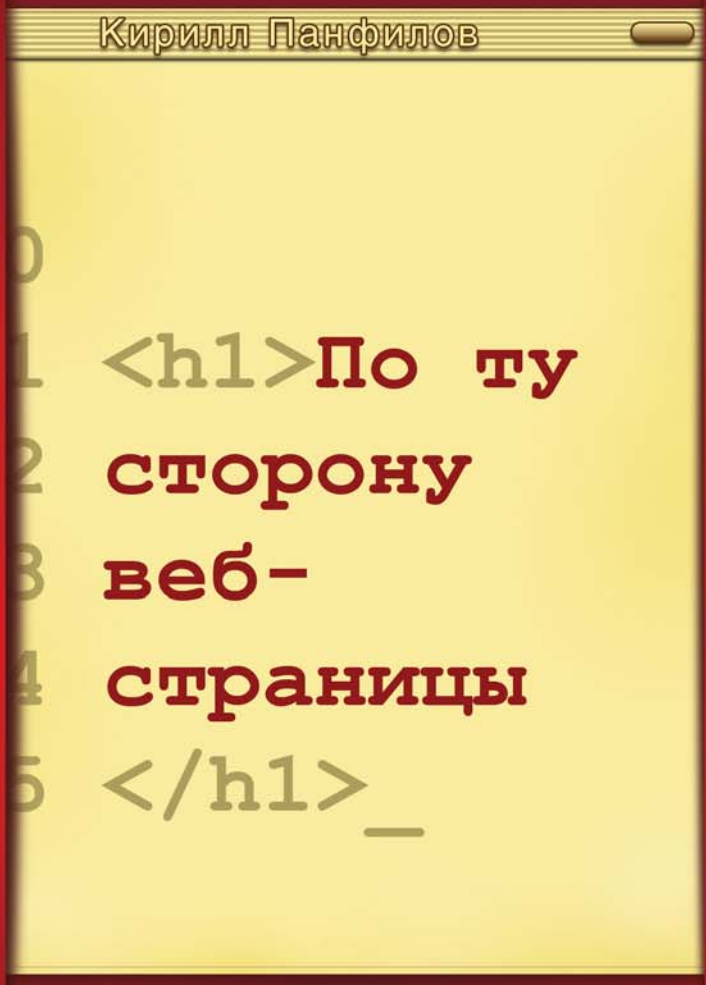
Кирилл Панфилов



ДМК ПРЕСС
издательство

УДК 004.4
ББК 32.973.26-018.2

Панфилов К.
П16 По ту сторону веб-страницы. – М.: ДМК Пресс, 2008. – 440 с.: ил.

ISBN 5-94074-392-7

Как создать сайт в одиночку, не упустив ни одной важной детали? Или, по крайней мере, проконтролировать весь процесс? На что нужно обращать внимание, а чего, наоборот, нужно остерегаться? Книга рассказывает о процессе разработки сайтов от замысла и проектирования до создания дизайна и кода. Особое внимание уделено проблемам современного веб-дизайна и нестандартным путям их решения.

Почему 95% дизайнеров рисуют похожие сайты? Потому что не хватает фантазии или знаний? Или потому что так предписывает Нильсен? Но существуют типы верстки, совершенно не похожие на привычные, но не менее удобные.

Что такое «мода в веб-дизайне»? Так ли необходимо ей следовать? И всегда ли нужно использовать инструменты, которые используют все? Не всегда для хранения информации удобны базы данных, и не для всех типов графики идеально подходит Adobe Photoshop.

А знаете ли вы, что языки серверного программирования — это пластилин, из которого можно лепить любые фигуры, были бы умения и фантазия. Создатели этих языков сами не догадывались, какие возможности заложены в их детища. В книге рассказывается о небольшой части этих возможностей. Знания языков HTML, XHTML, CSS, JavaScript, PHP и технологий SSI и RSS (их описания есть в книге) недостаточно: в первую очередь нужно понимать, что вы собираетесь сделать, а инструмент найти — дело второе.

Книга — по большому счету — об одном. Прежде чем сделать сайт, нужно подумать, зачем он нужен и как его задачи соотносятся с его внешним видом и его работоспособностью.

УДК 004.4
ББК 32.973.26-018.2

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 5-94074-392-7

© Панфилов К., 2007
© Оформление, ДМК Пресс, 2008

Содержание

0.
Вводная часть

Если уже есть десятки отличных и сотни других книг по веб-дизайну, то в этой должно быть что-то особенное.

Например, нет смысла переписывать в этой книге все атрибуты тэгов языка HTML, превращая его исключительно в справочник. Рассказывать, как в фотошопе правильно сделать имитацию трехмерного объекта для веб-страницы и как сделать так, чтобы изображение без потери качества и размера стало «весить» в 7 раз меньше.

А стоит рассказать о том, почему в российском веб-дизайне все больше сайтов строится на принципах «тянущейся» верстки, а западные до сих пор предпочитают фиксированные размеры. Рассказать, что плохого в использовании тэга `<div>` и базы данных MySQL при всех их неоспоримых достоинствах. Поведать, какие достойные альтернативы есть у баз данных и у модной технологии AJAX. И что в целом является модным, что ушло в прошлое, а на чем строится будущее веб-дизайна.

Еще очень важно рассказать о том, что сначала нужно знать, зачем создается сайт, а уж затем создавать его. Эта очевидная истина не принимается во внимание при создании четверти крупных сайтов и девяноста процентов самодельных. А при создании сайта нужно не только представлять, каким он будет, но и отбрасывать большую часть стереотипов и высвобождать свою фантазию. Потому что «так принято» не значит «лучше всего».

Книга — о современном положении дел в веб-дизайне, о том, как решить некоторые его проблемы, и о том, как сделать по-настоящему хороший и необычный сайт: в погоне за стандартами и удобством веб-дизайнеры забывают об эстетической составляющей.

Книга также о том, как сделать сайт не только необычным, но и удобным: в погоне за эстетической составляющей дизайнеры часто забывают о будущем посетителе.

Книга — не только сборник рекомендаций и наблюдений автора, но и краткий справочник. Например, в ней можно найти описания

HTML, XHTML, CSS, PHP и JavaScript, которыми вполне можно пользоваться. Кроме того, описаны технологии SSI, RSS и Atom. Для удобства в книге приводится не только оглавление, но и так называемый «Навигатор», или тематический указатель.

