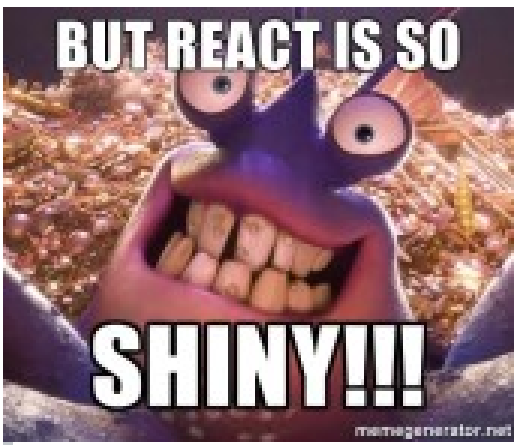
РАЗРАБОТКА ВЕБ-САЙТОВ*, REACTJS*, JAVASCRIPT*, HTML*, CSS*

Пробел в знаниях основ веб-разработки

ПЕРЕВОД

m1rko 3 ноября 2017 в 14:14 47,5k

Оригинал: Kevin Ball



Вчера я разговаривал с другом, который ищет разработчика на открытую вакансию. Он выразил некоторое разочарование, которое я тоже испытываю в последнее время:

> У меня проблемы с поиском фронтенд-разработчика, в основном, по WP, Foundation, CSS, JS, на низкоуровневую позицию. Не могу понять, в чём дело. Ни у кого из кандидатов нет «базовых знаний» ничего из перечисленного. Но они могут делать сайты на React или других JS-фреймворках, или на базе WP-шаблонов. Но если я говорю, что нужно сделать простые изменения в CSS, смотрят пустыми глазами… Или какую-нибудь мелочь на чистом JS, ничего.

Нет недостатка в учебных лагерях, курсах, полно ресурсов для изучения фронтенд-разработки. Но я собеседовал кучу ребят из этих учебных лагерей и думаю, что там серьёзно недооценивают важность CSS и основ JavaScript.

Конечно, есть ограничения на то, сколько можно усвоить за 12 недель обучения. Но огромная часть проблемы в том, что наша

индустрия восхищается новым, одержима самыми последними и прекрасными SPA-фреймворками, в то же время обесценив CSS и «старые» имплементации.

Восхищение новым

Наша индустрия восхищается новыми подходами к разработке. Чем ещё можно объяснить непрерывное стремление выбросить все наработки и «переделать с нуля» каждый раз, когда появляется новый, лучший и более сложный фреймворк? Каждый раз мы утверждаем, что это приведёт к более чистой, простой, идеально абстрагируемой архитектуре, и каждый раз всё заканчивается изобретением велосипедов, воссозданием багов и открытием заново всех пограничных ситуаций, которые снова приводят к такому же безобразному коду.

Это не означает, что никогда ничего не стоит переписывать или что новое никогда не бывает лучшим. Просто мы пали жертвами культа бесплатных вещичек и идеи идеальной абстракции. Каждая новая архитектура идеальна до тех пор, пока не сталкивается с жёсткими условиями реального мира. К сожалению, люди — довольно хаотичные существа, и весь наш софт создан для решения человеческих проблем. Поэтому каждая программа в реальном мире заканчивает дырявыми абстракциями, неуклюжими пограничными ситуациями и новыми компромиссами.

Эта гонка по бесконечной переделке и концентрации только на последнем и самом лучшем часто приводит к отказу от прежних решений тех проблем, которые в итоге появляются заново. Также

она приводит к тому, что мы применяем новые инструменты в абсолютно неподходящих областях, просто потому что они новые.

Одержимость самыми последними и прекрасными SPA-фреймворками

Публикуя список рассылки по фронтенд-разработке, я каждый день вижу эту проблему во время текущего бума SPA. Читаю огромное множество статей, где авторы пишут о разных технологиях, и поверьте, почти все в мире JavaScript пишут об одном или другом из этих фреймворков как будто это абсолютно новая и уникальная инновация. Хотя это отличные инструменты, но каждый из них создан для решения конкретной проблемы. Они основаны на схожем базисе и делают разный выбор в зависимости от задач, для которых оптимизированы.

