1 Основы адаптивного веб-дизайна

Всего лишь несколько лет назад сайты могли создаваться с фиксированной шириной в расчете на то, что все конечные пользователи получат удобные условия работы. Эта фиксированная ширина (обычно 960 пикселов или около того) была не слишком велика для экранов ноутбуков, а пользователи, имеющие мониторы с высоким разрешением, просто видели с обеих сторон большие поля.

Но в 2007 году на телефонах iPhone компании Apple впервые появились по-настоящему удобные условия просмотра информации и условия доступа людей к веб-данным и работы с ними изменились навсегда.

В первом издании этой книги было отмечено, что «за 12 месяцев, с июля 2010-го по июль 2011 года, процент использования мобильных браузеров во всем мире вырос с 2,86 до 7,02».

В середине 2015 года в той же статистической системе (gs.statcounter.com) утверждалось, что данный показатель увеличился до 33,47 %. Для сравнения, в компании North America Mobile этот показатель достиг 25,86 %.

Использование мобильных устройств растет по любым показателям, а на другом краю шкалы в общую практику применения входят 27- и 30-дюймовые дисплеи. Как никогда ранее растет и разрыв между наименьшими и наибольшими экранами, на которых просматривается веб-содержимое. Благо есть решение, подходящее для постоянно расширяющейся арены браузеров и устройств. Нормальная работа сайта на множестве устройств и экранов возможна благодаря так называемому адаптивному дизайну, построенному на основе применения HTML5 и CSS3. Эта технология позволяет разметке и возможностям сайта подстраиваться под имеющееся окружение (размер экрана, тип ввода и возможности устройства или браузера).

Более того, адаптивный веб-дизайн, создаваемый с помощью HTML5 и CSS3, может быть реализован без необходимости использования конечных решений на серверной стороне.

Итак, вперед к неизведанному

Я надеюсь, что глава 1 поможет достичь одной из двух целей независимо от того, в новинку ли вам такие понятия, как адаптивный дизайн, HTML5 или CSS3, или вы в них неплохо разбираетесь.

Если HTML5 и CSS3 уже используются вами для адаптивной веб-разработки, эта глава позволит быстро вспомнить их основы. А если вы новичок в этом деле,

воспринимайте ее в качестве своеобразного курса молодого бойца, охватывающего все с самого начала, так что данная глава пригодится всем без исключения.

К концу главы будет рассмотрено все необходимое для создания полностью адаптивной веб-страницы.

А зачем тогда остальные девять глав? Это также станет понятно к концу главы.

В текущей главе будут рассмотрены следующие вопросы:

- определение адаптивного веб-дизайна;
- способы настройки уровней поддержки браузеров;
- краткий обзор инструментальных средств и текстовых редакторов;
- создание первого адаптивного примера — простой страницы, созданной с использованием HTML5;
- важность метатега `viewport`;
- способ масштабирования изображений под их контейнер;
- написание медиазапросов CSS3 для создания контрольных точек дизайна;
- несовершенство нашего простого примера;
- почему мы находимся всего лишь в самом начале пути.

Определение адаптивного веб-дизайна

Понятие «адаптивный веб-дизайн» было введено Итаном Маркоттом (Ethan Marcotte) в 2010 году. В своей новаторской статье на сайте A List Apart (http://www.alistapart.com/articles/responsive-web-design/) он свел воедино три уже существовавшие на тот момент технологии (гибкий макет на основе сетки, подстраиваемые по размеру изображения и элементы мультимедиа, а также медиазапросы) в единый унифицированный подход, который он назвал адаптивным веб-дизайном.

Адаптивный веб-дизайн в кратком изложении. Адаптивный веб-дизайн является представлением веб-содержимого в наиболее удобном формате для окна просмотра и устройства, обращающегося за этим содержимым.

На ранней стадии развития наиболее характерным для адаптивного дизайна было построение, начинающееся с «рабочего стола», то есть дизайна с фиксированной шириной. Затем, чтобы этот дизайн работал на экранах меньшего размера, содержимое автоматически переформатировалось или удалялось. Но процесс не стоял на месте, и стало понятно, что все, от дизайна до содержимого и разработки, получается намного лучше, если действовать в обратном направлении, начиная с небольших экранов и работая по нарастающей.

Прежде чем вникать во все это, я хочу рассмотреть два вопроса, касающиеся поддержки браузера, а также текстовых редакторов и инструментальных средств.