0.1. Краткая история интернета

С течением времени менялась форма передачи данных по сетям интернета, но не менялась суть самого интернета. Сети как таковые были разработаны и созданы для обмена данными между различными компьютерами (логичнее было бы сказать: между людьми, сидящими за разными компьютерами, но в демагогию и казуистику мы ударяться не будем), и с тех пор мало что изменилось. Письма по электронной почте, ночное общение в «аське», торренты, загрузка веб-страниц, закачивание пиратских (и не только) версий программ и аудиофайлов на свой компьютер, проверка лицензионности Windows посредством веб-сайта, оповещение о новых заказах в интернет-магазине, ролик любимого исполнителя на сайте, обновление программы с официального сайта — все это передача данных по сетям. Но это современное положение дел.

В 1991 году был разработан язык HTML, без использования которого или потомков которого не обходится и еще долго не будет обходиться ни одна веб-страница. Даже если в основу дизайна сайта положен флэш, ролики как-то нужно включать на страницы: для этого используется HTML. Какими бы яростными ни были войны между последователями верстки на таблицах и верстки на блоках, оба типа верстки — это HTML. Он является основой для любой страницы, размещенной в веб-пространстве, за исключением страниц, построенных на XHTML, основной для которого все равно послужил HTML. И даже если данные для страницы записаны в формате XML или вообще в базе данных, на страницу они все равно выводятся в формате HTML.

С 1993 года, когда был разработан HTML 1.2, до настоящего времени прошла целая эпоха в том плане, что принципы верстки сильно изменились. Первые версии языка позволяли осуществлять только логическую разметку, едва позволяя касаться внешнего оформления. Обычно внешний вид отдавался на откуп браузерам: с самого начала были выработаны рекомендации по внешнему отображению логических элементов веб-страниц, большинства из которых производители браузеров придерживаются и до сих пор. Но если в самом начале выбора у веб-мастеров не оставалось, и им приходилось довольствоваться

тем, что дают, то в 1995 году ситуация изменилась с разработкой третьей версии языка HTML, в которую было включено большое количество тэгов визуального форматирования. Концепция HTML была нарушена: логический и визуальный код представляли такой сплав, который было трудно разделить. Но именно в это время появляются спецификации CSS и их зачаточная поддержка браузерами. Поскольку поддержка была именно зачаточной, никто о чистоте кода и не думал, и все продолжали верстать текст с помощью тэгов `<font>`, а уж о точном позиционировании элементов мало кто думал (разве что приверженцы Netscape). И только в последние несколько лет началась широкомасштабная, поглощающая ресурсы и истощающая силы борьба за чистый код. Дело в том, что CSS позволяет полностью контролировать внешний вид почти любого элемента (за мелкими исключениями), а значит, больше не нужны громоздкие конструкции вроде `<font color="red" size="+4" face="Tahoma, Arial, Helvetica, Sans-Serif"><b><i><u>Текст</u></i></b></font>`, когда то же самое можно записать в стилевом определении один раз, а потом повторять бесконечно, применяя, например, к тэгу `<span>`.

Итак, происходит возврат к логической разметке (правда, к этому пришли еще не все кодеры, да и придут нескоро, потому что, чтобы сверстать страницу нестандартно, все равно приходится прибегать к ухищрениям), но уже с применением дополнительной технологии. Но меняется не только это. За более чем 15 лет развития веб-дизайна после долгой и упорной работы производителей браузеров, наконец, прочно вошла в жизнь объектная модель документов и хорошая поддержка активных сценариев, что позволяет управлять внешним видом страницы в режиме реального времени, перемещать фрагменты страницы, скрывать их и создавать «на лету», производить проверки, выводить подсказки и совершать иные действия — проявлять реакцию на действия пользователя.

Получили мощное развитие и популярность благодаря очевидным удобствам серверные сценарии на языках Perl, PHP, ASP и некоторых других. Массу действий, которые приходилось выполнять вручную, теперь можно поручить сценариям. А в последние годы популярность приобрела и смешанная серверно-клиентская технология AJAX.

Буйным цветом цветет на страницах мультимедиа: кроме звука, фонового или полезного, веб-дизайнеры получили возможность включать на веб-страницы видео и анимацию в различных форматах (включая Flash). Что уж говорить об изображениях.

Часть технологий уходит в прошлое, едва появившись, часть остается надолго. Основой любой веб-страницы все же остается текст, и к нему нужно проявлять особое внимание.

Факторы, позволяющие внешней стороне веб-дизайна развиваться, — это люди и браузеры. Как правило, чем больше есть, тем больше хочется. Как только появляется новая возможность, появляются новые

потребности, капризы и желания. Когда-то мы могли передавать по интернету только текст. Потом — графику. Потом — музыку, видео, архивы, программы, целые сайты (скачанные для оффлайнового просмотра), базы данных, сверстанные полосы газет и журналов... В последнее время появились онлайновые редакторы, по функциональным возможностям приближающиеся к полноценным офисным пакетам. Уже давно существуют сервисы для рисования в режиме онлайн. Появилось огромное количество сервисов для хранения своих файлов: бесплатно предоставляется до пяти гигабайтов дискового пространства.

С развитием технологий появляется больше возможностей. Объемы жестких дисков растут (минимумом сейчас считается 40 Гб), оперативная память все больше и больше (гигабайтом никого не удивишь), мощности процессоров также нарастают, а широкополосные интернет-каналы проникают и в небольшие города и поселки. Это не может сказываться и на требованиях, которые предъявляют люди к сайтом. Формат «просто размеченный текст» потерял актуальность.

0.2. Инструментарий

Языками разметки и фотошопом веб-дизайн не ограничивается. Если коротко резюмировать все основные достижения человечества на ниве веб-разработок, то получится список из нескольких групп.

1. Языки разметки: HTML, XHTML, XML, CSS и др. Это единственная категория, которую при создании веб-страниц нельзя обойти в принципе. Стоит отметить, что, если HTML одновременно отвечает и за содержание, и за оформление, то современная тенденция — корректное использование (X)HTML и XML для передачи семантики веб-страницы, а CSS — для оформления.
2. Языки активных сценариев, исполняемые на стороне сервера, в самом браузере, а результат выполнения выводящие на экран: JavaScript, JScript (все вместе и в ногу со временем — ECMAScript), VBScript.
3. Языки серверных сценариев, код которых выполняется на сервере (и не виден конечному пользователю), а результат формирует окончательную страницу и выдает в браузер посетителя: PHP, Perl, ASP, JSP, Parser, ColdFusion и другие. Собственно, ЯСС не работают без специализированных серверов, например, Apache и IIS. Часто они работают в сочетании с базами данных — изначально упорядоченными массивами текстовой информации (MySQL, PostgreSQL, SQLite, Oracle, MSSQL, Firebird и др.), для обращения к которым используются разновидности языка SQL.

