KAPITOLA

5

Manipulace s modelem DOM

Témata kapitoly:

- **Pracujeme s atributy**
- **Manipulace se stromem DOM**
- **Kopírování elementů**
- **Metody pro načítání a změnu obsahu**
- **Metody pro manipulaci s modelem DOM v kostce**

Web vzniká díky spolupráci webových serverů a webových prohlížečů. Tvorba dokumentů, které můžeme zobrazovat ve webových prohlížečích, bývala obvykle výsadou webového serveru. Dosud popsané techniky však toto uspořádání lehce posunuly – již umíme měnit vzhled dokumentu HTML za běhu (změnou kaskádových stylů). Abychom však utužili naše znalosti jazyka JavaScript, musíme se naučit modifikovat samotný dokument.

S pomocí knihovny jQuery můžeme měnit dokument HTML, pokud použijeme rozhraní poskytované modelem DOM. Budeme moci vytvářet elementy a text, kdykoliv je budeme potřebovat. Rovněž je budeme umět odstraňovat, nebo je dynamicky upravovat přidáváním, odstraňováním nebo změnou jejich atributů.

Pracujeme s atributy

V prvních čtyřech kapitolách této knihy jsme používali metody `addClass()` a `removeClass()`. Ukázali jsme si na nich, jak měnit vzhled elementů na stránce. Tyto metody ve skutečnosti pouze ovlivňují atribut `class` (nebo v případě modelu DOM vlastnost `className`) daného elementu. Metoda `addClass()` vytváří nebo rozšiřuje tento atribut, kdežto metoda `removeClass()` odstraňuje nebo zkracuje daný atribut. Navíc máme k dispozici metodu `toggleClass()`, jež střídavě přidává a odstraňuje třídu, takže máme opravdu mnoho možností, jak nakládat s třídami.

Nicméně atribut `class` je jen jedním z mnoha atributů, s nimiž můžeme chtít pracovat – k dalším atributům patří například atributy `id`, `rel` a `href`. Knihovna jQuery poskytuje pro práci s nimi metody `attr()` a `removeAttr()`. Prostřednictvím těchto metod bychom dokonce mohli měnit atribut `class`, ale k tomuto účelu se lépe hodí specializované metody `addClass()` a `removeClass()`, pro-

tože se umí lépe vypořádat s více třídami u jediného elementu; jako v případě elementu `<div class="prvni druha">...</div>`.

Další atributy

S některými atributy nelze snadno pracovat bez pomoci knihovny jQuery. Kromě toho nám tato knihovna umožňuje upravovat více atributů současně; tj. podobně jako u vlastností jazyka CSS, které jsme předávali metodě `css()` v kapitole 4, „Styly a animace."

Kupříkladu bychom mohli nastavit zároveň atributy `id`, `rel` a `title` všem odkazům. Začněme ukázkovou stránkou HTML:

```
<h1 id="f-title">Flatland: vícerozměrný román</h1>
<div id="f-author">Edwin A. Abbott</div>
<h2>Část I, kapitola 3</h2>
<h3 id="f-subtitle">O obyvatelích Flatlandu</h3>
<div id="excerpt">výňatek</div>
<div class="chapter">
  <p class="square">Našimi profesionály a džentlemeny jsou čtverce (k nimž
    se řadím i já) a pětistranné útvary, nebo-li <a
    href="http://cs.wikipedia.org/wiki/P%C4%9Bti%C3%BAheln%C3%ADk">
    pětiúhelníky</a>.</p>
  <p class="nobility hexagon">Nad nimi stojí šlechta, kterou dělíme do
    několika tříd &ndash; nejnižší třídu tvoří šestistrané útvary, nebo-li
    <a href="http://cs.wikipedia.org/wiki/%C5%A0esti%C3%BAheln%C3%ADk">
    šestiúhelníky</a>, a jak se zvětšuje počet jejich stran, můžou
    dosáhnout až váženého titulu
    <a href="http://cs.wikipedia.org/wiki/Mnoho%C3%BAheln%C3%ADk">
    mnohoúhelný</a>, nebo také vícestranný. Jakmile začne být počet stran
    neúnosný a samotné strany příliš malé, nelze daný útvar rozlišit od
    <a href="http://cs.wikipedia.org/wiki/Kruh">kruhu</a>, jenž spadá do
    kulatého, nebo-li kněžského stavu, což je nejvyšší třída ze všech.</p>
  <p><span class="pull-quote">Zákon přírody <span class="drop">káže</span>,
    že syn by měl mít <strong>o jednu stranu více</strong> než jeho
    otec</span>, aby každá generace poskočila (zpravidla) o jeden stupeň
    nahoru v geometrickém žebříčku. Synem čtverce je tudíž pětiúhelník;
    synem pětiúhelníku je šestiúhelník atd.</p>
  <p>Toto pravidlo ale neplatí vždy pro obchodníky, ještě méně pro vojáky
    a dělníky; ti si ve skutečnosti stěží zaslouží označení lidské útvary,
    protože nemají všechny své strany stejně dlouhé. Pro ně tedy zákon
    přírody neplatí &ndash; syn rovnoramenného trojúhleníku (tj.
    trojúhelníku se dvěma shodnými stranami) zůstává rovnoramenným
    trojúhelníkem. Naději však neztrácí ani rovnoramenný trojúhelník
    a stále doufá, že jeho potomci se konečně odrazí od své
```

```
      základny&hellip;</p>
    ...
</div>
```

