2 ГИБКАЯ СЕТКА

ОДИН МОЙ ПРЕПОДАВАТЕЛЬ в колледже как-то сказал, что любое художественное действие — музыкальное, литературное или изобразительное — можно считать ответом на действие, ему предшествующее. Режиссеры шестидесятых сняли фильмы «Бонни и Клайд» и «Выпускник» в ответ на старые голливудские картины, такие как, например, «Звуки музыки». В «Потерянном рае» Джон Мильтон фактически помещает своих литературных предшественников в декорации ада — и это вряд ли можно считать тонкой насмешкой над их поэтическими идеалами. И если бы не музыка Дюка Эллингтона и Бенни Гудмена, Чарли Паркер, возможно, никогда бы и не затевал своих безумных экспериментов с бибопом.

Люди искусства всегда спорили друг с другом. Это в первую очередь касается художников-модернистов середины XX века. Модернисты смотрели на творческое наследие предшественников — романтиков конца XIX века — с некоторым,

Рис. 2.1. Модернисты провозглашали отрыв от чрезмерно разукрашенного реализма Уильяма Блейка и Эжена Делакруа и переход к более рациональному подходу Ханса Хофманна и Йозефа Мюллер-Брокманна

мягко говоря, презрением. Для них искусство романтиков было перегружено всей этой чепухой — бесполезным украшательством, которое сводило на нет художественную ценность произведения и не позволяло должным образом донести до зрителя его смысл (**рис. 2.1**).

Реакция модернистов проявлялась различными способами и охватывала практически все виды искусства. Так, в живописи это означало сведение картин до экспериментов с линиями, формой и цветом. Графические дизайнеры того времени, такие как Ян Чихольд, Эмиль Рудер и Йозеф Мюллер-Брокманн, популяризировали понятие типографской, или модульной, сетки — рациональной системы колонок и рядов, в которые можно было поместить модули с контентом (**рис. 2.2**). А благодаря дизайнерам Хою Виню и Марку Болтону нам удалось адаптировать эту старую концепцию к потребностям современного веб-дизайна.

В книге Grid Systems in Graphic Design («Системы сеток в графическом дизайне») Мюллер-Брокманн назвал этот процесс «созданием типографского пространства на странице», то есть разметкой сетки пропорционально размеру чистого листа бумаги.

Рис. 2.2. Типографская сетка, использующаяся для размещения содержимого и определения размеров страницы, — это мощный инструмент, помогающий и дизайнеру, и читателю

Но графический дизайн отличается от веб-дизайна одним ключевым моментом: размерами страницы. Наш же холст — окно браузера — может принимать любую форму и размеры в соответствии с прихотями читателя или размерами устройств, на которых этот холст отображается.

Обычно первый слой нашего макета выглядит следующим образом:

```
#page {
   width: 960px;
   margin: 0 auto;
}
```

Рис. 2.3. Созданный в Photoshop макет выглядит достаточно привлекательным, в отличие от сетки. Как можно сделать ее более гибкой?

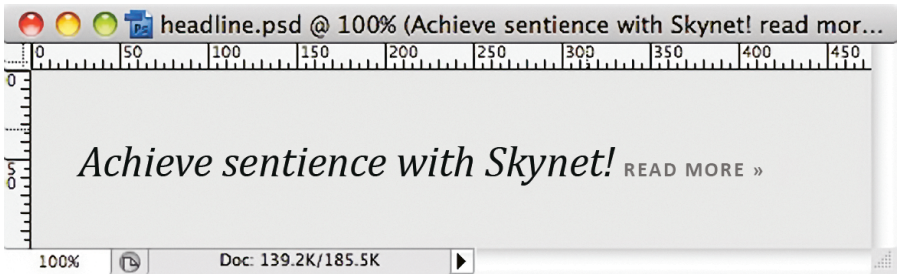

Рис. 2.4. Эскиз для нашего упражнения. По-хорошему, повторить его — минутное дело

То есть мы создали элемент в разметке, задали его фиксированную ширину в CSS и расположили на странице по центру. Если же мы решили создать гибкую сетку, мы должны перевести дизайн, созданный в Photoshop (**рис. 2.3**), во что-то более «резиновое», более *пропорциональное*.

С чего же начать?

ГИБКИЕ ШРИФТЫ

Чтобы ответить на этот вопрос, давайте сыграем в одну ролевую игру. Нет-нет, можете убрать реквизит, я говорю о чем-то более практичном, не имеющем отношения к игрищам «толкиенистов».