4. Смешанная клиентско-серверная технология AJAX, при использовании которой клиент (браузер) и сервер обмениваются данными без перезагрузки страницы (в отличие от языков серверных сценариев), что ускоряет работу.
5. Встраиваемая растровая графика. Современные браузеры воспринимают изображения в форматах JPG, GIF и PNG. Поддержка формата BMP, похоже, ушла в прошлое. Сочетание (X)HTML- и CSS-разметок с графикой и составляет основу верстки и дизайна большинства страниц.
6. Технология Flash (+ язык ActionScript). Используется как для анимированных векторных графических вставок в отдельных фрагментах веб-страниц, так и для построения целых сайтов. (Когда готовилась книга, компания Microsoft выпустила функционально подобную технологию Silverlight.) Собственно векторная графика на сайтах (например, формат SVG) поддерживается ограниченно.
7. Технология апплетов Java. Была популярной, поскольку позволяла создавать интересные визуальные эффекты. Ныне используется редко, например, для интерактивных карт.
8. Программы, исполняемые на сервере в помощь серверным языкам. Например, модули сервера для выполнения сценариев по времени (cron). Или программа по обработке графики, которая делает уменьшенные копии изображений по запросу сценария. **Примечание.** Многие их этих технологий имеют точки пересечения. Так, с базами данных можно работать не только с помощью языков серверных сценариев, но и с помощью AJAX. И серверные, и клиентские языки сценариев позволяют использовать регулярные выражения. Структурой документа, создаваемой с помощью языков разметки, можно управлять при помощи языков активных сценариев. Растровую графику могут включать в себя джава-апплеты и векторные флэш-ролики. Ряд примеров можно продолжать...

Любая из этих технологий — это медаль с двумя сторонами. Можно ужасно писать HTML-код. Заставлять при помощи активных сценариев текст моргать и прыгать по странице. С помощью серверных языков делать спам-рассылки. Ставить на страницы картинки, оптимизированные скорее для печати в глянцевом журнале (по весу), и побольше Java-апплетов, которые в половине браузеров не будут загружаться. Делать сайт целиком на AJAX'е, чтобы навигация по нему была ужасной. Флэш-ролики (только последней версии Flash) делать не меньше двух мегабайтов только ради того, чтобы по экрану изредка проезжал автомобиль. Наконец, ставить на компьютер посетителя вредоносные программы.

Но если их использовать по необходимости, по назначению и с умом, то получаются шедевры — не только по внешнему виду, но и по степени удобства сайта как инструмента.

Для каждой из этих групп технологий, в свою очередь, существуют целые наборы программного обеспечения — и очень хорошие программы, и очень плохие, и те, что выполняют очень хорошо только определенные задачи.

Для написания кода существует, наверное, наибольшее количество редакторов. Среди общеизвестных монстров — Adobe GoLive, Microsoft FrontPage, Macromedia Dreamweaver и Homesite (все коммерческого типа). Общий их плюс для начинающих творцов и одновременно общий минус в том, что они автоматически генерируют код, являясь редакторами визуального типа (или типа WYSIWYG, сокращение от «What you see is what you get» — «Что видишь, то и получаешь»). Текстовые блоки, иллюстрации и иные компоненты веб-страницы можно перетаскивать по рабочей области, работая, как в графическом редакторе. Как результат — автоматически генерируется код, чаще всего пригодный для очень ограниченного набора браузеров (так, сложная верстка в Microsoft FrontPage позволяет генерировать код, корректно обрабатываемый только браузером Microsoft же Internet Explorer). Кроме того, чаще получается только один тип верстки — жесткий, или фиксированный, а это далеко не всегда приемлемо.

Для профессионального веб-дизайнера намного важнее, чтобы программа позволяла писать код вручную, выполняя только рутинные и вспомогательные операции, например, автозамену, подсветку разных компонентов кода разными цветами, нумерацию строк кода, свертку блоков, отладку сценариев, сообщения об ошибках и т. п. В этом случае кодер получает намного больше контроля за конечным результатом, поскольку программа не «решает» за него. Отличие таких редакторов «блокнотного типа» от визуальных не только в том, что основной компонент программы — это поле для ввода текста, но и в том, что в них нет встроенного интерпретатора кода. Например, Homesite и Dreamwever генерируют внешний вид страницы, основываясь на внутренних интерпретаторах (средствах отрисовки страниц), а текстовый редактор EditPlus Text Editor (тоже коммерческий) при установке в систему позволяет открывать написанные страницы в тех браузерах, которые пользователь укажет в настройках.

Общее в таких программах — ориентированность на работу с кодом и веб-компонентами, будь то тэги языка HTML или визуально отображение кнопок отправки запроса, и масса вспомогательных инструментов — кроме уже упоминавшихся подсветки синтаксиса различных языков разметки и программирования и отладчиков это средства работы с протоколом FTP для загрузки файлов на сервер без файл-менеджера, палитры цветов, средства работы с набором страниц как с отдельным проектом и т.п. Различий больше. Визуальный и текстовый типы — это базовое разделение (хотя все визуальные редакторы позволяют редактировать код и вручную, а есть текстовые редакторы с зача-

точными средствами визуального редактирования). Остальные различия кроются в наборах функций.

Так, упоминавшийся EditPlus (его я считаю самым удобным, но это субъективно) позволяет, не выходя из программы, создавать и редактировать системные файлы для работы с любым языком программирования, даже созданным по ходу дела. AceHTML обладает встроенными справочниками по основным языкам разметки и сценариев и по технологии серверных вставок SSI, а также позволяет работать не только с внутренним интерпретатором для отрисовки результата выполнения кода, но и с внешними браузерами, и хранит множество DHTML-заготовок. Cool Page обладает большим количеством встроенных шаблонов. А вот захотите ли вы пользоваться симпатичным и простым, но функциональным Kiss HTML Editor'ом, если узнаете, что его название никак не связано с романтикой поцелуев, а расшифровывается как Keep It Simple Stupid («Делай проще, глупый»)? Из бесплатных популярных редакторов текстового и смешанного типа стоит также отметить CoffeCup Free HTML Editor («сайт за чашкой кофе») и 1st Page («первая страница»). Из менее известных визуальных редакторов можно отметить, например, Namu6, PersonalWebKit, Web Office и MoreMotion Web Express (по описанию — редактор, позволяющий разрабатывать сайты и их страницы даже тем пользователям, которые не знакомы с HTML, что уже нонсенс). Вред и польза таких редакторов в том, что они позволяют делать продукты разной степени полезности без знания собственно механизмов создания этих продуктов. Вред — потому что из поля зрения уплывает множество важных факторов, а польза — в возможности не тратиться на дорогих и циничных веб-дизайнеров при необходимости быстро разместить нужную информацию в сети. Шаблоны веб-страниц в этих программах аккуратны, но слишком просты и годятся только для непритязательных разработчиков и их заказчиков.