Можно взять React для примера, потому что он так сильно продвигается в последние несколько лет…

Не поймите меня неправильно, я люблю React. Это феноменально мощный инструмент. Он делает не только возможным, но и простым создание интерфейсов, которые казались нереальными, когда я начинал веб-разработку. Однако новички в индустрии приходят и видят всю эту шумиху вокруг React и предполагают, что это единственная истина, как следует писать на JavaScript. Сделать новое веб-приложение? Используй React! Нестандартный шаблон для блога? React! Переделать старый сайт? Переходи на React!

Это катастрофический подход к использованию технологии! И не слушайте меня, послушайте одного из самых видных разработчиков в сообществе React, Дэна Абрамова! Когда Кори Хаус попросил высказаться о недостатках React, Дэн дал наиболее подробное описание:

> Its performance is worse than templates if many things update extremely fast at the same time. For example stock trading apps.
> — Dan Abramov (@dan_abramov) September 16, 2017

> «Его производительность хуже шаблонов, если много вещей исключительно быстро обновляются одновременно. Например, приложения для биржевой торговли»

> It trades memory pressure for expressiveness. React apps typically do more short-lived allocations under heavy load.
> — Dan Abramov (@dan_abramov) September 16, 2017

> «Ради выразительных возможностей языка приносится в жертву память. Приложения React обычно делают больше краткосрочных размещений в памяти под большой нагрузкой»

> It’s mostly driven by the needs of Facebook. So if your apps are very different from what FB is building, may not satisfy your use case.
> — Dan Abramov (@dan_abramov) September 16, 2017

> «Он в основном спроектирован исходя из потребностей Facebook. Так что если ваши приложения сильно отличаются от

> того, что делает Facebook, то React может не подойти для ваших нужд»

Это только часть, но открытость Дэна подтолкнула Кори к ответу:

> Gotta say, I'm floored that the best input I've gotten on why "not* to use React is from you Dan. Really admire such candor and nuance.
> — Cory House (@housecor) September 17, 2017

> «Должен сказать, я поражён, что лучший список причин **не** использовать React пришёл именно от вас, Дэн. Действительно восхищаюсь столь откровенным и подробным ответом»

Очевидно, у Дэна нет иллюзий о том, что React идеален для всего; он хорошо знаком с компромиссами, на которые пошли разработчики! Но такая большая часть сообщества спешит перейти на SPA-фреймворки для всего подряд и полностью игнорируют факт, что эти инструменты решают конкретные области задач. Да, это феноменальные инструменты и огромное удовольствие с ними работать… но часто они абсолютно не подходят для решения задач в других областях.

Превознося фреймворки над всем остальным, мы не замечаем решений, которые гораздо более подходят для этих областей.

Обесценивание CSS

В индустрии появилась тенденция унижать HTML и CSS как «не настоящую разработку» и нечто низшего уровня. Думаю, это идёт

от того, что логику стали превозносить над графическим/пространственным мышлением… CSS и HTML воплощают иерархические, графические и пространственные отношения, в то время как JavaScript фокусируется в первую очередь на логике.

Но замечательная особенность логических языков/выражений состоит в том, что часто они могут *включать* другие типы отношений… что позволяет выражать пространственные отношения на логическом языке. Однако мы в индустрии часто неправильно интерпретируем такую большую широту возможного выражения как то, что выражение на этом языке строго превосходно.

Это НЕ так!

На самом деле, если посмотреть на примеры из математики и физики, то часто наоборот! В этих областях, если вы начинаете с логической модели, то часто отчаянно пытаетесь найти пространственную или графическую модель, которую можно здесь применить.

