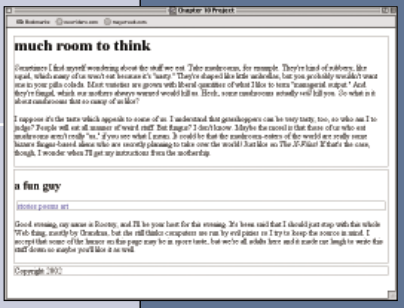
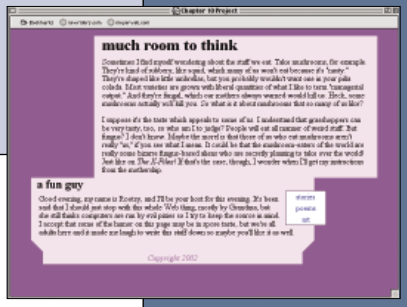
10

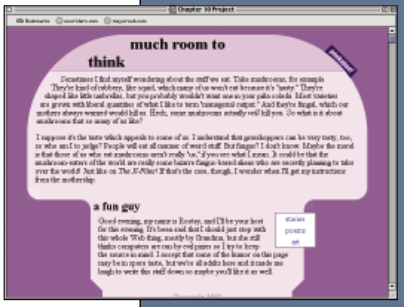
Jak vyzrát na box

Proměním ho v blechu, v malou bezbrannou blechu. Až se tak stane, čeká ji pobyt v krabici, kterou vložím ještě do jiné krabice, načež pošlu tuto krabici sám sobě. Jakmile mi bude doručena, uštědřím jí pořádnou ránu velkým kladivem. Opravdu skvěle vymyšleno, to vám povídám. Jsem prostě génius!

—THE EMPEROR'S NEW GROOVE (2000)







Historie designu v prostředí webu je již od počátku spjata s překonáváním rozličných omezení. Tato skutečnost je dobře patrná na značně skromné škále prostředků, které nám byly dány k dispozici pro tvorbu působivých designérských prvků. Důvod je ten, že elementy HTML nevytvářejí nic jiného, než – s trochou nadsázky řečeno – krabičky, přihrádky a zásuvky. nadpisy, odstavce a dokonce ani seznamy v sobě neskrývají nic jiného než určitý sled takovýchto boxů.

Možná vás již napadlo, že prapředkem těchto potíží je vlastně obyčejná tabulka. Leckdo se jistě snažil dělat téměř nemožné, jen aby přizpůsobil tuto – do značné míry nepoddajnou – strukturu připomínající pravidelnou mřížku momentálním okolnostem a potřebám. Kolik designérů rozdělilo obrázek na více částí tak, aby bylo možno s využitím různého uspořádání buněk tabulky navodit iluzi, že layout stránky není sestaven z pouhého sledu boxů? Kolik z nás alespoň jednou nepomyslelo na buňku tabulky, která by mohla mít zaoblené rohy?

Rovněž je pravdou, že v CSS nelze spatřovat původ nějakých zásadních změn. Důkazem toho je vlastně i kapitola 8 ze specifikace kaskádových stylů úrovně 2 (`www.w3.org/TR/REC-CSS2/`), která se zabývá rozvržením elementů. I když CSS

stále vychází z obdélníkových tvarů, jejichž zdrojem je HTML, není možné tvrdit, že by nenabízelo pomocnou ruku. Díky některým téměř neznámým jemnostem CSS je možné vytvořit šikmé čáry bez zapojení grafických prvků a rovněž přizpůsobit směr textu zaoblenému tvaru. V následujícím projektu se společně podíváme na několik možností, jak vykouzlit oba zmíněné efekty.

Cíle projektu

Řekněme, že jeden z našich přátel, který má na starosti webové stránky nabízející vtipné postřehy a komentáře, nás požádal o vytvoření chytlavého a neobvyklého layoutu, který by mohl použít na svém webu. Poté, co jsme se ujistili, že návštěvníci jeho stránek používají vyšší než čtvrté verze prohlížečů, požádali jsme našeho přítele o zaslání ukázkového dokumentu.

K tomuto projektu máme od zákazníka jenom velmi zběžně stanovené požadavky. Předně víme, že je velkým příznivcem purpurové barvy a barevné ladění podstatné části jeho webu se točí právě kolem různých odstínů této barvy. Tuto skutečnost potřebujeme promítnout do vzhledu našeho návrhu. (Protože nemáme pevně stanovenou sadu odstínů, můžeme si pro tento projekt zvolit odstíny vlastní.) Kromě toho, náš přítel vyjádřil přání, aby layout byl „prostě působivý a praktický, aniž by byl současně okoukaný. To pro tebe bude určitě hračka, nemám pravdu?“

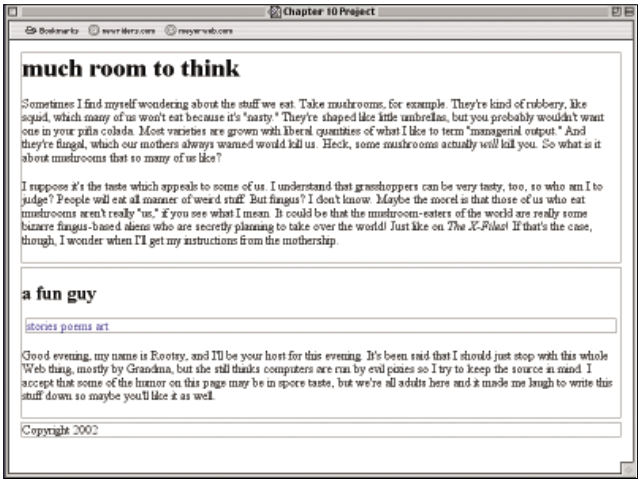
Příprava

Instrukce ohledně stažení potřebných souborů z webu pro tento projekt naleznete v úvodu knihy.

Z webové stránky této knihy si stáhněte soubory pro Projekt 10. Pokud hodláte spolupracovat doma, načtěte si do vašeho oblíbeného HTML editoru soubor `ch10proj.html`, který budete při postupující práci na projektu editovat, ukládat a znovu načítat.

Budujeme základy

Jak už jsme zvyklí z předchozích projektů, opět se jedná o nepříliš rozsáhlý projekt, který však má tentokráte poněkud neobvyklé členění. Na první pohled jsou zde patrné dvě části pro text – každý je zastřešen jedním prvkem `div`, poslední `div` je pak využit pro potřeby zápatí. Kromě toho, zhruba v polovině stránky je umístěno menu. Výpis 10.1 zachycuje výchozí kód stránky, obrázek 10.1 ukazuje stránku s dočasným orámováním, které je nastaveno pro všechny prvky `div`.



Obr. 10.1

Prvotní stav projektu, kde lze stěží hovořit o nějaké přítomnosti stylu.

Výpis 10.1 – Základní struktura stránky

```
<body>
<div class="wrap" id="p1">
</div>
<div class="wrap" id="p2">
  <div id="menu">
  </div>
</div>
<div class="wrap" id="footer">
</div>
</body>
```

V této konkrétní části kódu stojí za zmínku několik zajímavých skutečností:

- Stránka je rozdělena do tří hlavních částí, které sdílejí třídu pojmenovanou jako `wrap`. To bude velmi užitečné pro použití společné definice stylů.
- Prvek `div` náležející k menu je potomkem (descendant) druhého ze tří prvků `div` na nejvyšší úrovni hierarchie. V této fázi projektu se vlastně jedná o jediný prvek `div`, u kterého není uvedena výše zmíněná třída.
- Pro každý `div` je specifikována konkrétní hodnota identifikátoru `id`. Tím je usnadněno použití jakéhokoliv stylu pro konkrétní prvek `div`, aniž by došlo k nechtěnému ovlivnění prvků ostatních.

S povědomím o těchto záležitostech můžeme přejít k tvorbě stylů stránky.

