Оглавление

Предисловие ... 16

Введение ... 18

Благодарности ... 20

Об этой книге ... 22
- Для кого предназначена книга ... 22
- Структура издания ... 23
- Условные обозначения и файлы примеров ... 24
- Версии браузеров ... 25
- Об авторе ... 25
- Иллюстрация на обложке ... 26

Часть I. Обзор основных принципов

Глава 1. Каскадность, специфичность и наследование ... 28
- 1.1. Каскадность ... 29
  - 1.1.1. Источник стилей ... 33
  - 1.1.2. Специфичность селекторов ... 36
  - 1.1.3. Исходный порядок ... 41
  - 1.1.4. Два правила ... 44
- 1.2. Наследование ... 45
- 1.3. Специальные значения ... 48
  - 1.3.1. Ключевое слово inherit ... 48
  - 1.3.2. Ключевое слово initial ... 49

1.4. Сокращенная запись свойств ... 50
1.4.1. Остерегайтесь сокращений, скрыто переопределяющих другие стили ... 51
1.4.2. Порядок записи сокращенных значений ... 52
Итоги главы ... 54

Глава 2. Работа с относительными единицами ... 55
2.1. Мощь относительных значений ... 55
2.1.1. Борьба за pixel-perfect-дизайн ... 56
2.1.2. Конец эпохи pixel-perfect ... 56
2.2. Единицы em и rem ... 58
2.2.1. Единицы em для указания размера шрифта ... 59
2.2.2. Указание размера шрифта в единицах rem ... 63
2.3. Перестаньте думать в пикселах ... 65
2.3.1. Установка адекватного размера шрифта по умолчанию ... 66
2.3.2. Делаем панель адаптивной ... 68
2.3.3. Изменение размера отдельного компонента ... 69
2.4. Единицы измерения, относящиеся к размеру экрана устройства ... 71
2.4.1. Единицы vw для указания размера шрифта ... 74
2.4.2. Функция calc() для указания размера шрифта ... 74
2.5. Числа без единиц измерения и свойство line-height ... 75
2.6. Пользовательские свойства (или CSS-переменные) ... 77
2.6.1. Динамическое изменение пользовательских свойств ... 79
2.6.2. Изменение пользовательских свойств с помощью JavaScript ... 82
2.6.3. Экспериментирование с пользовательскими свойствами ... 83
Итоги главы ... 84

Глава 3. Знакомство с блочной моделью ... 85
3.1. Трудности с шириной элемента ... 86
3.1.1. Избегаем волшебных чисел ... 89
3.1.2. Настройка блочной модели ... 90
3.1.3. Глобальное применение свойства box-sizing: border-box ... 91
3.1.4. Добавление зазора между колонками ... 93
3.2. Проблемы высоты элементов ... 94
3.2.1. Управление переполнением ... 95
3.2.2. Применение альтернатив к высотам, указанным в процентах ... 96
3.2.3. Свойства min-height и max-height ... 101
3.2.4. Центрирование контента по вертикали ... 102

3.3. Отрицательные значения полей ... 104
3.4. Схлопывание полей ... 105
3.4.1. Схлопывание между текстом ... 106
3.4.2. Схлопывание многочисленных полей ... 106
3.4.3. Схлопывание вне контейнера ... 107
3.5. Расстояние между элементами в контейнере ... 109
3.5.1. Учет изменения контента ... 111
3.5.2. Универсальное решение: селектор лоботомированной совы ... 112
Итоги главы ... 116

Часть II. Разметка

Глава 4. Плавающие элементы ... 118
4.1. Предназначение плавающих элементов ... 118
4.2. Схлопывание контейнера и clearfix ... 125
4.2.1. Что такое схлопывание контейнера ... 125
4.2.2. Что такое clearfix ... 128
4.3. Неожиданный «захват» плавающего элемента ... 131
4.4. Медиаобъекты и блочный контекст форматирования ... 134
4.4.1. Что такое блочный контекст форматирования ... 135
4.4.2. Использование блочного контекста форматирования для разметки медиаобъектов ... 136
4.5. CSS-сетки ... 138
4.5.1. Принципы CSS-сетки ... 139
4.5.2. Создание CSS-сетки ... 139
4.5.3. Добавление зазоров ... 144
Итоги главы ... 148

Глава 5. Flexbox-верстка ... 149
5.1. Принципы flexbox-верстки ... 150
5.1.1. Создание базовой flex-навигации ... 153
5.1.2. Добавление отступов и промежутков ... 156
5.2. Размеры flex-элементов ... 157
5.2.1. Свойство flex-basis ... 160
5.2.2. Свойство flex-grow ... 160
5.2.3. Свойство flex-shrink ... 162
5.2.4. Практические примеры ... 163