Причина в том, что эти пространственные модели зачастую раскрывают гораздо более интуитивные или лаконичные способы представления проблем — и они приводят к важным озарениям и выводам, которые мы затем кропотливо пытаемся перевести обратно в логическую форму.

CSS представляет невероятно мощный фреймворк для выражения графических и пространственных отношений, иногда

исключительно сложных!

Сохранение сложности

Это приводит меня к моему ключевому тезису о разработке ПО — **сохранение сложности**.

В любой решаемой проблеме есть некий присущий уровень сложности, и эта сложность должна быть где-то учтена.

Это всплывает во многих случаях, но в данном примере речь идёт о сложности выражения графических и пространственных отношений. Различные элементы на странице пространственно взаимосвязаны друг с другом *невероятно* сложным образом на разных уровнях, особенно с учётом манипуляций и перемещений этих элементов. Эта сложность должна быть где-то учтена, а в случае CSS браузер делает почти всю работу за вас!

Существует *невероятно много* кода JavaScript, написанного просто потому что разработчик недостаточно хорошо знает CSS.

> Truth. I worked on a project that had 2000 lines of JS to do what position: absolute already does because they didn't understand it
>
> — Sarah Drasner (@sarah_edo) September 16, 2017

> «Это правда. Я работала над проектом, где 2000 строк JS делали то, что уже реализовано в `position: absolute`, просто потому что разработчики не понимали этого»

Что делать?

Я не пытаюсь сказать, что нам не следует использовать или учить людей новейшим и самым лучшим инструментам. SPA-фреймворки вроде React, Angular, Vue и Ember позволяют создавать в вебе невероятно мощные интерфейсы, которые просто были невозможны всего несколько лет назад. Эти инструменты действительно изменили представление о том, что возможно сделать в онлайне.

Но я считаю, что следует искоренить элитизм SPA и вновь подчеркнуть важность фундаментальных знаний и выбора правильного инструментария.

Создатели этих фреймворков редко заявляют, что они хороши для всего подряд, но мы в индустрии настолько их возвысили, что новички полностью пропускают изучение базовых основ и предполагают, что только этот сложный инструментарий — единственный «правильный» способ решения их проблем.

Если все новые разработчики будут с презрением смотреть на CSS, то мы придём к тому, что 2000 строчек JavaScript будут пытаться заново реализовать `position: absolute;`.

Если все новые разработчики решат, что новый код HTML и JavaScript можно писать только через SPA, то мы придём к исключительно перепроектированным, глючным и тормозным блогам, маркетинговым сайтам и всему остальному, что сейчас хорошо реализовано на старых технологиях.

Нам следует серьёзно поговорить о том, какие навыки мы ожидаем от разработчиков в нашей отрасли и чему мы их обучаем. Вполне нормально начать обучение новичка в каком-нибудь учебном лагере, но есть серьёзный разрыв между современными выпускниками этих лагерей и тем, что требует индустрия. Может быть нереально ожидать соответствие этим требованиям после 8-12 недель обучения, но нужно хотя бы направить их по правильному пути.

К учебному плану основ

Я не составлял учебник основ веб-разработки, но несколько вещей определённо включил бы туда. Они перечислены ниже со ссылками на бесплатные и платные ресурсы для более глубокого изучения, но что бы ВЫ включили в список? Напишите в комментариях.

CSS

1. **Базовая блочная модель в CSS** (box model). Это главная из основ, которую вы ОБЯЗАНЫ понять. В качестве бесплатного ресурса можно порекомендовать отличную статью Шея Хоу из его бесплатного курса Learn to code HTML & CSS. С платной стороны, особенно если вы легче воспринимаете аудиовизуальный контент, у моего друга Джеймса Стоуна есть неплохой видеокурс по блочной модели в CSS.
2. **Специфичность CSS**. Это ещё одна важная основа, которую нужно понять, если собираетесь заняться веб-разработкой. Без

прочного понимания специфичности вы непрерывно будете пытаться заставить свой код CSS работать так, как нужно. В Smashing Magazine есть отличный обзор для начала.