Class a ID

Ačkoliv se nejedná o příliš obvyklý postup, je zcela pořádku, když prvek `div` doplníme o dva identifikátory `class` a `id`. Jak uvidíme později, za určitých okolností to může být obzvláště užitečné.

Definice stylu dokumentu

Protože nemáme na tomto místě přesně stanoveno jak postupovat, většinou se budeme muset s konkrétní záležitostí vypořádat až tehdy, jakmile přijde na řadu. Jistě nám bude k užitku, že na tomto poli již máme určité zkušenosti. Ze všeho nejdříve vytvoříme základní styly pro pozadí a barvy, trošku si pohrajeme s okraji pro různé části dokumentu a uvidíme, kam nás naše počínání zavede.

Potíže s mezerami

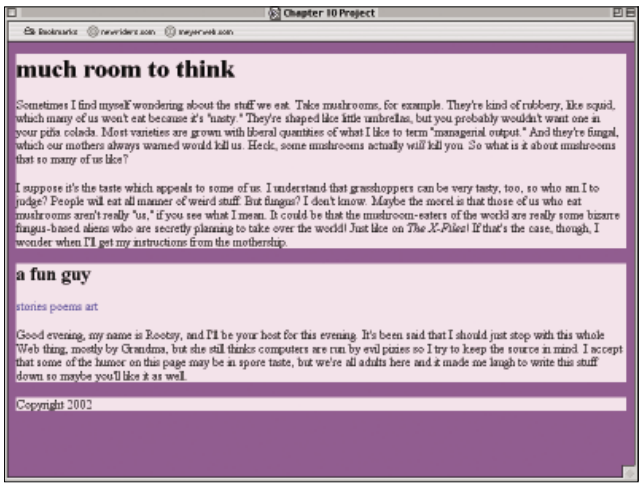
Naše úsilí je vedeno snahou – jak bylo ostatně již dříve rozhodnuto – získat stránku vyvedenou v purpurové barvě. Současně by bylo ale vhodné zachovat černý text na světlém pozadí, což přímo volá po použití světlejší barvy pro třídu `wrap`.

Nyní především potřebujeme odstranit světle šedé orámování, jehož hlavním úkolem bylo usnadnit pochopení struktury stránky (viz obrázek 10.1). Vínově zbarvené pozadí získáme tak, že prvku `body` nadefinujeme barvu pozadí s hodnotu `#969`, která představuje přiměřeně tmavý odstín purpurové. Pro vlastní obsah bude ale lepší nastavit barvu pozadí na hodnotu `#FDF`. Každopádně, v obou případech budeme chtít, aby text byl zobrazen černou barvou. Na obrázku 10.2 vidíme dopad našich změn:

```
<title>Chapter 10 Project</title>
<style type="text/css">
body {background: #969; color: black;}
div.wrap {background: #FDF; color: black;}
</style>
</head>
```

Obr. 10.2

Stránka získá mírně vínový nádech.



Bezpečná barva

Ačkoliv barva pro pozadí stránky patří mezi tzn. webové bezpečné barvy (web-safe), u barvy pro prvek `div` tomu tak již není. Nejbližší bezpečná barva by totiž byla dána hodnotou `#FCF`. V tomto konkrétním případě bude všechno v pořádku, nicméně, s problémy bychom se mohli setkat u každého projektu, kde je požadavek na společné použití většího množství obrázků a barev založených na CSS.

Nyní můžeme takřka na první pohled odhalit, v čem spočívá náš problém – mezery mezi jednotlivými prvky `div`. Tento nechtěný efekt jde na vrub výchozím hodnotám okrajů pro odstavce, což vede k vytváření odstupu mezi jednotlivými prvky. Nedělají to sice všechny prohlížeče, nicméně, jejich počet je dostatečný na

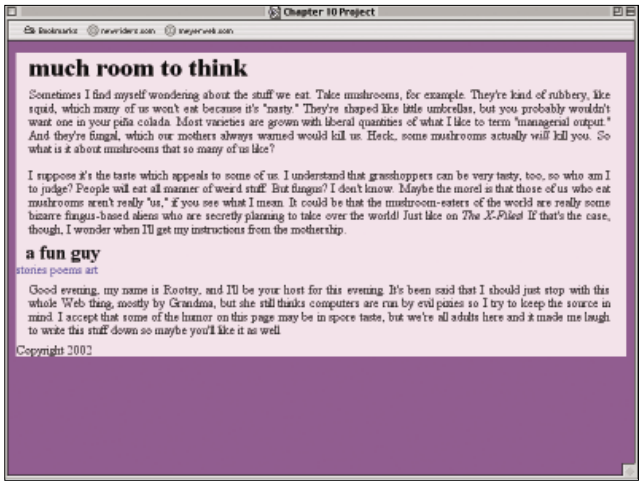
to, abychom se tím zde zabývali. Řešení je zcela prozaické – ve stylu pro odstavce použijme vlastnost `margin` a nastavíme jí nulovou hodnotu. Odstup mezi jednotlivými odstavci pak můžeme, v případě potřeby, zdůraznit pomocí vlastnosti `padding`.

```
div.wrap {background: #FDF; color: black;}
p {margin: 0; padding: 0.5em 1em;}
</style>
```

Toto nové pravidlo nejenom, že nám pomůže vyřešit problém s mezerou, ale současně také odsadí obsah umístěný v odstavcích od okrajů jednotlivých prvků div. Jak je vidět na dalším obrázku, vypadá to určitě lépe. A abychom předešli eventuálnímu výskytu dalších problémům s mezerami, nastavíme nulové okraje i pro nadpisy `h1` a `h2`.

```
p {margin: 0; padding: 0.5em 1em}
h1, h2 {margin: 0; padding: 0 0.5em;}
</style>
```

Díky provedení těchto úprav je možno dosáhnout efektivní spolupráce všech prvků `div`, což nám zobrazí stránku pomocí jednoho celistvého bloku namísto tří odlišných sekcí. Výsledek je názorně zobrazen na obrázku 10.3.



Obr. 10.3

Spojení jednotlivých částí stránky vytváří iluzi jednoho velkého elementu.

Nyní, když jsme se konečně dopracovali k tomuto vizuálnímu souladu, je načase všechno opět rozdělit, tentokráte ale ku prospěchu věci!

▶▶ Okraje, výplň a mezera

Pro zachování obvyklé mezery o velikosti `1em` mezi jednotlivými odstavci textu jsme museli u každého nastavit horní a spodní výplň na hodnotu `0.5em`. To je nezbytné učinit, protože výplň se nepřekrývá – svisle přiléhající okraje již ano. Tento aspekt layoutu zůstává mnoha autory ponechán zcela bez povšimnutí, nicméně, jde o důležitou stránku věci.