Чаще всего подобные редакторы, кроме чисто визуальных, способны работать с несколькими языками (зачастую в одном файле). Однако есть редакторы, предназначенные только для создания и редактирования таблиц стилей (CSS, например, Top Style или CSS Magic) или обработки XML-документов (к примеру, Microsoft XML Notepad). С одной стороны, удобно работать с минимальным набором многофункциональных инструментов, не загромождая рабочее пространство, с другой — если во главу угла ставится, к примеру, только программирование на PHP, то, возможно, стоит установить специализированный редактор именно для работы с этим языком.

Отдельно стоит отметить, что редакторы не со встроенными браузерами удобны более корректной работой с активными сценариями. Дело в том, что у каждого браузера достаточно много особенностей в интерпретации сценариев на JavaScript, а язык VBScript поддерживается преимущественно одним браузером — Internet Explorer. Таким образом,

если разработчик хочет, чтобы сценарии были совместимы со всеми распространенными браузерами, ему нельзя полагаться на встроенный интерпретатор какой-либо программы. Логичным выводом является необходимое условие: «зоопарк браузеров» на компьютере разработчика (помимо собственно редакторов). О конфликтах браузеров и их производителей будет рассказано в следующих главах.

Если редактор кода веб-мастер волен выбирать по своему вкусу, то браузеры для тестов выбирать не приходится: выбирают пользователи. А поскольку разработчик только в очень редких случаях может предугадать, какими программами просмотра будут пользоваться посетители сайта, ему нужно тестировать страницы во всех возможных браузерах. Под операционной системой Windows нужно обязательно проверять сайт в браузерах Internet Explorer (на момент написания книги последней версией является седьмая, а наиболее популярной шестая, обе обнаруживают достаточные отличия в интерпретации веб-страниц от пятой и тем более шестой версии; изредка посетители заходят на сайт из-под третьего Internet Explorer), Mozilla Firefox (1, 1.5 и 2 версии), просто Mozilla (и других браузерах на основе механизма Gecko, например, K-Meleon), Netscape (3 и 4 версии работали на одном «движке», а начиная с 6 — на основе того же Gecko, различия в отображении страниц гигантские) и в браузере Opera (последние и наиболее корректно интерпретирующие код версии — 8 и 9; глобальные различия в прорисовке страниц есть с шестой версией). В тех же браузерах, но под другими операционными системами страницы могут выглядеть по-другому из-за различной работы со шрифтами и из-за разницы в исходном коде браузеров. Например, Internet Explorer 5 для Windows и Internet Explorer 5 для MacOS — это два разных браузера. Firefox и Opera, правда, дают не столь различающиеся результаты. Для MacOS (начиная с 10 версии) характерно наличие еще двух браузеров — Safari (браузер по умолчанию, версии 1-2, и планируется третья) и Camino. Safari базируется на том же коде, что и браузер Konqueror для семейства операционных систем Linux, стремительно набирающих популярность (однако «линуксоиды» реже пользуются им, чем браузером Firefox). Безусловно, вне студийных условий трудно тестировать сайт на разных системах, но для этого существуют онлайновые сервисы (список в конце книги прилагается).

Для работы с языками серверных сценариев (ЯСС) необходимы серверы. Серверы — это не только металлические ящики с дисковыми массивами, но и программы, определенным образом организующие работу с данными для многих пользователей и имеющие средства интерпретации ЯСС. Например, для работы с сайтами на PHP (а именно на этом языке работает очень много готовых веб-архитектур, или «движков») требуется сервер Apache (как наиболее подходящий), сам PHP (ведь язык — это тоже программа) и — желательно — база данных, например, MySQL. А для работы с БД существует удобное средство, на-

писанное на языке PHP — программа PHPMyAdmin. Все это нужно держать под рукой и правильно настроить. А для языка ASP (точнее, это не язык, а средство разработки, поскольку включает сценарии на других языках) больше подходит сервер IIS. Правда, есть мощные отладочные пакеты, например, «Денвер» от российских программистов. Компактный дистрибутив (работающий под Windows, тогда как традиционно Apache ставится на Linux или другую Unix-подробную систему) включает собственно сервер Apache с большинством необходимых модулей, поддержку языков PHP, Perl и Parser (последний не в основном дистрибутиве) и базы данных MySQL с инструментом PHPMyAdmin, что делает его незаменимым отладочным средством: кусочек Linux'а в системе Windows. Тем не менее, если требуются специфические настройки сервера, особые расширения, а также тест дополнительных программ, которые будут запускаться на сервере, то стоит не просто установить сервер, язык и БД по отдельности, но и воссоздать условия их функционирования на рабочем сервере, где будет располагаться сайт, — нужная операционная система нужной версии, утилиты, модули и т. п. Для написания кода на ЯСС пригодятся специализированные текстовые редакторы (в том числе из перечислявшихся выше). Например, для PHP подойдут PHP Expert Editor, HTML Expert, CatsHtml KissCool. А редакторы EditPlus Text Editor, PHP Edit, Aditor, Web Development Studio и некоторые другие позволяют работать с языками HTML, ASP, PHP, Perl, Java, JavaScript, VBScript, CSS, XML, C/C++, Python, WML, qBasic, Pascal, Lisp, ActionScript, ColdFusion, Parser, и их можно «научить» работать и с остальными языками.

Растровая графика — один из важнейших компонентов веб-страниц. Сложно (но все-таки возможно) встретить современный сайт, где разработчики обошлись без картинок в форматах JPG, GIF или PNG. (Замечу в скобках, в чем коренное различие этих форматов. JPG пригоден для фотоизображений и обеспечивает весьма разнообразные степени сжатия; GIF предпочтителен для изображений с большими однородными областями одного цвета, поддерживает прозрачные области и анимацию; PNG обеспечивает хорошее сжатие данных и также поддерживает прозрачные области и даже полупрозрачность, но не во всех браузерах корректно, а в старых браузерах — например, в Netscape 4 — изображения формата PNG вообще игнорируются.) Изображения используются как иллюстрации (фотографии, схемы и прочие материалы визуального представления данных), строительный и декоративный материал (фон страницы, «однопиксельные распорки» — о них пойдет речь в разделе про верстку, элементы оформления таблиц и страниц в целом) и функциональный материал (рисованные кнопки, пиктограммы и прочие элементы интерфейса). При создании и обработке изображений следует всегда держать баланс между качеством изображения и его размером. Огромный процент посетителей интернета пользуется модемами

с низкими скоростями, так что загрузка страницы, нагруженной графикой, превращается в томительные минуты ожидания. Очень большой максимум суммарного «веса» изображений для одной страницы — всего 100 килобайтов, а желательно не более пятидесяти. В этом случае ожидание даже на невысоких скоростях не будет настолько критичным, чтобы из-за недозагрузки функциональной графики посетитель сайта не выдержал и ушел на другой ресурс. При этом никогда нельзя сжимать изображения настолько, чтобы на них появлялись дефекты, потому что эстетическая составляющая использования графики на сайте играет не последнюю роль. Использование графики, соотносящейся с тематикой сайта, и нестандартной (но оправданной) верстки зачастую создает лицо сайта, его неповторимый стиль. Поэтому для обработки и создания веб-графики существует огромное количество приложений.