Представьте на мгновение, что вы разработчик пользовательских интерфейсов. (Если вы и так разрабатываете пользовательские интерфейсы, то представьте себя еще и в пиратской шляпе.) Дизайнер из вашей команды попросил вас преобразовать простой макет в разметку и CSS. Вы бросаете быстрый взгляд на макет, который он вам прислал (**рис. 2.4**).

Содержимое достаточно скромное, но даже небольшие проекты требуют пристального внимания к мелочам. Итак, вы углубляетесь в изучение дизайна. Оценив типы контента в макете, вы пишете следующий HTML-код:

Рис. 2.5. Разметка без стилей. Именно так создается мечта (и веб-сайт)

```
<h1>Achieve sentience with Skynet! <a href="#">Read More
&raquo;</a></h1>
```

Заголовок с включенной в него ссылкой — прекрасная основа для семантической разметки, не правда ли? После обнуления стилей вы получаете в браузере следующий результат (**рис. 2.5**). По чуть-чуть продвигаемся вперед. Теперь мы можем начать добавлять свой стиль оформления. Давайте впишем в элемент `body` некоторые базовые правила:

```
body {
   background-color: #DCDBD9;
   color: #2C2C2C;
   font: normal 100% Cambria, Georgia, serif;
}
```

Ничего особенного: светло-серый фон (`#DCDBD9`) для всего документа и черный текст (`#2C2C2C`). И конечно, характеристики шрифта — жирность (по умолчанию обычная, `normal`) и семейство шрифтов с засечками (`Cambria, Georgia, serif`).

Вы, вероятно, заметили, что кегль (размер шрифта) был установлен `100%`. Поступив таким образом, мы привели базовый кегль к величине, принятой в браузере по-

умолчанию, который в большинстве случаев составляет 16 пикселей. Теперь мы всегда сможем изменить кегль по отношению к этой относительной базовой величине с помощью единиц измерения `em`. Но прежде чем мы это сделаем, давайте посмотрим, что у нас уже получилось (**рис. 2.6**).

Удивлены, почему `h1` не выглядит как нормальный заголовок? Мы используем обнуление стилей, нивелирующее стили браузера по умолчанию для элементов HTML, чтобы обеспечить их соответствие в различных браузерах. Лично мне больше всего нравится способ обнуления от Эрика Мейера (http://bkaprt.com/rwd/9/), но вы можете выбрать какой-нибудь другой, благо выбор сейчас достаточно большой.

В любом случае наш `h1` выглядит таким маленьким именно по этой причине: он наследует стиль `font-size: 100%`, который мы задали родительскому элементу `body`, а установленный в браузере по умолчанию кегль — 16 пикселей.

Теперь, если пиксели нас устраивают, мы можем перевести значения из оригинал-макета непосредственно в CSS:

```
h1 {
   font-size: 24px;
   font-style: italic;
   font-weight: normal;
}

h1 a {
   color: #747474;
   font: bold 11px Calibri, Optima, Arial, sans-serif;
   letter-spacing: 0.15em;
   text-transform: uppercase;
   text-decoration: none;
}
```

Нет ничего плохого или неправильного в определении размера текста с помощью пикселей. Но в целях нашего эксперимента давайте начнем думать пропорционально и переведем значения кегля (`font-size`) из пикселей в относительные единицы, а вместо пикселей и будем использовать знакомые нам `em`.

Контекстное восстановление

Сейчас будет немного математики: берем *целевое* значение кегля и делим на кегль (`font-size`) его контейнера, то есть контекста. В результате мы получаем желаемый кегль, выраженный в относительных и достаточно гибких единицах `em`.

Другими словами, относительный кегль можно рассчитать по следующей формуле:

```
target ÷ context = result
```

(Отвлечемся на минутку. Лично у меня слово «математика» вызывает немедленный и серьезный приступ паники. У вас тоже? Стойте, не убегайте с криками — все не так страшно. Это говорит вам человек, который заменил курс математики в кол-

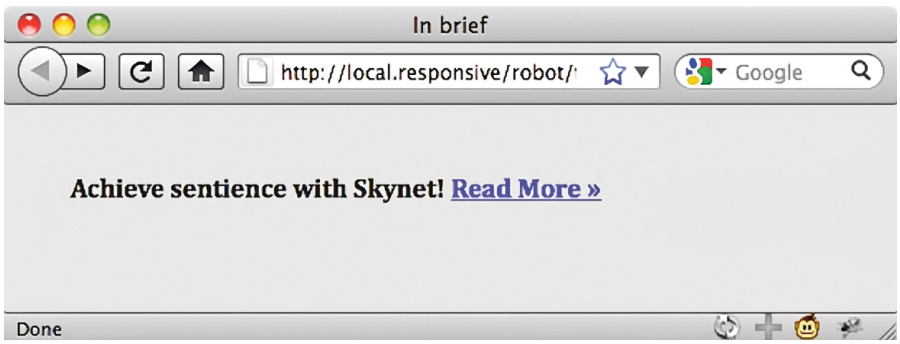

Рис. 2.6. Применив одно простое правило CSS, мы можем придать эскизу несколько другой вид

ледже курсом философии. Делайте, как я: просто посчитайте все на калькуляторе и скопируйте результат в CSS. Калькуляторы — просто спасение для таких, как мы, правда?)

