Профессия front-end архитектор

Эволюция веб-дизайна как профессии

Показать графически взаимосвязь визуального, технического и интерактивного дизайна

В начале было слово.

Потом появился интернет.

Интернет состоял из слов и был создан для того, чтобы одни слова ссылались на другие слова.

Но потом стало возможно использовать изображения…

Белый экран, большой пустой квадрат, надпись spacer.gif, отъезжаем и заполняем экран пустыми квадратиками. Их очень много — они образуют фон. На этом фоне появляется надпись:

...и началась эра веб-дизайнеров.

Дизайнер-ремесленник

Фокусируясь в первую очередь на том, как тот или иной сайт *выглядит* дизайнеры часто забывают о *целях и задачах*, на которые должен быть *ориентирован веб-сайт*. О том, что *сайтом должны пользоваться люди*, чтобы *искать и находить* ту или иную информацию, *делать покупки*, *читать новости*, *участвовать в обсуждениях* и многое другое.

Стремление дизайнеров быть *оригинальми* зачастую приводит к необычным результатам.

Показать несколько скриншотов аляповатых сайтов.

Дизайн ради дизайна, а не ради пользователей, не ради их целей и задач, не ради их *удовлетворенности продуктом или услугой*.

Рассказать про генерирование идей для дизайна в виде набора “10 фишек”.

> *Работая в свое время в достаточно известной дизайн-студии, у нас было правило, которому следовали дизайнеры перед тем как начать работу над дизайн-макетом. Правило это заключалось в том, что дизайнеры “генерировали” идеи визуальных “фишек” для каждого макета. Каждая фишка — это необычный элемент или графический прием. Конечно же, никакие реальные задачи в этот момент не учитывались. Это был чистой воды “креатив”, решавший лишь одну проблему — работы студии хоть как-то должны были отличаться друг от друга.*

Почему так происходит, спросите вы? Очень часто дизайнеры начинают думать, что результат их работы — самый важный при создании ваб-сайта. Однако это не так. Графический дизайн — лишь один из аспектов веб-разработки. Такой же, как, например, юзабилити, доступность использования, семантика и контент.

Кроме того, многие "дизайнеры-ремесленники" не пишут код разметки страниц останавливаясь лишь на визуальном аспекте работы. Поэтому после создания макетов, дизайнер, как правило, выводится из проекта, а реализацией веб-сайта по этим макетам занимаются другие люди — HTML-кодеры, зачастую имеющие слабое представление об *эстетике, удобстве использования, доступности*. Дизайнер *теряет контроль* над конечным продуктом.

Показать сравнение макета и скриншота получившегося веб-сайта.

Можно относиться к дизайну *как к искусству*, но это имеет мало общего с *реальным миром* и его *потребностями*. Добавьте сюда *низкое качество кода* и станет понятно, *как мы оказались там, где мы оказались* — каша, сваренная из HTML тэгов, отсутствие семантики, огромные объемы нечитаемого кода.

Веб-стандартист

Поэтому, когда в сети появились люди, называющие себя "веб-стандартисты" и выступающие за использование таких принципов, как

Фотография Джефри Зельдмана в шапке :) Все последующее он как бы произносит — показавыем элементы списка в балуне.

- создание семантического и валидного кода,
- разделение представления и содержания,
- использование безтабличной верстки (и использования таблиц исключительно по-назначению),

веб-сообщество приняло эти принципы как реальное *решение накопившихся проблем*.

Многие дизайнеры заявляют, что *веб-стандарты*, не являясь *выразительным средством коммуникации*, *мешают их творческой работе*.

Однако, *важно понимать, что графический дизайн никак не зависит от веб-стандартов.*

> *Например, если думать о Вебе как о мировом океане, то можно увидеть лишь то, что он синий, плоский и покрывает большую часть нашей планеты. Однако, что для океана, что для веба, под видимой их поверхностью находится богатая экосистема, требующая своих специфических потребностей и ограничений. Учитывать эти потребности и ограничения — задача дизайнера.*

Две картинки — сначала "плоский" синий океан. Потом подводная фотография с рыбам, коралами и т.д.