Растровые графические редакторы можно разделить на две группы: выполняющие ограниченное количество функций (условно «простые», потому что по количеству исходного кода и размеру системных файлов их не всегда можно назвать простыми) и полнофункциональные, зачастую включающие такие операции, которые могут пригодиться всего несколько раз («сложные»).

«Простые» графические редакторы (например, разные варианты ACDSee, i.Mage, Photo Razor, Image Enhance, Image Explorer, NeoPaint и другие) являются одновременно программами просмотра и основного редактирования: позволяют кадрировать изображения, изменять их размер и иногда степень сжатия, применять простейшие фильтры и изменять яркость, контрастность и насыщенность цветов. Некоторые из них умеют создавать уменьшенные копии изображения (thumbnails в принятой англоязычной терминологии). Такая обработка достаточно удобна, если требуется быстро обработать одну или несколько фотографий для размещения в виртуальной фотогалерее в короткий срок. Очевидно, что для веб-дизайна такие программы могут пригождаться ограниченно: они вряд ли помогут в создании коллажей или оригинальной графики, и их ценность сводится к быстрому редактированию и пакетной обработке (удобно реализовано в Photo Razor).

Более мощные редакторы (полноценные с огромным количеством функций — Adobe PhotoShop, Corel PhotoPaint, Macromedia Fireworks, Ulead PhotoImpact, The Gimp, Jasc Paint Shop Pro — и более ограниченные в возможностях Photobie, PhotoFiltre, Pixia, Project Dogwaffle, VCW VicMan's Photo Editor и другие), во-первых, поддерживают технологии слоев и работу с множеством объектов на рабочем поле, что позволяет комбинировать изображения, а во-вторых, обладают большим арсеналом инструментов (кисти, фигуры, инструменты клонирования, выделение и редактирование фрагментов, наборы фильтров и эффектов и т.п.), что дает возможность создавать изображения с нуля. А эти два факта в сочетании с богатыми возможностями коррекции уже существующих

изображений позволяют подготавливать графику для веб-страниц без каких-либо ограничений. Из указанных редакторов Adobe Photoshop наиболее сильно сжимает изображения при подготовке для веб-страниц без существенной потери качества. В остальных редакторах нужно вручную проставлять нужное разрешение (72 dpi вместо 300), нужный размер изображения и степень сжатия, а после этого еще использовать утилиты вроде JPGCleaner или PureJPG для того, чтобы из изображения была вычищена вспомогательная информация.

Нужно добавить, что есть совсем уж узкоспециализированные программы: например, MicroArt предназначена только для создания иконок для сайтов и программ в формате .ico, а программы Zx Color Spy или EyeDropper нужны только для выбора нужного цвета и определения его кодов в разных цветовых моделях.

Создание анимации в формате GIF поддерживают далеко не все редакторы. Удобные инструменты для этого — программы Adobe ImageReady и Ulead Gif Animator. Сохраняя графику в рабочем формате, они способны на выходе давать оптимизированное для веба изображение с поддержкой анимации. Кроме того, создавать анимированные изображения позволяют программы Atani, Babarosa Gif Animator, Pivot Stickfigure Animator,

В процессе подготовки изображений не стоит недооценивать и роль векторных редакторов (из которых наиболее популярны Adobe Illustrator, CorelDraw и Macromedia Freehand, а менее известными, но не менее функциональными аналогами со своими особенностями являются Xara и Creature House Expression, новые версии которой ныне выпускаются компанией Microsoft). Дело в том, что для свободного рисования линиями, для создания нефотореалистичной графики они подходят даже больше, чем растровые, и при этом все умеют экспортировать изображение в большинство популярных растровых форматов, в том числе пригодных и для публикации на веб-страницах.

Векторная графика на сайтах пока используется ограниченно, и, несмотря на то, что стандарт SVG для описания векторных фигур появился достаточно давно, он не поддерживается большинством браузеров без дополнительных модулей. Единственный векторный формат, используемый на вебе — это Flash (файлы в формате .swf), дитя компании Macromedia, ныне принадлежащий Adobe. Он позволяет создавать анимированные интерактивные ролики, которые используются как встраиваемая графика, как меню и прочие элементы на веб-страницах, но которые также могут являться практически самостоятельными сайтами (практически — потому что для помещения ролика на страницу нужен небольшой HTML-код). В последнем случае ролик включает навигацию, способен подгружать вспомогательные ролики, текстовые файлы, сценарии, изображения и аудиофайлы. Подробнее о таком подходе будет рассказано далее.

Основная и наиболее удобная программа, использующаяся для создания флэш-роликов, — Macromedia Flash (ныне Adobe Flash). Кроме этого, работать с флэшем позволяют программа Corel R.A.V.E. и последние версии Adobe PhotoShop.

И последнее, но, пожалуй, одно из важнейших замечаний. Оно касается работы с текстом.

На большинстве веб-страниц основным носителем информации является текст. Новости, блоги, книги, справочные страницы, форумы, чаты, электронные газеты и журналы, книги отзывов и другие типы страниц — основой содержимого всех их является текст. И если посетители вводят тексты комментариев, отзывы, реплики в форумах и чатах как бог на душу положит (без пробелов после знаков препинания, с массой ошибок и опечаток, с интернетовскими жаргонными словечками, со знаками дюйма вместо кавычек и с дефисами вместо тире), то создатели и администраторы сайтов должны следить за верстkой и грамотностью текстов, размещаемых на страницах. Помочь в этом могут три инструмента: грамотность или желание писать грамотно (обычно оно способствует грамотности), словари (толковые, иностранных слов, орфографические и иные) и справочники (грамматика, орфография, пунктуация) и, наконец, специализированные онлайновые сервисы: часть из них проверяет правописание (как, например, «Орфограф» — *www.artlebedev.ru/tools/orfograf/*, или проект «Орфус» — *www.dklab.ru/chicken/nablas/24.html*), а часть — выполняет рутинные операции, связанные с типографикой текста: расставляют правильные виды кавычек и тире, удаляют ненужные пробелы и восстанавливают пропущенные, расставляют неразрывные пробелы и тэги запрета переносов и т.п. Среди них наиболее популярны «Типограф» (*www.artlebedev.ru/tools/typograf/*), «Автотипографика» (*http://at.webcode.ru/online/*), еще один «Типограф» (*http://rmc.net.ru/typo/*) других авторов, а также три инструмента, работающих в режиме реального времени: «Корректор типографики» (*www.pixel-apes.com/typografica/corrector*), Jevix (*www.jevix.ru*) и «Devanagari» (*www.erlang.com.ru/devanagari*), написанный мной (его можно также сохранить на жесткий диск своего компьютера и пользоваться без подключения к интернету: файл-обработчик весит около 25 килобайт). С помощью таких инструментов можно достаточно быстро привести текст в порядок, забыв о ручной расстановке тэгов абзаца и прочих обязательных вещей.