Uvažujme například situaci, kdy je jeden `div` *následován dalším, přičemž u obou dvou je nastaven okraj.*

```
<div style="margin: 1em;">div one</div>
<div style="margin: 1em;">div one</div>
```

Kolik místa se podle vás objeví mezi textem uzavřeným do těchto dvou prvků `div`*? Odpověď většiny lidí by nejspíše zněla jako* `1em`*, přičemž ve stejném duchu hovoří i CSS. Je ale správné se ptát – pokud má první div spodní okraj o velikosti* `1em` *a druhý* `div` *horní okraj o téže velikosti, proč není velikost mezery mezi těmito* `divy` *rovna* `2em`*?*

Důvod, proč to tak je, spočívá v tom, že svisle přiléhající okraje – podobně jako v předchozím příkladě – se skutečně překrývají. Vzdálenost mezi dvěma elementy tak bude odpovídat větší hodnotě z obou sousedících okrajů. Tudíž, pokud měl první prvek `div` *spodní okraj o velikosti* `1px` *a druhý prvek* `div` *horní okraj o velikosti třeba* `15px`*, vzdálenost mezi nimi by pak odpovídala právě těm 15 pixelům.*

Tohle všechno vám bude naprosto zřejmé, jakmile nad tím přestanete zbytečně hloubat. Bohužel – výplň se chová odlišně. Předpokládejme, že v našem příkladu zaměníme okraje za výplň.

```
<div style="padding: 1em;">div one</div>
<div style="padding: 1em;">div one</div>
```

Nyní bude mezi těmito dvěma elementy skutečně mezera o velikosti `2em`*. Abychom tedy nijak nenarušili soudržnost layoutu, musíme zmenšit velikost horní a spodní výplně na polovinu – a právě to jsme v našem projektu udělali.*

Hrátky s okraji

V tomto okamžiku je nejvyšší čas vzít si na paškál náš velký box a pokusit se jej učinit trošku méně hranatým. V prvním kroku jednoduše posuneme jeho okraje.

```
div.wrap {background: #FDF; color: black; margin: 0 2em;}
```

Tímto jednoduchým způsobem se docílí mírného zúžení obsahu. Na to lze elegantně navázat v následujícím kroku – proč by měly mít všechny prvky `div` stejné okraje? Pokud budou v jednotlivých částech odlišné, box již nebude vzbuzovat dojem obyčejného boxu, což je ukázáno na obrázku 10.4.

```
h1, h2 {margin: 0; padding: 0 0.5em;}
div#p1 {margin: 0 2em 0 10em;}
div#p2 {margin: 0 10em 0 2em;}
div#footer {margin: 0 11em 0 2em;}
</style>
```

V určitém slova smyslu představují námi dvě vytvořené hlavní části stránky navzájem zrcadlově převrácený obraz. První část má pravý okraj nastavený na hodnotu `2em`, pro levý okraj je to pak `10em`; u druhé části jsou hodnoty mezi sebou zaměněny. A abychom zachovali mírně odlišný vzhled zápatí, bylo – kromě zarovnání jeho levého okraje vůči shora sousedícímu prvků `div` – nutné lehce posunout jeho pravý okraj (konkrétně o `11em`).

Obr. 10.4

Přerušení boxu.

Při pohledu na obrázek 10.4 vystupují do popředí dvě skutečnosti, které si přímo říkají o to, aby se nenechaly jenom tak ležet – menu a zápatí. Bez dlouhého přemýšlení se můžeme pustit do úpravy zápatí, protože všechno, co zde budeme potřebovat udělat, se týká přidání výplně a zarovnání textu na střed. Kromě toho, text bude zobrazen kurzívou, přičemž jeho barvu přizpůsobíme podle elementu `body`.

```
div#footer {margin: 0 11em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;}
```

Jakmile jsme s tím hotovi, můžeme směřovat naše úsilí směrem k úpravě menu.

Kouzla s menu

S pouhými třemi odkazy v rámci příslušného prvku `div` jde o více než jednoduché menu. Zkusme nyní tento prvek `div` posunout doprava a podívejme se, jak to dopadne (viz obrázek 10.5). Mějte ale na zřeteli, že kdykoliv provádíme posun textového prvku, nesmíme zapomenout na jeho šířku. Vzhledem k docela krátkým odkazům může být menu poměrně úzké. Kromě toho mu také doplníme černé orámování a bílou barvu pozadí, čímž menu na stránce vynikne o něco lépe.

```
div#p2 {margin: 0 10em 0 2em;}
div#menu {float: right; width: 5em;
  border: 1px solid black; background: white;}
div#footer {margin: 0 11em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;}
```

Ovšem to je pouze začátek. Tím, že textu nastavíme vhodné okraje, zajistíme, aby se nedostal příliš blízko k okraji menu. Záporná velikost okraje menu jej popostrčí směrem doprava. To způsobí, že bude částečně zasahovat do tmavě purpurového pozadí. Pro všechny případy rovněž nastavíme hodnotu výplně na nulu.

```
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid black; background: white;}
```

Vedlejší efekty

Z nějakého důvodu považuje prohlížeč IE5.x/Win záporný levý a pravý okraj za větší, než jak tomu ve skutečnosti je. V některých případech se udaná velikost mezery dokonce zdvojnásobí. Tato skutečnost poněkud omezuje možnost uplatnění záporných okrajů pro účely posunutí, nicméně, za určitých okolností (jako je tato) však mohou nalézt uplatnění bez větších potíží. Nicméně, v jakémkoliv designu, kde je požadavek na přesné umístění elementů, by se pro jistotu nemělo používat posunutí pomocí záporné hodnotou okrajů.

Obr. 10.5

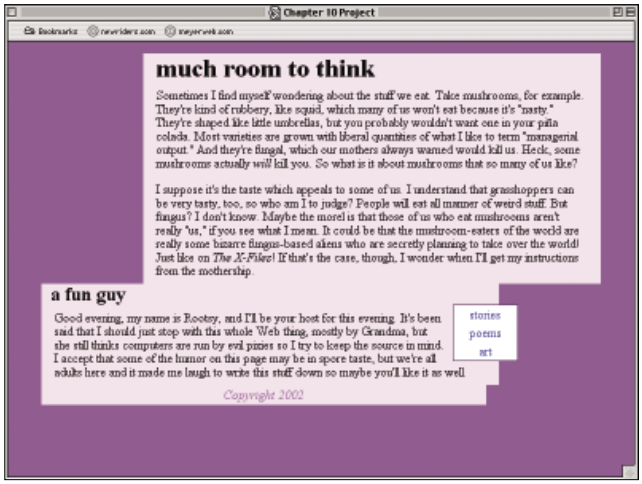
Posunutí menu doprava.

Černé orámování menu působí trochu nevlídným dojmem, což je navíc umocněno purpurovou barvou kolem dokola. Pokusme se to napravit změnou barvy orámování na maličko tmavší odstín než má nastaveno pozadí stránky. Jednoduše vyjdeme z hodnoty barvy pozadí pro element `body`, což je `#969`, a od každé číslice odečteme dvojku.

```
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid #747; background: white;}
```

Samozřejmě, jednotlivé odkazy v menu na sebe navzájem navazují jako na sebe navazují slova běžného textu, protože se jedná o řádkové elementy. Pokud chceme umístit každý odkaz na vlastní řádek, musíme pro to něco udělat. Namísto vkládání klasických elementů `<br>` se zkrátka vydáme cestou vytvoření blokových elementů (pro více informací – viz Projekt 5). Přidejme ještě k dobru nějakou tu výplň a když už si s tím dáváme takovou práci, vycentrujme text jednotlivých odkazů; výsledek je zachycen na obrázku 10.6.

```
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid #747; background: white;}
div#menu a {display: block; text-align: center;
  padding: 0.2em 0.5em;}
div#footer {margin: 0 11em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;}
```



Obr. 10.6

Posunutí menu mimo oblast a umístění odkazů do řádků.

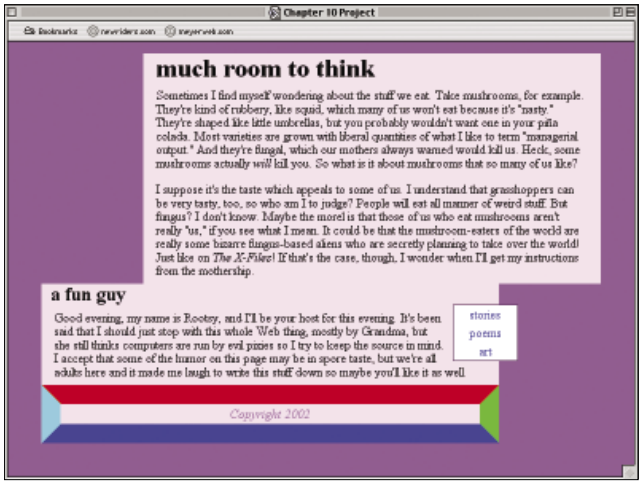
Myslíme v úhlech

Celkově vzato – náš design si nestojí vůbec špatně. Připusťme, že stále máme před sebou množství rovných čar a ostrých rohů, nicméně, náš efekt spočívá v použití nepravidelného polygonu namísto běžných obdélníkových tvarů. Přesto by nám určitě přišlo vhod několik šikmých čar jako alternativa vůči rovným čarám, které mají v našem layoutu stále dominantní postavení. Bylo by ovšem ještě více působivější, kdybychom tyto šikmé čáry vytvořili zkombinováním obvyklých HTML elementů a CSS namísto klasického použití obrázků.