5.3. Направление flex-элементов ... 165
5.3.1. Изменение направления flex-элементов ... 167
5.3.2. Стилевое форматирование формы авторизации ... 168
5.4. Выравнивание, промежутки и другие штрихи ... 170
5.4.1. Свойства flex-контейнеров ... 174
5.4.2. Свойства flex-элементов ... 175
5.4.3. Выравнивание flex-блоков ... 176
5.5. Пара вещей, о которых следует знать ... 178
5.5.1. Flex-баги ... 178
5.5.2. Полноформатная разметка ... 179
Итоги главы ... 179

Глава 6. CSS-сетки ... 181
6.1. Веб-разметка уже здесь ... 182
6.1.1. Создание базовой сетки ... 183
6.2. Анатомия сетки ... 185
6.2.1. Нумерация линий сетки ... 191
6.2.2. Совместная работа с flex-блоками ... 192
6.3. Альтернативный синтаксис ... 196
6.3.1. Присвоение имен линиям сетки ... 196
6.3.2. Присвоение имен областям сетки ... 198
6.4. Явная и неявная сетка ... 200
6.4.1. Внесем разнообразие ... 204
6.4.2. Подгонка элементов для заполнения полосы сетки ... 207
6.5. Запросы функций ... 210
6.6. Выравнивание ... 213
Итоги главы ... 216

Глава 7. Контексты позиционирования и наложения ... 217
7.1. Фиксированное позиционирование ... 218
7.1.1. Создание модального окна с фиксированным позиционированием ... 218
7.1.2. Управление размером позиционированных элементов ... 221
7.2. Абсолютное позиционирование ... 222
7.2.1. Абсолютное позиционирование кнопки Закрыть ... 222
7.2.2. Позиционирование псевдоэлементов ... 223
7.3. Относительное позиционирование ... 225
7.3.1. Создание раскрывающегося меню ... 226
7.3.2. Создание треугольника CSS ... 229

7.4. Контексты наложения и z-индекса ... 231
7.4.1. Процесс рендеринга и порядок наложения ... 232
7.4.2. Управление наложением с помощью свойства z-index ... 234
7.4.3. Контексты наложения ... 235
7.5. Липкое позиционирование ... 239
Итоги главы ... 242

Глава 8. Адаптивный дизайн ... 243
8.1. Подход Mobile First ... 244
8.1.1. Создание мобильного меню ... 252
8.1.2. Добавление метатега viewport ... 257
8.2. Медиазапросы ... 258
8.2.1. Типы медиазапросов ... 260
8.2.2. Добавление контрольных точек на страницу ... 262
8.2.3. Добавление адаптивных колонок ... 266
8.3. Резиновые макеты ... 268
8.3.1. Добавление стилей для большой области просмотра ... 269
8.3.2. Работа с таблицами ... 272
8.4. Адаптивные изображения ... 273
8.4.1. Использование нескольких изображений для экранов разных размеров ... 274
8.4.2. Использование атрибута srcset для передачи нужного изображения ... 275
Итоги главы ... 276

Часть III. Масштабируемый CSS-код

Глава 9. Модульный CSS ... 278
9.1. Базовые стили: закладываем основы ... 279
9.2. Простой модуль ... 281
9.2.1. Вариации модуля ... 282
9.2.2. Модули с множеством элементов ... 287
9.3. Составление крупных структур из модулей ... 290
9.3.1. Разделение ответственности между модулями ... 290
9.3.2. Именование модулей ... 295
9.4. Вспомогательные классы ... 296
9.5. Методологии CSS ... 297
Итоги главы ... 299

Глава 10. Библиотеки компонентов ... 300
10.1. Введение в KSS ... 301
10.1.1. Установка KSS ... 302
10.1.2. Написание KSS-документации ... 304
10.1.3. Документирование вариаций модуля ... 308
10.1.4. Создание начальной страницы ... 311
10.1.5. Документирование модулей, которым требуется JavaScript ... 311
10.1.6. Упорядочение контента библиотеки компонентов по разделам ... 314
10.2 Инновационный способ верстки CSS ... 316
10.2.1. Метод CSS First ... 317
10.2.2. Библиотека компонентов в качестве API ... 318
Итоги главы ... 324