Наконец, еще один класс программ, без которого не обойдется разработчик сайта, — менеджеры файлов, поддерживающие FTP-доступ к сайту. Есть файловые менеджеры общего назначения с таким доступом (например, Total Commander), есть специализированные FTP-менеджеры (FTP Commander, Core FTP и другие), есть технология SVN (с помощью которой можно организовать совместную работу с файлами сайта на различных компьютерах), а можно написать файл-менеджер самому на одном из серверных языков программирования.

0.3. Процесс

Процесс создания сайта обычно включает несколько этапов: проектирование, разработка дизайна, верстка, программирование и тестирование. С одной стороны, все эти составляющие процесса обязательно должны присутствовать, а с другой — они не всегда выстраиваются в такой последовательности. Они могут протекать параллельно (особенно если над проектом работает группа людей), могут меняться местами (например, если «движок» для сайта уже написан как компонент предыдущего проекта, и нужна лишь его доработка для текущего проекта), а могут постоянно переплетаться, когда стиль разработки проекта таков, что проектирование отдельных фрагментов сайта происходит уже во время работы. Строгих правил тут нет и не может быть.

Единственное правило, которого нужно придерживаться для того, чтобы процесс не затягивался и не приходилось переделывать уже почти законченную работу, заключается в том, чтобы тестирование проходило не только в конце, но и на протяжении всей работы.

При проектировании нужно ориентироваться не только на свой вкус (поскольку разработчики обычно лучше, чем рядовые пользователи, ориентируются в интерфейсах, в интернете и в собственных разработках), а советоваться с потенциальными посетителями будущего ресурса — если, конечно, мастерство не достигло такого уровня, когда разработчик намного лучше пользователя знает, что последнему нужно. Это самое лучшее тестирование будущего проекта: показывать эскизы, советоваться, принимать к сведению все замечания (необязательно все их воплощать в жизнь). То же самое с дизайном. Типичная ошибка российских дизайнеров без большого опыта — забывать о том, что внешний вид веб-страницы является не только произведением искусства (и демонстрацией степени владения фотошопом), но и интерфейсом, служащим для работы с сайтом. Напротив, западные веб-дизайнеры (апологеты Нильсена) делают аскетичные веб-страницы, в которых невозможно запутаться, но с эстетической точки зрения такие сайты выглядят шаблонно и непривлекательно. Найти золотую середину — задача-максимум еще на этапе проектирования.

Наибольшая проблема при верстке сайта — написание такого кода, который давал бы одинаковый или максимально близкий результат во всех современных и устаревших браузерах в разных операционных системах, на разных мониторах с различным разрешением и при разных условиях (отключенные или включенные активные сценарии, таблицы стилей, изображения и дополнения вроде Java или

Flash). В таких условиях тестирование приобретает особую важность. При программировании же тесты важны в двух случаях: во время написания кода при «обкатке» его в условиях, приближенных к реальным (на домашнем или тестовом сервере) и после размещения проекта на рабочем сервере. Файлы конфигурации (например, *.htaccess* на сервере Apache), переменные окружения, пути к файлам, работа модулей (например, количество переадресаций в модуле `mod_rewrite`) и прочие нюансы могут различаться на тестовом и реальном серверах. Все эти факторы делают постоянное тестирование совершенно необходимым.

Рассмотрим, как может протекать процесс создания сайта в условиях, когда все функции (дизайнер, кодер, программист и т. п.) выполняет один и тот же человек.

Грамотное проектирование определяет, сколько времени будет затрачено на создание сайта, переделку его под влиянием заказчика, советчиков и здравого смысла, а также на редизайн и изменение структуры в дальнейшем. Важно представить себе каждый из последующих процессов и понять, что и в какой последовательности нужно делать. Перед началом работы нужно задать себе ряд вопросов (насколько длинным будет ряд — зависит от добросовестности разработчика) и максимально подробно ответить на них.

Для чего нужен этот сайт? Нужен ли он вообще?

Какая информация будет представлена на нем? Много ли ее там будет?

Как лучше логически разделить разные порции этой информации на группы, чтобы посетитель без ошибок понял, в какой группе ему стоит искать нужное?

Что в нем будет особенного по сравнению с подобными существующими? Если ничего, то все же почему он имеет право на существование? А если есть особенное, то почему бы не сделать это лейтмотивом работы?

Для кого создается ли этот сайт? Для самой широкой аудитории (тогда эксперименты с интерфейсом и дизайном должны будут свестись к минимуму), для молодежи (приветствующей эксперименты и обожающей игры) или для бизнес-аудитории (для которой важнее всего оперативный доступ к удобно рубрицированной и правильно дозированной информации)?

Какого рода информацию будет искать посетитель на этом сайте? Нужно ли будет на нем что-то искать?

Как помочь ему, если он сразу не нашел нужную информацию?

Нужно ли снабжать его обилием дополнительной информации? Будет ли дополнительная информация подгружаться на ту же самую страницу в указанное место или потребует загрузки дополнительной страницы (страшно представить — в новом окне)?

Нужна ли на сайте декоративная и иллюстративная графика? В каком объеме? Могут ли стратегически важные ссылки или фрагменты изображения быть решены в виде изображений?

Нужен ли на сайте флэш? Почему без него нельзя обойтись? Какие функции он будет выполнять?

Нужна ли на сайте анимация?

Что стоит вынести на передний план, что дать анонсами, а что вообще спрятать на внутренних страницах сайта?

Как именно информация будет разнесена на разные страницы? В чем логика такого разделения? Может ли сайт быть размещен только на одной странице с динамически подгружаемыми блоками информации? Что будет, если у пользователя будет отключен JavaScript?

Будет ли на сайте время от времени изменяющаяся информация? Насколько часто она будет меняться? В каком виде лучше сделать архив старых сообщений (новостей, постов, объявлений), нужно ли его делать вообще?