Выверенный и хорошо выполненный *графический дизайн действует* на пользователей лишь через *визуальный канал*. Однако *семантическая разметка*, на которую "наложена" *графическая составляющая*, будет также взаимодействовать с важными *технологиями и механизмами Веба*, а через них, с еще большим количеством людей, которые их используют.

Создать уникальный графический дизайн и реализвать его, используя веб-стандарты, — это реальная задача.

Кроме того, хорошая разметка значительно ускоряет время разработки и отладки веб-приложений. Что, согласитесь, в текущей рыночной ситуации, является неоспоримым преимуществом.

> *Когда я только перешел на свою текущую работу, мы как раз начали разработку крупного социального проекта. Кроме всего прочего, я писал HTML/CSS код для этого проекта. Когда началась реализация проекта, серверные разработчики сначала были удивлены, увидев семантический код без использования таблиц. Но когда они начали интегрировать его с приложением, их восторгу не было предела — настолько просто было "оборачивать" такой код сверверными логикой и создавать из него модульные компоненты. И то лишнее время, которое я потратил на создание хорошего кода, не раз окупилось позже.*

Время front-end архитекторов

Сегодняшняя веб-разработка немыслима без использования разнообразных серверных фреймворков, таких как *Rails, .Net,* их Java-аналогов, которые призваны упрощать и ускорять разработку и отладку веб-приложений. Усложняется логика работы приложений. Растут объемы данных, от которых зависят и с которыми работают приложения. Все это сделало стандартом де-факто наличие в больших проектах системного архитектора.

Професиия системного архитектора не нова и такие люди есть во многих средних и крупных проектах. Как правило, это человек, владеющий большим багажом знаний и опыта, представляющий, как используемые технологии интегрированы и взаимодействуют друг с другом, способный решать проблемы высокого уровня: архитектура БД, интеграция со сторонними компонентами и многое другое. Архитектор оценивает ситуацию в целом, помогая команде принимать правильные интеграционные и технологические решения.

Серверные фреймворки *предполагают*, что разработчик должен больше *думать о логике приложения*, о *красоте* своего *кода*, его *объектной модели*, нежели о деталях реализации под различными браузерами и платформами. Но результат работы фреймворка не всегда идеален и зачастую должен быть вовремя откорректирован.

> *Взять, к примеру, ASP.Net. Идея фреймворка хороша с позиции разработчика — этот фреймворк заботится о многих аспектах приложения: взаимосвязи клиентских и серверных контролов и многом другом. Но, используя этот инструмент, приложение может оказаться недоступным многим пользователям из-за использования JavaScript для обработки и отправки данных на сервер (а если JavaScript у пользователя отключен?), смешения логики и презентации, использования серверных ID для элементов (вида "ctl01_blah-blah-blah") и, как следствие, невозможности использовать "нормальные" IDs (например, "header", "content" и т.д.) при создании HTML/CSS кода.*

Вместе с *развитием серверных фреймворков*, становятся все *более сложными* и применяемые *фронт-енд технологии: Ajax и динамическое изменение содержание страниц, drag'n'drop, различные визуальные эффекты, сложные элементы пользовательского интерфейса*. Добавьте сюда такие понятия, как *user experience, usability и accessibility*, которые также приобрели большое значение в последнее время, и станет понятно, что современные задачи уже *не могут быть решены* лишь *с помощью HTML и CSS*, использованием *безтабличной верстки* и умением писать *валидный код*. Это был лишь первый шаг. Пришло время *front-end архитектуры*.

Front-end архитектура

Учитывая то количество различных технологий и способов решения тех или иных задач на фронт-енде, очень часто бывает сложно принять решение, какой из этих способов следует использовать в данном конкретном случае.

Например, использовать CSS или JavaScript для создания выпадающих меню? Использовать текстовые, графические или sIFR загловки? Каждый из способов имеет свои плюсы и минусы, которые могут иметь разное значение в зависимости от конкретной ситуации. Необходимо знать не только слабые и сильные стороны технологии, но и владеть этой ситуацией.

Для принятия правильного решения требуется человек, способный оценить ситуацию в целом, учитывая большое количество факторов: удобство использования, accessibility, серверную реализацию, цели пользователей, бизнес-задачи и многое другое.

Вот несколько примеров:

Разметка страниц

Основой фронт-енда является HTML-разметка страниц. Под разметку пишется CSS, JavaScript, серверный код. В зависимости от целей и задач, разметка может быть сделана разными способами.