Часть IV. Темы повышенной сложности

Глава 11. Фоны, тени и режимы смешивания ... 326
11.1. Градиенты ... 327
11.1.1. Использование нескольких цветовых узлов ... 329
11.1.2. Использование радиального градиента ... 332
11.2. Тени ... 334
11.2.1. Создание объема с помощью градиентов и теней ... 335
11.2.2. Элементы с плоским дизайном ... 336
11.2.3. Создание кнопок с более современным дизайном ... 337
11.3. Режимы смешивания ... 338
11.3.1. Изменение оттенка изображения ... 341
11.3.2. Виды режимов смешивания ... 342
11.3.3. Добавление текстуры изображению ... 343
11.3.4. Микширование режимов смешивания ... 345
Итоги главы ... 347

Глава 12. Контраст, цвета и интервалы ... 348
12.1. Царство контраста ... 349
12.1.1. Создание шаблона ... 351
12.1.2. Реализация дизайна ... 352

12.2. Цвета ... 353
12.2.1. Разбираемся с нотациями цветов ... 360
12.2.2. Добавление цветов в палитру ... 364
12.2.3. Применение контраста при выборе цвета текста ... 367
12.3. Интервалы ... 368
12.3.1. Единицы em или пикселы? ... 369
12.3.2. Вычисление высоты строки ... 371
12.3.3. Интервалы между строчными элементами ... 374
Итоги главы ... 377

Глава 13. Шрифтовое оформление ... 378
13.1. Веб-шрифты ... 380
13.2. Сервис Google Fonts ... 381
13.3. Как работает свойство @font-face ... 386
13.3.1. Форматы шрифтов и замена при необходимости ... 387
13.3.2. Варианты начертания в одной гарнитуре ... 388
13.4. Управление интервалами в целях повышения читаемости ... 390
13.4.1. Интервалы основного текста сайта ... 391
13.4.2. Заголовки, мелкие элементы и интервалы ... 392
13.5. Вспышки нестилизованного и невидимого текста ... 397
13.5.1. Библиотека Font Face Observer ... 398
13.5.2. Откат к системным шрифтам ... 400
13.5.3. И наконец, свойство font-display ... 402
Итоги главы ... 403

Глава 14. Переходы ... 404
14.1. Отсюда сюда ... 404
14.2. Функции времени ... 407
14.2.1. Изучение кривых Безье ... 409
14.2.2. Шаги ... 412
14.3. Неанимируемые свойства ... 413
14.3.1. Свойства, которые нельзя анимировать ... 416
14.3.2. Появление и исчезновение ... 417
14.4. Переход к автоматическому выравниванию высоты ... 419
Итоги главы ... 421

Глава 15. Трансформации ... 422
15.1. Вращение, масштабирование, смещение и наклон ... 422
15.1.1. Изменение точки трансформации ... 426
15.1.2. Применение нескольких трансформаций ... 426
15.2. Анимированные трансформации ... 427
15.2.1. Масштабирование значков ... 433
15.2.2. Создание «вылетающих» меток ... 436
15.2.3. Поэтапные переходы ... 438
15.3. Производительность анимации ... 439
15.3.1. Рендеринг страницы ... 440
15.4. Трехмерные (3D) трансформации ... 442
15.4.1. Контроль перспективы ... 443
15.4.2. Профессиональные приемы 3D-трансформации ... 446
Итоги главы ... 449

Глава 16. Анимация ... 450
16.1. Ключевые кадры ... 451
16.2. Анимация 3D-трансформаций ... 454
16.2.1. Создание макета без анимации ... 454
16.2.2. Добавление анимации в макет ... 459
16.3. Задержка запуска анимации и режим заполнения ... 461
16.4. Передача смысла с помощью анимации ... 464
16.4.1. Реакция на действие пользователя ... 464
16.4.2. Привлечение внимания пользователя ... 468
16.5. Совет напоследок ... 471
Итоги главы ... 472

Приложения

Приложение А. Селекторы ... 474
А.1. Базовые селекторы ... 474
А.2. Комбинаторы ... 474
А.3. Селекторы псевдоклассов ... 475
А.4. Селекторы псевдоэлементов ... 477
А.5. Селекторы атрибутов ... 478

Приложение Б. Препроцессоры 479
Б.1. Препроцессор Sass 480
Б.1.1. Установка препроцессора Sass 480
Б.1.2. Запуск препроцессора Sass 481
Б.1.3. Важные функции препроцессора Sass 482
Б.2. PostCSS 491
Б.2.1. Использование инструмента Autoprefixer 491
Б.2.2. Применение cssnext 492
Б.2.3. Использование cssnano 492
Б.2.4. Использование PreCSS 493

1 Каскадность, специфичность и наследование

В этой главе

- Четыре компонента каскадности.
- Разница между каскадностью и наследованием.
- Управление стилями, применяемыми к элементам.
- Распространенные недоразумения с сокращенными объявлениями.