Domníváte se, že je to zhola nemožné? Popravdě řečeno – k něčemu takovému není potřeba o nic více, než to, co nám nabízí CSS1. Pojďme tedy v současném projektu mírně upravit konec stránky. Zápatí vzhledem k části `div`, která mu předchází, mírně zúžíme. Jejich společným pojítkem pak budou šikmé čáry. Nejlepší na tom je ovšem to, že stačí upravit styl u prvku `div`, jehož prostřednictvím je zápatí definováno.

Začneme přizpůsobením okrajů zápatí tak, aby byly shodné s `div#p2`. Následně doplníme zápatí o patřičně silné orámování v úžasně křiklavé barevné kombinaci, což v celé kráse zachycuje obrázek 10.7.

```
div#footer {margin: 0 10em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: red lime blue cyan;}
```



Obr. 10.7

Velmi křiklavé orámování kolem zápatí.

Popravdě řečeno, teď to vypadá docela ošklivě. Jde však pouze o přechodný stav, kdy se vám snažíme usnadnit pochopení toho, jak to všechno funguje. Pozorně se podívejte na rohy zápatí. Všimli jste si šikmých předělů v místě, kde se potkávají barvy? Jsou vždy přítomny v rozích rámečku každého elementu. Protože jen velmi málo designerů nastavuje šířku orámování na větší hodnotu než 1 nebo 2 pixely nebo používá odlišnou barvu pro každou jeho část, zůstávají tyto šikmé čáry prakticky bez povšimnutí. Nyní, když jsme se s nimi seznámili, můžeme je s úspěchem využít pro naše potřeby.

Klíč k úspěchu leží v manipulaci s barvou a šířkou jednotlivých částí orámování, čímž lze dosáhnout požadovaného efektu. Namísto barevné kombinace červená/zelená/modrá/azurová musíme použít právě ty dvě barvy, kterými je obklopen `div` zastřešující zápatí. Tudíž, potřebujeme orámování na obou stranách barevně sladit s barvou pozadí pro prvek `body` a horní orámování pak s barvou pozadí prvku `div` ležícího bezprostředně nad ním.

```
div#footer {margin: 0 10em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969;}
```

Kromě tohoto nastavení barev se musíme postarat ještě o jednu záležitost. Spodní orámování zápatí právě rozšířilo oblast vyplněnou světle purpurovou barvou, což není přesně to, co bychom zamýšleli.

Máme na výběr ze dvou možnosti, jak toto vyřešit. Jedna z nich spočívá v tom, že se velikost orámování spodní části zápatí se nastaví na nulovou hodnotu. Tím bychom se zbavili přebytečné oblasti se světle purpurovou barvou. Druhou možností je zvolit pro orámování spodní částí zápatí stejnou barvu, jaká je nadefinována pro pozadí prvku `body` – přebytečná oblast pak „zmizí“. Vůbec nezáleží na tom, pro kterou z možností se nakonec rozhodneme, nicméně, změna barvy vyžaduje kratší zápis a tudíž použijeme toto řešení. Výsledek viz obrázkek 10.8.

```
div#footer {margin: 0 10em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969 #969;}
```

Co za tím vězí?

Zdroj inspirace pro tento efekt pochází z ukázky nazvané „An Exercise in Regular Polygons“, kterou je možno zhlédnout na `www.tantek.com/CSS/Examples/`. Tantek Celik zde brilantním způsobem demonstroval možnosti tvorby šikmých čar s využitím CSS, což položilo základy pro vytváření efektů podobného ražení, jako byl právě ten, který jsme použili v našem projektu.

Větší mezera

Povšimněte si, že nyní je mezi textem a zápatím více prázdného místa. Za to je zodpovědné horní orámování a nedá se tím tedy nic dělat. Prázdný prostor, který zde máme jaksi navíc, je sice možné zredukovat zmenšením tloušťky rámečku, nicméně, to by vedlo k nechtěnému efektu v podobě zúžení orámování po stranách a v horní části zápatí.

Obr. 10.8

Šikmé čáry pro potěchu oka.

Tato finta hovoří doslova sama za sebe, nicméně, opravdu funguje pouze v situacích podobné té naší, kdy je potřeba překrýt mezeru oddělující dva elementy buďto rozšířením nebo zúžením společné barvy pozadí. Očekávaný efekt se prostě nedostaví v případě rozdílných barev nebo v případě existence orámování prvku. Rovněž nepochodíme ve většině případů, ve kterých je použitý obrázek na pozadí.

Výsledky naší dosavadní práce shrnuje výpis zdrojového kódu 10.2. A již za okamžik nás čeká další krok – zaoblené tvary.

Výpis 10.2 – Shrnutí dosavadního stylu

```
<style type="text/css">
body {background: #969; color: black;}
div.wrap {background: #FDF; color: black; margin: 0 2em;}
p {margin: 0; padding: 0.5em 1em;}
h1, h2 {margin: 0; padding: 0 0.5em;}
div#p1 {margin: 0 2em 0 10em;}
div#p2 {margin: 0 10em 0 2em;}
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid #747; background: white;}
div#menu a {display: block; text-align: center;
  padding: 0.2em 0.5em;}
div#footer {margin: 0 10em 0 2em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969 #969;}
</style>
```

Příchod zaoblených tvarů

Do této chvíle jsme si ukázali, jak lze s pomocí vhodně spojených boxů vytvářet nepravidelné tvary a jak využít orámování boxu pro tvorbu šikmých čar. Ale co třeba zaoblené tvary? Je možné HTML elementy přinutit k tomu, aby něco takového zobrazovaly?

Prvotní odpověď zní, že to nejde. Ale s využitím několika obyčejných obrázků a troškou nápaditého přístupu při vytváření stylů, je možné vytvořit zaoblené rohy nebo eventuálně dosáhnout obtékání textu podél zaobleného tvaru namísto vnitřku boxů.

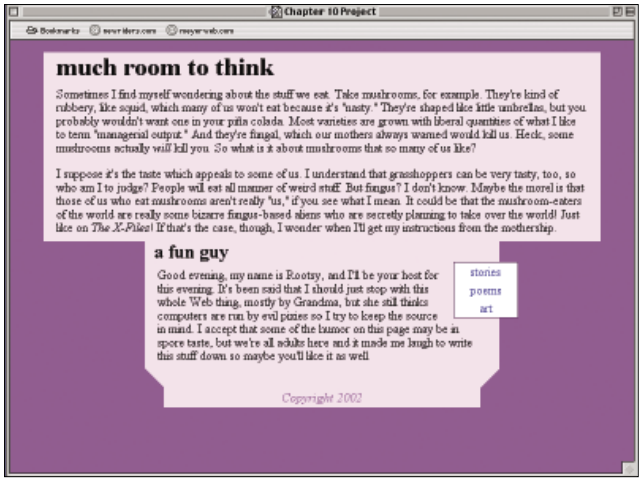
Naše vlastní muchomůrka

Vzhledem k tématu, o kterému se věnuje námi upravovaná webová stránka, by jistě bylo zajímavou ukázkou důmyslnosti vytvořit layout ve tvaru houby. O tom nejsou pochyby. Pojďme si tedy trochu malinko pohrát s okraji, čímž vtiskneme našemu layoutu jakýsi náznak tvaru muchomůrky (viz obrázek 10.9).

```
div#p1 {margin: 0 2em;}
div#p2 {margin: 0 10em;}
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid #747; background: white;}
div#menu a {display: block; text-align: center;
  padding: 0.2em 0.5em;}
div#footer {margin: 0 10em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969 #969;}
```

Obr. 10.9

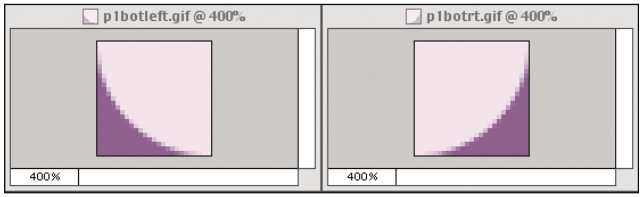
Několik jednoduchých úprav vede k zásadním změnám layoutu.



