

Оглавление

Краткое оглавление	5
Об авторе	15
О научных редакторах	15
Предисловие.....	16
Для кого эта книга	16
Структура.....	16
Что нужно для работы с книгой	17
Загрузка файлов с примерами кода	17
Цветные изображения	18
Типографские соглашения.....	18
От издательства	18
Глава 1. Основы отзывчивого веб-дизайна	19
Браузеры и устройства	20
Что такое отзывчивый дизайн	21
Коротко об отзывчивом веб-дизайне	21
Поддержка браузерами.....	21
Текстовые редакторы	23
Инструменты для разработки софта	23
Первый пример отзывчивого дизайна	24
Базовый файл HTML	24
Укрощение изображений	28
Ввод медиазапросов.....	31
Несовершенство нашего примера	37
Итоги	37
Глава 2. Создание HTML-разметки	38
Правильное начало работы с HTML-страницами	40
Элемент doctype	40
Тег html и атрибут lang	40
Кодировка символов	41

Добрый HTML5	41
Разумный подход к HTML-разметке	42
Да здравствует могущественный тег <a>!	42
Новые семантические элементы в HTML5	43
Элемент main	44
Элемент section	45
Элемент nav	45
Элемент article	45
Элемент aside	46
Элемент header	46
Элемент footer	46
Алгоритм схемы HTML5	47
Замечания относительно элементов h1–h6	48
Элемент <div>	48
Элемент <p>	49
Элемент <blockquote>	49
Элементы <figure> и <figcaption>	49
Элементы <details> и <summary>	50
Элемент <address>	52
Семантика HTML на уровне текста	52
Элемент 	53
Элемент 	53
Элемент 	53
Элемент 	54
Элемент <i>	54
Устаревшие функции HTML	54
Практическое применение HTML-элементов	55
Использование WCAG и WAI-ARIA для повышения доступности веб-приложений	56
WCAG	56
WAI-ARIA	57
Медиавозможности, встроенные в HTML5	59
Добавление видео и аудио в разметку HTML	59
Отзывчивое HTML5-видео и iFrames	61
Итоги	62
Упражнение	63

Глава 3. Медиазапросы — поддержка различных окон просмотра ...	65
Метатег <code>viewport</code>	67
Зачем нужны медиазапросы	69
Основная условная логика в CSS	69
Синтаксис медиазапроса	70
Медиазапросы в тегах <code>link</code>	71
Медиазапросы с использованием правила <code>@import</code>	71
Медиазапросы в файле CSS	71
Инверсия логики медиазапросов.....	72
Объединение медиазапросов	72
Ряд различных медиазапросов	72
Повседневные медиазапросы	73
Что можно тестировать с помощью медиазапросов	73
Использование медиазапросов для изменения веб-дизайна	74
Расширенные рекомендации по медиазапросам	77
Организация медиазапросов	78
Практические аспекты разделения медиазапросов	78
Вложение медиазапросов через встраивание	79
Что делать — объединять медиазапросы или же записывать их там, где они пригодятся?	79
Спецификация Media Queries Level 4	81
Медиафункции взаимодействия со страницей	82
Медиафункция <code>prefers-color-scheme</code>	83
Итоги	84
Глава 4. Fluid Layout, Flexbox и отзывчивые изображения	85
Преобразование фиксированного пиксельного дизайна в пропорциональный резиновый макет	86
Зачем нужен Flexbox	90
Представляем Flexbox	92
Тернистый путь Flexbox	92
Кому-то все еще нужны префиксы	92
Возможность динамического изменения	93
Различные макеты Flexbox внутри разных медиазапросов	98
Свойство <code>inline-flex</code>	98
Свойства выравнивания, предоставляемые Flexbox	100
Простой зафиксированный подвал	107
Изменение порядка следования исходных элементов	109