CSS отличается от многих средств разработки программного обеспечения. Строго говоря, это не язык программирования, хотя он требует абстрактного мышления. Это не исключительно инструмент дизайна, хотя понадобится толика креатива. Каскадные таблицы стилей обеспечивают обманчиво простой декларативный синтаксис, но если вы использовали его при работе над крупными проектами, то знаете, что он способен перерасти в серьезную проблему.

Если требуется научиться делать что-то в обычном программировании, вы чаще всего знаете, что искать (например, «как найти элементы типа *x* в массиве»). С CSS не всегда легко решить проблему с помощью только одного запроса в Интернете. Даже когда вы можете это сделать, ответ часто от чего-то зависит. Чтобы добиться чего-либо, нередко нужно уметь обрабатывать разнообразные ситуации.

Первая часть книги начинается с рассмотрения фундаментальных принципов языка: каскадности, блочной модели и широкого набора доступных единиц измерения. Большинство веб-разработчиков знают о каскадности и блочной модели. Они знают о пикселах как о единицах, в которых можно измерять размер элементов, и, вероятно, слышали, что должны использовать единицы em. Но суть в том, что подобных тем очень много и поверхностного их рассмотрения недостаточно. Если вы решили изучить CSS, то должны сначала разобраться в фундаментальных понятиях, причем досконально.

Я знаю, что вам не терпится начать изучать современные и красивые правила CSS. Это очень интересный материал. Но сначала обратимся к основам. Я кратко расскажу о базовых вещах, с которыми вы, вероятно, уже знакомы, а затем мы подробно разберем каждую тему. Моя цель — укрепить фундамент, на котором строится ваше понимание CSS.

В этой главе начнем с каскадности. Я сформулирую, как она действует, а затем покажу, как работать с каскадностью на практике. После этого мы рассмотрим смежную тему — наследование. Я буду придерживаться этого плана, говоря о свойствах и некоторых распространенных недоразумениях, возникающих по поводу них.

Все эти темы касаются применения желаемых стилей к нужным элементам. Здесь можно допустить много нелепых ошибок. Глубокое понимание этих тем позволит лучше контролировать работу вашего CSS-кода, чтобы он делал то, что вы хотите. Если повезет, вам даже понравится работать с CSS.

1.1. Каскадность

По сути, CSS — это объявленные правила: мы хотим, чтобы в различных условиях происходили определенные вещи. Если этот класс добавлен к тому элементу — применить такие-то стили. Если элемент *X* является потомком элемента *Y* — вот эти. Затем браузер принимает эти правила, определяет, какие из них действуют, в каком порядке и в каком месте, и использует их при визуализации страницы.

Когда рассматриваются небольшие примеры, этот процесс обычно не вызывает сложностей. Но по мере увеличения размера таблицы стилей или количества веб-страниц, к которым вы ее применяете, код может неожиданно быстро усложниться. В CSS существует несколько способов делать одно и то же. При изменении структуры HTML или использовании стилей на разных страницах результаты могут оказаться совершенно разными. Это зависит от того, какое решение задействовано. Ключевая часть разработки CSS сводится к написанию правил таким образом, чтобы они были предсказуемыми.

Первый шаг — разобраться в том, как именно браузер понимает ваши правила. Каждое из них может быть простым, но что происходит, когда два правила противоречат друг другу в том, как форматировать элемент? Вы можете обнаружить, что одно из ваших правил не делает того, что вы ожидаете, поскольку с ним конфликтует другое правило.

Предсказывать, как поведут себя правила, реально, если понимать принципы каскадности. Продемонстрирую это. Создайте простенькую шапку типа той, которая находится в верхней части веб-страницы (рис. 1.1). В шапке указано название сайта, под ним — навигационные ссылки. Последняя ссылка окрашена в оранжевый цвет — так выделяются важные элементы. (Внимание: в печатной версии книги изображения черно-белые.)

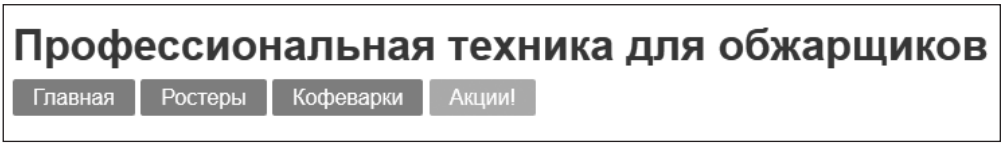


Рис. 1.1. Название страницы и навигационные ссылки

Для начала создайте HTML-документ и таблицу стилей с именем `styles.css`. Добавьте код, показанный в листинге 1.1, в HTML-файл.

