ссылке, вам кажется, что вы переместились с одной страницы на другую. На самом деле вы отправили новый запрос на сервер, получили ответ, браузер снова нарисовал страницу. Вы никуда не переместились, вы лишь получили новый кусочек интернета.

Теперь вопрос — откуда появились эти кусочки интернета, которые вам прислали?

Статичные сайты

Изначально сайты в интернете хранились и отдавались так:

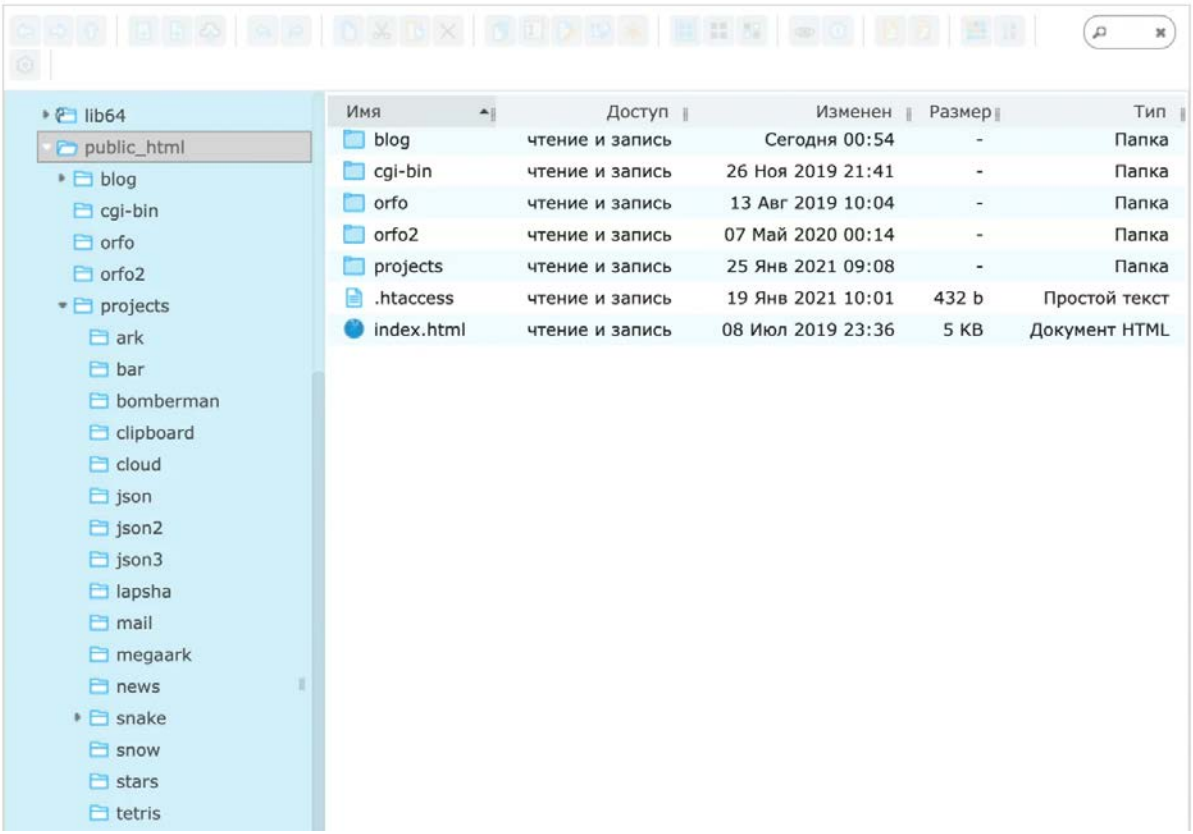


На удалённом компьютере лежали документы. В буквальном смысле: документы с текстовой информацией. Один документ соответствовал одной странице.

