

KAPITOLA 15

Případová studie: FooReader.NET

Kapitola 7 se zabývala syndikací obsahu pomocí RSS a Atom a tím, jak jednoduše sdílet informace. Za účelem zobrazit informace z několika různých zdrojů, volala aplikace tzv. agregátor, který zkombinoval různé zdroje do jediné lokace. Agregátor usnadňuje získávání aktuálních informací posbíraných z celého webu, což je jednodušší, než navštěvovat spoustu stránek každý den.

FooReader.NET je webově založený agregátor .NET RSS/Atom vycházející z ColdFusionbased Fooreader (<http://reader.forgetfoo.com/>) od ForgetFoo. Proč ovšem budovat webový RSS/Atom agregátor, když už existuje mnoho tradičních aplikací se stejnou funkcionalitou (včetně e-mailových aplikací a prohlížečů)? Důvodů je několik:

- **Web je multiplatformní.** Agregátorem založeným na webu zajistíte, že jej bude moci používat kterýkoliv uživatel s Internet Explorerem 6+, Firefoxem nebo Operou.
- **Web je umístěný centrálně.** Jedním z problémů tradičních agregátorů, které jsou nainstalovány na počítači, je oblast správy dat při více lokacích. Pokud chcete číst dané zdroje v práci a současně doma, musíte nainstalovat agregátor na dvou počítačích a nastavit jim požadované zdroje. Agregátor založený na webu tento problém odstraňuje, protože každá změna v seznamu zdrojů je vidět vždy, bez ohledu na to, odkud uživatel přistupuje k agregátoru.

Tato kapitola popisuje, jak je FooReader.NET vytvořen pomocí Ajaxu. A jak už nyní asi tušíte – jako každá webová aplikace se bude i tato aplikace skládat ze dvou hlavních druhů komponent: z komponent na straně klienta a z komponent na straně serveru.

Pokud máte nainstalovanou nějakou verzi Visual Studia, otevřete ji a vytvořte novou webovou stránku nazvanou FooReader. Ujistěte se, že použitý jazyk je C#.

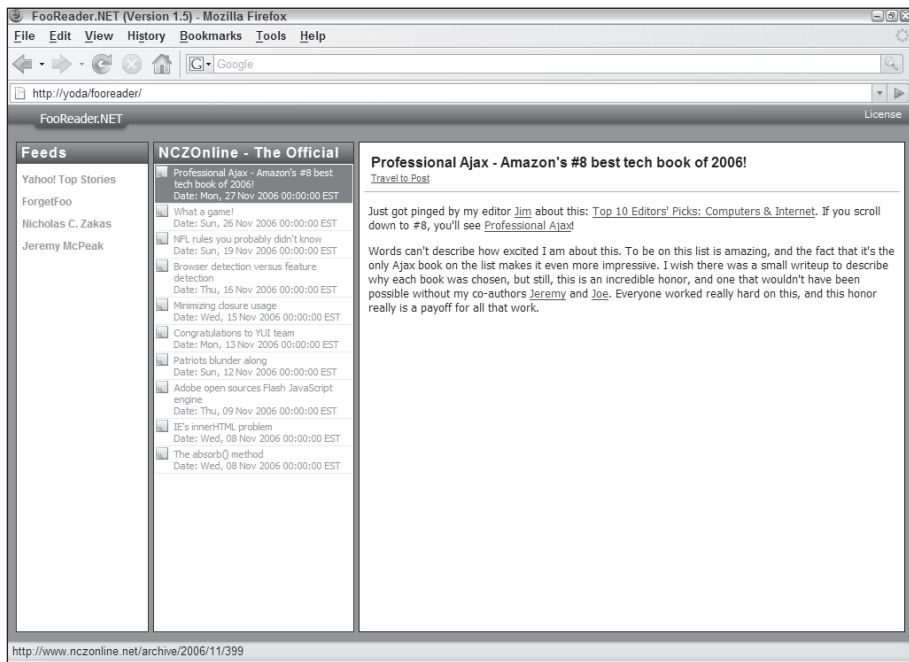
Komponenty na straně klienta

Komponenty na straně klienta jsou v tomto ajaxovém řešení zodpovědné za komunikaci se serverem a zobrazování přijatých dat uživateli. Pro FooReader.NET je nezbytných několik následujících komponent na straně klienta, aby se dosáhlo těch správných uživatelských prožitků.