ПРИМЕЧАНИЕ

Репозиторий, содержащий все примеры для этой книги, доступен для загрузки по адресу github.com/CSSInDepth/css-in-depth. В нем вы найдете HTML-файлы со встроенными правилами CSS.

Листинг 1.1. Разметка для шапки страницы

```
<!doctype html>
<head>
  <meta charset="utf-8">
  <link href="styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <header class="page-header">
    <h1 id="page-title" class="title">Профессиональная
      техника для обжарщиков</h1>
    <nav>
      <ul id="main-nav" class="nav">
        <li><a href="/">Главная</a></li>
        <li><a href="/coffees">Ростеры</a></li>
        <li><a href="/brewers">Кофеварки</a></li>
        <li><a href="/specials" class="featured">Акции!</a></li>
      </ul>
    </nav>
  </header>
</body>
```

Название страницы

Список навигационных ссылок

Важная ссылка

Если к одному и тому же элементу страницы применяется два правила или больше, это может привести к конфликту объявлений. Листинг 1.2 показывает, как это происходит. В нем три набора правил задают шрифт для названия страницы. Название не может отображаться тремя разными шрифтами одновременно. Какой конкретно будет использован? Добавьте этот листинг в свой CSS-файл, чтобы узнать ответ.

Инструкции с конфликтующими объявлениями могут следовать одна за другой или оказаться разбросанными по всей таблице стилей. В любом случае, учитывая HTML-код, все они нацелены на один и тот же элемент.

Листинг 1.2. Конфликтующие объявления

```
h1 {                                  <-- Селектор тега
  font-family: serif;
}

#page-title {                         <-- Селектор идентификатора
  font-family: sans-serif;
}

.title {                              <-- Селектор класса
  font-family: monospace;
}
```

Все три набора правил пытаются установить для названия собственное семейство шрифтов. Кто победит? Чтобы получить ответ, браузер следует конкретному набору правил, поэтому результат оказывается предсказуем. В этом случае правила указывают, что выигрывает второе объявление, с селектором идентификатора. Название будет оформлено рубленым шрифтом (sans-serif), применяемым по умолчанию (рис. 1.2).

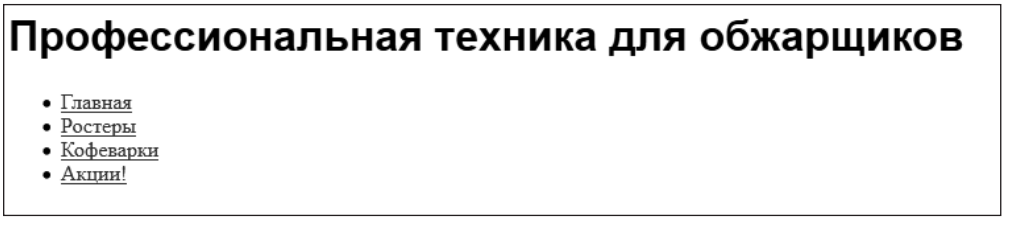


Рис. 1.2. Селектор идентификатора выигрывает у других наборов правил, отображая название страницы рубленым шрифтом

Каскадность — так называется этот набор правил. Каскадность определяет способы разрешения конфликтов, и это фундаментальная часть работы CSS. Большинство опытных разработчиков имеют общее представление о каскадности, однако не всегда правильно интерпретируют ее.

Разберемся с каскадностью. Когда объявления конфликтуют, для устранения проблемы нужно учесть три показателя.

1. *Источник стилей* — место их расположения. Ваши стили накладываются на стили браузера, применяемые по умолчанию.
2. *Специфичность селекторов* — то, какие селекторы имеют приоритет над другими.
3. *Исходный порядок* — порядок, в котором стили объявляются в таблице стилей.

Правила каскадирования рассматривают их именно в этом порядке. Рисунок 1.3 обобщенно показывает, как они работают.

