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.)

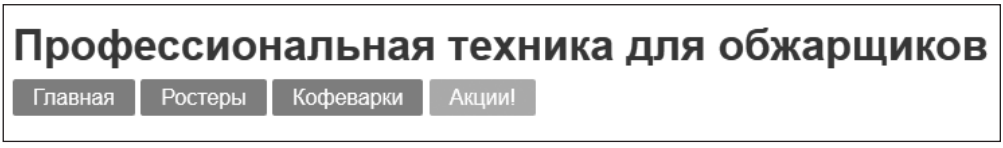


Рис. 1.1. Название страницы и навигационные ссылки

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).

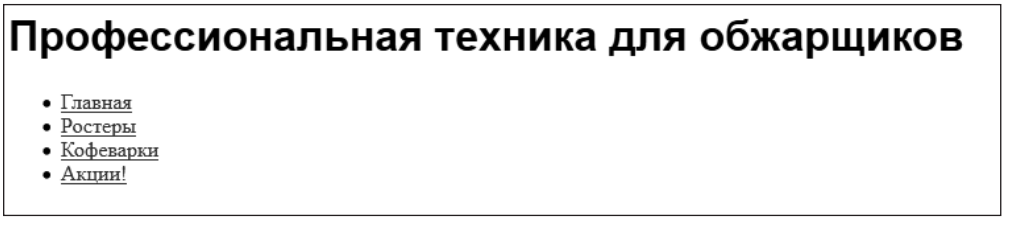


Рис. 1.2. Селектор идентификатора выигрывает у других наборов правил, отображая название страницы рубленым шрифтом

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.

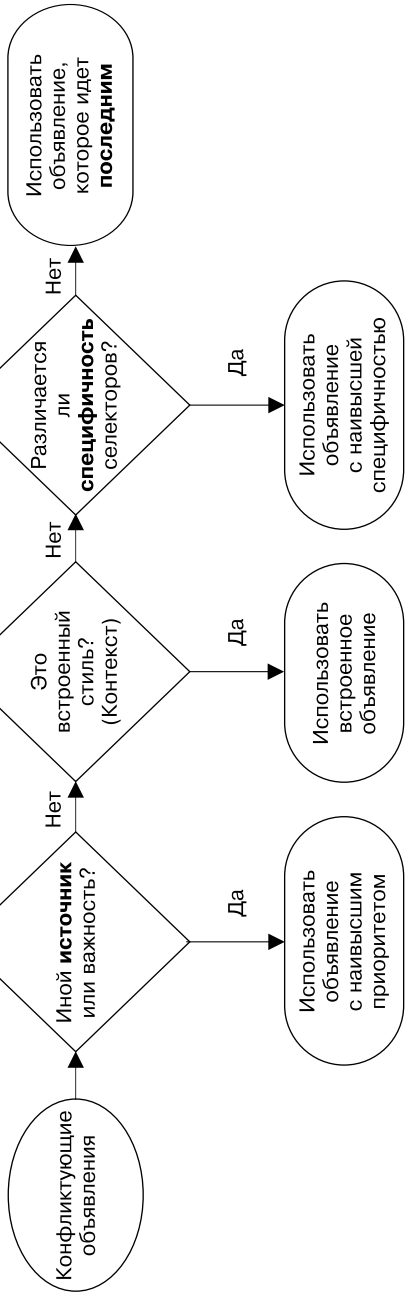


Рис. 1.3. Обобщенная блок-схема каскадности с указанием приоритета объявления

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.

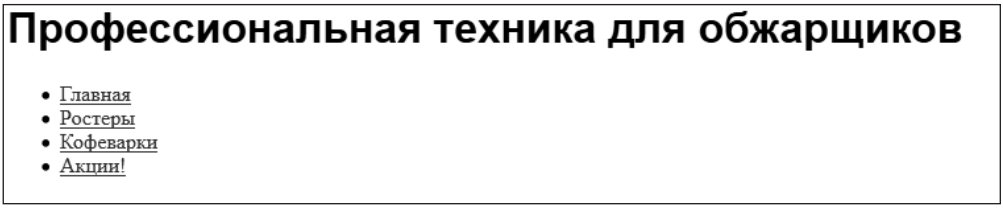


Рис. 1.4. Браузерные стили устанавливают форматирование по умолчанию для элементов на веб-странице

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.

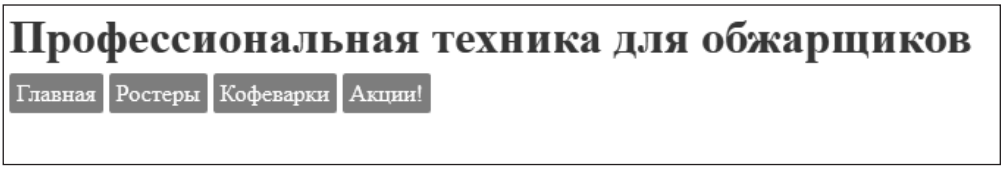


Рис. 1.5. Авторские стили переопределяют браузерные, поскольку имеют более высокий приоритет

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.

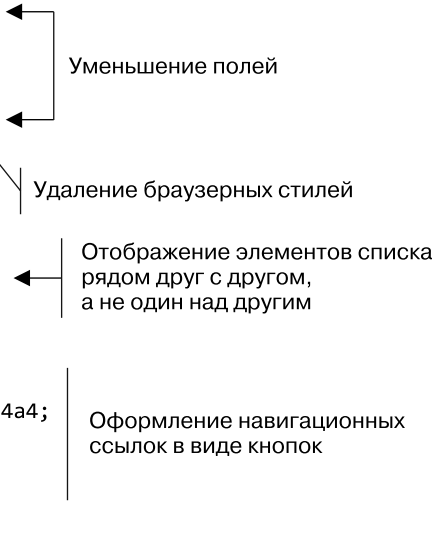
Листинг 1.3. Переопределение браузерных стилей

```
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).

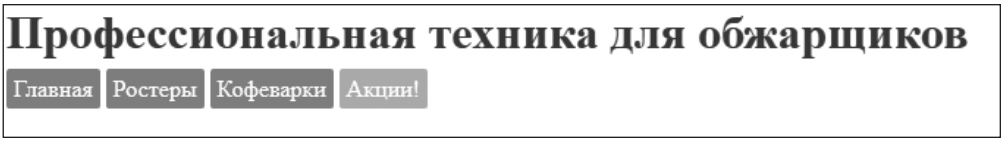


Рис. 1.6. Встроенные стили переопределяют стили, применяемые с помощью селекторов

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.