No – zatím to spíše připomíná písmeno „T“ než tvar nějaké muchomůrky. Pro lépe vyhlížející tvar houby potřebujeme zaoblit celou škálu ostrých rohů, v našem layoutu jde zejména o oblast samotného „kloboučku“ (`div#p1`). Tohle už sice nebude tak jednoduché, nicméně, bude to pořádně zajímavé.

Spodek kloboučku

Ze všeho nejdříve provedeme zaoblení spodních rohů kloboučku houby. Pro tento efekt budeme potřebovat dva samostatné obrázky, které zaujmou místo v příslušném elementu `div`. Jejich vzhled je ve čtyřnásobném zvětšení zachycen na obrázku 10.4.



Obr. 10.10

S těmito dvěma soubory typu GIF provedeme zaoblení kloboučku.

První krok bude spočívat v doplnění samotného dokumentu o malou část kódu. Potřebujeme vložit prvek `div`, do kterého pak vložíme obrázek pro pravý spodní roh (nikoliv však pro levý spodní roh).

```
<p>
I suppose it's the taste which appeals to some of us. I under-
stand that grasshoppers can be very tasty, too, so who am I to
judge? People will eat all manner of weird stuff. But fungus?
I don't know. Maybe the morel is that those of us who eat mush-
rooms aren't really "us," if you see what I mean. It could be
that the mushroom-eaters of the world are really some bizarre
fungusbased
aliens who are secretly planning to take over the world! Just
like on <cite>The X-Files</cite>! If that's the case, though,
I wonder when I'll get my instructions from the mothership.
</p>
<div id="p1end"><img src="p1botrt.gif" alt=" "></div>
</div>
```

V tomto okamžiku máme odpovídajícím způsobem označený `div`, který obsahuje daný obrázek. Pochopitelně, až do té doby, než se rozhodneme učinit nějakou změnu, bude příslušný obrázek zarovnán doleva. Do pravého rohu jej můžeme umístit za přispění jednoduchého `text-align`. A když už jsme v tom, můžeme dát na příslušné místo další zbývající roh.

```
div#p1 {margin: 0 2em;}
div#p1end {text-align: right; margin: 0; padding: 0;
background: #FDF url(p1botleft.gif) bottom left no-repeat;}
div#p2 {margin: 0 10em;}
```

Zde je důležité si uvědomit, že jsme přesunuli obrázek `p1botrt.gif` na pravou stranu elementu `div`, přičemž současně byl obrázek `p2botleft.gif` umístěn do levého spodního rohu. Poněvadž tento `div` ukončuje oblast s kloboučkem, oba dva obrázky budou nakonec umístěny v rozích. Je možné se přesvědčit při pohledu na obrázek 10.11.

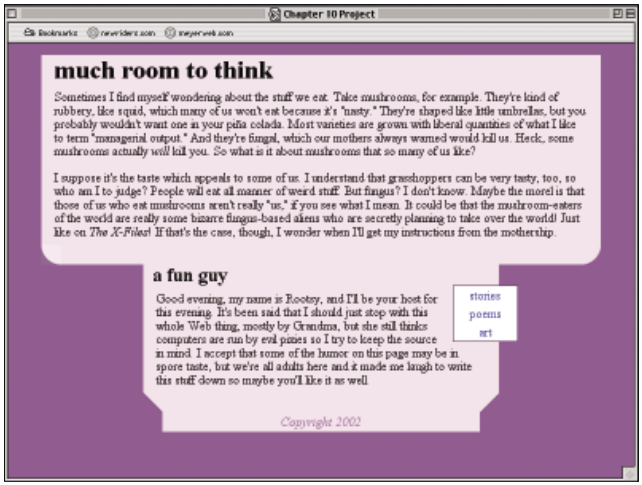
Přesně tak – pomocí několika obrázků jsme dokázali vylepšit náš design. V této myšlence budeme pokračovat dále – zaoblíme rohy v místě, kde se klobouček setkává s nožkou.

Odstranění prázdných míst

Odstraněním všech mezer mezi prvky `img` a `div` jsme zajistili, že `div` nezobrazuje žádné dodatečné mezery, které nepotřebujeme. Některé prohlížeče, obzvláště IE5.x/Win, považují za mezeru, kterou je zobrazit na stránce, i stisk kláves Enter ve zdrojovém kód. U jiných prohlížečů může být vše – nebo taktéž nemusí – v pořádku, ale my raději eventuálním komplikacím předejdeme tak, že odstraníme všechny nepotřebné mezery v dokumentu.

Obr. 10.11

Zaoblení spodní části kloboučku odlehčí design.



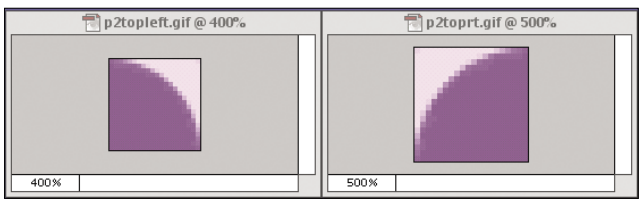
Zaoblený přechod

Nyní můžeme vložit další prvek `div` do horní části `div#p2`, kam bychom pak umístili požadované zaoblení, ale v zásadě, nic takového není potřeba. Máme tady totiž element, který se nám bude pro daný účel náramně hodit. Vše, co bude prvek `h2` potřebovat, jsou obrázky a nějaký ten styl.

Budeme muset použít dva nové obrázky, protože ty, které jsme měli k dispozici před tím, pro klobouček houby, nám zde nebudou moc platné. Obrázky, které vměstnáme do elementu `h2`, zachycuje obrázek 10.12.

Obr. 10.12

Dva obrázky, které použijeme ke spojení kloboučku a nožky.



Prázdný `alt`

U těchto obrázků nespecifikujeme žádnou hodnotu u atributu `alt`, protože tyto obrázky se přímo nepodílejí na obsahu stránky – není nutné je tedy popisovat způsobem určeným pro čtečky obrazovky (screenreader), které používají uživatelé s postižením zraku. Postup by pochopitelně lišil v případě umístění loga či tlačítek, ovšem, v našem designu nic takového není.

Překvapivě ale není možné se držet stejného postupu, který předtím fungoval u `div#p2`. V okamžiku, kdy se pokusíme zarovnat prvek `h2` doprava, abychom mohli umístit zaoblení, dojde ve stejný okamžik u prvku `h2` k zarovnání textu doprava, což nechceme. To vysvětluje, proč bude obrázek pro levý horní okraj nožky umístěn na začátku elementu `h2`.

```
<h2><img src="p2topleft.gif" alt=" "> a fun guy</h2>
```

Povšimněte si mezery vložené mezi obrázek a text. Účelem je zabránit textu v tom, aby ocitnul hned vedle obrázku.