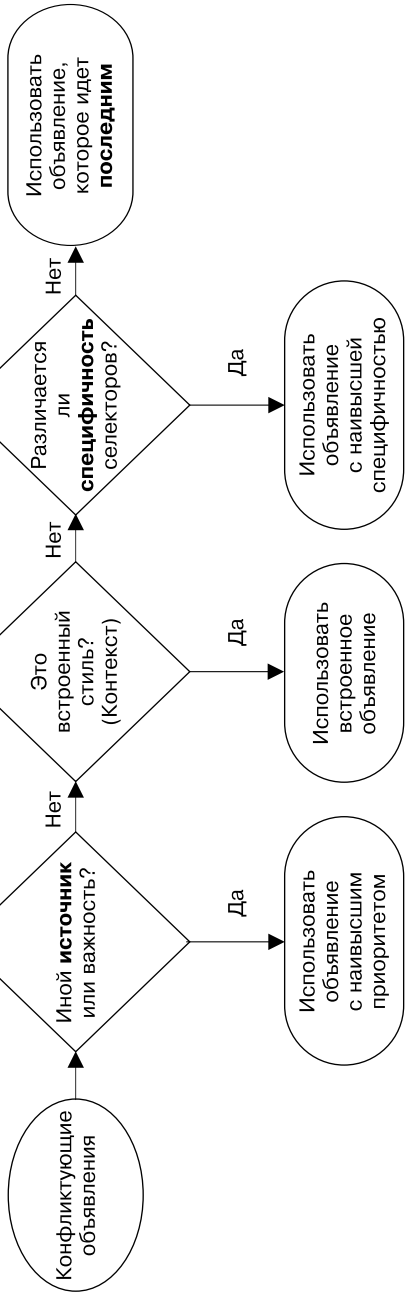


Рис. 1.3. Обобщенная блок-схема каскадности с указанием приоритета объявления

Эти правила определяют поведение браузера при появлении какой-либо неясности с CSS. Давайте разберемся с ними.

Быстрое ознакомление с терминологией

Возможно, вы уже знакомы с основной терминологией, относящейся к синтаксису CSS. Я не буду углубляться в эту тему, но, поскольку стану использовать термины в книге, напомню, что они означают.

Далее приведена строка CSS-кода. Она называется *объявлением*. *Объявление* состоит из *свойства* (`color`) и *значения* (`black`):

```
color: black;
```

Свойства не следует путать с *атрибутами*, которые относятся к синтаксису HTML. Например, в элементе `<a href="/">` часть `href` — это атрибут элемента `a`.

Группа объявлений внутри фигурных скобок называется *блоком объявлений*. Блоку объявлений предшествует *селектор* (в данном случае `body`):

```
body {
  color: black;
  font-family: Helvetica;
}
```

В совокупности селектор и блок объявлений называются *набором правил*. Набор правил также называют *правилом*, хотя, по моему мнению, это слово редко используется в таком смысле. Обычно оно стоит во множественном числе и обозначает набор стилей.

Наконец, *@правила* — это языковые конструкции, начинающиеся с символа `@`, такие как запросы `@import` или `@media`.

1.1.1. Источник стилей

Таблицы стилей, добавляемые вами на вашу веб-страницу, не единственные, к которым обращается браузер. Существуют разные типы, или источники, таблиц стилей. Созданные вами таблицы называются *авторскими*, существуют также *браузерные* стили — стили браузера по умолчанию. Браузерные стили имеют более низкий приоритет, поэтому ваши стили заменяют их.

ПРИМЕЧАНИЕ

Некоторые браузеры позволяют пользователям определять свои стили. Это третий источник стилей, по приоритету они занимают промежуточное положение между браузерными и авторскими стилями. Пользовательские стили применяются редко и не поддаются контролю, поэтому для упрощения я их пропускаю.

Браузерные стили различаются в разных браузерах, но обычно выполняют одни и те же задачи: форматируют заголовки (от `h1` до `h6`) и абзацы (`p`), задают верхнее и нижнее значения полей, форматируют списки (`ol` и `ul`), устанавливают отступ слева, определяют цвет ссылок и размеры шрифта по умолчанию.

Браузерные стили

Снова взглянем на страницу из примера (рис. 1.4). Название выполнено рубленым шрифтом, что определено теми стилями, которые вы добавили. Браузерные стили определяют ряд других параметров: для списка задан левый отступ, для отображения маркеров — свойство `list-style-type` со значением `disc`. Ссылки окрашены в синий цвет и подчеркнуты. У заголовка и списка есть верхние и нижние поля.