Nyní musíme projít všechny odkazy uvnitř elementu `div` s třídou `chapter` a přidělit jim postupně atributy. Kdybychom chtěli nastavit jedinou hodnotu atributu všem odkazům, mohli bychom to udělat pomocí jediného řádku kódu uvnitř obsluhující funkce v konstrukci `$(document).ready()`.

Výpis 5.1

```
$(document).ready(function() {
  $('div.chapter a').attr({rel: 'external'});
});
```

Metoda `attr()` přijímá dva argumenty podobně jako metoda `css()` – prvním z nich je název atributu a druhým je nová hodnota. Mnohem častěji však budeme používat objekt s dvojicemi složenými z klíče a hodnoty jako ve výpisu 5.1. Díky tomuto zápisu můžeme jednoduše upravovat několik atributů současně.

Výpis 5.2

```
$(document).ready(function() {
  $('div.chapter a').attr({
    rel: 'external',
    title: 'Dozvěďte se více v encyklopedii Wikipedia.'
  });
});
```

Hodnotové funkce zpětného volání

Technika předávání objektu metodě `attr()` je dostačující, pokud chceme atributu nebo atributům přiřadit stejnou hodnotu pro všechny vyhledané elementy. Někdy však musí mít přidávané atributy také jiné hodnoty. Typickým příkladem je hodnota atributu `id`, která musí být v rámci daného dokumentu jedinečná, aby se náš kód jazyka JavaScript choval předvídatelně. Jedinečný identifikátor lze nastavit všem odkazům tak, že použijeme další prvek metod typu `css()` nebo `attr()`, a to **hodnotovou funkci zpětného volání**.

Hodnotová funkce zpětného volání je obyčejná funkce, kterou zapisujeme místo hodnoty atributu. Knihovna jQuery volá tuto funkci pro každý element z vybrané skupiny elementů. Novou hodnotou daného atributu potom bude návratová hodnota, kterou tato funkce vrátí. S použitím této techniky můžeme kupříkladu generovat různé hodnoty atributu `id` pro jednotlivé elementy.

Výpis 5.3

```
$(document).ready(function() {
  $('div.chapter a').attr({
    rel: 'external',
```

```
    title: 'Dozvědět se více v encyklopedii Wikipedia.',
    id: function(index, oldValue) {
      return 'wikilink-' + index;
    }
  });
});
```

Kdykoliv knihovna jQuery spustí naši hodnotovou funkci zpětného volání, předává jí dva argumenty. První z nich je celé číslo označující pořadí průchodu – tentokrát ho používáme pro sestavení identifikátoru, který tudíž bude mít posloupnost `wikilink-0`, `wikilink-1` atd. Druhý argument jsme v předchozím kódu nepoužili, avšak obsahuje hodnotu, kterou daný atribut obsahoval před úpravou.

Pomocí atributu `title` lákáme uživatele, aby se dozvěděli více o odkazovaném termínu v encyklopedii Wikipedia. V kódu jazyka HTML, který jsme si doposud představili, ukazují všechny odkazy na encyklopedii Wikipedia. Uvážíme-li možnost existence jiných typů odkazů, měli bychom náš selektor trochu upřesnit.

Výpis 5.4

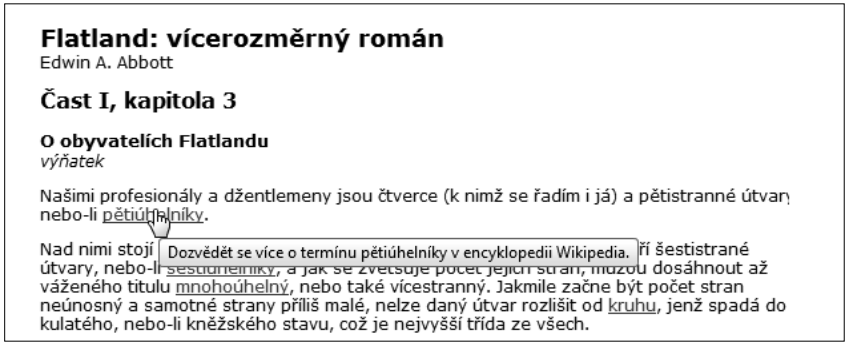
```
$(document).ready(function() {
  $('div.chapter a[href*="wikipedia"]').attr({
    rel: 'external',
    title: 'Dozvědět se více v encyklopedii Wikipedia.',
    id: function(index, oldValue) {
      return 'wikilink-' + index;
    }
  });
});
```

Průzkum metody `attr()` ukončíme tak, že vylepšíme atribut `title` těchto odkazů, aby lépe informoval o cíli daného odkazu. K tomuto účelu se opět skvěle hodí hodnotová funkce zpětného volání.