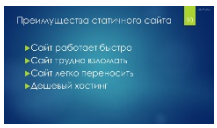
На этом же компьютере работала специальная программа — веб-сервер. Она знала, что, если у неё запросили какой-то сайт, нужно отдать какой-то конкретный документ. Можно представить, что сервер — это сотрудник архива: вы запросили документ — вам его отдали.

Следовательно, скачать из интернета можно было только те документы, которые физически лежали где-то на серверах.

Если вам на сайте нужно было иметь 30 страниц, вам нужно было иметь 30 документов, которые физически будут лежать на сервере.



Например, вот так выглядят документы сайта, В «корне» нам доступен только документ index.html — это главная страница. Ещё слева видны папки с проектами. Чтобы завести новый проект, мы делаем новую папку и складываем в неё нужные документы.



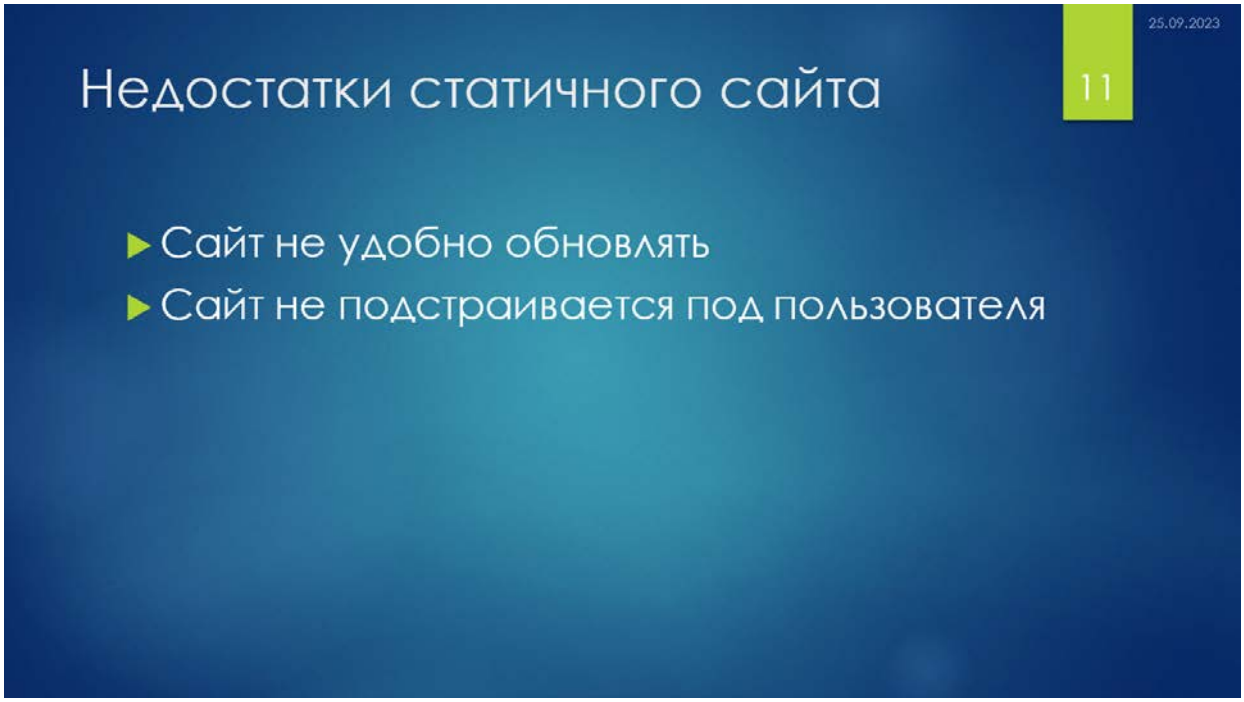
Что нам даёт статичный сайт (и в чём мешает)

✅ Сайт работает молниеносно — отдавать заранее заготовленные документы очень легко, с этим справится даже маломощный компьютер типа Arduino. А мощные веб-сервера — и подавно: странички будут прилетать мгновенно.

