18.08.2015

Почему JavaScript и PHP победили в Интернете

Ни PHP, ни JavaScript не являются великой технологией. Если вы смотрите на оба языка с точки зрения дизайна, вы не найдете много интересного. На самом деле, вы найдете много разочарования.

Это не теоретический анализ с точки зрения исследовательского языка; множество широко используемых языков избегают формальных методов, не раскрывают основного формализма, или даже избегают базовой безопасности типов и вывода из 1970-х годов. Действительно, одним из самых популярных языков, которые до сих пор широко используются, является язык программирования C, который берет свое начало с конца 1960-х годов. (Несмотря на обновления в последние десятилетия, он все еще по существу тот же язык, что и в своей первоначальной форме.)

Почему PHP добился успеха?

Первоначальная цель дизайна PHP была сделать создание веб-страниц простым. В его основе - это язык шаблонов. Возьмите веб-страницу. Добавьте пару циклов и условий. Добавьте некоторые переменные. У вас есть PHP. Уберите все это, и у вас все равно будет PHP.

Это означает, что любая когда-либо написанная веб-страница является действительным документом PHP. Это действительный шаблон PHP. Тот, кто управляет вашим веб-сервером, может настроить этот сервер на обслуживание этой страницы с помощью PHP, и она будет работать идеально. Кроме того, веб-сервер Apache httpd вскоре получил расширение под названием mod_php, которое добавило поддержку PHP. Когда Apache httpd был самым популярным веб-сервером на планете, любой сервер мог включить PHP без проблем.

PHP не всегда был очевидным лидером в серверном веб-программировании. Администраторы систем должны были сами включить PHP. Однако это было логично. Модель выполнения PHP — способ, которым PHP превращает веб-страницу с командами PHP в веб-страницу — очень проста. У каждого запроса есть своя память. Она инициализируется, страница обслуживается, а затем эта память освобождается. Отсутствие каких-либо ошибок в самом PHP позволяет иметь тысячу различных сайтов на одном сервере, каждый из которых работает на PHP, и ни один из них не будет знать ничего о других.

Да, PHP, работающий через mod_php, был более эффективным, чем серверные программы, работающие в модели выполнения CGI. Если вы запускали сто различных сайтов на одном хосте и любой из них имел какую-то популярность, накладные расходы CGI могли перегрузить веб-сервер.

Установка mod_php была дешевой и простой, а обучение пользователей тому, как изменить существующую HTML-страницу для использования нескольких директив PHP, было намного проще, чем объяснение того, как настроить скрипт CGI на Perl или shell или C, включая установку прав доступа и перемещение вещей в правильный файл и добавление необходимых библиотек, поэтому

веб-серверам было проще использовать mod_php.

Если вы платили 4 доллара в месяц за хостинг вашего сайта, скорее всего, у вас был доступный PHP. Если вы установили свою собственную дистрибутив Linux, скорее всего, установка Apache httpd, которую он имел в конфигурации по умолчанию, включала mod_php.

PHP стал успешным, потому что сначала было очень легко его получить. Он действительно преуспел, потому что он уже был там. Не недооценивайте ценность того, чтобы быть там. Да, было относительно легко начать — проще, чем Perl или shell или C, потому что любая существующая HTML-страница является действительным документом PHP — но более важно, что он уже был там. Люди изучали его, потому что он уже был там, и люди устанавливали его, потому что люди изучили его и хотели его.

Наем программистов PHP

Работодатели хотят нанимать программистов PHP, потому что эта технология повсеместна (поэтому риск выбора стандартной технологии минимален) и потому что есть много программистов PHP, которых можно

нанять (что делает их дешевыми и легкими для замены). Программисты хотят изучать PHP, потому что его легко учить и доступно много вакансий. Конечно, эта ситуация была предрасположена к тому, чтобы стать лимонным рынком принятия языка программирования - и так оно и случилось.

Почему JavaScript добился успеха?

Когда интернет только начинал свое развитие, в нем даже не было изображений. Были только ссылки между страницами, и это было практически все. Это было огромным преимуществом перед тем, что существовало ранее.

Затем появились формы и серверное программирование. Вы могли взаимодействовать с этими документами. Могли вводить свой адрес электронной почты и получать письма с сайтов. Могли добавлять текущую дату на страницу или общее количество всех, кто ее когда-либо прочитал. Возможности были безграничными.

Потом появились апплеты Java, которые были небольшими кусочками кода, который ваш веб-браузер загружал с сервера и запускал на вашем компьютере. Это должно было быть быстрым и простым способом делать вещи без ожидания медленного интернета. К сожалению, апплеты Java склонны были к сбоям, использовали много памяти и работали медленно.

Потом появился JavaScript, который не был расширением браузера. Это была часть самого браузера. Вы могли иметь браузер без Java, но с Netscape Navigator у вас был браузер, включающий JavaScript. Если вы хотели отображать текущее время или перемещать текст или показывать здесь или там крутой эффект (когда вы нажимаете одну кнопку, появляется окно, если вам не нравится ответ!), вы могли положиться на то, что у людей есть JavaScript.