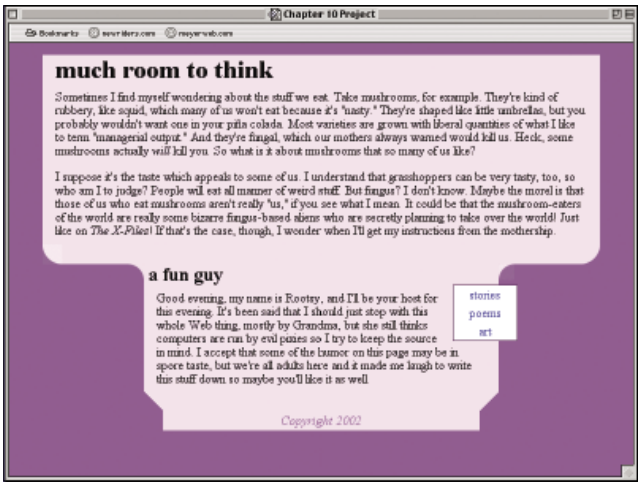
Nyní je potřeba na pozadí elementu h2 vložit do pravého horního rohu odpovídající zaoblení nožky. Rovněž musíme zařídit, aby byl současně levý zaoblený horní roh zarovnán vůči horní hraně elementu h2.

```
h1, h2 {margin: 0; padding: 0 0.5em;}
h2 {background: transparent url(p2toprt.gif) right top no-repeat;}
h2 img {vertical-align: top;}
div#p1 {margin: 0 2em;}
```

V tomto okamžiku se jedno zaoblení usídlilo v levém horním rohu prvku h2 coby jeho obsah. Druhé zaoblení pak zaujímá pravý horní roh, ovšem, tváří se jako pozadí tohoto elementu. Je zde pouze jeden problém: obě zaoblení jsou umístěna uvnitř nožky houby. Aby byla iluze dokonalá, potřebujeme je mít vně.

Víme, že zaoblení jsou tvořena obrázky se stranami o velikosti 20 pixelů a kolem elementu h2 nemůže být žádná výplň. Výplň by totiž odsunula levý horní roh z jeho určeného místa. To vysvětluje, proč musí být výplň nastavena na nulu a levý a pravý okraj h2 na -20px. V důsledku toho se levý a pravý okraj h2 ocitne mimo oblast nožky, přičemž je budou doprovázet i rohové obrázky. Tento efekt si můžeme prohlédnout na obrázku 10.13.

```
h2 {margin: 0 -20px; padding: 0;
  background: transparent url(p2toprt.gif) right top no-repeat;}
```



Obr. 10.13

Záporné okraje a nepřítomnost výplně umístí obrázky tam, kam patří.

Při pohledu na klobouček a nožku se před námi objeví několik ladných křivek, ale zdaleka nejsme ještě hotovi. Spodek kloboučku se již nemusí nijak upravovat – ale co jeho vršek? Stále je příliš hranatý.

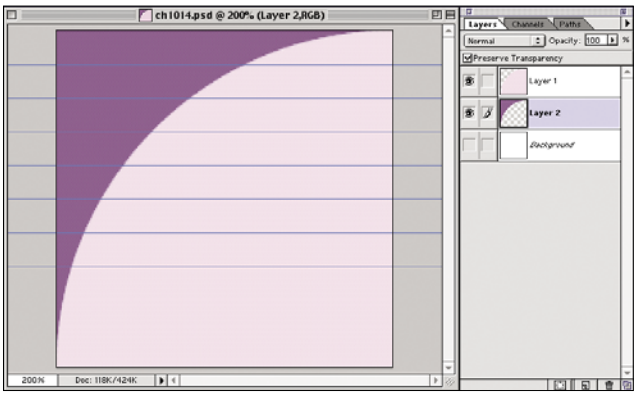
Průhlednost

Barva pozadí byla pro element h2 nastavena zcela záměrně jako průhledná, aby přes ni mohla prosvítat jakákoliv jiná barva pozadí. Tato skutečnost bude hrát před dokončením projektu klíčovou roli.

Uzavření kloboučku

Pohled na vršek kloboučku nás zcela určitě přivede k myšlence, že jednoduše vezmeme zaoblené rohy, vsuneme je do prvku `h1` a prohlásíme vše za hotové. Nicméně, touto cestou se nevydáme. Namísto toho pozvedneme myšlenku oblých tvarů pro potřeby webového designu na zcela novou úroveň. Vedle přikrášlení designu velkými oblouky v horních rozích, se nám dostane odměny v podobě obtékání textu podle těchto oblouků. Opravdu.

Ze všeho nejdřív je ale nutné si obstarat zaoblený grafický prvek, který je dostatečně velký pro naše účely. Řekněme, že rozměr 200x200 px by mohl být tak akorát. Další krok bude spočívat v jeho rozdělení do několika vrstev. Bohatě nám postačí, když jej rozdělíme na osm částí, jak je znázorněno na obrázku 10.14.

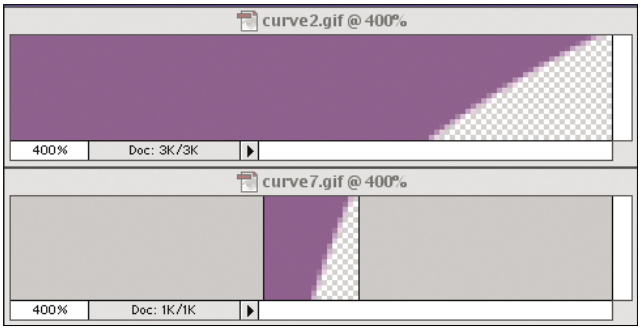


Obr. 10.14

Levý horní oblouk (zobrazen v prostředí Adobe Photoshop).

Dalo by se sice uvažovat o rozdělení na 10 částí, kde by každá část měla výšku 20 pixelů, nicméně, vůbec nevadí, když spodní část oblouku bude vysoká celkem 60 pixelů – je totiž dost úzká na to, aby nám to nevadilo.

Nyní náš čeká poměrně nudná část. Pro každou část oblouku se vybere dostatečně velká oblast tak, aby zasahovala do oblasti s tmavě purpurovou barvou (za okamžik pochopíte, proč tomu tak je). Zvolená oblast bude uložena jako obrázek ve formátu GIF, přičemž světle purpurová barva se nastaví jako transparentní. Jak to bude vypadat v praxi, ukazuje obrázek 10.15, kde jsou zachyceny dva grafické prvky získané z oblouku.



Obr. 10.15

Dvě části oblouku; šachovnicový vzorek značí průhlednost.

Jakmile budeme mít pohromadě všech osm částí (které pojmenujeme jako `curve1.gif až curve8.gif`), uložíme je do adresáře nazvaného jako `curve-1`. Pro vložení obrázků do dokumentu je potřeba vytvořit následující krátký úsek kódu.

```
<body>
<div class="wrap" id="p1">
<img src="curve-1/curve1.gif" alt=" " class="curve-1">
<img src="curve-1/curve2.gif" alt=" " class="curve-1">
<img src="curve-1/curve3.gif" alt=" " class="curve-1">
<img src="curve-1/curve4.gif" alt=" " class="curve-1">
<img src="curve-1/curve5.gif" alt=" " class="curve-1">
<img src="curve-1/curve6.gif" alt=" " class="curve-1">
<img src="curve-1/curve7.gif" alt=" " class="curve-1">
<img src="curve-1/curve8.gif" alt=" " class="curve-1">
<h1>much room to think</h1>
```

Když tento kód máme v dokumentu, sestavíme z něj požadovaný oblouk. Protože se ale jedná o řádkové elementy, jsou všechny tyto části poskládány jedna za druhou. Tento stav je nutno změnit – musíme však na to jít chytře. Před vytvářením blokově orientovaných elementů dáme přednost použití vlastnosti `float`, která je posune doleva.

```
div#footer {margin: 0 10em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969 #969;}
img.curve-1 {float: left;}
</style>
```

To však nestačí. Pokud je naším záměrem poskládat jednotlivé části pěkně nad sebe, musíme zařídit, že každá část zaujme místo vlevo pod obrázkem, který jí předcházel. Pro tento účel nebude potřeba použít nic jiného, než vlastnost `clear`. Aby bylo na první pohled jasné, jak se věci mají, nastavíme těmto obrázkům dočasné orámování (viz obrázek 10.16).

```
img.curve-1 {float: left; clear: left;
  border: 1px solid gray;}
```

Obr. 10.16
Jednotlivé části vytvoří oblouk.