Výpis 5.5

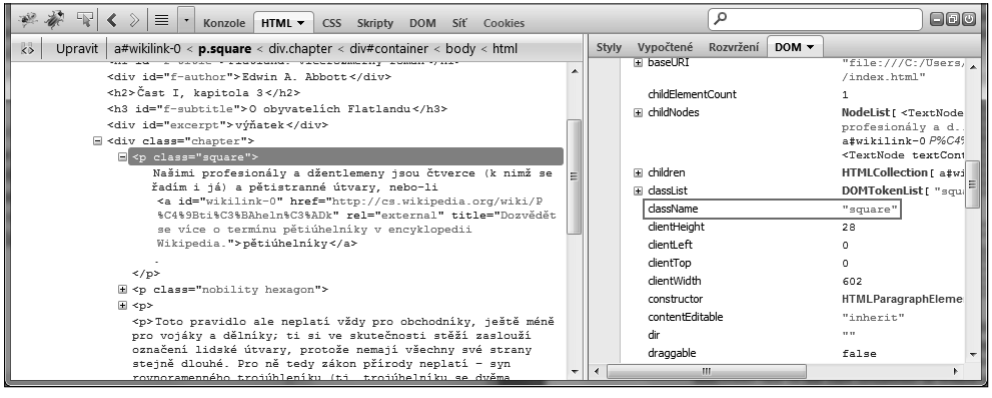
```
$(document).ready(function() {
  $('div.chapter a[href*="wikipedia"]').attr({
    rel: 'external',
    title: function() {
      return 'Dozvědět se více o termínu ' + $(this).text() +
        ' v encyklopedii Wikipedia.';
    },
    id: function(index, oldValue) {
      return 'wikilink-' + index;
    }
  });
});
```

Tentokrát jsme použili **kontext** hodnotové funkce zpětného volání. Klíčové slovo `this` totiž ukazuje na element, s nímž pracujeme v době volání této funkce (podobně jako obsluhujících funkcí událostí). Zde zabalujeme tento element do objektu knihovny jQuery, abychom mohli načíst jeho textový obsah pomocí metody `text()` (kterou jsme si představili v kapitole 4, „Styly a animace). Tím jsme pěkně upřesnili popisky našich odkazů:



Vlastnosti elementů modelu DOM

Dříve jsme se zmínili, že existuje rozdíl mezi **atributy** jazyka HTML a **vlastnostmi** modelu DOM. Atributy jsou hodnoty zapisované mezi uvozovky v kódu jazyka HTML pro danou stránku, zatímco vlastnosti jsou hodnoty, ke kterým přistupujeme z kódu jazyka JavaScript. Atributy i vlastnosti můžeme jednoduše sledovat ve vývojářském nástroji, jakým je například doplněk Firebug na následujícím obrázku:



Nástroj Firebug nám ukazuje, že zvýrazněný element `p` má třídu `class` s hodnotou `square`. V pravém panelu lze vidět, že k tomuto elementu patří odpovídající vlastnost `className` taktéž s hodnotou `square`. Právě můžeme pozorovat poněkud vzácnou situaci, v níž se atribut jmenuje jinak než jemu odpovídající vlastnost.

Ve většině případů fungují související atributy a vlastnosti stejně a knihovna jQuery rovněž řeší rozdíly v pojmenování za nás. Nyní se ale zaměříme na rozdíly mezi nimi. Kromě názvů se můžou lišit také datové typy jejich hodnot – například hodnotou atributu `checked` je textový řetězec, kdežto vlastnost `checked` obsahuje pravdivostní hodnotu. U takových vlastností

s pravdivostní hodnotou je lepší pracovat přímo s touto vlastností, a ne s atributem, čímž dosáhneme konzistentního chování mezi webovými prohlížeči.

Hodnoty vlastností můžeme načítat a nastavovat v knihovně jQuery pomocí metody `prop()`:

```
// Načítáme aktuální hodnotu vlastnosti checked.
var currentlyChecked = $('.muj-checkbox').prop('checked');

// Nastavujeme novou hodnotu vlastnosti checked.
$('.muj-checkbox').prop('checked', false);
```

Metoda `prop()` se chová ve všech ohledech stejně jako metoda `attr()`, což znamená, že jí můžeme předávat objekt s více hodnotami a hodnotové funkce zpětného volání.