Как лучше организовать архив основных материалов? Все ли нужно держать на виду или часть стоит опускать за пределы видимости при помощи ссылок («Все материалы по теме», «Остальные статьи», «Отчеты за прошлый год», «Новости за прошлую неделю»)?

Нужна ли на сайте обратная связь с посетителями? В какой форме ее лучше сделать? Нужны ли пресловутая «книга отзывов» или форум, или достаточно будет автоматически отсылающегося письма разработчику, если посетитель заходит на отсутствующую страницу по «битой» ссылке?

Насколько обширной будет система статистики? Будет ли она регистрировать (помимо общего количества посетителей) ежедневную и постраничную посещаемость? Будет ли фиксироваться география пользователей и время, проводимое ими на каждой из страниц?

Что на сайте должно быть необычно? Где именно можно проявить новаторство? Не повредит ли новаторство работе с сайтом как с инструментом получения информации?

Что будет, если посетитель отключит активные сценарии, таблицы стилей, изображения, плагины (флэш и Java) или все вместе?

Как содержимое сайта будет смотреться на экранах разной ширины?

Как удобнее сделать меню доступа к остальным страницам?

Как разграничить функциональные (на запуск сценариев) и навигационные (на другие страницы) ссылки? Как разделить ссылки, открывающие страницу в новом или том же самом окне?

Какие технологии будет удобнее всего использовать при создании сайта? (Речь идет не о том, что разработчик лучше знает фотошоп, чем остальные растровые программы, и поэтому будет пользоваться имя для всех операций, а о том, что в каждом случае он сознательно выбирает

инструмент, пригодный в данной ситуации.) Есть ли смысл скрывать от посетителей то, что сайт написан на PHP, а не на Perl'е?

Честно и непредвзято ответив на эти вопросы, разработчик неожиданно для себя на следующих этапах создания сайта столкнется с тем, что часть проблем он уже решил (точнее, он уже просто не заметит этих проблем). В процессе обдумывания будущего сайта рисуется не только внешний вид главной страницы, но и абстрактная структура — разделы, страницы, динамические данные. При этом проектирование вполне может сопутствовать уже начавшейся работе. Я глубоко убежден, что веб-дизайн — это не только ремесло, требующее навыков, множества разнообразных знаний и профессионализма, но еще и искусство, позволяющее создавать работы, не повторяющие друг друга.

Этап разработки важен в первую очередь тем, что встречают все-таки по одежке. Что важно в дизайне? В первую очередь уместность его элементов, удобство использования и внешняя привлекательность — об этом нужно думать при создании каждой мелочи, и именно в такой последовательности. По этой причине разработка дизайна — одновременно и наиболее, и наименее творческий этап. С одной стороны, именно в этот момент можно проявить свое творческое начало, новаторство, изысканный вкус, нестандартное видение и отсутствие стереотипов. С другой — именно «уместность и удобство» должны сдерживать разработчика при попытке создать что-то очень новаторское. Даже при очень необычном построении страницы посетитель должен хорошо представлять себе, как он может добраться до нужной информации, а также сразу увидеть то, ради чего и замышлялся сайт.

Новаторство в первую очередь должно проявляться в том, чтобы найти новые, более удобные пути решения обычных проблем. Почему, допустим, навигационное меню должно обязательно располагаться слева или сверху? Почему оно должно быть прямоугольной формы?

Вопрос: «Какой дизайн нужен для данного сайта?» — можно поставить и по-другому. Нужно понять, что будет на веб-странице, а затем решать, как это будет выглядеть. Потому что, только располагая какими-то объектами, можно решать, как их располагать. С другой стороны, ничего не мешает придумать идею для сайта до того, как разработчик точно решит, какие разделы будут присутствовать на сайте. Если приветствуется творческий подход, то заранее рожденная идея даже поможет создать концепцию всего сайта в целом.

В таких случаях незаменимым средством оказывается мозговой штурм. Он помогает найти самые неожиданные ассоциации к заданной теме. Например, слово «навигация» пришло из мореходной терминологии; отсюда вывод: навигационное меню делаем в виде штурвала или карты. Платные материалы («товары») группируем в рубрике «трюм», а раздел «карта сайта» оформляем в виде развернутого свитка навигационной карты. Страницу поиска оформляем в виде «марса» (места, откуда в средневековом мореходстве наблюдали за горизонтом), функциональную ссылку «Добавить в Избранное» (или «Поставить закладку») снабжаем пиктограммой якоря, а раздел «Персонал» — тельняшками или бескозырками. В этом случае появляется выдержанность идеи оформления сайта. Конечно, если сайт нейтрален по тематике либо относится к мореходному делу. Вопрос, нужно ли подобным образом оформлять сайт, посвященный недвижимости в Подмосковье, не возникает.

Процесс разработки дизайна может быть разложен на составляющие. В этом случае стоит выделить этапы: поиска решения, выработки концепции, набросков (эскизов), согласования вариантов, разработки окончательного макета, воплощения этого макета в жизнь, окончательной проработки. Первый этап подразумевает поиск направления, в котором будет двигаться дизайнер: будет ли это сайт в академическом стиле «ничего лишнего» либо взрыв эмоций на экране, будет ли дизайн по смыслу ориентирован на содержание или к высочайшему рассмотрению будет принят один из бесплатных шаблонов. Выработка концепции являет собой детальную проработку той идеи, которая пришла в голову на предыдущем этапе. Не стоит недооценивать ни один из них. Под воплощением макета (о котором пойдет речь чуть ниже) в жизнь подразумевается перевод графического представления в вид веб-страницы: очевидно, что представление в виде статичной картинки заметно отличается от сверстанной страницы в браузере. Наконец, окончательная проработка необходима, потому что изначально всех мелочей предусмотреть нельзя; остальные этапы в комментариях не нуждаются. Часть из этих этапов могут быть пропущены (если сайт для себя, то какое уж тут согласование), а часть — совмещены по времени. Никаких строгих рамок и предписаний тут нет и быть не может, а рекомендации можно составлять для себя самому.

Например, я чаще всего делаю наброски и макеты в графических редакторах. В силу привычки думать, что называется, «с карандашом в руке», набросать примерную схему будущего дизайна, одновременно продумывая детали, удобно в какой-нибудь векторной программе (например, в Creature House Expression или Macromedia Freehand). А вот уже окончательный макет логичнее готовить в растровом редакторе (Adobe Photoshop как общепризнанный стандарт, но возможен и другой — вопрос вкуса, привычек и конкретных задач), поскольку основой для внешнего вида сайта будут (помимо разметки на HTML и CSS) как

раз растровые изображения. Распространенные растровые редакторы дают как минимум две возможности, чтобы нарезать из макета изображения, которые в дальнейшем будут загружаться с веб-страницей (встроенная технология разделения на фрагменты и последующего экспортирования в отдельные изображения, которая, например, в фотошопе называется Slices — «ломтики», а в Jasc Paint Shop Pro — очень похоже, Image Slicer, и — вторая возможность — элементарное выделение, копирование и вставка в нужного размера файлы, сохраняемые для веб-страниц).