Какой doctype стоит использовать? Каким (x)HTML элементом кодировать тот или иной блок на странице? Использовать атрибут id или class? Это зависит от того, какой серверный

фреймворк будет использоваться, какая функционаьность будет на страницах, потребуется ли менять внешний вид одних и тех же страниц?

Например, если используется ASP.Net, то лучше использовать атрибут class, нежели ID, так как этот фреймворк использует свою динамическое присвоения элементам ID. Если потребуется создавать "темы" для настройки внешнего вида страниц, то лучше сделать код немного более насыщенным (использовать дополнительно div и span), нежели привязывать CSS непосредственно к структуре HTML документа.

CSS

Отвечая за то, как страницы выглядят, CSS является презентационным уровнем во фронт-енд модели приложения.

Как организовать CSS документы? Использовать conditional comments или CSS-хаки? Создавать дополнительные имена классов или привязать внешний вид непосредственно к элементам DOM? Использовать наследование или писать дублирующий код? И опять нет однозначного ответа.

В большом проекте напследование может сильно попортить нервы, если потребуется изменить представление какого-то блока. Привязка к структуре документа тоже может оказаться проигрышной стратегией, если разметка вдруг измениться. CSS-хаки могут перестать работать в следующей версии браузера и т.д.

JavaScript

JavaScript и DOM scripting в частности — это логика фронт-енд приложения. С появлением JavaScript фреймворков, этот язык получил очень большое распространение. При правильном использовании это очень мощное средство создания насыщенных пользовательских интерфейсов. При неправильном — причина многих проблем.

Использовать onclick или "unobtrusive" обработчиков событий для элементов? Реализовать валидацию на стороне клиента или сервера? Назначать стиль отображения inline через JavaScript или использовать className? Организовать код через namespaces или использовать global scope?

Любое решение требует ответа на ряд вопросов: что решает эта функциональность, как именно она должна работать, поддерживают ли данную реализацию необходимые браузеры? Кроме этого, существуют определенные практики, знание и владение которыми позволит избежать множества проблем при использовании JavaScript.

Ajax

С популяризацией Ajax эта технология заполонила веб-сайты по поводу и без. В некоторых случаях использование ее оправдано, в некоторых — нет. Зависит от ситуации.

Использовать Ajax или стандартный механизм с обновлением страницы? Какой из этих подходов наиболее удобен для пользователей в контексте решаемой задачи? Как это

отразится на доступности приложения? Смогут ли использовать эту функциональность пользователи мобильных устройств?

Ведь если у пользователя отключен JavaScript или пользователь работает с сайтом на мобильном устройстве, то Ajax функциональность перестанет работать и пользователь не сможет решить свои задачи. Конечно, этот недостаток можно обойти, но это потребует дополнительных усилий. К тому же очень немного людей задумываются о нем в момент реализации.

И снова требуется кто-то, кто сможет дать рекомендации по использованию, кто сможет принять решение о необходимости Ajax в конкретном случае.

Кто же это? Это — фронт-енд архитектор.

> *Когда меня спрашивают, кем я работаю, обычно я отвечаю — "фронт-енд архитектором". Но это не отвечает вопрос, чем я занимаюсь. Чтобы объяснить людям, я говорю: "я занимаюсь всем тем, с чем взаимодейтсвуют люди, глядя на экран компьютера — как это работает, как это выглядит, что это значит и помогает ли это достичь их целей удобным и быстрым способом". Список не полный, но он дает представление о моей деятельности. ;)*

Какими *знаниями и навыками* должен обладать фронт-енд архитектор, чтобы *успешно решать эти задачи*?

- XHTML & CSS
- Кросс-браузерная и кросс-платформенная совместимость
- JavaScript разработка (DOM scripting, Ajax, UI)
- Flash и ActionScript
- Progressive Enhancement and Graceful Degradation
- Доступность использования (Accessibility)
- Удобство использования (Usability)
- Информационная архитектура
- Дизайн пользовательских интерфейсов
- Визуальный дизайн
- Логика генерации страниц (ASPX, Rails Views, etc.)
- Бизнес-логика

Фронт-енд архитектор должен обладать знаниями во всех перечисленных областях.