Manipulace se stromem DOM

Metody `attr()` a `prop()` jsou velmi užitečné, když potřebujeme změnit samotný dokument. Co když ale budeme chtít změnit strukturu dokumentu? Abychom mohli pracovat se stromem DOM, musíme důkladněji poznat funkci, která leží v samotném srdci knihovny jQuery.

Funkce $() v novém kabátu

Od začátku této knihy hledáme elementy v dokumentu pomocí funkce `$()`. Již víme, že tato funkce funguje jako továrna produkující objekty knihovny jQuery, které ukazují na elementy odpovídající zadanému selektoru jazyka CSS.

To však není všechno, co tato funkce umí. Tato funkce se totiž může pochlubit schopností měnit stránku nejen po vizuální, ale také obsahové stránce. Pokud totiž funkci `$()` předáme kód jazyka HTML, vytvoří z něj novou strukturu modelu DOM.

TIP

Opět nesmíme zapomínat na to, že může být nebezpečné měnit vzhled nebo poskytovat informace jen v těch webových prohlížečích, v nichž je povolený jazyk JavaScript. Důležité informace bychom měli zpřístupnit všem lidem, a ne jen těm, kteří používají správný software.

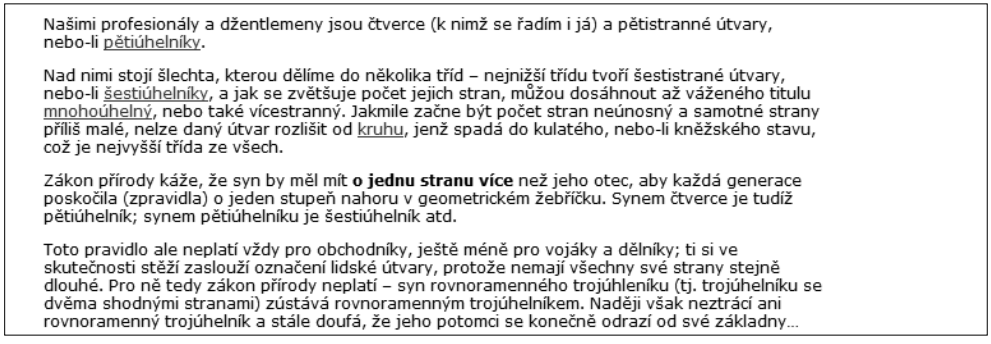
Tvorba nových elementů

Obvyklým prvkem stránek s často kladenými otázkami je odkaz **zpět nahoru**, jenž se zobrazuje za každou dvojicí otázek a odpovědí. Lze tvrdit, že tyto odkazy nemají žádný podstatný význam, a proto je můžeme vytvořit jazykem JavaScript jako vylepšení pro jistou podmnožinu návštěvníků. V našem příkladu přidáme odkaz **zpět nahoru** za všechny odstavce a navíc vytvoříme kotvu, na kterou budou tyto odkazy směřovat. Začneme vytvořením těchto nových elementů.

Výpis 5.6

```
$(document).ready(function() {
  ...
  $('<a href="#top">zpět nahoru</a>');
  $('<a id="top"></a>');
});
```

Na prvním řádku jsme vytvořili odkaz **zpět nahoru** a cílovou kotvu na druhém řádku. Na naší stránce se však zatím žádné odkazy neobjevily:


Našimi profesionály a džentlemeny jsou čtverce (k nimž se řadím i já) a pětistranné útvary, nebo-li pětiúhelníky.

Nad nimi stojí šlechta, kterou dělíme do několika tříd – nejnižší třídu tvoří šestistrané útvary, nebo-li šestiúhelníky, a jak se zvětšuje počet jejich stran, můžou dosáhnout až váženého titulu mnohoúhelný, nebo také vícestranný. Jakmile začne být počet stran neúnosný a samotné strany příliš malé, nelze daný útvar rozlišit od kruhu, jenž spadá do kulatého, nebo-li kněžského stavu, což je nejvyšší třída ze všech.

Zákon přírody káže, že syn by měl mít **o jednu stranu více** než jeho otec, aby každá generace poskočila (zpravidla) o jeden stupeň nahoru v geometrickém žebříčku. Synem čtverce je tudíž pětiúhelník; synem pětiúhelníku je šestiúhelník atd.

Toto pravidlo ale neplatí vždy pro obchodníky, ještě méně pro vojáky a dělníky; ti si ve skutečnosti stěží zaslouží označení lidské útvary, protože nemají všechny své strany stejně dlouhé. Pro ně tedy zákon přírody neplatí – syn rovnoramenného trojúhleníku (tj. trojúhelníku se dvěma shodnými stranami) zůstává rovnoramenným trojúhelníkem. Naději však neztrácí ani rovnoramenný trojúhelník a stále doufá, že jeho potomci se konečně odrazí od své základny...