Свертка flex-элементов	113
Итоги по Flexbox	116
Отзывчивые изображения	116
Проблема, присущая отзывчивым изображениям	116
Простое переключение разрешения с помощью <code>srcset</code>	117
Более совершенный вариант переключения с помощью <code>srcset</code> и <code>sizes</code>	117
Управление элементом <code>picture</code>	119
Итоги	120
Глава 5. CSS Grid	121
Что такое CSS Grid и какие проблемы она решает	121
Базовый синтаксис Grid	122
Терминология, специфичная для Grid	122
Настройка сетки	123
Явное и неявное позиционирование элементов	127
Размещение и изменение размера элементов сетки	130
Свойство <code>gap</code>	133
Функция <code>repeat()</code>	133
Единица <code>fr</code>	133
Размещение элементов в сетке	134
Ключевое слово <code>span</code>	134
Ключевое слово <code>dense</code>	135
Именованные линии сетки	136
Свойство <code>grid-template-areas</code>	139
Применение ваших знаний на практике	140
Ключевые слова <code>auto-fit</code> и <code>auto-fill</code>	141
Функция <code>minmax()</code>	143
Сокращения в синтаксисе	144
Сокращение <code>grid-template</code>	145
Сокращение <code>grid</code>	145
Итоги	148
Глава 6. CSS-селекторы, типографика, цветовые режимы и многое другое	149
Селекторы, единицы измерения и возможности	150
Анатомия правила CSS	150
Псевдоэлементы и псевдоклассы	151
Селекторы CSS Level 3 и как их использовать	152

Структурные псевдоклассы	156
Отзывчивые меры длины, выражаемые в процентных отношениях, применительно к окнам просмотра (vmax, vmin, vh, vw)	166
Функция calc()	167
Пользовательские свойства CSS	167
Использование @supports для создания альтернативных вариантов	171
Веб-тиографика	173
Системные шрифты	173
CSS-правило @font-face	174
Реализация веб-шрифтов с помощью @font-face	175
Оптимизация загрузки шрифтов с помощью @font-face	176
Вариативные шрифты	178
Форматы цвета в CSS и альфа-прозрачность	183
Цвет в формате RGB	183
Цвет в формате HSL	184
Итоги	186
Глава 7. Потрясающая эстетика с помощью CSS	187
Создание теней для текста средствами CSS	188
Если размытие не нужно, его значение можно опустить	189
Создание теней для блоков	189
Тень внутри элемента	190
Создание нескольких теней	190
Понятие протяженности	191
Градиентные фоны	192
Запись линейного градиента	192
Радиальные градиентные фоны	195
Удобные ключевые слова «распространения» для задания размеров отзывчивых конструкций	196
Повторяющиеся градиенты	197
Паттерны градиентных фонов	198
Использование нескольких фоновых изображений	199
Размер фона	200
Позиционирование фона	200
Краткая запись настроек фона	201
Фоновые изображения с высоким разрешением	202
CSS-фильтры	202
Доступные CSS-фильтры	203
Объединение CSS-фильтров	208

Предупреждения, касающиеся CSS-производительности	209
CSS-свойство <code>clip-path</code>	210
Свойство <code>clip-path</code> с URL-адресом	210
Базовые формы CSS	210
Анимация с помощью свойства <code>clip-path</code>	214
Изображение в качестве маски	215
Пример изображения-маски	215
Свойство <code>mix-blend-mode</code>	217
Итоги	218
 Глава 8. SVG и независимость от разрешения	219
Краткая история SVG	221
Изображение — считываемый веб-документ	222
Корневой элемент SVG	223
Пространство имен	224
Теги <code>title</code> и <code>desc</code>	225
Тег <code>defs</code>	225
Элемент <code>g</code>	225
Фигуры SVG	225
SVG-пути	226
Создание SVG-графики с помощью популярных пакетов и сервисов редактирования изображений	226
Сервисы SVG-значков экономят время	226
Вставка SVG-графики в веб-страницы	227
Использование тега <code>img</code>	228
Использование тега <code>object</code>	228
Вставка SVG-графики в качестве фонового изображения	229
Краткое отступление о URI-идентификаторах данных	229
Создание спрайтов изображений	230
Непосредственная вставка SVG	231
Повторное использование графических объектов из символов	231
Встраиваемая в код SVG-графика позволяет задавать разные цвета в разных контекстах	233
Изменение цветов SVG-изображений с помощью пользовательских свойств CSS	234
Повторное использование графических объектов из внешних источников	236
На что влияет способ вставки SVG-данных?	237
Особенности браузеров	238