Из всех этих этапов наиболее длительный — отрисовка окончательного макета, поскольку он должен получиться таким, чтобы через два-три года самому не было мучительно стыдно. В этом случае особенно полезными оказываются навыки рисования с натуры, четкое представление функций и свойств изображаемых объектов, а также хорошее понимание того, где остановиться в создании красоты и не перейти грань, за которой интерфейс сайта становится неудобным.

Наиболее же неприятным этапом чаще всего оказывается согласование вариантов дизайна с заказчиком. Очень редко встречаются идеальные заказчики, полностью довольные предоставленным макетом. Основные беды работы с заказчиком (с точки зрения веб-дизайнера) следующие:

1. Я не знаю, чего хочу...
2. Вот это левее подвиньте, это перекрасьте в синий цвет, фоном побольше фигур вставьте, шрифт какой-нибудь менее строгий подберите...
3. Хочу так же, как у вон той фирмы...
4. Понимаете, я тоже веб-дизайном занимаюсь...
5. Хочется, чтобы как у всех...

С первой бедой можно справиться, приведя множество аргументации с непонятными терминами и уверениями, что сейчас так модно и что это будет работать. Но примерно в 50% случаев. С остальными бедами бороться очень тяжело. С шаблонным мышлением, на мой взгляд, вообще бороться почти невозможно — если только использовать какой-то нешаблонный метод.

Когда дизайн разработан (и согласован), наступает звездный час верстальщика. С одной стороны, он должен придерживаться двух правил: верстать, как принято в его компании (или в его убеждениях), и верстать так, чтобы это корректно работало в большинстве браузеров на большинстве операционных систем при учете разных разрешений экрана и сопутствующих условий. Для этого нужно просто хорошо знать правила верстки и языки разметки и постоянно тестировать каждый фрагмент кода. На деле все оказывается несколько сложнее по той простой причи-

не, что у каждого браузера есть свои специфические условия, при которых тот или иной фрагмент кода будет работать некорректно. Например, все верстальщики знают, что при указании фонового изображения средствами CSS в скобках нельзя использовать кавычки. То есть нужно писать `background:URL(image.gif)` вместо логичного `background:URL("image.gif")` — потому что браузер Microsoft Internet Explorer в среде операционной системы MacOS, встречая эти кавычки, отказывается работать дальше. (Правда, все также знают, что на «макинтошах» сейчас все больше используют Safari и Mozilla Firefox, а также изредка Camino — все три построены на других «движках»; но по традиции кавычек не ставят.) Еще все знают, что Netscape 4 перезагружается при изменении размеров страницы пользователем... Всем известно, что священные тексты спецификаций HTML и CSS разработчики всех браузеров трактуют несколько по-разному (и только к выходу IE 7, Firefox 2 и Opera 9 война конфессий приняла более спокойное течение), и что рамки (указанные как «`border`») браузеры Opera и Firefox будут помещать снаружи блока, а Internet Explorer — внутри. Остается держать в уме не только спецификации, но и таблицы несовместимостей браузеров с точки интерпретаций тэгов HTML и свойств CSS. И еще много раз тестировать, потому что даже эти знания не спасают от обнаружения новых неожиданностей.

Программирование может протекать не только параллельно верстке страниц, но даже параллельно разработке дизайна и рождению идей дизайна сайта. Смысл в том, что структура сайта разрабатывается в самом начале, а грамотное построение структуры сайта — как раз задача программиста.

Если сайт статичный и состоит всего из нескольких страниц, то задача программиста сводится к минимуму: грамотно организовать файлы и директории на сервере, чтобы документы и изображения, архивы и стилевые файлы не были свалены в кучу, а образовывали структуру, в которой можно легко разобраться. По мере возрастания требований к сайту увеличивается и количество задач программиста. Типичные задачи — разработка структуры сайта (логичной, безопасной, удобной, масштабируемой и модифицируемой), системы управления сайтом (загрузка файлов на сайт, выбор схем оформления, редактирование разделов сайта, модерирование интерактивных разделов, добавление и удаление разделов и другие операции над компонентами сайта), поисковой системы (все эти три задачи взаимосвязаны), а также разработка всех разделов, формирование которых происходит динамически. Сюда входит собственно создание интерактивных страниц (чаты, форумы, блоги, книги отзывов и т. п.), отчетов, написание сценариев для вывода однотипно выглядящих страниц с разным содержимым, новостных лент и еще множество задач. Никто не запрещает писать код во время разработки дизайна, только особую важность в этом случае приобретает изначальная разработка концепции.

1. Проектирование

Если не знаешь, зачем сайт и куда его применить — не делай его совсем. Это, пожалуй, ключевое правило проектирования сайта. Потому что во всех остальных случаях есть материал, идеи и хотя бы приблизительное представление, как воплотить идею в жизнь.

После того, как идея появилась и в достаточной мере оформилась, а материалы оказались подготовленными, следует решить кроссворд.

Кроссворды, как известно, обладают одной особенностью: все без исключения должно встать на свои места так, чтобы не осталось пустых мест. В случае с разработкой сайта действует такое же правило: все компоненты сайта должны вступать между собой в такие отношения, чтобы не оставалось смысловых пробелов. Чтобы ничего не было упущено, но и слишком многого бы не получилось.

Тривиальный пример — построение главной страницы. С одной стороны, на ней должно уместиться все основное, что есть на сайте, хотя бы в виде ссылок и анонсов. С другой стороны, если на главной странице будет переизбыток информации, ею будет трудно пользоваться. Остается найти золотую середину: либо частью ссылок пожертвовать, либо сделать изящное иерархическое меню, которое позволит расположить на главной странице большинство ссылок, если не все.

Процесс создания сайта после проектирования включает, как правило, три фазы (не обязательно протекающие последовательно — часто и параллельно). Это дизайн, верстка и программирование. Зачастую три фазы выполняют три разных человека (или даже три группы людей). И смысл проектирования в этом случае — сделать так, чтобы все трое могли сразу начать свою работу, потому что общая концепция уже решена, и можно работать уже по готовому техническому заданию. Даже если всю работу делает один человек, важно изначально представлять себе процесс, чтобы не переделывать десятки раз уже готовую работу.

Но цель проектирования не только в том, чтобы в первом приближении разработать структуру сайта и внешний вид, но в том, чтобы учесть десятки мелочей, нужных для того, чтобы сайт выглядел полно-