Přestože výše uvedeným kódem vytváříme nové elementy, nevkládáme je do stránky. Musíme prohlížeči sdělit, kam by měl tyto elementy umístit. K tomuto účelu můžeme používat spoustu metod knihovny jQuery pro vkládání elementů.

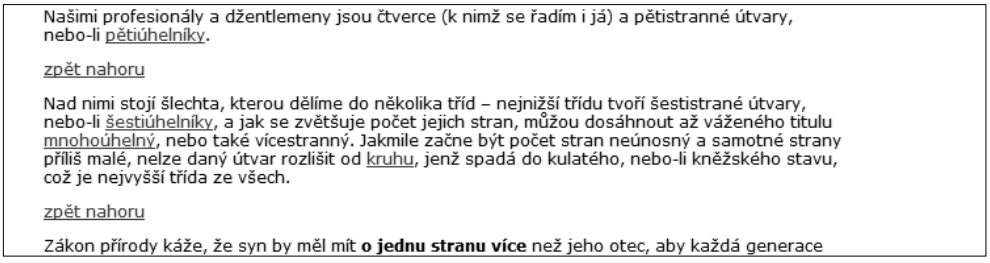
Vkládáme nové elementy

Knihovna jQuery nabízí několik metod pro vkládání elementů do dokumentu. S jejich pomocí definujeme vztah mezi novým a existujícím obsahem. Kupříkladu naše odkazy **zpět nahoru** by se měly objevit za každým odstavcem, proto použijeme vhodně pojmenovanou metodu `insertAfter()`.

Výpis 5.7

```
$(document).ready(function() {
  ...
  $('<a href="#top">zpět nahoru</a>').insertAfter('div.chapter p');
  $('<a id="top"></a>');
});
```

Nyní jsme vložili odkazy **zpět nahoru** do stránky (a do modelu DOM) za všechny odstavce uvnitř elementu `div` s třídou `chapter`, takže je webový prohlížeč zobrazí podobně jako na tomto obrázku:



Všimněte si, že nové odkazy se zobrazují na samostatném řádku, a ne uvnitř odstavce. Je tomu tak z toho důvodu, že metoda `insertAfter()` (a obrácená metoda `insertBefore()`) přidává obsah mimo uvedený element.

Přidané odkazy bohužel prozatím nefungují. Stále musíme vložit do stránky kotvu s identifikátorem `top`. Tentokrát použijeme metodu, která vkládá elementy do jiných elementů.

Výpis 5.8

```
$(document).ready(function() {
  ...
  $('<a href="#top">zpět nahoru</a>').insertAfter('div.chapter p');
  $('<a id="top"></a>').prependTo('body');
});
```

Pomocí nového kódu vkládáme naši kotvu na začátek elementu `body`; jinými slovy – na začátek stránky. S metodami `insertAfter()` a `prependTo()` už odkazy **zpět nahoru** fungují správně.

Jakmile si představíme metodu `appendTo()`, získáme kompletní sadu metod pro vkládání nových elementů:

1. Metoda `insertBefore()` vkládá obsah před (tj. mimo) vybrané elementy.
2. Metoda `prependTo()` vkládá obsah na začátek (tj. dovnitř) vybraných elementů.
3. Metoda `appendTo()` vkládá obsah na konec (tj. dovnitř) vybraných elementů.
4. Metoda `insertAfter()` vkládá obsah za (tj. mimo) vybrané elementy.

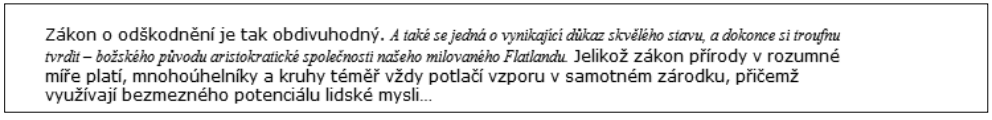
Přesouvání elementů

Při tvorbě odkazů **zpět nahoru** jsme vytvořili nové elementy, které jsme vložili do naší stránky. Rovněž můžeme vyjmout elementy z jednoho místa ve stránce a vložit je na jiné místo. V praxi se tento postup používá pro dynamické umístění a formátování poznámek pod čarou. Jedna poznámka pod čarou se objevuje také v původním textu románu Flatland, který používáme v následujícím příkladu, ale pro účely tohoto příkladu jsme označili také jiné části textu jako poznámky pod čarou:

```
<p>Zákon o odškodnění je tak obdivuhodný.
  <span class="footnote">
    A také se jedná o vynikající důkaz skvělého stavu, a dokonce si troufnu
```

```
      tvrdit &ndash; božského původu aristokratické společnosti našeho
      milovaného Flatlandu.
    </span>
    Jelikož zákon přírody v rozumné míře platí, mnohoúhelníky a kruhy téměř
    vždy potlačí vzporu v samotném zárodku, přičemž využívají bezmezného
    potenciálu lidské mysli&hellip;
</p>
```