✅ Сайт очень трудно взломать: единственный способ навредить сайту — это получить прямой доступ в файловую систему сервера и вручную напакостить в каждом файле (или стереть их). Это не невозможно, но в современных реалиях довольно трудно.

✅ Сайт элементарно переносить: если сломался один сервер, просто копируете файлы из резервной копии на новый сервер, и сайт работает как раньше. Никакой дополнительной настройки, кроме перенаправления адреса для запросов. Никаких баз данных, версий движка и глобальных переменных.

✅ Дёшево хостить: услуга хостинга файлов — самая дешёвая из всех, потому что хранение и раздача файлов расходуют мало ресурсов.



❌ Сайт неудобно обновлять: чтобы обновить информацию на какой-то странице, нужно отредактировать нужный файл и загрузить его на сервер. Если нужно внести изменения в 30 страниц, эту операцию придётся повторить 30 раз. Например, если вы хотите добавить новый пункт меню.

❌ Сайт не подстраивается под пользователя: статичный сайт может только выдавать вам заранее заготовленную информацию. Если нужно сформировать корзину покупок или поискать что-то конкретно под ваш запрос, статики недостаточно. Исключение — скрипты, которые можно выполнить прямо в браузере.

👉 Статичные сайты идеальны для сайтов-визиток или сайтов-книг, где один раз написал информацию, и она там долго лежит.

Динамические сайты

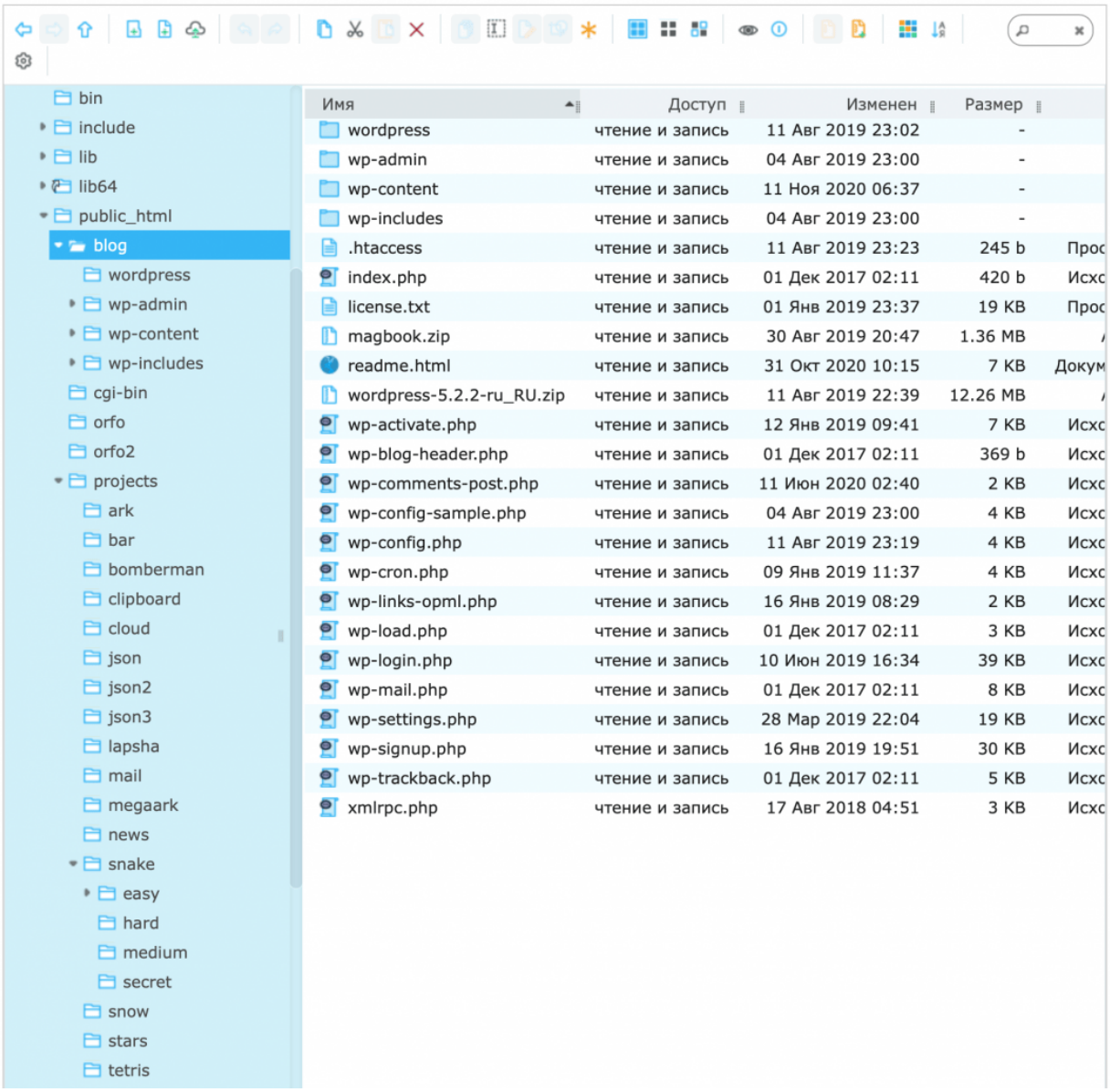
Чтобы обойти ограничения статичных сайтов, придумали такую схему:

1. Вместо файлов с контентом на сервере работает специальная программа — назовём её поваром.
2. Повар получает заказ, например, «Приготовь мне главную страницу этого сайта». Повар берёт нужные ингредиенты и собирает из них главную страницу, отдаёт.
3. Получается, что каждая страница сайта как бы «готовится» под ваш запрос с помощью специальной программы.

Если в статичных сайтах сервер выполнял роль работника архива, то в динамических — это такой официант: он принимает у вас запрос, относит на кухню, даёт задание повару, а когда блюдо готово, приносит вам.

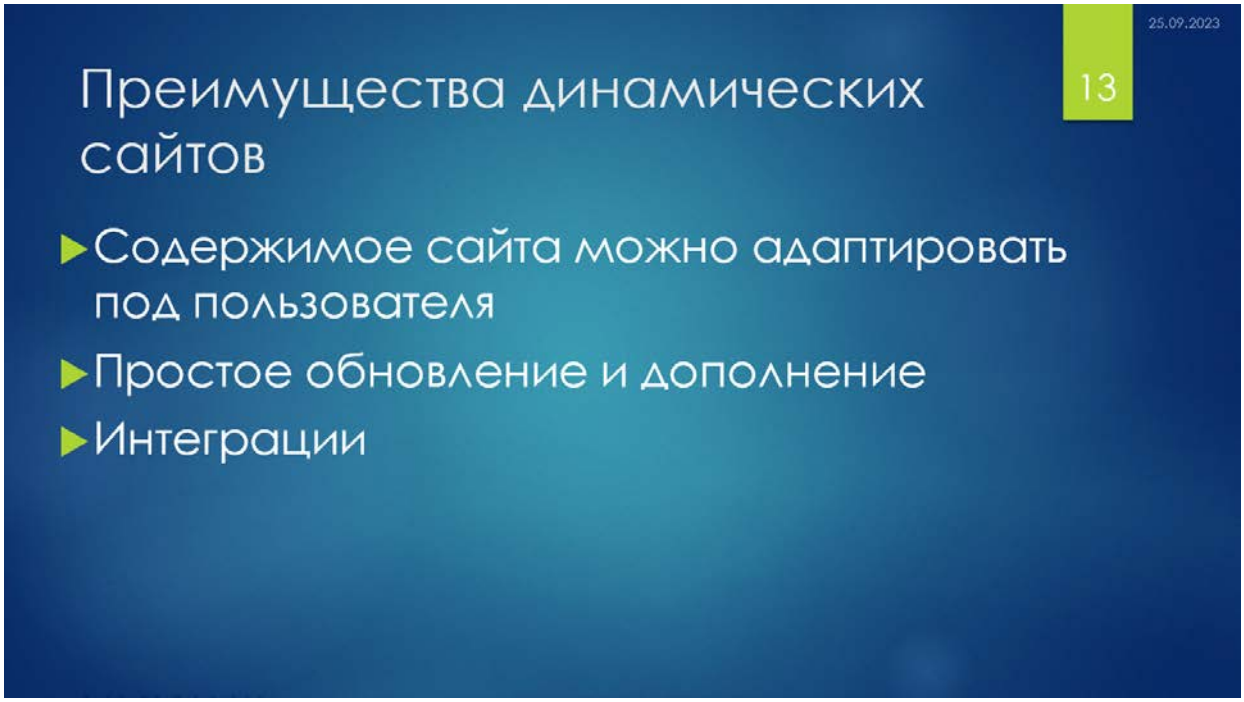
На самом сервере больше не хранятся как таковые документы — теперь они собираются под индивидуальный запрос из существующих заготовок.

