1 Начало

Пока мы не погрузимся в написание программ для iPhone, позвольте быстро познакомить вас с открывающимся перед нами полем боя. В этой главе я определю основные термины, сравню достоинства и недостатки двух главных методов разработки, а также проведу интенсивный курс по трем основным веб-технологиям, используемым в данной книге.

Сравнение веб-приложений и нативных приложений

Сначала определим, что такое «веб-приложение» и что такое «нативное приложение», а также рассмотрим достоинства и недостатки каждого.

Что такое веб-приложение?

С моей точки зрения, веб-приложение — это, по сути, просто сайт, оптимизированный для iPhone. Такой сайт может представлять собой что угодно — от проспекта небольшой компании до ипотечного калькулятора или программы для ежедневного контроля калорий — контент в данном случае неважен. Определяющими характеристиками веб-приложения являются следующие: пользовательский интерфейс создается при помощи стандартных веб-технологий, приложение доступно по гиперссылке (открытой, закрытой или под логином). Кроме того, программа оптимизирована с учетом специфики iPhone. Веб-приложение не устанавливается на телефоне, отсутствует в App Store iTunes и написано без помощи Objective-C.

Что такое нативное приложение?

Нативные приложения, напротив, устанавливаются на iPhone, имеют доступ к оборудованию (динамику, акселерометру, камере и т. д.) и пишутся на Objective-C. Однако самой важной характеристикой нативного приложения является то, что оно присутствует в App Store iTunes — хранилище, содержащем плоды работы бесчисленного множества программистов со всего мира, в том числе вашего покорного слуги.

Достоинства и недостатки

Программы бывают разными, различаются и предъявляемые к ним требования. Некоторые программы реализуются при помощи веб-технологий лучше, чем

другие. Зная достоинства и недостатки обоих методов, вы сможете правильно решить, какой метод лучше избрать в данной конкретной ситуации.

Достоинства разработки нативных приложений:

- миллионы зарегистрированных обладателей кредитных карточек находятся в одном щелчке от вашей программы;
- Xcode, Interface Builder и фреймворк Cocoa Touch — это очень хорошая среда разработки;
- вы имеете доступ ко всем великолепным аппаратным функциям устройства.

А вот недостатки:

- чтобы стать разработчиком Apple, потребуется заплатить;
- вы зависите от принятого в Apple процесса утверждения программ;
- при разработке придется использовать язык Objective-C;
- вы будете вынуждены работать на Mac;
- вам не удастся оперативно исправлять обнаруженные ошибки;
- цикл разработки достаточно медленный, а на процесс тестирования накладываются специфические ограничения App Store.

Достоинства разработки веб-приложений:

- вы можете пользоваться привычным инструментарием веб-разработчика;
- можете применять имеющиеся у вас навыки веб-дизайна и веб-разработки;
- вы можете писать программы не только для операционной системы Mac;
- ваша программа будет работать на любом устройстве, которое имеет браузер;
- вы сможете исправлять замеченные ошибки в режиме реального времени;
- цикл разработки достаточно быстр.

И недостатки:

- отсутствует доступ к аппаратному обеспечению телефона;
- если вы хотите продавать программу, то вам придется задействовать собственную платежную систему;
- возможно, будет сложно реализовать некоторые изысканные эффекты при написании пользовательских интерфейсов.

Какой метод подходит вам?

Вот это уже интересно. Поскольку природа iPhone такова, что устройство практически все время находится онлайн, создается среда, в которой границы между веб-приложениями и нативными приложениями размываются. В iPhone есть некоторые малоизвестные возможности, позволяющие при желании переводить веб-приложение в автономный режим (то есть использовать без Интернета, подробнее см. гл. 6). Более того, некоторые сторонние проекты (из которых следует особо отметить PhoneGap) являются активно разрабатываемыми решениями, позволяющими веб-разработчику брать веб-приложение и создавать пакет с аналогичным нативным приложением для iPhone и других мобильных платформ.