Náš dokument HTML obsahuje celkem tři poznámky pod čarou, ale zde jsme si uvedli jen jednu z nich. Text poznámky pod čarou se nachází přímo uvnitř odstavce, od něhož ho oddělujeme elementem `span` s třídou `footnote`. Pomocí tohoto označení v kódu jazyka HTML, vymezujeme oblast této poznámky pod čarou. Pravidlo stylu v naší šabloně stylů jí přiřazuje kurzívu, takže výsledný odstavec vypadá jako na následujícím obrázku:



Nyní musíme vyjmout poznámky pod čarou a přesunout je do spodní části dokumentu. Konkrétně je umístíme mezi element `div` s třídou `chapter` a element `div` s třídou `footer`.

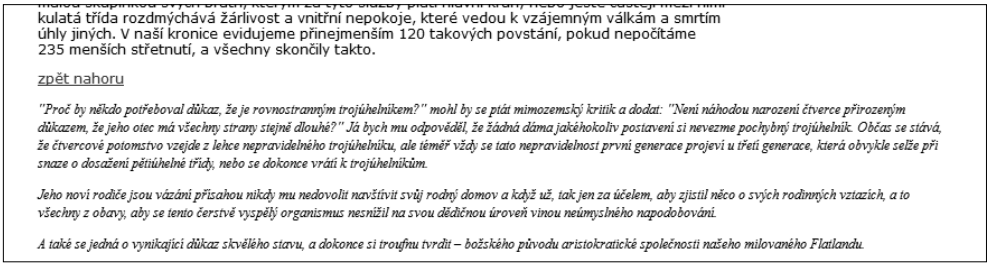
Nezapomínejme, že i s implicitní iterací je pořadí zpracovávání elementů pevně dané – začíná u kořene stromu DOM a pokračuje směrem dolů. Protože bychom měli zachovat správné pořadí poznámek pod čarou i po jejich přesunu, měli bychom volat `insertBefore('#footer')` jako v níže uvedeném výpisu. Tím umístíme poznámku pod čarou bezprostředně před element `div` s identifikátorem `footer`, takže první poznámka pod čarou se přesune mezi element `div` s třídou `chapter` a element `div` s identifikátorem `footer`, druhá poznámka pod čarou se umístí mezi první poznámku pod čarou a element `div` s identifikátorem `footer` atd. Kdybychom totiž zavolali `insertAfter('div.chapter')`, obrátili bychom pořadí poznámek pod čarou.

Doposud náš zdrojový kód vypadá podobně jako v následujícím výpisu.

Výpis 5.9

```
$(document).ready(function() {
  ...
  $('span.footnote').insertBefore('#footer');
});
```

Poznámky pod čarou jsou obalené elementem `span`, jenž se standardně zobrazuje jako řádkový – tzn. na jediném řádku jeden za druhým. To jsme však vyřešili v naší šabloně stylů, v níž jsme elementům `span` s třídou `footnote` přiřadili vlastnost `display` s hodnotou `block`, pokud se nacházejí mimo element `div` s třídou `chapter`. Díky tomu tyto poznámky pod čarou vypadají takto:



Poznámky pod čarou se zobrazují na správném místě, ale ještě s nimi budeme mít spoustu práce. Robustní řešení poznámek pod čarou by mělo:

1. Číslovat jednotlivé poznámky pod čarou.
2. Označovat číslem poznámky pod čarou místo v textu, z něhož jsme ji vyjmuli.
3. Vytvořit odkaz ze zástupného čísla v textu, který bude odkazovat na příslušnou poznámku pod čarou, a také odkaz z poznámky pod čarou, jejž nasměrujeme zpět na původní místo v textu.

Zabalování elementů

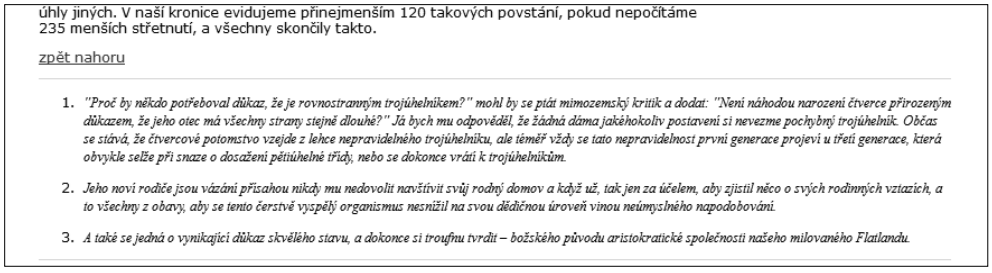
Poznámky pod čarou bychom mohli očíslovat tak, že bychom napsali čísla přímo do kódu dokumentu, ale zde využijeme běžný uspořádaný seznam, jenž očísluje poznámky za nás. Za tímto účelem vytvoříme obalující element `ol`, do něhož zabalíme všechny poznámky pod čarou, přičemž každou z nich bude obepínat element `li`. Tentokrát nám přijdou pomoci **balicí metody**.