Например, в базе данных лежит тысяча товаров, а вы просите показать 10 самых популярных. Не проблема: шеф-повар делает запрос в базу данных, получает ответ, собирает под вас страничку и отдаёт.



Когда сайт динамический, его фактические файлы никак не связаны с его содержимым. Например, на скриншоте — папка системы динамических сайтов WordPress. Каждый из 14 файлов — это, условно говоря, повар, который делает свою часть работы. Но эти 14 поваров вместе могут подать нам бесконечное количество динамических документов. То, что здесь 14 файлов, никак не связано с тем, сколько и каких страниц может выдать наш сайт.

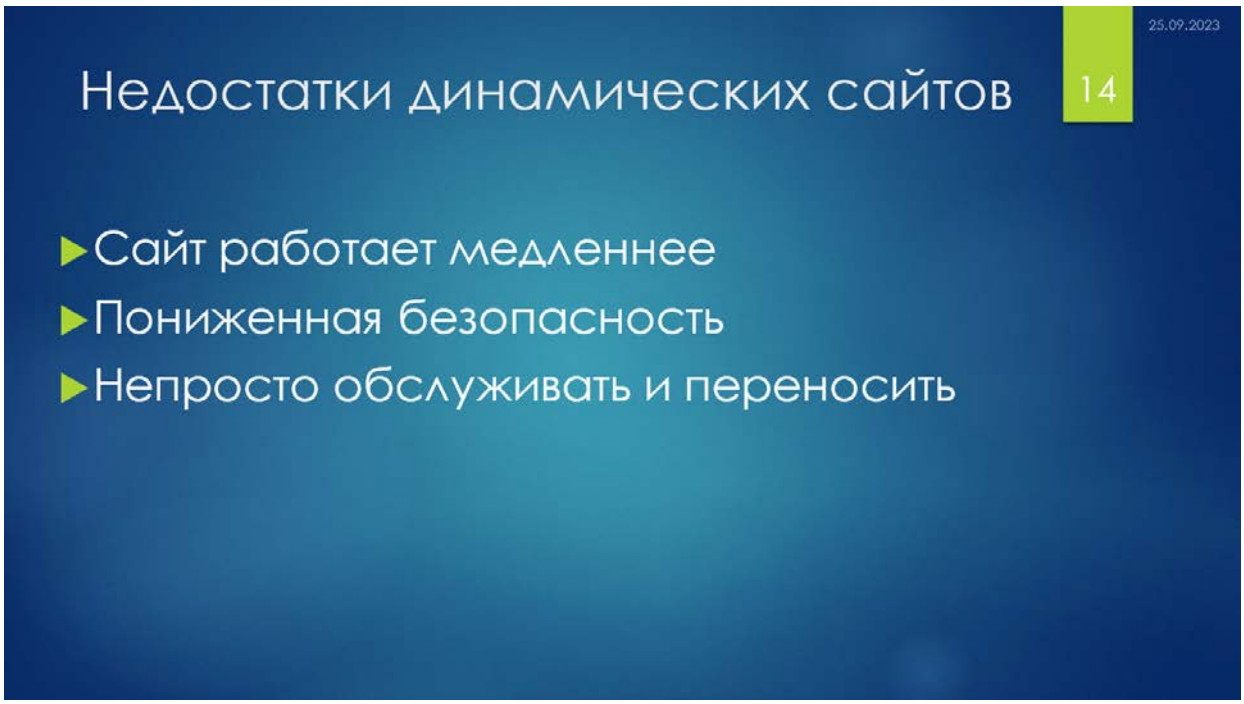
Что даёт динамический сайт (и в чём мешает)