3. **Flexbox**. Спецификации Flexbox произвели революцию в методах вёрстки, и вам непременно нужно воспользоваться преимуществами этой технологии. Лучший ресурс по Flexbox, который я видел, — этот справочник от CSS Tricks. Если ищете видеокурс по Flexbox, посмотрите этот на Treehouse или этот на Udemy.
4. **Сетка в CSS**. Хотя это новый стандарт и он не поддерживается старыми браузерами, но CSS Grid — исключительно мощная технология и определённо шаг вперёд в создании макетов. Среди отличных бесплатных ресурсов — руководство по трюкам CSS и Сетка CSS на примерах от Рэйчел Эндрю. С платной стороны, Рэйчел Эндрю — гуру, и определённо стоит посмотреть её курс, а если ищете менее дорогой вариант, то Treehouse предлагает хороший вводный курс.
5. **Базовая архитектура CSS**. Это по-настоящему глубокая тема, и я не хочу слишком в неё углубляться, но для начала я бы прочитал бесплатную книгу SMACSS. Она даст отличное понимание некоторых фундаментальных принципов организации CSS и концепций — оттуда можно перейти к другим подходам и конвенциям вроде BEM, ITCSS и т.д.

JavaScript

1. **Типы и приведения типов**. В JavaScript странные типы. Всё сводится к этому. Они обеспечивают как бы «слабую типизацию» и довольно причудливое поведение. Если хотите

углубиться в JavaScript, то их нужно понять. Лучший ресурс, который мне встречался — это первые четыре главы «Вы не знаете JS: типы и грамматика».

2. **Контексты и `this`**. Понять `this` фундаментально важно для более глубокого понимания, что происходит в коде JavaScript. Непонимание `this` — причина путаницы № 1 у начинающих программистов JavaScript, частично потому что многие библиотеки используют и злоупотребляют ею (я смотрю на тебя, jQuery!). Для создания фундамента в понимании рекомендую великолепную статью Зелла «*Это* в JavaScript».
3. **Замыкания**. Замыкания откроют вам все типы концепций и покажут, почему практически одинаковый код настолько разно работает, и вы начнёте понемногу представлять функции как объекты первого класса. Вот хорошо написанное введение от Free Code Camp: «Изучим замыкания в JavaScript».
4. **События и цикл событий**. Модель цикла событий отличает JavaScript от многих других парадигм программирования, и без хорошего понимания у вас могут быть серьёзные неприятности, когда впервые начнёте работать с нетривиальными взаимодействиями между асинхронными событиями. Я не нашёл ресурс, который был бы мне абсолютно по душе на эту тему (дайте знать, если вам известен такой), но эта статья Carbon Five неплоха: «Объяснение цикла событий JavaScript».

Комментарии (228)

Похожие публикации

FrontFest.Kvartirniki — говорим о будущем JavaScript и судьбе фронтенд-разработчика 0

tth • 2 ноября 2017 в 11:22

FrontFest.Vyorstka: поговорим о доступности, экономии трафика и будущем CSS 17

tth • 10 октября 2017 в 09:48

5 практических примеров для изучения фреймворка React 46

ПЕРЕВОД

jojo97 • 13 июля 2014 в 03:37

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на русском языке 14

alexkuku • вчера в 16:12

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада

4

lahmatiy • вчера в 13:05

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе

25

ПЕРЕВОД

Smileek • вчера в 10:32

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации

2

ПЕРЕВОД

ru_vds • вчера в 12:04

Как адаптировать игру на Unity под iPhone X к апрелю

0

P1CACHU • вчера в 16:13

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

33

HostingManager • вчера в 13:49

Обзор рынка моноколес 2018

70

lozga • вчера в 06:58

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

40

alizar • вчера в 15:14

Стивен Хокинг и его работа — что дал ученый человечеству?

8

marks • вчера в 14:46

Sunlike — светодиодный свет нового поколения

17

AlexeyNadezhin • вчера в 20:32