Для меня эта смесь подходит идеально. Я могу писать программы на языке, который хорошо знаю, выпускать продукт именно как веб-приложение (для iPhone или других устройств, на которых стоят современные браузеры), не проходя через процесс допуска от Apple. Затем при помощи уже имеющейся кодовой базы создавать улучшенную нативную версию программы, которая будет иметь доступ к оборудованию устройства и потенциально сможет продаваться через App Store. А если Apple забракует программу? Ничего страшного, так как у меня сохраняется онлайновая версия. Я могу продолжать доработку нативной версии, пока пользователи работают с сетевой.

Вводный курс веб-программирования

Три основные технологии, при помощи которых мы будем писать программы, — это HTML, CSS и JavaScript. Кратко рассмотрим каждую из них, чтобы гарантировать, что перед началом работы над более сложными темами все читатели обладают нужными базовыми знаниями.

Введение в HTML

Путешествуя по Интернету, мы переходим от сайта к сайту, то есть просматриваем веб-страницы, которые являются просто текстовыми документами, находящимися на каком-то другом компьютере. Текст на типичной веб-странице заключен в теги HTML, сообщающие браузеру, какую структуру имеет документ. С помощью этой информации браузер определяет, как отображать контент страницы, чтобы донести его смысл.

Рассмотрим фрагмент веб-страницы, приведенный в примере 1.1. В первой строке мы видим строку «Привет!», заключенную в теги `h1` (обратите внимание, что открывающий и закрывающий теги немного различаются: во втором теге есть слэш, а в открывающем — нет).

Текст, заключенный в теги `h1`, означает для браузера заголовок, который будет отображаться сравнительно крупным полужирным шрифтом в отдельной строке. Существуют также теги заголовков `h2`, `h3`, `h4`, `h5` и `h6`. Чем меньше номер, тем важнее текст — это значит, что текст, заключенный в `h6`, будет мельче, чем текст в `h3`.

После тега `h1` в примере 1.1 идут две строки, заключенные в теги `p`. Это теги абзацев. Браузер будет начинать каждый абзац с новой строки. Если абзац достаточно широкий и не помещается целиком по ширине окна браузера, текст будет продолжаться со следующей строки. В любом случае после каждого абзаца вставляется пустая строка, отделяющая его от следующего элемента страницы.

Пример 1.1. Фрагмент HTML

```
<h1>Привет!</h1>
<p>Спасибо, что зашли на мой сайт.</p>
<p>Надеюсь, он вам нравится.</p>
```

Кроме того, одни HTML-теги могут находиться внутри других. В примере 1.2 показан тег ненумерованного списка (`ul`), в котором находятся три элемента

списка (li). В таком случае браузер отобразит маркированный список, каждый элемент которого будет расположен в отдельной строке. Если внутри тега находится другой тег (или несколько тегов), то внутренние теги называются дочерними элементами, а внешние — родительскими элементами. В данном примере элементы li являются дочерними относительно ul, а ul — родительским для li.

Пример 1.2. Ненумерованный список

```
<ul>
   <li>Пицца</li>
   <li>Пиво</li>
   <li>Хот-доги</li>
</ul>
```

Все теги, рассмотренные выше, являются *метками блоков*. Важнейшей характеристикой метки блока является то, что содержание такого тега отображается в отдельной строке, без других элементов слева или справа. Вот почему заголовки, абзацы и списки добавляются по вертикали страницы, а не по горизонтали. Кроме меток блока существуют также *внутритекстовые теги*, которые, как следует из названия, могут появляться прямо в строке. Тег визуального выделения (em) относится именно к этой группе и выглядит так:

```
<p>Я <em>правда</em> надеюсь, что вам нравится.</p>
```

«Прадедушка» внутритекстовых тегов и, пожалуй, самый классный элемент HTML — это тег a. Тег a означает *точку привязки* (anchor), но чаще этот тег называют ссылкой или гиперссылкой. На тексте, заключенном в теге a, можно щелкнуть (например, кнопкой мыши), и в таком случае в браузере откроется новая HTML-страница.