JavaScript не был везде — многие отключали его, потому что он склонен к раздражающим вещам — но он мог быть везде. Когда веб увидел свой огромный рост популярности, веб-браузеры были везде, и они все привносили в себя JavaScript.

(В те времена веб был диким местом. Один браузер добавлял функцию, а другие спешили ее скопировать. Попечители JavaScript мудро представили язык стандартному агентству ECMA, которое помогало контролировать фрагментацию, даже когда Microsoft пыталась захватить веб с помощью Internet Explorer.)

JavaScript не является идеальным языком. Он был разработан за деся

ть дней, и это заметно. Кроме того, он сталкивается со многими теми же проблемами, что и HTML: современный веб-браузер должен быть способен отображать веб-страницы, написанные в 1993 году, наряду с веб-страницами, написанными в 2013 году. За 20 лет многое изменилось, но старые страницы все еще должны работать. С JavaScript та же история; старый код, написанный для Internet Explorer 4, все еще должен работать. Это усложняет реализацию языка, потому что он должен поддерживать странные старые крайние случаи, и это усложняет программирование на языке, потому что программисты JavaScript должны учитывать эти странные крайние случаи.

Это не должно было сделать JavaScript успешным. На самом деле, его успех был неизбежен. Как и ни один веб-браузер действительно не подходит для не технических конечных пользователей, если он не поддерживает изображения, так и ни один веб-браузер действительно не подходит для не технических конечных пользователей, если он не поддерживает JavaScript. Когда Google Maps продемонстрировала удивительные вещи, которые вы могли сделать с JavaScript, игра была окончена. JavaScript победил. Вот почему планшет у вас под рукой или телефон в вашем кармане (не говоря уже о ноутбуке на вашей кухонной столешнице и настольном компьютере на вашем рабочем столе) все имеют по крайней мере один веб-браузер, который поддерживает JavaScript.

Это означает две вещи.

Во-первых, несмотря на свои недостатки (одним из первых среди которых является полное отсутствие системы библиотек, насчет которой можно утверждать, что существует несколько конкурирующих систем библиотек, которые взаимодействуют почти совсем не взаимодействуют, если они все не очень

дисциплинированны и которые или увеличивают объем загрузки одной страницы, или полагаются на очень точные версии и системы кеширования браузера и прокси-сервера, которые слишком византийские, чтобы объ

яснить даже в этой технической скобочной заметке), вы можете рассчитывать на то, что JavaScript будет присутствовать на каждом устройстве, способном к интернету, которое позволяет неограниченный доступ к веб-браузеру. Каждый из них.

В отличие от случая с PHP, где возможно, что установка веб-сервера не включает mod_php или где администратор запрещает использование PHP, вы можете с уверенностью предположить, что на любом устройстве, способном к интернету, будет какая-то форма JavaScript. Телефон, планшет, компьютер, телевизор, консоль, что угодно.

Таким образом, если вы хотите написать программное обеспечение, которое работает везде, где есть конечный пользователь, JavaScript - это опция. Его легко развернуть и он поддерживается везде.

Во-вторых, каждая из этих сред с доступом к клавиатуре (даже если это одна из этих ужасных экранных клавиатур, которые у вас есть на телефонах, планшетах или консолях) потенциально является средой программирования на JavaScript. Вам не нужно учиться об IDE и компиляторах или о том, как их устанавливать, и вам почти не нужно учиться о путях к файлам на вашем устройстве. Вы можете начать программировать на JavaScript прямо там. Как и PHP, документы JavaScript могут начинаться как простые HTML-документы. Вы можете добавлять к ним столько или насколько мало, сколько вам нравится, и они все равно будут работать.

Размывание границ при использовании серверного JavaScript и клиентских фреймворков JavaScript

Программисты так любят последовательность, что готовы потратить невероятное количество времени, усилий и ресурсов, чтобы сделать несколько казалось бы несовместимых вещей более последовательными. Это как если бы вы весь день распутывали запутанные шнурки на обуви. Конечно, это удовлетворяет ваши эстетические чувства, но что вы в итоге добились?

При создании сложных приложений, таких как Google Maps, которые написаны в основном на JavaScript, неизбежно возникает вопрос. "В чем польза сервера, если все, что он делает, - это отправляет некоторые данные, которые приложение на стороне клиента должно как-то перестраивать?"

Вместо того, чтобы JavaScript служил улучшением для идеально работающей веб-страницы - небольшой специей, добавляющей вкус к питательному блюду - веб-страница стала просто средством доставки целой программы, написанной на JavaScript, выполняемой на клиенте, которая производит реальный вывод, который веб-браузер будет отображать клиенту. Это уже не улучшение. Это главный ингредиент.