✅ Содержимое сайта можно адаптировать под пользователя: страницы заказов, личные кабинеты, результаты поиска, соцсети, форумы, комментарии — всё это динамические продукты. В статике реализовать всё это невозможно.

✅ Простое обновление и дополнение: при желании разработчик может сделать удобный интерфейс добавления материалов на сайт — как в тех же соцсетях. Вам не нужно загружать файлик в «Фейсбук» или VK.com, вы просто набираете текст прямо в браузере.

✅ Интеграции: сайт может сам обращаться к базам данных, спрашивать данные у других сервисов, на лету подгружать ваши данные из разных источников.



❌ Сайт работает медленнее: сборка страниц на лету требует больше времени, чем отдача документа. Счёт идёт на десятые доли секунды, но, когда у тебя очень много пользователей, это может быть проблемой.

❌ Есть пространство для взлома: хакер может завернуть в свой запрос какой-нибудь вредоносный код (в духе «повар, приготовь мне торт с ключами от твоей квартиры»). Если специально не предусматривать такие атаки, можно получить дыру в безопасности.

❌ Непросто обслуживать и переносить. Динамические сайты требуют установки на сервер особенного сборочного софта (например, PHP или Python). Этот софт должен быть определённой версии, с определённым набором модулей. Их нужно правильно между собой увязать. Это не так просто, как скопировать файлики и перекинуть на другой сервер.

Гибридный вариант: кеширование

Допустим, нам нужно совместить плюсы двух подходов. Мы хотим, чтобы сайт был предельно быстрый и взломостойкий, но при этом его было легко обновлять. На этот случай придумали кеширование.

Кэширование похоже на отдел кулинарии в супермаркете: повара заранее наготовили салатов и нажарили курицы, а мы берём эту еду с прилавка. С одной стороны, не нужно ждать, что повар приготовит курицу конкретно под нас. С другой стороны, эта курица может быть не самой свежей.

Кэширование работает похожим образом: берётся движок сайта, на его основе генерируется куча документов. И дальше эта куча ведёт себя как статичный сайт. Когда это может быть полезно:

В блоге, где есть статьи, но нет комментариев. Кеш нужно обновлять, только когда добавилась новая статья.

В принципе, с комментариями тоже можно: если кто-то комментирует ваши статьи раз в час, можно сразу перестраивать кеш, и у вас будет актуальная копия сайта.

На сайте-каталоге, где нельзя оформить заказ. Кеш обновляется только при добавлении нового товара.

В справочниках, энциклопедиях, книгах: исправил опечатку — кеш обновился.

Что дальше

На основе идеи кеширования придумали технологию генерации статичных сайтов. Это когда у вас есть куча исходного текста (например, инструкция от сложной системы). Вы говорите: «Оформи мне эту кучу текста по такому-то шаблону». Специальный генератор оформляет эту кучу, а вы получаете пачку статичных документов, которые работают как молниеносный статичный сайт.