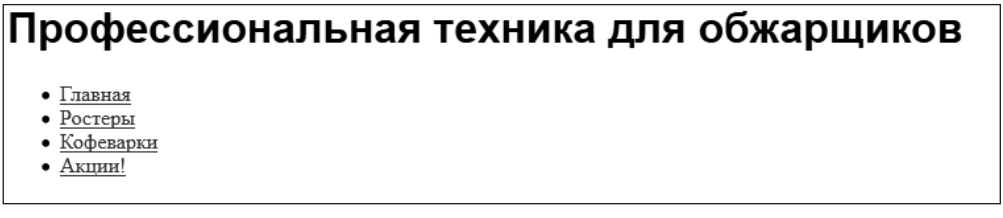


Рис. 1.4. Браузерные стили устанавливают форматирование по умолчанию для элементов на веб-странице

После применения браузерных стилей браузер задействует ваши авторские стили. Благодаря этому указанные вами объявления замещают браузерные. Если вы в своем HTML-файле ссылаетесь на несколько таблиц стилей, все они имеют один источник — авторский.

Обычно браузерные стили устанавливают подходящие параметры, поэтому ничего неожиданного не происходит. Если вам не нравится, как они влияют на то или иное свойство, задайте собственное значение в таблице стилей. Сделаем это сейчас. Можете переопределить некоторые браузерные стили, добавляющие не то форматирование, какое вы хотите, чтобы ваша страница выглядела, например, как на рис. 1.5.

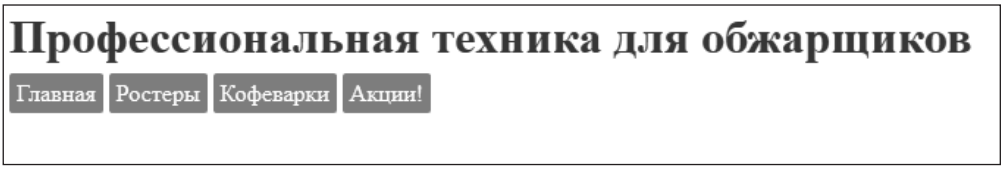


Рис. 1.5. Авторские стили переопределяют браузерные, поскольку имеют более высокий приоритет

В листинге 1.3 я удалил конфликтующие объявления `font-family` из предыдущего примера, добавил новые для установки цветов и переопределил браузерные настройки полей, отступа списка и маркеров.

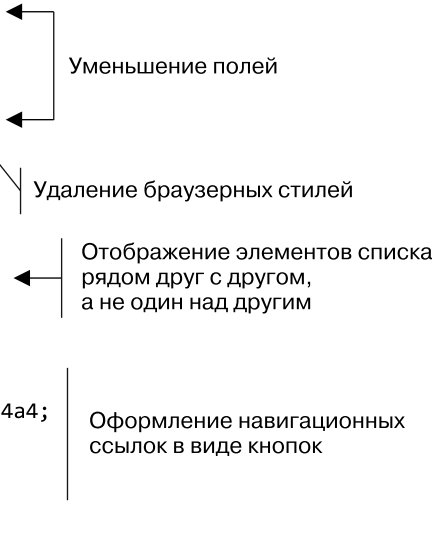
Листинг 1.3. Переопределение браузерных стилей

```
h1 {
  color: #2f4f4f;
  margin-bottom: 10px;
}

#main-nav {
  margin-top: 10px;
  list-style: none;
  padding-left: 0;
}

#main-nav li {
  display: inline-block;
}

#main-nav a {
  color: white;
  background-color: #13a4a4;
  padding: 5px;
  border-radius: 2px;
  text-decoration: none;
}
```



Отредактируйте таблицу стилей, чтобы она соответствовала этим изменениям.

Если вы уже давно работаете с CSS, то, вероятно, уже переопределяли браузерные стили. В этот момент вы пользовались таким принципом каскадности, как определение источника стиля. Ваш стиль всегда будет переопределять браузерный, потому что у них разные источники.

ПРИМЕЧАНИЕ

Возможно, вы заметили, что в данном коде я использовал селекторы идентификатора. Но в работе лучше не делать этого, о чем я вкратце еще расскажу.

Ключевое слово important

Объявления, отмеченные как *важные*, стоят особняком при определении источника стилей. Объявление может быть помечено как важное с помощью слова `!important`, указанного в конце, перед точкой с запятой:

```
color: red !important;
```

Объявление со словом `!important` рассматривается как источник с более высоким приоритетом. Далее стили перечислены в порядке убывания приоритета.

1. Важные авторские стили.
2. Авторские стили.
3. Браузерные стили.

Каскадность самостоятельно разрешает конфликты для всех свойств любого элемента на странице. Например, вы установили полужирный шрифт для абзацев текста. Тогда значения верхнего и нижнего полей будут применены согласно браузерным стилям, если вы явно не переопределите их. Приоритеты начинают играть важную роль, когда речь заходит об анимациях и переходах, так как те вводят дополнительные источники стилей. Аннотация `!important` — интересная причуда CSS, к которой мы еще вернемся.