Итак, формула у нас есть, давайте вернемся к нашему заголовку в `24px`. Если предположить, что базовый кегль (`font-size`) элемента `body` равен 16 пикселям при `100%`, мы можем подставить эти значения непосредственно в формулу. В результате получим отношение целевого кегля заголовка `h1` (24 пикселя, `24px`) и кегля его контекста (16 пикселей, `16px`):

```
24 ÷ 16 = 1.5
```

Так как 24 пикселя в 1,5 раза больше 16 пикселей, это значит, что кегль равен `1.5em`.

```
h1 {
   font-size: 1.5em; /* 24px / 16px */
   font-style: italic;
   font-weight: normal;
}
```

Вуаля! Размер нашего заголовка прекрасно совпадает с размером, указанным в оригинал-макете, но при этом выражен в относительных, масштабируемых единицах (**рис. 2.7**).

(Обычно я оставляю комментарий с расчетами с правой стороны (`/* 24px / 16px */`). Вносить изменения становится намного проще.)

С этим закончили, давайте вернемся к нашей одинокой маленькой ссылке Read More (*узнать больше*). Хотя, если посмотреть на **рис. 2.7**, она не такая уж и маленькая. И это проблема. Нам нужно уменьшить заданные 11 пикселей, и довольно существенно, поскольку размер его шрифта принял значение `1.5em` от его контекста, `h1`.

И вот здесь требуется внимание. Поскольку текст заголовка установлен равным `1.5em`, любые элементы внутри этого за-

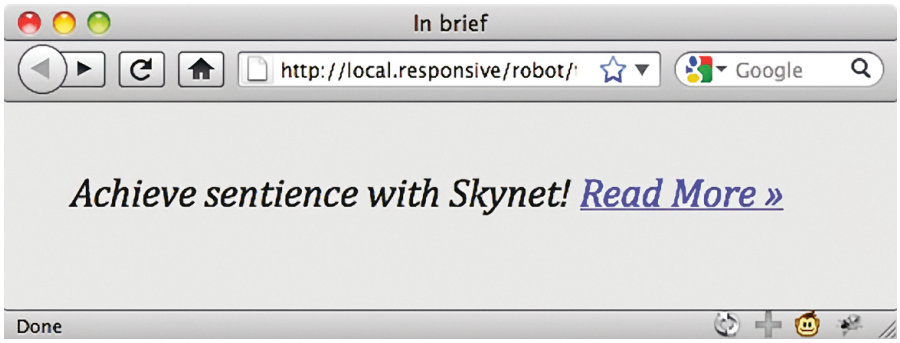

Рис. 2.7. Размер нашего заголовка правильно выражен в гибких, масштабируемых единицах em. (Но что за ерунда творится со ссылкой?)

головка должны быть выражены в виде доли этого значения. Другими словами, *изменился наш контекст*.

Поэтому, чтобы установить кегль ссылки в единицах `em`, мы делим целевые 11 пикселей (`11px`) не на `16` (значение, установленное в `body`), а на `24` — кегль заголовка, наш новый контекст:

```
11 ÷ 24 = 0.458333333333333
```

Мы получили какое-то совсем некрасивое число. Может быть, самое некрасивое, которые вы сегодня видели. (Но подождите, эта глава еще не окончена.)

Вам захочется округлить его до более-менее приемлемого числа — скажем, `0.46 em`. Даже не думайте! Может, ваши глаза и устанут смотреть на `0.458333333333333`, но именно это число идеально представляет желаемый кегль в пропорциональном отношении. К тому же браузеры мастерски владеют искусством округления лишних десятичных знаков, когда преобразовывают значения в пиксели. Поэтому нужно дать им больше, а не меньше, и в конце вас будет ожидать отличный результат.