Když zabalujeme elementy do jiného elementu, musíme mít jasno, jestli chceme každému elementu přidělit samostatný obalující element, nebo všechny elementy zabalit do jediného obalujícího elementu. Pro číslování poznámek pod čarou potřebujeme oba typy obalujících elementů.

Výpis 5.10

```
$(document).ready(function() {
  ...
  $('span.footnote')
    .insertBefore('#footer')
    .wrapAll('<ol id="notes"></ol>')
    .wrap('<li></li>');
});
```

Potom, co jsme vložili naše poznámky pod čarou před zápatí, je teď všechny zabalujeme do elementu `ol` voláním metody `wrapAll()`. Posléze balíme jednotlivé tyto poznámky do samostatných elementů `li` pomocí metody `wrap()`. Na následujícím obrázku lze vidět, že jsme tím správně očíslovali naše poznámky pod čarou.


úhly jiných. V naší kronice evidujeme přinejmenším 120 takových povstání, pokud nepočítáme 235 menších střetnutí, a všechny skončily takto.

zpět nahoru

1. *"Proč by někdo potřeboval důkaz, že je rovnostranným trojúhelníkem?" mohl by se ptát mimozemský kritik a dodat: "Není náhodou narození čtverce přirozeným důkazem, že jeho otec má všechny strany stejně dlouhé?" Já bych mu odpověděl, že žádná dáma jakéhokoliv postavení si nevezme pochybný trojúhelník. Občas se stává, že čtvercové potomstvo vzejde z lehce nepravidelného trojúhelníku, ale téměř vždy se tato nepravidelnost první generace projeví u třetí generace, která obvykle selže při snaze o dosažení pětiúhelné třídy, nebo se dokonce vrátí k trojúhelníkům.*
2. *Jeho noví rodiče jsou vázáni přísahou nikdy mu nedovolit navštívit svůj rodný domov a když už, tak jen za účelem, aby zjistil něco o svých rodinných vztazích, a to všechny z obavy, aby se tento čerstvě vyspělý organismus nesnížil na svou dědičnou úroveň vinou neúmyslného napodobování.*
3. *A také se jedná o vynikající důkaz skvělého stavu, a dokonce si troufnu tvrdit – božského původu aristokratické společnosti našeho milovaného Flatlandu.*

Nyní už nám nic nebrání v tom, abychom označili a očíslovali místa, ze kterých jsme tyto poznámky vyjmuli. Pokud toho chceme dosáhnout jednoduše, musíme přepsat náš současný kód tak, aby se nespoléhal na implicitní iteraci.

Metoda `each()` pro explicitní iteraci se velmi podobá cyklu `for`. Můžeme ji použít v případě, že by kód, jejž chceme provést pro každý z vyhledaných elementů, byl příliš složitý na zápis s implicitní iterací. Této metodě předáváme funkci zpětného volání, kterou knihovna jQuery zavolá pro každý vyhledaný element.

Výpis 5.11

```
$(document).ready(function() {
  ...
  var $notes = $('<ol id="notes"></ol>').insertBefore('#footer');
  $('span.footnote').each(function(index) {
    $(this).appendTo($notes).wrap('<li></li>');
  });
});
```

Důvod, proč měníme náš kód, si objasníme za chvíli. Nejprve musíme pochopit, co sdělujeme nové funkci zpětného volání uvnitř metody `each()`.

Podobně jako u předchozích funkcí zpětného volání ukazuje klíčové slovo `this` na aktuální element modelu DOM. V kódu z výpisu 5.11 vytváříme objekt knihovny jQuery ukazující na jedinou poznámku pod čarou (element `span`), kterou vkládáme na konec poznámek (element `ol`) a nakonec ji zabalujeme do elementu `li`.

Až budeme označovat místa, z nichž jsme tyto poznámky vyjmuli, můžeme použít parametr funkce zpětného volání pro metodu `each()`. Tento parametr určuje pořadí průchodu začínající nulou a zvyšující se o jedna pro každé volání této funkce. To znamená, že toto pořadí bude vždy o jedna menší než číslo aktuální poznámky pod čarou. Díky této skutečnosti vytvoříme vhodné popisky v textu.