1.1.2. Специфичность селекторов

Если конфликт объявлений не может быть разрешен на основании их источника, браузер попытается решить проблему, рассматривая их *специфичность*. Понять принципы специфичности очень важно. Вы можете долго работать, не вникая в источники стилей, потому что 99 % стилей на вашем сайте происходят из одного источника. Но если вы не знакомы с понятием специфичности, то дальше вам будет трудно. К сожалению, это часто упускают из виду.

Браузер оценивает специфичность в два этапа: сначала проверяет стили, встроенные в HTML-код (их еще называют строчными), затем стили, применяемые с помощью селекторов.

Встроенные стили

Если для задания стилей конкретного элемента в HTML-коде используется атрибут `style`, то объявления применяются только к данному элементу. Это, по сути, объявления с ограниченной областью действия, которые переопределяют любые объявления, задаваемые в вашей таблице стилей или теге `<style>`. Во встроенных стилях нет селектора, потому что они действуют непосредственно на элемент, на который нацелены.

Вы хотите, чтобы ссылка Акции! в меню навигации была оранжевой (рис. 1.6)? Рассмотрим несколько способов решения задачи, начиная с применения встроенных стилей (листинг 1.4).

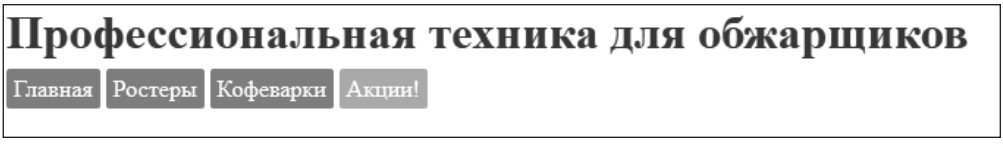


Рис. 1.6. Встроенные стили переопределяют стили, применяемые с помощью селекторов

Листинг 1.4. Встроенные стили, переопределяющие объявления из других источников

```
<li>
  <a href="/specials" class="featured"
    style="background-color: orange;">
    Акции!
  </a>
</li>
```

Встроенный стиль, применяемый с помощью атрибута style

Чтобы увидеть результат в своем браузере, отредактируйте код в соответствии с приведенным листингом. (Вы отмените это изменение за секунду.)

Чтобы переопределить встроенные объявления стилей, необходимо добавить к нужному объявлению ключевое слово `!important`, изменив его приоритет на более высокий. Если же встроенные стили отмечены как важные, их не получится так переопределить. Рекомендуется сделать это из таблицы стилей. Отмените изменение, и мы рассмотрим более удачные способы.

Специфичность селекторов

Вторая составляющая специфичности касается селекторов. Например, селектор с двумя классами более специфичен, чем с одним. Если одно объявление задает оранжевый фон, а другое, более специфичное, меняет его, скажем, на бирюзовый, то в браузере вы увидите бирюзовый цвет.

Посмотрим, что произойдет, если мы попытаемся окрасить ссылку в оранжевый цвет с помощью простого селектора классов. Обновите заключительную часть таблицы стилей, чтобы она соответствовала коду, приведенному в листинге 1.5.

Листинг 1.5. Селекторы разной специфичности

```
#main-nav a {
  color: white;
  background-color: #13a4a4;
  padding: 5px;
  border-radius: 2px;
  text-decoration: none;
}

.featured {
  background-color: orange;
}
```

Более специфичный селектор

Фоновый цвет — бирюзовый

Переопределение на оранжевый цвет не меняет бирюзовый цвет из-за специфичности селектора

Не сработало! Все ссылки остаются бирюзовыми. Почему? Первый селектор здесь специфичнее, чем второй. Он состоит из идентификатора и тега, в то время как второй — из класса. Однако можно сделать еще кое-что вместо того, чтобы учитывать, какой селектор длиннее.

Различные типы селекторов также имеют особенности. К примеру, селектор идентификатора специфичнее селектора класса. На самом деле один идентификатор специфичнее селектора с любым количеством классов. Аналогично селектор класса специфичнее селектора тега, называемого также *селектором типа*.

Итак, существуют следующие правила специфичности.

1. Наиболее специфичным будет селектор с идентификаторами. Чем больше идентификаторов, тем специфичнее будет селектор.
2. Далее идет селектор с наибольшим количеством классов.
3. Следующий по специфичности будет селектор с наибольшим количеством тегов.

Рассмотрим селекторы, показанные в листинге 1.6 (но не добавляйте их в свой код). Они написаны в порядке возрастания специфичности.