Povšimněte si, jakým způsobem vyplňuje text prostor vedle obrázků. Výsledný efekt je vlastně názorným důkazem obtékání textu podél oblouku. Dost dobré, není-liž pravda?

Přirozeně zde existuje několik dalších věcí, na které je dobré dát si pozor. Obrázky musí být úplně zarovnány k levému okraji `div#p1`; pokud tomu tak není, iluze obtékání se zhroutí jako domeček z karet. Neměli bychom rovněž zapomenout na přidání nevelkého okraje vpravo, aby se text neoctnul příliš blízko oblouku. A konečně můžeme odstranit to odporné šedé orámování.

```
img.curve-l {float: left; clear: left; margin: 0 0.5em 0 0;}
```

Rozhodně nebude na škodu, když trochu změníme styl nadpisu `h1`. Trochu tmavší pozadí a orámování ve spodní části společně zanechávají příjemný dojem, a navíc ukazují jednu udivující vlastnost právě vytvořeného oblouku.

```
h1, h2 {margin: 0; padding: 0 0.5em;}
h1 {background-color: #EBE; border-bottom: 1px solid #969;}
h2 {margin: 0 -20px; padding: 0;
background: transparent url(p2toprt.gif) right top no-repeat;}
```

Vzpomeňte si, že prvek `h1` ve skutečnosti vytvořený oblouk obtéká. Pokud jste prošli přes Projekt 9, asi tušíte, k čemu zde dojde. Při pohledu na obrázek 10.17 si můžete ověřit, zdali byl váš odhad správný.

Nyní se v celé kráse ukazuje, proč jsme se k poněkud pracnějšímu nastavení transparentnosti u světle purpurových oblastí grafických prvků tvořících oblouk. Pokud bychom to neučinili, měli bychom nyní co do činění s velkými plochami světle purpurové barvy, které by překryly pozadí elementu `h1`. Nyní ovšem po nich není ani památky – jsou totiž jakoby zasunuty pod plovoucí elementy. Velkou výhodu lze spatřovat v tom, že vůbec nezáleží na velikosti použitého textu – a tudíž ani na výšce, kterou má element `h1` – tento efekt to nijak neovlivňuje.

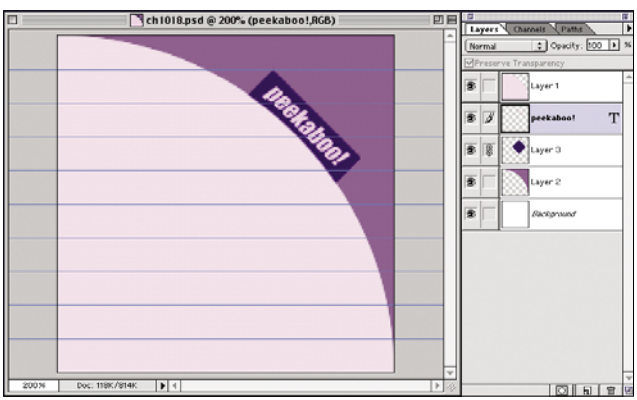
Obr. 10.17

Celistvý oblouk se symbolickým klobouček s titulkem.

Druhý oblouk kloboučku

Tato finta hovoří doslova sama za sebe, ovšem, naše práce je hotova teprve z poloviny. Pravý horní roh kloboučku je stále silně hranatý. Budeme zde potřebovat umístit zrcadlově otočenou variantu předchozího oblouku. Postup je na první pohled zřejmý, nicméně, tentokrát půjdeme na věc trochu jinak.

Zřejmě se opět nevyhneme nutnosti rozdělit větší grafický prvek na jednotlivé dílčí úseky Mohli bychom sice ve vodorovném směru otočit levý oblouk a znovu jej použít, nicméně, my si to trošku ztížíme a nahradíme jej motivem zachyceným na obrázku 10.18.



Obr. 11.18

Zbývající oblouk kloboučku (zobrazen v prostředí Adobe Photoshop).

Všimli jste si boxu s textem „peekaboo!“, který vyčnívá mimo oblouk? Samozřejmě – je to nedílná součást grafického prvku. Rozhodně ji ale nenecháme ležet ladem a nějak ji využijeme. Nejprve ale musíme absolvovat stejnou proceduru jako s prvním obloukem: z každého úseku vezmeme přiměřeně velkou část tak, aby zahrnovala oblast s tmavě purpurovou barvou a uložíme ji jako obrázek ve formátu GIF, kde světle purpurovou složku nastavíme jako transparentní. Všechny tyto soubory pak shromáždíme v adresáři s názvem `curve-r`.

Nyní doplníme něco málo HTML kódu. Kód musí být do stránky vložen přesně stejným způsobem, jak to ukazuje následující výpis:

```
<img src="curve-l/curve1.gif" alt=" " class="curve-l">
<img src="curve-r/curve1.gif" alt=" " class="curve-r">
<img src="curve-l/curve2.gif" alt=" " class="curve-l">
<img src="curve-r/curve2.gif" alt=" " class="curve-r">
<img src="curve-l/curve3.gif" alt=" " class="curve-l">
<img src="curve-r/curve3.gif" alt=" " class="curve-r">
<img src="curve-l/curve4.gif" alt=" " class="curve-l">
<img src="curve-r/curve4.gif" alt=" " class="curve-r">
<img src="curve-l/curve5.gif" alt=" " class="curve-l">
<img src="curve-r/curve5.gif" alt=" " class="curve-r">
<img src="curve-l/curve6.gif" alt=" " class="curve-l">
<img src="curve-r/curve6.gif" alt=" " class="curve-r">
<img src="curve-l/curve7.gif" alt=" " class="curve-l">
<img src="curve-r/curve7.gif" alt=" " class="curve-r">
<img src="curve-l/curve8.gif" alt=" " class="curve-l">
<img src="curve-r/curve8.gif" alt=" " class="curve-r">
<h1>much room to think</h1>
```

Co vězí za tímto střídáním kódu pro levý a pravý oblouk? Nuže, podle CSS nesmí žádný plovoucí element sahat výše, než kde se nachází vrchol nějakého předchozího elementu. Z toho tedy plyne, že pokud bychom umístili všechny obrázky tvořící pravý oblouk za obrázky, které patří k levému oblouku, vrchol pravého oblouku by se nacházel na stejné úrovni, jako spodek levého oblouku. A to opravdu není výsledek, který chceme.

A pochopitelně, bez správných stylů se nedopracujeme k požadovanému efektu. Vše, co potřebujeme, jsou v podstatě zrcadlově otočené hodnoty stylů z levého oblouku. Výsledek je pak ukázán na obrázku 10.19.

```
img.curve-l {float: left; clear: left; margin: 0 0.5em 0 0;}
img.curve-r {float: right; clear: right; margin: 0 0 0 0.5em;}
</style>
```

Obr. 10.19

Naše houba má hezky tvarovaný klobouček.

Aby vše zůstalo vcelku

V současné podobě hrozí našemu designu jedno potenciální nebezpečí. Při dostatečném zmenšení textu by se klobouček stal kratším než dva velké oblouky, které mu dominují. V takovém případě by sice oblouky zůstaly nedotčeny, nicméně by vystupovaly ze spodního okraje kloboučku.