Установка уровней поддержки браузеров

Благодаря популярности и возможности повсеместного использования адаптивного веб-дизайна теперь стало намного легче торговать с клиентами и ключевыми

партнерами. Некоторые представления об адаптивном веб-дизайне сложились уже у большинства людей. Понятие единого кода, способного работать практически на всех устройствах, становится весьма привлекательным.

При запуске проекта с адаптивным дизайном почти всегда возникает вопрос поддержки со стороны браузеров. При наличии столь широкого спектра браузеров и устройств вряд ли имеет смысл реализовывать полную поддержку изменений каждого отдельно взятого браузера. Вероятно, сдерживающим фактором является время, возможно — деньги. А может быть, и то и другое.

Как правило, чем старее браузер, тем больший объем работы и кода требуется для достижения желаемого результата или выравнивания эстетического восприятия с тем, которое пользователь получает при работе с современными браузерами. Поэтому рациональнее иметь менее вариативный и благодаря этому более быстродействующий код за счет распределения создаваемых впечатлений по уровням, обеспечивая получение самых совершенных визуальных эффектов и возможностей только на наиболее восприимчивых к ним браузерах.

В предыдущем издании этой книги некоторое время уделялось рассмотрению вопросов обслуживания очень старых браузеров, работающих исключительно на настольных компьютерах. В новом издании тратить время на это мы не будем.

В середине 2015 года мне уже приходилось писать о том, что времена браузеров Internet Explorer 6, 7 и 8 прошли. Даже IE 9 на всемирном рынке браузеров занимает весьма скромные 2,45 % (у IE 10 только 1,94 %, а у IE 11 уже более привлекательный показатель, равный 11,68 %). Если вам не остается ничего другого, как вести разработку под Internet Explorer 8 и более ранние версии, то я вам искренне сочувствую и заявляю, что ничего особо полезного из этой книги вы для себя не почерпнете.

Все остальные должны объяснить своему клиенту или заказчику, финансирующему проект, ошибочность разработки под скудные по своим возможностям браузеры и причины, по которым выделение времени и ресурсов преимущественно на разработку под современные браузеры и платформы во всех отношениях имеет более определенный финансовый смысл.

Но в конечном счете реальный вес имеет только ваша собственная статистика. За исключением экстремальных ситуаций, создаваемые нами сайты должны работать как минимум на каждом из самых распространенных браузеров. Кроме основных функциональных возможностей, для каждого веб-проекта имеет смысл заранее определить, на каких платформах нужно в полной мере создать наилучшие впечатления от его работы, а на каких вполне возможно будет согласиться на визуальные или функциональные отступления.

Вы также поймете, что на практике легче начинать с представления, создаваемого самым простым, базовым уровнем, и заниматься его усложнением (этот подход называется **постепенным усложнением**), чем подходить к решению проблемы с противоположной стороны — заниматься сначала созданием представления самого высокого уровня, а затем предпринимать попытки отступления для работы на платформах с более скромными возможностями (этот подход называется **постепенным упрощением**).

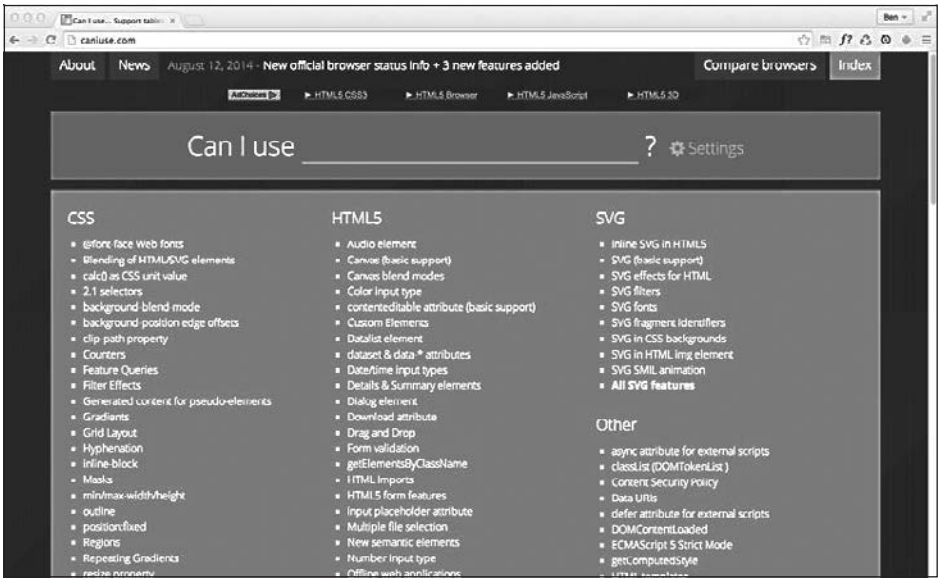
Чтобы пояснить причину, по которой это важно знать заранее, подумайте о том, что, если вам не повезет и 25 % посетителей вашего сайта будут пользоваться, к примеру, Internet Explorer 9, придется учесть, какие функции поддерживаются