Кроме всех вышеперечисленных знаний и навыков, фронт-енд архитектор обязательно *должен уметь общаться на языке разработчиков и уметь принимать критически-важные интеграционные решения*. Написать *хороший клиентский код* — это *часть задачи*. Необходимо, чтобы этот код начал *взаимодействовать* со своей *серверной частью* в *реальных условиях*. Без навыков сделать это практически невозможно. Ведь *требуется учитывать специфику серверного приложения*.

Фронт-енд архитектор, как и системный архитектор, должен владеть ситуацией на высоком уровне, уметь объективно и взвешенно оценить преимущества и недостатки того или иного подхода, руководствуясь множеством факторов. Умение писать клиентский (HTML, CSS, JavaScript) код, проектировать удобные пользовательские интерфейсы, владеть современными технологиями и следить за их развитием — это не опциональные, а обязательные качества специалиста.

Будущее профессии

Пусть этим занимаются системные архитекторы?

Системные архитекторы, как правило, полностью поглощены техническими аспектами разработки и имеют крайне мало времени для обучения специфике фронт-енд технологий. И даже если они имеют необходимые знания, найдется ли у них время на решение вопросов разработки фронт-енда?

Когда такие профессионалы будут востребованы?

Вчера! Специалисты такого плана требуются рынку *уже не первый год*. Но лишь не так давно это *стало очевидным фактом*. В настоящее время российский рынок тоже начинает переживать бум веб 2.0 разработки — появляются *аналоги западных веб 2.0 сервисов*, создаются *новые проекты*, *"социальный аспект"* начинают хотеть большие компании, работающие *на рынке электронной коммерции*.

Обязательно ли это связано с *интернетом* и *"социальными" или "веб2.0" сервисами*?

Отнюдь! Веб технологии активно и очень широко *применяются* в заказной *разработке ПО* благодаря более *простой интеграции* в существующию инфраструктуру, более *низкой себестоимости разработки и поддержки*. Не стоит забывать и про *мобильные платформы*, *разработка* под которые также ведется очень активно.

Российский рынок труда.

Российский рынок веб-разработки сильно отличается от западного. Не секрет, что мы находнимся позади и пытаемся "догнать" западный уровень. Это отставание привело в частности и к тому, что многие *компании не доверяют многопрофильным специалистам* и предпочитают *нанимать на работу узконаправленных* фронт-енд *специалистов* — графических дизайнеров, кодировщиков HTML/CSS, JavaScript разработчиков. Но даже будучи очень хорошими в своих областях, они *не смогут решить 100% возникающих проблем*. Поэтому, мне кажется, что спрос на фронт-енд архитекторов в России *появится в ближайшее время* — ведь *кто-то должен будет управлять узкопрофильными специалистами, координировать их действия, отвечая за конечный результат*?

В настоящий момент *явного* спроса на фронт-енд архитекторов в России *не существует*. Связано это с несколькими факторами:

- **Первый** — это сетереотип, что "многостаночник" не может хорошо владеть всеми заявленными знаниями. Другими словами — знает много, а делать умеет мало. Одна из целей моего выступления — сломать этот стереотип. :)
- **Второй** — относительная дороговизна таких сотрудников по сравнению с узкопрофильными кадрами. Но ведь системные архитекторы — дорогостоящие кадры, и это никого не смущает. :)
- **Третий** — пока еще в России не уделяется должного внимания таким вопросам, как удобство использования, accessibility (в то время как в некоторых странах есть законы, обязывающие делать веб-сайты в соответствии со стандартами), user experience и др. Но как показывает время, ситуация эта быстро меняется. Взять, к примеру, тех же юзабелистов. Рост рынка юзабилити в России за последние год или два очень большой.

Итог

Подводя итог, хочется сказать, что современному фронт-енду *требуются архитекторы* также, как серверной разработке требуется люди, которые решают проблемы *правильного разделения уровня баз данных, бизнес-модели и презентации*. И *чем более сложные приложения* будут *разрабатываться* с использованием *веба как платформы*, тем больше эта *потребность будет расти*. Адекватно *отреагировать на* эти *изменения рынка — задача* сегодняшних узкопрофильных *специалистов*.

Спасибо! :)