- **Uživatelské rozhraní.** UI svazuje uživatele s daty. Protože UI je v podstatě webová stránka, jsou použity obvyklé technologie podporované ve webových prohlížečích. Kód stránky je vytvořen pomocí HTML. CSS je pak použito pro vylepšení celkového dojmu.
- **XParser.** Javascriptová knihovna odpovědná za požadavky na zdroje a jejich analýzu.
- **Javascriptový kód.** Řídí UI, získává data od XParseru a zobrazuje je uživateli. Tento kód bude obsažen v souboru `fooreader.js`.

Uživatelské rozhraní

Klíč ke každé úspěšné aplikaci spočívá v návrhu uživatelského rozhraní. Když uživatel neumí aplikaci ovládat, není důvod, aby taková aplikace existovala. FooReader.NET je navržen pro jednoduché použití. Ve skutečnosti dost silně těží z uživatelského rozhraní Microsoft Outlook 2003 (a pozdějších verzí). Jak sami vidíte na obrázku níže, jeho rozhraní se skládá ze tří částí. První dvě části mají pevnou šířku, zatímco třetí část má šířku proměnlivou (viz obrázek 15-1).



Obrázek 15-1. Uživatelské rozhraní aplikace.

Rozhraní je obsaženo v souboru `default.htm` a jeho kód je následující:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xml:lang="en" lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>FooReader.NET (Version 1.5)</title>
    <link rel="stylesheet" type="text/css" href="css/FooReader.css" />
    <script type="text/javascript" src="js/zxml.src.js"></script>
    <script type="text/javascript" src="js/XParser.js"></script>
    <script type="text/javascript" src="js/FooReader.js"></script>
  </head>
  <body>
    <div id="divLoading">
      
    </div>
    <div id="divTopBar">
      
      <div id="divLicense">
        <a href="http://creativecommons.org/licenses/by-nc-sa/2.5/"
          title="Some Rights Reserved" target="_blank">License</a>
      </div>
    </div>
    <div id="divPaneContainer">
      <div id="divFeedsPane">
        <div class="paneheader">Feeds</div>
        <div id="divFeedList"></div>
      </div>
      <div id="divItemsPane">
        <div id="divViewingItem" class="paneheader">Items</div>
        <div id="divItemList"></div>
      </div>
      <div id="divReadingPane">
        <div id="divMessageContainer">
          <div id="divMessageHeader">
            <div id="divMessageTitle"></div>
            <a href="" id="aMessageLink" title="Click to goto posting."
              target="_new">Travel to Post</a>
          </div>
          <div id="divMessageBody"></div>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

    </div>
  </body>
</html>

```

Prvními dvěma přímými potomky těla dokumentu jsou prvky `<div>` s identifikátory `divLoading` a `divTopBar`. První prvek zajišťuje ten úkol uživatelského rozhraní, který uživateli říká, že aplikace načítá zdroj. Objevuje se vždy, když je učiněn požadavek na server aplikace a schová se, když server odpoví a odpověď je zpracována. Prvek s ID `divTopBar` se snaží napodobit titulkový řádek z tradičních aplikací Windows a říká uživateli, že aplikace, kterou nyní používá, je FooReader.NET (a současně poskytuje odkaz na licenci Creative Commons, pod níž je aplikace zveřejněna).

Dále je tu prvek s identifikátorem `divPaneContainer`. Jak už jeho jméno napovídá, v tomto prvku jsou obsaženy všechny tři panely aplikace. Tento kontejnerový prvek tak dovoluje, aby bylo možné ostylevat všechny tyto tři panely současně (na rozdíl od aplikace stylu pro každý prvek zvlášť).

První panel, který je pojmenován jako `divFeedsPane`, zobrazuje jednotlivé zdroje v podobě odkazů, na něž může uživatel kliknout. Vnořený prvek `<div>` s identifikátorem `divFeedList` v panelu se zdroji dovoluje, aby seznam zdrojů byl do dokumentu vepsán dynamickým způsobem. Samotné zdroje jsou bloky prvků `<a>` s CSS třídou `feedlink`. HTML kód těchto odkazů je zde:

```

<a href="[url of feed]" class="feedlink"
  title="Load [feed title]">Feed title</a>

```

Když uživatel klikne na nějaký zdroj, bude daný zdroj načten do druhého prostředního panelu. Tento panel obsahuje dva hlavní prvky `<div>`, které jsou používány pro zobrazování informací:

- První je prvek `<div>` s id nastaveným na `divViewingItem`. Jedná se vlastně o hlavičku tohoto druhého panelu. Tento prvek totiž zobrazuje, který zdroj je právě načten.
- Druhý prvek je `<div>`, jehož atribut `id` je nastaven na `divItemList`. Tento prvek bude obsahovat seznam RSS prvků `<item>` nebo Atom prvků `<entry>`. Oba tyto druhy prvků jsou do `divItemList` přidány dynamicky.

HTML struktura jednotlivých položek z druhého panelu je následující:

```

<a class="itemlink" href="[item url]"
  frfeeditem="[item number]" id="item[number]">
  <div class="itemheadline">[Headline]</div>
  <div class="itemdate">[Date]</div>
</a>

```

Toto HTML je zcela standardní, kromě atributu `frfeeditem` prvku `<a>`. Když je zdroj načten a položka je přidána do stránky, je každé položce přiřazeno číslo, pomocí kterého je identifikována.

Když uživatel klikne na nějakou položku, bude načtena do posledního třetího panelu s identifikátorem `divReadingPane`. Tento panel se skládá ze tří prvků, které zobrazují informace z dané položky. První prvek s identifikátorem `divMessageTitle` je místem, kde jsou zobrazeny prvky `<title>` ze zdroje RSS nebo Atom. Druhý prvek má ID `aMessageLink` a jeho atribut `href` je

měněn dynamicky. A konečně – poslední prvek je `divMessageBody` a zobrazuje samotný obsah prvků `<rss:description>` a `<atom:content>`.

Tato stránka vyžaduje jeden stylový předpis (soubor `FooReader.css`) a tři následující javascriptové soubory: knihovnu `zXML`, `XParser` a soubor `FooReader.js`, který obsahuje veškerou funkčnost u klienta.

Protože FooReader.NET používá XParser, nebude tato aplikace pracovat v Safari. Ovšem bude bez problémů pracovat v IE 6+, ve Firefoxu 1+ a v Opeře 9+, což jsou ty nejpoužívanější webové prohlížeče.

Kompletního uživatelského rozhraní je dosaženo kombinací CSS a JavaScriptu. CSS nastavuje velikost těla dokumentu (což je nezbytné pro Operu), velikost všech tří panelů a samozřejmě i vzhled všech ostatních prvků. JavaScript pak umisťuje prvky do panelů.

Stylování rozhraní

Jediný použitý stylový předpis ve FooReader.NET je v souboru `FooReader.css`, který je umístěn v adresáři `css`. Pro naši aplikaci je klíčová velikost zobrazovacího pole v prohlížeči. Stránka je explicitně nastavena tak, aby používala všechno dostupné vertikální místo v okně prohlížeče.

```
html, body {  
height: 100%;  
margin: 0px;  
overflow: hidden;  
background-color: gray;  
font: 11px verdana, arial, helvetica, sans-serif;  
}
```

Vlastnost `height` je nastavena na 100%. Tato vlastnost je nezbytná kvůli Opeře. Vlastnost `overflow` je nastavena na hodnotu `hidden`, protože některé prvky ve stránce způsobují zobrazení posuvníků v okně prohlížeče. Naším cílem je dosáhnout toho, aby aplikace FooReader.NET vypadala jako normální aplikace a viditelné posuvníky na úrovni dokumentu takovému dojmu logicky brání. Posuvníky v jednotlivých panelech samozřejmě budou k dispozici.

Horní pruh

Následující pravidla jsou pro `divTopBar` a pro další prvky v něm vnořené:

```
/* Horní pruh */  
#divTopBar {  
background: gray url("../img/top_bg.gif") repeat-x;  
height: 31px;  
padding-left: 25px;
```

```
position: relative;
}
```

Horní pruh bude mít šedou barvu pozadí, která bude ve shodě s šedým pozadím stránky. Pro tento horní pruh je dále specifikován obrázek na pozadí (soubor `top_bg.gif`). Levá strana má nastavenou výplň na 25 pixelů, což posune prvek `` s logem doprava. Relativní pozicování horního pruhu dovoluje všem dceřiným prvkům (kterým je například prvek `divLicense`), aby mohly být pozicovány vůči tomuto prvku:

```
#divLicense {
position: absolute;
right: 10px;
top: 3px;
}
#divLicense a {
color: white;
padding: 1px 4px 2px 4px;
}
#divLicense a:hover {
background: blue url("../img/toolbar_back.gif") repeat-x;
border: 1px solid #000080;
color: #000080;
padding: 0px 3px 1px 3px;
}
```

Horní pruh obsahuje pouze jeden odkaz – licenci. Tento odkaz je pozicován absolutně 10 pixelů od pravé hrany a 3 pixely od horní hrany prvku `divTopBar`. Tento odkaz s identifikátorem `divLicense` má nastavenou bílou barvu textu. Když na něj uživatel ukáže kurzorem myši, odkaz změní své pozadí na obrázek `toolbar_back.gif`, který se opakuje v horizontálním směru. Dále se objeví 1 pixel široké modré orámování, barva textu se změní na tmavě modrou, a výplň se zmenší o jeden pixel na každé straně. Tato změna velikosti výplně je nezbytná, protože tento prvek má specifikováno orámování. Z tohoto vyplývá, že kdybychom hodnoty výplně nezmenšili, odkaz by se ve skutečnosti zvětšil o 1 pixel na každé straně.

Informace o načítání stránky

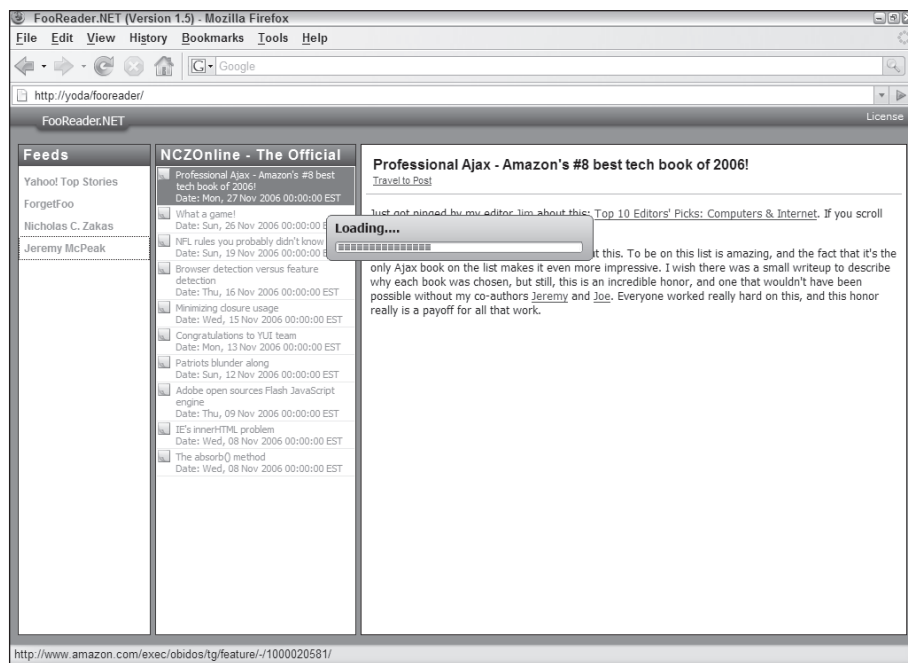
Ve zdrojovém HTML kódu je v prvku `<div>` s identifikátorem `divLoading` umístěn obrázek nazvaný jako `progress.gif`. Tento obrázek, který je 5 až 10 pixelů vysoký a široký něco málo pod 300 pixelů, zobrazuje animaci v podobě ukazatele s průběhem načítání stránky.

Tento prvek `<div>` s ID `divLoading` je pozicován absolutně. Protože má vlastnost `display` nastavenou na hodnotu `none`, bude kompletně vyjmut z toku dokumentu, takže bude neviditelný.