Výpis 5.12

```
$(document).ready(function() {
  ...
  var $notes = $('<ol id="notes"></ol>').insertBefore('#footer');
  $('span.footnote').each(function(index) {
    $('<sup>' + (index + 1) + '</sup>').insertBefore(this);
```

```
    $(this).appendTo($notes).wrap('<li></li>');
  });
});
```

Než vytrhneme poznámku z kontextu a umísťujeme ji na spodek stránky, vytváříme element `sup` obsahující číslo této poznámky, jejž vkládáme do textu. Pořadí prováděných akcí je důležité – číslo musíme vložit předtím, než přesuneme poznámku, jinak bychom ztratili přehled o její původní pozici. Výraz `(index + 1)` musíme uzavřít do závorek, aby ho prohlížeč interpretoval jako sčítání, a ne zřetězování textových řetězců.

Na naší stránce se již zobrazují značky poznámek pod čarou:

Narození rovnostranného potomka rovnoramenných trojúhelníkům je v naší zemi důvodem k oslavám po mnoho let. Pokud důkladné přezkoumání takového novorozence prokáže jeho pravidelnost, je po slavnostním obřadu přijat do třídy rovnostranných trojúhelníků. Tohoto potomoka okamžitě odeberou stále hrdým, avšak zarmouceným, rodičům a přidělí ho do rodiny bezdětných rovnostraných trojúhelníků. [2]

zpět nahoru

Zákon o odškodnění je tak obdivuhodný. [3] Jelikož zákon přírody v rozumné míře platí, mnohoúhelníky a kruhy téměř vždy potlačí vzporu v samotném zárodku, přičemž využívají bezmezného potenciálu lidské mysli...

Obrácené metody pro vkládání

Ve výpisu 5.12 jsme vložili obsah před element, který jsme následně přemístili na jiné místo v dokumentu. Při práci s elementy v knihovně jQuery většinou používáme řetězení více metod. Zde jsme to ale udělat nemohli, protože objekt `this` je cílovým objektem metody `insertBefore()`, ale „předmětem" metody `appendTo()`. Obrácené metody pro vkládání nám pomůžou překonat toto omezení.

Ke všem metodám pro vkládání (jako jsou například metody `insertBefore()` a `appendTo()`) existují obrácené metody. Tyto obrácené metody vykonávají stejné úlohy jako jejich standardní verze, ale prohazují „předmět" a „cíl" vkládání. Například:

```
$('<p>Ahoj</p>').appendTo('#container');
```

se shoduje s:

```
$('#container').appendTo('<p>Ahoj</p>');
```

Použijeme-li metodu `before()`, převrácenou variantu metody `insertBefore()`, můžeme doplnit řetězení.

Výpis 5.13

```
$(document).ready(function() {
  ...
  var $notes = $('<ol id="notes"></ol>').insertBefore('#footer');
  $('span.footnote').each(function(index) {
    $(this)
      .before('<sup>' + (index + 1) + '</sup>')
```

```
      .appendTo($notes)
      .wrap('<li></li>');
  });
});
```

TIP

Obrácené vkládací metody přijímají funkci jako argument; podobně jako metody `attr()` a `css()`. Tato funkce se spouští jedenkrát pro každý cílový element, a ta by měla vracet textový řetězec s vkládaným kódem jazyka HTML. Tuto techniku bychom mohli použít, ale jelikož u každé poznámky pod čarou nastává tato situace vícekrát, metoda `each()` je mnohem lepším řešením.

Nyní jsme připraveni implementovat náš poslední cíl – vytvořit odkaz z číselné značky směřující na poznámku pod čarou a odkaz z poznámky pod čarou na číselnou značku v textu. K dosažení tohoto cíle budeme potřebovat čtyři prvky pro každou poznámku pod čarou v našem dokumentu – dva odkazy (jeden v textu a druhý pod danou poznámkou pod čarou) a dva atributy `id` na stejných místech. Jelikož argument předávaný metodě `before()` bude složitější, měli bychom si představit nový způsob zápisu pro tvorbu textových řetězců.

Ve výpisu 5.13 jsme připravovali číselnou značku řetězením textových řetězců prostřednictvím operátoru +. Jedná se o velmi užitečnou techniku, ale s větším množstvím zřetězování začíná být poněkud nepřehledná. Velké textové řetězce můžeme sestavovat zavoláním metody `join()` na pole. Níže uvedené příkazy produkují stejné textové řetězce:

```
var str = 'a' + 'b' + 'c';
var str = ['a', 'b', 'c'].join('');
```

Přestože v tomto příkladu musíme napsat o něco více znaků, metoda `join()` může zpřehlednit kód v případech, kdy by sčítací znaménko při řetězení textových řetězců mohlo mást. Opět přepišme náš kód; tentokrát s metodou `join()`.

Výpis 5.14

```
$(document).ready(function() {
  ...
  var $notes = $('<ol id="notes"></ol>').insertBefore('#footer');
  $('span.footnote').each(function(index) {
    $(this)
      .before([
        '<sup>',
        index + 1,
        '</sup>'
      ].join(''))
      .appendTo($notes)
      .wrap('<li></li>');
  });
});
```