Теперь просто скопируйте это непритязательное число в CSS:

```
h1 a {
   font: bold 0.4583333333333333em Calibri, Optima, »
     Arial, sans-serif; /* 11px / 24px */
   color: #747474;
   letter-spacing: 0.15em;
   text-transform: uppercase;
   text-decoration: none;
}
```

Посмотрим на результат. Наша страничка закончена, она полностью соответствует желаемому дизайну, а текст задан в масштабируемых единицах em (**рис. 2.8**).

От гибких шрифтов к гибкой сетке

Вы, наверное, сейчас зеваете со скуки. Здесь ведь должна была быть глава про гибкие макеты, а этот тип Итан все талдычит про *математику* и *размеры шрифтов*. Надоело!

Но когда я впервые делал гибкую сетку, я понятия не имел, с чего начинать. Поэтому, столкнувшись с неразрешимой проблемой, я сделал то, что у меня получается лучше всего: полностью ее проигнорировал и начал работать над тем, что знаю.

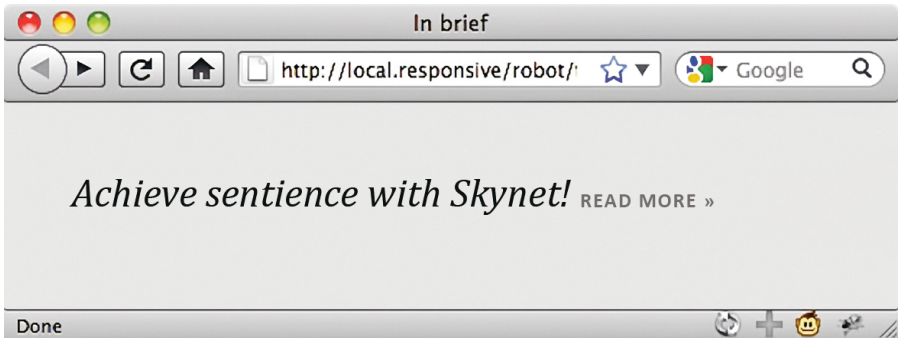

Рис. 2.8. С помощью простой математики мы создали более красивый дизайн — и без всяких пикселей

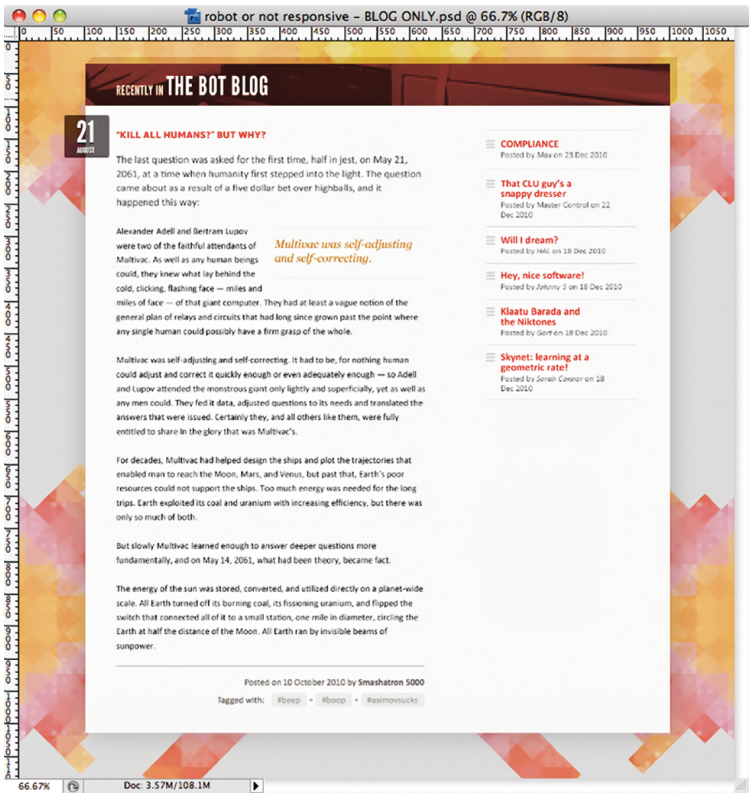

Рис. 2.9. Новое задание — превращение эскиза в гибкий макет

Когда я понял, как устанавливать размеры текста в единицах `em`, на меня снизошло прозрение: ведь мы можем применять тот же принцип пропорционального мышления и в отношении самих макетов. Другими словами, все элементы нашей сетки — строки, колонки и накладываемые на них модули — могут быть выражены как *пропорция* содержащихся в них элементов, а не в неизменных, жестких пикселях.

И в этом нам снова поможет наша формула `target ÷ context = result`. Идем дальше.

издательство
МАНН, ИВАНОВ И ФЕРБЕР
Максимально полезные книги