этим браузером, и подстраивать свое решение под эти качества. Те же меры станут необходимы, если большой контингент ваших пользователей посещает сайт с устаревших платформ мобильных телефонов, таких как Android 2. Что именно рассматривать в качестве базового представления, будет сильно варьироваться в зависимости от проекта.

Если подходящие данные недоступны, то для определения того, нужно ли тратить время на разработку версии под конкретную платформу или браузер, я прибегаю к простому логическому приему: если стоимость разработки под браузер X и дальнейшей поддержки превышает выручку или выгоду, создаваемую пользователями браузера X, то вести разработку конкретных решений под браузер X не стоит.

Надо реже задаваться вопросом о том, возможно ли вести подгонку под устаревшую платформу или браузер, и чаще спрашивать себя о целесообразности этого.

Если при рассмотрении вопроса о том, какие функции какими платформами и версиями браузеров поддерживаются, вы еще не ознакомились с сайтом http://caniuse.com, я настоятельно советую вам сделать это. Там предоставляется весьма простой интерфейс для выявления, поддержкой какого браузера пользуются интересующие нас функции.



Краткая справка по инструментарию и текстовым редакторам. Неважно, какими именно текстовым редактором или IDE-системой вы пользуетесь для создания своих адаптивных веб-конструкций. Если самые простые текстовые редакторы позволяют вам успешно записывать код HTML, CSS и JavaScript, значит, все в порядке. Аналогично этому нет никакой особой оснастки, играющей важную роль в получении на выходе адаптивного веб-дизайна. Фактически вам

нужно лишь что-то позволяющее записывать код HTML, CSS и JavaScript. Чему вы отдадите предпочтение, Sublime Text, Vim, Coda, Visual Studio или Блокноту, не играет практически никакой роли. Работайте в той среде, которая вас больше всего устраивает.

Тем не менее следует знать, что, в отличие от прежних времен, сейчас имеется множество инструментальных средств (зачастую бесплатных), способных существенно облегчить выполнение рутинных, затратных по времени задач создания сайтов. Например, CSS-процессоры (Sass, LESS, Stylus, PostCSS) способны помочь с организацией кода, с переменными, в работе с цветовыми решениями и арифметикой. Такие средства, как PostCSS, способны также автоматизировать решение трудоемких и неприятных задач, например установку в коде CSS префиксов производителей. Кроме того, проверочные средства могут сравнить ваш код HTML, JavaScript и CSS со стандартами, по которым вы работаете, сэкономив массу времени на выявление опечаток и синтаксических ошибок.

Постоянно появляются все новые и новые инструментальные средства с более совершенными свойствами. Поэтому, какие бы актуальные и полезные средства ни упоминались в настоящее время, следует иметь в виду, что где-то рядом на выходе уже есть что-то более интересное. Следовательно, в своих примерах мы не можем полагаться на что-либо иное, кроме стандартов на основе HTML и CSS. Но вы можете воспользоваться доступными средствами для создания своей интерфейсной части кода как можно быстрее и надежнее.

Первый пример придания адаптивности

В начале главы я обещал, что к ее завершению вы узнаете обо всем необходимом для создания полностью адаптивной веб-страницы. До сих пор вокруг этого вопроса велись лишь досужие разговоры. Настало время выйти на прогулку.

ПРИМЕРЫ ПРОГРАММНОГО КОДА

Все примеры кода из этой книги можно загрузить по адресу rwd.education/download.zip или через GitHub на сайте https://github.com/benfrain/rwd. Имейте в виду, что окончательные версии отдельных примеров, приводимых в главе, имеются только в загружаемом коде. Так, если загрузить примеры кода из главы 2, они будут иметь тот вид, в котором приводятся в конце данной главы. В отличие от текста самой главы, в загружаемых примерах кода никаких промежуточных состояний не приводится.