Дополнительные возможности и особенности технологии SVG	239
SMIL-анимация	239
Задание стилей SVG с помощью внешней таблицы стилей	241
Задание стилей SVG с помощью внутренних стилей	242
Анимация SVG-графики с помощью CSS	242
Анимация SVG-графики с помощью JavaScript	244
Простой пример анимации SVG-графики с помощью GreenSock	244
Оптимизация SVG	246
Использование SVG в качестве фильтров	247
Медиазапросы внутри SVG	250
Советы по внедрению	251
Итоги	251
Дополнительные ресурсы	252
 Глава 9. Переходы, преобразование и анимация	253
Что такое CSS-переходы и как ими можно воспользоваться	254
Свойства перехода	256
Краткая форма записи перехода с помощью свойства transition	257
Переходы различных свойств за разные периоды	257
Функции развития процесса перехода по времени	258
2D-преобразования в CSS	260
Масштабирование (scale)	262
Перемещение (translate)	262
Поворот (rotate)	265
Наклон (skew)	265
Матрица (matrix)	266
Свойство transform-origin	267
3D-преобразования в CSS	268
Свойство transform3d	272
Использование преобразований при постепенном улучшении на примере функции translate3d	273
Эффекты анимации с помощью CSS	276
Свойство animation-fill-mode	278
Упражнения и практика	280
Итоги	280

Глава 10. Освоение форм с помощью HTML5 и CSS	281
Формы HTML5	281
Основные сведения о компонентах формы HTML5	283
Атрибут placeholder	283
Стилизация указателя ввода с помощью свойства caret-color	284
Атрибут required	285
Атрибут autofocus	285
Атрибут autocomplete	287
Атрибут list и элемент datalist	287
Типы вводимой информации, определяемые HTML5	289
Тип email	289
Тип number	291
Тип url	293
Тип tel	294
Тип search	296
Атрибут pattern	296
Тип color	298
Типы date и time	298
Тип range	301
Стилизация форм HTML5 с помощью CSS	302
Обозначение полей, требующих обязательного заполнения	305
Создание эффекта заливки фона	307
Итоги	308
Глава 11. Бонусные техники и советы	309
Разбивка длинных URL-адресов	310
Обрезка текста	311
Создание панелей с горизонтальной прокруткой	312
Создание панелей с горизонтальной прокруткой с помощью Grid	314
Модуль CSS Scroll Snap	315
Свойство scroll-snap-type	315
Свойство scroll-snap-align	316
Свойство scroll-padding	317
Плавная прокрутка с помощью свойства scroll-behavior	319
Привязка брейкпоинтов CSS к JavaScript	319
Обкатка веб-дизайна в браузере на самых ранних стадиях	323
Тестирование на реальных устройствах	323

Использование принципа постепенного улучшения	324
Определение матрицы браузерной поддержки	324
Функциональное, но не эстетическое единство.....	325
Выбор поддерживаемых браузеров	325
Создание нескольких уровней пользовательского восприятия ..	326
Отказ от использования сред разработки CSS при создании конечного продукта	326
Скрытие, показ и загрузка содержимого для разных окон просмотра ..	327
Средства контроля качества кода	328
Повышение производительности	329
Инструменты для оценки производительности	330
В преддверии великих перемен	331
Итоги	332

Медиавозможности, встроенные в HTML5

Многие услышали об HTML5, когда Apple отказалась добавлять поддержку Flash на iOS-устройства. Технология Flash доминировала на рынке (некоторые утверждают, что она его и вовсе захватила) в качестве дополнительного модуля для обслуживания видео в браузере. Но вместо использования технологии от Adobe Apple в вопросах вывода на экран сложного медиаcontentа решила положиться на HTML5. HTML5 неплохо справился с задачей, а публичная поддержка со стороны Apple дала этой версии могучий толчок и сделала медиаинструменты HTML5 привлекательными для широкой аудитории.