Díky dřívějšímu efektu designu tomu můžeme jednoduše zabránit. Spodní část kloboučku, který má na starosti prvek `div` s hodnotou selektoru `p1end`, může být upravena tak, aby se po jeho stranách nemohl objevit žádný plovoucí element.

```
div#p1end {text-align: right; margin: 0; padding: 0; clear: both;
  background: #FDF url(p1botleft.gif) bottom left no-repeat;}
```

Tento postup nám zajistí celistvost kloboučku, i když to v krajním případě povede k velkému množství volného místa mezi jeho koncem a začátkem nožky.

Hrátky s odkazy

Samo o sobě by nemělo moc velký smysl doplnit oblouk nápisem „peekaboo!“ a nechat ho tam ležet jen tak. Ostatně, vždyť vypadá jako záložka, kterou by mělo být možné použít, že ano? V odkaz jej proměníme doslova ve vteřině.

Podstata finty spočívá v tom, že určíme, ze kterých částí pravého oblouku je nutno udělat odkazy. Bližší pohled nám prozradí, že „peekaboo!“ zasahuje do obrázků `curve2.gif` až `curve5.gif`. Takže, tyto obrázky nyní zabalíme do elementů definujících hypertextové odkazy.

```
<img src="curve-l/curve2.gif" alt=" " class="curve-l">
<a href="peekaboo.html"><img src="curve-r/curve2.gif"
  alt="peekaboo!" class="curve-r"></a>
<img src="curve-l/curve3.gif" alt=" " class="curve-l">
<a href="peekaboo.html"><img src="curve-r/curve3.gif"
  alt=" " class="curve-r"></a>
<img src="curve-l/curve4.gif" alt=" " class="curve-l">
<a href="peekaboo.html"><img src="curve-r/curve4.gif"
  alt=" " class="curve-r"></a>
<img src="curve-l/curve5.gif" alt=" " class="curve-l">
<a href="peekaboo.html"><img src="curve-r/curve5.gif"
  alt=" " class="curve-r"></a>
<img src="curve-l/curve6.gif" alt=" " class="curve-l">
```

Tímto se „peekaboo“ (a části purpurového oblouku po jeho pravé straně) promění ve funkční odkaz. Je zde ovšem nebezpečí, že by webový prohlížeč mohl přijít s nápadem zobrazit kolem těchto obrázkových odkazů orámování. To by poničilo náš pěkný oblouk, a tak raději doplníme styl, jehož úkolem bude tomu zabránit.

```
img.curve-r {float: right; clear: right; margin: 0 0 0 0.5em;}
img {border-width: 0;}
</style>
```

Úprava velikosti obrázků

Jakmile úspěšně dokončíte vytváření designu, jako je tento, není zcela od věci přidat pro každý obrázek HTML atributy `width` a `height`. Je sice možné nastavit rozměry obrázků s využitím CSS, nicméně, nestálo by to za vynaložené úsilí, poněvadž byste museli pro každý obrázek specifikovat vlastní `id` a následně vytvořit i speciální pravidla. Použití HTML atributů představuje mnohem více přímočařejší postup.

Tento styl ve skutečnosti zabrání výskytu orámování kolem všech obrázků na stránce, včetně mírného zaoblení u spodní části kloboučku a vrcholu nožky. A to jistě potěší, protože v žádném případě nechceme mít tyto prvky orámované.

Trocha zarovnání neuškodí

Jeden z posledních zásahů bude spočívat v zarovnání textu do bloku. Tím se získá co největší možné roztáhnutí textu. V konečném důsledku to povede k tomu, že pravé a levé okraje každého řádku budou v zákrytu – samozřejmě by tomu tak bylo, pokud by do toho nezasahovaly oblouky. Mimo jiných věcí dosáhneme také zvýraznění toku textu podél pravého oblouku, což je možné vidět na obrázku 10.20.

```
p {margin: 0; padding: 0.5em 1em; text-align: justify;}
```

Obr. 10.20

V našem layoutu bude text zarovnán do bloku.

Ve výpisu 10.3 je uvedena kompletní definice námi vytvořených stylů.

Výpis 10.3 – Finální definice stylů

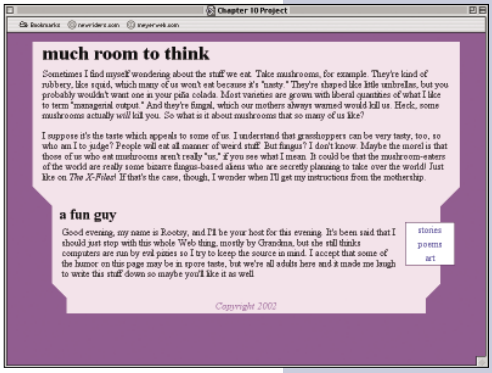
```
<style type="text/css">
body {background: #969; color: black;}
div.wrap {background: #FDF; color: black; margin: 0 2em;}
p {margin: 0; padding: 0.5em 1em; text-align: justify;}
h1, h2 {margin: 0; padding: 0 0.5em;}
h1 {background-color: #EBE; border-bottom: 1px solid #969;}
h2 {margin: 0 -20px; padding: 0;
  background: transparent url(p2toprt.gif) right top no-repeat;}
h2 img {vertical-align: top;}
div#p1 {margin: 0 2em;}
div#p1end {text-align: right; margin: 0;
  background: #FDF url(p1botleft.gif) bottom left no-repeat;
  clear: both;}
div#p2 {margin: 0 10em;}
div#menu {float: right; width: 5em;
  padding: 0; margin: 0 -1.5em 0.25em 0.5em;
  border: 1px solid #747; background: white;}
```

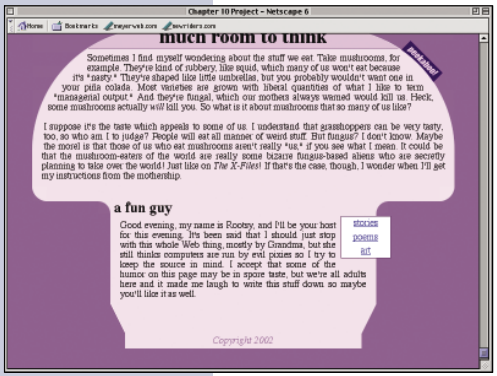
```
div#menu a {display: block; text-align: center;
  padding: 0.2em 0.5em;}
div#footer {margin: 0 10em; padding: 0.25em;
  text-align: center; font-style: italic; color: #969;
  border: 1.5em solid; border-color: #FDF #969 #969;}
img.curve-l {float: left; clear: left; margin: 0 0.5em 0 0;}
img.curve-r {float: right; clear: right; margin: 0 0 0 0.5em;}
img {border-width: 0;}
</style>
```

Možnosti dalšího rozšíření

Pochopitelně existuje nepřeberné množství způsobů, jak zužitkovat principy vyzkoušené v tomto projektu. Zde je několik námětů:

1. Vraťte se k layoutu na obrázku 10.8 (před nápadem s houbou) a přepracujte jej tak, že jednotlivé části webu jsou navzájem spojeny pomocí šikmých čar (jedna část už tak ostatně spojena je). S tím souvisí změna prvků `div` na nejvyšší úrovni a doplnění druhé části o rámečky. V souvislosti se spodním okrajem u `div#p2` si vzpomeňte na další způsob, jak nechat zmizet orámování.





2. Zkuste změnit šířku orámování kolem horní části zápatí a pozorujte, jak to ovlivní způsob vykreslení šikmých čar. Například, použijte horní orámování o šířce `3em` a `0.75em`. Zamyslete se nad tím, jak vzájemné poměry šířky horního a bočního orámování ovlivní výsledné úhly.

3. Zkuste umístit menu ze své současné pozice někam úplně jinam, například úplně dovnitř nožky s následným s posunutím nahoru. Nebo je možné menu přesunout úplně mimo nožku, pokud se vám bude zdát, že to vypadá lépe. Kromě toho, zamyslete se nad tím, jak je možné upravit styl odkazů v menu tak, aby lépe vynikly.