Формат взаимодействия этих данных, конечно, был JSON, который сам по себе является способом представления структур данных JavaScript. Учитывая сложное клиентское приложение на JavaScript, запрашивающее структуры данных JavaScript от сервера, имеет ли смысл, чтобы Perl, Python, Java или даже PHP брали данные из базы данных, обрабатывали их как структуры данных Perl, Python, Java или PHP, а затем преобразовывали их в структуры данных JavaScript и отправляли их в приложение на JavaScript, работающее на клиенте?

(Все становится еще забавнее, когда это не настоящая база данных, что бы это ни значило в данном контексте, и это приложение на JavaScript, которое содержит документы JavaScript, но это только подтверждает мой пункт.)

Почему бы вместо этого не запускать JavaScript на сервере, отправлять сериализованные данные JavaScript клиенту и выполнять JavaScript на клиенте? Позвольте программистам использовать один и тот же язык на сервере и на клиенте. В конце концов, на сервере вы должны использовать что-то, а на клиенте у вас есть только один выбор.

Наем программистов JavaScript

Работодатели хотят нанимать программистов JavaScript, потому что у этой технологии нет конкурентов. (Java апплеты не получили широкого распространения, Flash ушел, и если вы не хотите писать один и тот же код три раза для пользователей настольных компьютеров, Android и iOS, то для планшетов, телефонов и других мобильных устройств есть только JavaScript.) Программисты учатся JavaScript частично потому, что у него нет конкурентов, и частично потому, что это всепроникающая программная среда на всех их устройствах.

В отличие от PHP, у JavaScript нет достойных конкурентов. Как и в случае с PHP, нет недостатка желающих разработчиков. В отличие от PHP, отсутствие конкуренции не ухудшило качество доступных разработчиков. Сложность поддержки нескольких браузеров с JavaScript-реализациями сомнительного качества привела к появлению вспомогательных библиотек, таких как jQuery, которые помогают обеспечить соблюдение стандартов кодирования (хотя бы в том, что они часто предоставляют приемлемые системы расширения и стабильное поведение на протяжении всего многообразия реализаций). Поскольку разработчики реализаций JavaScript обычно хотят сохранить обратную совместимость веба, программа на JavaScript, написанная в 1996 году, с большой вероятностью будет работать без изменений в веб-браузере, выпущенном в 2014 году.

JavaScript не столкнулся с таким же явлением "рынка лимонов" при найме, как PHP, вероятно, из-за отсутствия конкуренции. Хороший разработчик JavaScript должен оставаться разработчиком JavaScript или полностью отказаться от разработки на стороне клиента. (Это в некоторой степени может объяснить желание использовать JavaScript на серверной стороне разработки: это альтернативный способ продолжить программировать на JavaScript, избегая работодателей низкого качества, заинтересованных исключительно в разработке на стороне клиента.)

Качество, принятие и победа

PHP и JavaScript стали доминирующими в своих областях похожим образом, но важные детали, обусловившие это, различны. Оба языка позволяют пользователям начать с существующего HTML-документа и постепенно расширять его с помощью программирования. PHP требует установки (хотя можно предположить, что он доступен повсюду, потому что его установка и настройка дешевы и все хотят его), в то время как JavaScript доступен повсюду так же, как веб-браузер доступен повсюду.

Ни одно из этого не касается качества реализации, пользовательского опыта, дизайна или опыта. Техническое качество является второстепенным аспектом в сравнении с соответствием продукта рынку. С технической точки зрения сложно утверждать, что JavaScript или PHP лучше своих конкурентов. Во-первых, оба языка и экосистемы имеют недостатки, которых нет в других языках и экосистемах. Во-вторых, и самое главное, у них почти нет конкуренции.

Оба языка стали успешными, потому что они повсюду, и оба языка продолжают оставаться повсюду потому, что они повсюду.

Если это звучит круговым рассуждением, то это так. Если это звучит обескураживающе для потенциальных конкурентов, то, вероятно, так оно и есть. Единственный вероятный способ вытеснить PHP или JavaScript с позиции доминирования - это предоставить нечто неотъемлемое на новой платформе, которая будет еще более доминирующей, чем существующая платформа. Однако, пока существующая платформа продолжает развиваться - сравните HTML 5 с HTML 2.0 - шансы заменить что-либо из них на что-то лучшее невелики.

Возможно, появится конкурент, который предложит качественное улучшение в производительности, всепроникновенности или простоте поддержки (заметьте, что ни одно из этих качеств не является специфическим техническим аспектом качества дизайна). Однако, пока это не произойдет, веб будет оставаться прикованным к PHP

и JavaScript - и если эта тезис верна, то скорее к последнему, чем к первому.

Вывод, который можно сделать из этого, может быть очень простым, поскольку его выполнение чрезвычайно сложно. Найдите доминирующую платформу, которая все еще развивается. Сделайте себя неотъемлемой частью этой платформы. Продолжайте быть неотъемлемыми.