Чтобы сообщить браузеру, какую именно страницу следует загрузить, нужно добавить к тегу так называемый *атрибут*. Атрибуты — это именованные значения, вставляемые в открывающий тег. В теге привязки используется атрибут href, указывающий расположение страницы, на которую следует перейти. Вот ссылка на главную страницу Google:

```
<a href="http://www.google.com/">Google</a>
```

Если вы не привыкли читать HTML, это может показаться немного путаным, но вы как минимум должны узнать URL главной страницы Google. В книге нам встретится много тегов a и атрибутов href, поэтому лучше сейчас задержитесь на минутку и разберитесь в этом вопросе, если сразу не все поняли.

Необходимо помнить несколько моментов относительно атрибутов. Различные теги могут иметь разные атрибуты. К открывающему тегу можно добавлять несколько атрибутов, разделяя их пробелами. Атрибуты никогда не добавляются к закрывающему тегу. Существуют сотни возможных комбинаций атрибутов и тегов, но не волнуйтесь — в этой книге вам придется столкнуться примерно с десятком вариантов.

Рассмотренный выше фрагмент кода на языке HTML обычно находится в теге body готового HTML-документа. Документ на языке HTML состоит из двух частей: заголовка (head) и тела (body). В теле документа записывается вся информация,

которую вы хотите показать пользователю. В заголовке содержится информация о странице, бóльшая ее часть невидима пользователю.

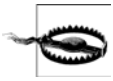
Тело и заголовок документа всегда заключаются в элемент `html`. В примере 1.3 приведен фрагмент из правильно оформленного HTML-документа. На данный момент в разделе `head` есть элемент `title`, сообщающий браузеру, что этот текст должен отображаться в окне на месте заголовка.

Пример 1.3. Правильно оформленный HTML-документ

```
<html>
   <head>
      <title>Моя потрясающая страничка</title>
   </head>
   <body>
      <h1>Привет!</h1>
      <p>Спасибо, что зашли на мой сайт.</p>
      <p> Надеюсь, он вам нравится.</p>
      <ul>
         <li>Пицца</li>
         <li>Пиво</li>
         <li>Хот-доги</li>
      </ul>
   </body>
</html>
```

Как правило, в браузере открываются веб-страницы, расположенные в Интернете. Однако браузеры отлично показывают и локальные HTML-документы (то есть те, которые находятся прямо на устройстве). Чтобы пояснить, что я имею в виду, откроем текстовый редактор и воспроизведем в нем пример 1.3. Когда все будет готово, сохраним этот пример на Рабочем столе как `test.html` и откроем его в Safari. Для этого либо перетащим документ на ярлычок Safari, находящийся на Рабочем столе, либо откроем Safari и выполним команду File ▸ Open File (Файл ▸ Открыть файл). Кроме того, может сработать обычный двойной щелчок на `test.html`. Также, можно открыть файл в текстовом редакторе или в другом браузере в зависимости от действующих настроек.

Даже если вы работаете не в Mac OS X, а в другой операционной системе, при тестировании программ для iPhone следует использовать браузер Safari, так как он наиболее похож в техническом отношении на Mobile Safari — браузер, установленный на iPhone. Версия Safari для Windows находится по ссылке http://www.apple.com/safari.



Некоторые текстовые редакторы не очень удобны при написании документов на HTML. В частности, не рекомендуется пользоваться инструментами, поддерживающими развитое редактирование текста (rich text editing), к которым относятся, например, Microsoft Word и TextEdit. Редакторы такого типа могут сохранять файлы не в виде обычного текста (plain text), а в других форматах, которые просто сломают структуру HTML. Если вы ищете хороший текстовый редактор, то для Mac мне больше всего нравится TextMate (http://macromates.com/). Я также слышал, что существует клон этой программы для Windows. Он называется E Text Editor (http://www.e-texteditor.com/). Если вам нужна бесплатная программа, то можете скачать TextWrangler для Mac (http://www.barebones.com/products/TextWrangler/) либо работать с обычным Блокнотом Windows.