Мы уже обсуждали, что сегодня люди склонны использовать термин «HTML» вместо «HTML5», но в отношении медиа различие имело значение. До HTML5 добавление видео и аудио в разметку было не таким удобным. Теперь эта задача упростилась.

Добавление видео и аудио в разметку HTML

Работать с видео и аудио в HTML довольно просто. Вот пример ссылки на видеофайл из разряда «проще некуда»:

```
<video src="myVideo.mp4"></video>
```

В HTML со всей трудной работой справляется один-единственный элемент `<video></video>` (или `<audio></audio>` для аудио). Между открывающим и закрывающим тегами можно вставить текст: он будет отображаться, если у пользователя возникнут проблемы с загрузкой. Можно добавить дополнительные атрибуты, например задать высоту и ширину. Сделаем это:

```
<video src="myVideo.mp4" width="640" height="480">If you're reading this either the video didn't load or your browser is waaaayyyyy old!</video>
```

Теперь, если добавить предыдущий фрагмент кода на страницу и посмотреть на его работу в любом браузере, видео появится, но без элементов управления его воспроизведением. Чтобы получить возможность управлять видео, нужно добавить атрибут `controls`. В качестве примера можно добавить атрибут `autoplay`, но вообще этого лучше не делать: автоворонение выбесит кого угодно! Вот как выглядит код с этими атрибутами:

```
<video src="myVideo.mp4" width="640" height="480" controls autoplay>If you're reading this either the video didn't load or your browser is waaaayyyyy old!</video>
```

Результат выполнения этого фрагмента показан на скриншоте.



Рис. 2.3. Мы вставили видео на страницу, добавив минимум кода

В число прочих атрибутов входят `preload` для управления предварительной загрузкой медиа, `loop` для повторного воспроизведения и `poster` для определения кадра заставки для видео (изображение, отображающееся во время загрузки видео). Для применения атрибута достаточно вставить его в тег. Вот как выглядит пример, включающий все эти атрибуты:

```
<video src="myVideo.mp4" width="640" height="480" controls autoplay  
preload="auto" loop poster="myVideoPoster.png"> If you're reading  
this either the video didn't load or your browser is waaaayyyyyy  
old!</video>
```

Резервные возможности

Элемент `<source>` позволяет при необходимости добавить резервные возможности. Например, наряду с видео в формате MP4 можно обеспечить поддержку другого формата. Кроме того, если у пользователя в браузере нет подходящей технологии проигрывания, можно предоставить ссылки на скачивание видеофайлов. Рассмотрим такой пример:

```
<video width="640" height="480" controls preload="auto" loop  
poster="myVideoPoster.png">  
    <source src="myVideo.sp8" type="video/super8" />
```

```
<source src="myVideo.mp4" type="video/mp4" />
<p><b>Download Video:</b> MP4 Format: <a href="myVideo.mp4">"MP4"</a></p>
</video>
```

Здесь мы сначала указали выдуманный видеоформат `super8`. Браузер считывает код сверху вниз, решая, какой файл воспроизводить. Если он не поддерживает формат `super8`, то переходит к следующему — в нашем случае `mp4`. В итоге браузер переходит по ссылкам для скачивания, если не может согласовать какой-либо из перечисленных форматов. Атрибут `type` сообщает браузеру MIME-тип файла. Без этого атрибута браузер получит контент и все равно попытается воспроизвести его. Но если вы знаете MIME-тип, лучше добавьте его. Предыдущий пример кода и образец видеофайла в формате MP4 (по чистой случайности это фрагмент сериала Coronation Street, в котором я снимался; тогда у меня еще были волосы и надежды сыграть главную роль вместе с Де Ниро) находятся в разделе `example2` кода для этой главы.

Работа `audio` и `video` практически ничем не различается

Элемент `<audio>` работает по таким же принципам и с такими же атрибутами (исключая `width`, `height` и `poster`), но у `<audio>` отсутствует область проигрывания визуального содержимого.