Наш исходный файл HTML

Начнем с простой структуры HTML5. Не обращайте внимания на предназначение абсолютно каждой строки (особенно на содержимое контейнера `<head>`, которое более подробно будет рассматриваться в главе 4).

Предлагаю пока сконцентрироваться на элементах внутри тега `<body>`. Я абсолютно уверен, что вы не увидите там ничего необычного: всего лишь несколько `div`-контейнеров, символ логотипа, изображение (симпатичная булочка), один-два текстовых абзаца и список ингредиентов.

Вам будет представлена сокращенная версия кода. Для краткости я удалил из показанного далее кода текстовые абзацы, поскольку нас интересует лишь структура. Но вы должны знать, что это рецепт и описание способа изготовления булочек, относящихся к типично британской выпечке.

Если есть желание просмотреть весь файл HTML, его можно загрузить с сайта rwd.education:

```
<!doctype html>
<html class="no-js" lang="en">
  <head>
    <meta charset="utf-8">
    <title>Our first responsive web page with HTML5 and CSS3</title>
    <meta name="description" content="A basic responsive web page
      - an example from Chapter 1">
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <div class="Header">
     <a href="/" class="LogoWrapper"><img src="img/SOC-Logo.png" alt="Scone O'Clock
logo" /></a>
      <p class="Strap">Scones: the most resplendent of snacks</p>
    </div>
    <div class="IntroWrapper">
      <p class="IntroText">Occasionally maligned and
         misunderstood; the scone is a quintessentially British classic.</p>
      <div class="MoneyShot">
        <img class="MoneyShotImg" src="img/scones.jpg" alt="Incredible scones" />
        <p class="ImageCaption">Incredible scones, picture from Wikipedia</p>
      </div>
    </div>
    <p>Recipe and serving suggestions follow.</p>
    <div class="Ingredients">
      <h3 class="SubHeader">Ingredients</h3>
      <ul>

      </ul>
    </div>
    <div class="HowToMake">
      <h3 class="SubHeader">Method</h3>
      <ol class="MethodWrapper">

      </ol>
    </div>
  </body>
</html>
```

Изначально веб-страницы обладают гибкостью. Если открыть страницу примера даже в теперешнем ее состоянии (без медиазапросов) и изменить размер окна браузера, станет видно, что текст будет подвергнут необходимой перекомпоновке.

А как она себя поведет на других устройствах? При полном отсутствии CSS на iPhone получится следующее изображение.

Как видите, на iPhone она отобразилась точно так же, как и обычная веб-страница. Дело в том, что изначально iOS отображает веб-страницы шириной 980 пикселов и ужимает их в области просмотра.

Область просмотра браузера в технической терминологии известна как окно просмотра (viewport). Иногда это окно соответствует размеру экрана устройства, особенно в тех случаях, когда у пользователя есть возможность изменить размеры окна браузера. Поэтому впредь при обозначении пространства, доступного для нашей веб-страницы, как правило, будет использоваться эта, более точная терминология.

Эта заранее предполагаемая проблема легко решается путем добавления в `<head>`-контейнер следующего фрагмента кода:

```
<meta name="viewport" content="width=device-width">
```

Фактически этот метатег с именем `viewport` не считается стандартным способом указания браузеру способа отображения страницы (хотя и является стандартом

де-факто). В данном случае наш метатег `viewport` представляет собой четкое предписание «отобразить содержимое во всю ширину экрана устройства». Легче, наверное, просто показать вам действие этой строки кода на воспринимающих ее устройствах.

Отлично! Теперь тест отобразился и разлился до более естественного размера. Пойдем дальше.

Различные настройки и сочетания метатега (и стандарты, послужившие основой для версии подобных функциональных возможностей) будут рассмотрены в главе 2.

Укрощение изображений

Как говорится, лучше один раз увидеть, чем сто раз услышать. Это относится и к булочкам на нашей взятой для примера странице, на которой изображение всей этой красоты пока что отсутствует. Я собираюсь поместить изображение булочки ближе к началу страницы в качестве своеобразной приманки для пользователей, чтобы им захотелось прочитать ее содержимое.