Отзывчивое HTML5-видео и iFrames

Единственная проблема со столь понравившейся нам реализацией видео в HTML5 — в ней нет отзывчивости. И действительно, в книге приведен пример отзывчивой HTML5- и CSS-разметки, которая не реагирует на изменение условий просмотра. К счастью, для встроенного HTML-видео это можно легко исправить. Просто уберите из разметки все атрибуты высоты и ширины (например, удалите `width="640"` `height="480"`) и добавьте в CSS следующий код:

```
video {
  max-width: 100%;
  height: auto;
}
```

Да, с файлами, которые могут храниться на локальном устройстве, этот прием работает вполне успешно. Но он не решает проблемы видео, встроенного в iFrame (низкий поклон YouTube, Vimeo и другим сайтам). Следующий код позволяет добавить трейлер фильма «Успеть до полуночи» из YouTube:

```
<iframe width="960" height="720" src="https://www.youtube.com/embed/B1_N28DA3gY" frameborder="0" allowfullscreen></iframe>
```

Но если добавлять этот код к странице в неизменном виде, то даже при использовании CSS-правила произойдет обрезка, если ширина окна просмотра будет менее 960 пикселей.

Проще всего решить эту проблему с помощью небольшого CSS-трюка, впервые примененного французским CSS-специалистом Тьерри Кобленцем (Thierry Ko-

blentz), который создал, по сути, блок с правильным соотношением сторон для содержащегося в нем видео. Собственные объяснения этого мага можно найти по ссылке <http://alistapart.com/article/creating-intrinsic-ratios-for-video>.

Если лень самим вычислять и подставлять соотношение сторон, можно не заморачиваться: один онлайн-сервис способен сделать это за вас. Просто откройте <http://embedresponsively.com/> и поместите в адресную строку этого сайта URL-адрес вашего iFrame. В результате вы получите простой фрагмент кода, который можно вставить в собственную разметку.

К примеру, для трейлера фильма «Успеть до полуночи» ресурс выдает следующее (обратите внимание на значение `padding-bottom` для определения соотношения сторон):

```
<style>
  .embed-container {
    position: relative;
    padding-bottom: 56.25%;
    height: 0;
    overflow: hidden;
    max-width: 100%;
    height: auto;
  }
  .embed-container iframe,
  .embed-container object,
  .embed-container embed {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
  }
</style>
<div class="embed-container">
  <iframe
    src="http://www.youtube.com/embed/B1_N28DA3gY"
    frameborder="0"
    allowfullscreen
  ></iframe>
</div>
```

Вот и все! Просто добавьте этот фрагмент в код и получите полностью адаптируемое YouTube-видео (примечание: дети, не берите пример с мистера Де Ниро — курить вредно!).

Итоги

В этой главе мы рассмотрели много всего, начиная с основ создания страницы, проходящей проверку на соответствие стандартам HTML5, и заканчивая встраи-

ваемым в разметку сложным медиа содержимым (видео) и вопросами обеспечения его отзывчивого поведения. Все это и не имеет отношения к отзывчивому веб-дизайну, но зато мы познакомились с семантикой в коде, смысловым наполнением страниц и обеспечением их пригодности для пользователей, применяющих вспомогательные технологии.

Упражнение

Мы рассмотрели множество HTML-элементов. Это лишь малая часть всех существующих элементов, но мы, безусловно, рассмотрели те из них, которые могут вам понадобиться в ежедневной работе. Предлагаю сделать небольшое упражнение, чтобы увидеть, как много вы поняли. Вот скриншот с веб-дизайном сайта для этой книги:



Рис. 2.4. Веб-дизайн сайта, разработанного для этой книги



Если у вас macOS, вы можете получить исходный файл Sketch: он включен в загружаемый код под названием RWD3e_design.sketch.

Взгляните на дизайн и попробуйте создать для него HTML-страницу. Рассмотрите часть заголовка. Подумайте о языке контента и о метатегах, которые могут вам понадобиться. Затем подумайте о визуальных эффектах. Какие элементы лучше использовать для создания раздела навигации? Как быть с каждым из этих маленьких разделов под изображением книги? А как насчет блока DOWNLOAD CODE? Есть идеи, как его разметить?

Сайт <https://rwd.education> покажет, что выбрал я. Но постарайтесь не заглядывать туда, пока не попробуете сами!