```
/* The loading <div/> */
#divLoading {
```

```
position: absolute;
display: none;
top: 20%;
left: 35%;
width: 302px;
z-index: 5;
background: transparent url("../img/loading.gif") no-repeat;
padding: 30px 10px; }
```

Pravděpodobně nejdůležitějším pravidlem v tomto stylu je vlastnost `z-index`. Protože tento prvek `<div>` je úplně prvním prvkem v těle dokumentu (z hlediska struktury HTML), je logicky překryt ostatními následujícími prvky ve stránce. Ovšem díky tomu, že nastavíte vlastnost `z-index` na hodnotu větší než 0, bude tento prvek `<div>` s obrázkovým ukazatelem průběhu umístěn "nad" ostatními prvky (z hlediska osy `z`), takže se pro uživatele stane viditelným (samozřejmě až po změně hodnoty vlastnosti `display` na `block`, což si popíšeme o několik stránek dále).



Obrázek 15-2. Informace o stavu načítání.

Styly pro panely

Všechny tři panely jsou obsaženy uvnitř kontejnerového prvku `<div>` s ID `divPaneContainer`. Tento prvek umísťuje hromadně všechny panely na požadovanou pozici, což je 10 pixelů od levé hrany prohlížeče a 10 pixelů od horního pruhu s logem aplikace a licencí:

```
#divPaneContainer {  
position: relative;  
top: 10px;  
left: 10px;  
}
```

Pozicování tohoto jediného kontejnerového prvku vás osvobozuje od jednotlivého otravného pozicování všech tří panelů.

Panely se zdroji a položkami rovněž mají hlavičku ve své horní části. Tato hlavička dává uživateli vědět, jaký typ dat daný panel obsahuje. Hlavičky jsou 20 pixelů vysoké a jejich text je bílý a tučný. Zde je jejich styl:

```
.paneheader {  
height: 20px;  
background-image: url("../img/header_background.gif");  
font: bold 16px arial;  
color: white;  
padding: 2px 0px 2px 5px;  
letter-spacing: 1px;  
overflow: hidden;  
}
```

Levý panel se zdroji

První panel úplně vlevo je panel, který obsahuje zdroje. CSS kód pro HTML prvek s identifikátorem `divFeedsPane` je velmi jednoduchý, jak ostatně ukazuje následující pravidlo:

```
#divFeedsPane {  
float: left;  
width: 148px;  
border: 1px solid navy;  
background-color: white;  
overflow: hidden;  
}
```

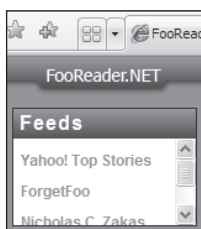
Prvek je nastaven tak, aby byl 148 pixelů široký a byl plovoucí vlevo. Nastavení vlastnosti `float` posune prvek doprava nebo doleva. V tomto případě je `divFeedsPane` posunut doleva. To umístí prostřední panel s položkami doprava, ačkoliv jsou to oba blokové elementy. Aby panel měl pořad stejnou velikost, je jeho vlastnost `overflow` nastavena na `hidden`. Toto nastavení skryje jakýkoliv obsah, který by se mohl rozšířit za hranice panelu. Nicméně je žádoucí, aby obsahem panelu bylo možné rolovat. Tohle zajišťuje prvek s ID `divFeedList`, jehož definice stylu je následující:

```
#divFeedList {  
padding: 5px 1px 5px 1px;
```



```
overflow: auto;
}
```

Tento prvek obsahuje odkazy na jednotlivé zdroje. Jeho vlastnost `overflow` je nastavena na hodnotu `auto`. Toto umožňuje, aby se v prvku `divFeedList` mohl objevit posuvník, pokud bude obsah tohoto prvku větší než je rozměr `divFeedsPane`. Posuvníky se objeví uvnitř `divFeedsPane`, a tím bude možné obsahem panelu rolovat, zatímco panel samotný se nezmění (viz obrázek 15-3).



Obrázek 15-3. Rolovací lišta uvnitř panelu.

Odkazy v tomto panelu jsou stylovány jako blokové prvky. Mají specifikovanou výplň o velikosti 5px, aby byly vizuálně odděleny od okolních odkazů.

```
a.feedlink {
display: block;
padding: 5px;
font: bold 12px arial;
text-decoration: none;
color: #5583d3;
}
a.feedlink:hover {
color: #3768B9;
text-decoration: underline;
}
```

Prostřední panel s položkami

CSS styly pro prostřední panel s položkami se trochu podobají stylům pro levý panel se zdroji:

```
#divItemsPane {
float: left;
width: 225px;
border: 1px solid navy;
background-color: white;
margin-left: 5px;
margin-right: 5px !important;
margin-right: 2px;
```

```
overflow: hidden;
}
```

Tento panel je také plovoucí vlevo a jeho vlastnost `overflow` je rovněž nastavena na `hidden`. Pravý a levý okraj přidávají malou mezeru mezi oběma panely. Zde se ovšem vyskytne drobný problém, protože IE6 obsahuje chybu ohledně správné velikosti okrajů. Aby každý prohlížeč vykreslil UI stejně, musí být použita deklarace `!important`. Tu má nastavenou první vlastnost `margin-right` ve stylu. Toto v praxi znamená, že IE7, Firefox a Opera použijí hodnotu této vlastnosti bez ohledu na skutečnost, že ve stylu je ještě specifikována jedna vlastnost `margin-right`. Tuto druhou vlastnost `margin-right` použije IE6. Ano – je to jeden řádek ve stylu navíc, který tam v podstatě být nemusí, nicméně odměnou vám bude jednotný vzhled UI ve všech čtyřech prohlížečích.

Položky v tomto panelu mají rovněž specifikovány své vlastní styly. Pokud se podíváte o několik stránek zpět na strukturu HTML kódu stránky, zjistíte, že jednotlivé položky jsou vlastně obyčejné prvky `<a>` s CSS třídou `itemlink`. Pro tyto odkazy chceme, aby měly dva stavy. První stav je pochopitelně jejich výchozí stav (před kliknutím kurzorem myši). Pokud uživatel ovšem klikne na nějakou položku, pak kód JavaScriptu změní jejich CSS třídu na `itemlink-selected`. Oba stavy pro odkazy (resp. položky) používají tuto shodnou definici CSS:

```
a.itemlink, a.itemlink-selected {
border-bottom: 1px solid #EAE9E1;
background-image: url("../img/item_icon.gif");
background-repeat: no-repeat;
cursor: pointer;
text-decoration: none;
display: block;
padding: 2px;
font: 11px tahoma;
}
```

Následující styly pro položky prostředního panelu slouží pro vzájemné odlišení těchto dvou stavů. Pověsimně si, že je zde také definována pseudotřída `:hover`, která je určena pro normální stav:

```
a.itemlink {
background-color: white;
color: #808080;
}
a.itemlink:hover {
background-color: #D3E5FA;
}
a.itemlink:hover .itemheadline {
color: black;
}
.itemheadline,.itemdate {
```

```
margin-left: 20px;
}
a.itemlink-selected {
background-color: #316AC5;
color: white;
}
```

Třetí panel pro čtení

Třetí panel se od předchozích dvou panelů liší v tom, že nemá pevně specifikovanou šířku, jak to ostatně ukazuje následující kód CSS:

```
#divReadingPane {
margin: 0px 20px 0px 0px;
border: 1px solid black;
background-color: white;
height: 100%;
overflow: hidden;
}
```

Prohlížeč automaticky nastaví šířku tohoto prvku tak, aby vyplnila zbývajícím volný prostor prvku `divPaneContainer`. Nastavení vlastnosti `margin` vytvoří pravý okraj o velikosti 20 pixelů. Stejně jako u ostatních panelů je i zde vlastnost `overflow` nastavena na `hidden`. Ovšem na rozdíl od ostatních dvou panelů je zde nastavena výška na 100%. Díky tomu bude mít tento panel stejnou výšku jakou má `divPaneContainer`.

Přímým potomkem `divReadingPane` je `divMessageContainer`, který obsahuje jednotlivé prvky zpráv. Následující styl nastavuje výplň na 5 pixelů z každé strany.

```
#divMessageContainer {
padding: 5px;
}
```

První částí zprávy je hlavička, která obsahuje nadpis článku a odkaz pro přesměrování uživatele ke kompletnímu textu článku. CSS kód je následující:

```
#divMessageHeader {
height: 34px;
background-color: white;
border-bottom: 1px solid #ACA899;
padding: 8px;
}
#divMessageTitle {
font: bold 16px arial;
}
```

```
#aMessageLink {  
font: 11px arial; }
```

Prvek s ID `divMessageBody` je místem, kde se zobrazuje obsah jednotlivých položek. Na tento prvek je aplikováno následující pravidlo CSS.

```
#divMessageBody {  
background-color: white;  
padding: 0px 0px 0px 5px;  
font: 13px tahoma;  
overflow: auto;  
}
```

Tento styl mimo jiné zajišťuje, že se objeví posuvníky pro rolování obsahu, pokud výška obsahu bude větší, než výška třetího panelu pro obsah.

Výška jednotlivých prvků, které obsahují obsah (tzn. prvky s identifikátory `divFeedList`, `divItemList` a `divMessageBody`, viz struktura HTML kódu, jež byla uvedena dříve v této kapitole) není nastavována prostřednictvím CSS – je řízena pomocí JavaScriptu.

Řízení UI

Kód JavaScriptu, který je obsažen v souboru `fooreader.js`, ovládá všechny aspekty UI. Získává seznam zdroje, analyzuje ho a naplňuje panel se zdroji. Dále vytváří objekty `XParser` pro požadavky, přijímá a analyzuje zdroje RSS a Atom a používá tyto informace pro naplnění položek a panelu pro čtení. Dále nastavuje velikost mnoha prvků UI při změně velikosti okna. Řečeno ve zkratce – jedná se o páteřní komponentu na straně klienta.

Pomocné funkce

Ačkoliv je většina kódu obsažena v objektu `fooReader`, dvě funkce zůstanou samostatně, aby nám pomohly se změnou velikostí prvků. První funkce se jmenuje `getStyle()` a zajišťuje změnu hodnoty konkrétní vlastnosti stylu bez ohledu na použitý prohlížeč. Funkce přijímá dva argumenty – prvek a název CSS vlastnosti.

```
function getStyle(oElement, sProperty) {  
    var sStyle;  
    if (typeof window.getComputedStyle == "undefined") {  
        sStyle = oElement.currentStyle[sProperty];  
    } else {  
        sStyle = getComputedStyle(oElement, "")[sProperty];  
    }  
    return sStyle;  
}
```

Kód používá vlastnost Internet Exploreru `currentStyle` a W3C metodu DOM `getComputedStyle()` pro získání hodnoty dané vlastnosti.

Druhá funkce je `getStyleName()` a vykonává podobnou aktivitu. Rozdílem je to, že vrací celé číslo (integer) místo řetězce (string):

```
function getStyleNumber(oElement, sProperty) {  
    return parseInt(getStyle(oElement, sProperty));  
}
```

Objekt `fooReader`

Jak už bylo zmíněno dříve, objekt `fooReader` obsahuje většinu javascriptového kódu, čímž tak tvoří hlavní část aplikace. Obsahuje různé vlastnosti a metody, které jsou nezbytné pro správný běh UI. Je to jediný objekt svého druhu v aplikaci:

```
var fooReader = {  
    parser : null,  
    feeds : [],  
    //HTML prvky  
    divFeedList : null,  
    divViewItem : null,  
    divItemList : null,  
    divMessageTitle : null,  
    aMessageLink : null,  
    divMessageBody : null,  
    divLoading : null,  
    selectedItem : null,  
    //zde bude další kód  
}  
//zde bude další kód
```

Vlastnosti, které jsou použity v této definici, jsou následující:

- První vlastnost je `parser` a obsahuje objekt `XParser`.
- Další je pole nazvané `feeds` a obsahuje seznam zdrojů.
- Dalších sedm vlastností jsou odkazy na objekty `HTMLElement`. Tyto prvky jsou používány skrze celou relaci aplikace, což je dobrý důvod, aby byly cachovány.
- Poslední vlastnost je `selectedItem`, která slouží jako ukazatel na položku (prvek `<a>`), na niž uživatel naposledy kliknul.

Všechny tyto vlastnosti jsou inicializovány na `null`, aby se předešlo případným chybám.

Inicializace UI

Předtím, než uživatel může pracovat s UI, musí být inicializovány vlastnosti `HTMLElement`. K tomu slouží metoda `init()`, která kromě přiřazení vlastností pro prvky nastavuje velikost těch prvků UI, jež potřebují dynamickou velikost. Tato metoda je volána při událostech `load` a `resize` okna. Proto funkce existuje jako metoda objektu `fooReader`, ale její definice leží mimo definici hlavního objektu. To je jediný vizuální rozdíl mezi touto metodou a ostatními členy `fooReader`.

```
var fooReader = {
    parser : null,
    feeds : [],
    //HTML prvky
    divFeedList : null,
    divViewingItem : null,
    divItemList : null,
    divMessageTitle : null,
    aMessageLink : null,
    divMessageBody : null,
    divLoading : null,
    selectedItem : null,
    //zde bude další kód
}

fooReader.init = function (evt) {
    evt = evt || window.event;
    if (evt.type == "load") { //Tyto inicializace proběhnou
        //jen při události load
        fooReader.divFeedList = document.getElementById("divFeedList");
        fooReader.divViewingItem = document.getElementById("divViewingItem");
        fooReader.divItemList = document.getElementById("divItemList");
        fooReader.divMessageTitle = document.getElementById("divMessageTitle");
        fooReader.aMessageLink = document.getElementById("aMessageLink");
        fooReader.divMessageBody = document.getElementById("divMessageBody");
        fooReader.divLoading = document.getElementById("divLoading");
        //zde bude další kód
    }
    var divPaneContainer = document.getElementById("divPaneContainer");
    var divReadingPane = document.getElementById("divReadingPane");
    var divMessageContainer = document.getElementById("divMessageContainer");
    var divMessageHeader = document.getElementById("divMessageHeader");
    //zde bude další kód
};

window.onload = fooReader.init;
window.onresize = fooReader.init;
```

Protože vývojáři musí pořád zápasit s rozdíly mezi různými implementacemi modelu událostí, první řádek této metody získává správný objekt události. Dále je ověřeno, jestli typ události byl `load`. Pokud ano, jsou pomocí metody `getElementById()` přiřazeny různé vlastnosti `HTMLElement`.

Mimo blok `if` jsou získány ostatní vlastnosti `HTMLElement` a přiřazeny proměnným. Tyto proměnné jsou používány pro operace s velikostí prvku:

```
//zde je kód objektu fooReader
fooReader.init = function (evt) {
    evt = evt || window.event;
    if (evt.type == "load") { // Tyto inicializace proběhnou
        // jen při události load
        fooReader.divFeedList = document.getElementById("divFeedList");
        fooReader.divViewItem = document.getElementById("divViewItem");
        fooReader.divItemList = document.getElementById("divItemList");
        fooReader.divMessageTitle = document.getElementById("divMessageTitle");
        fooReader.aMessageLink = document.getElementById("aMessageLink");
        fooReader.divMessageBody = document.getElementById("divMessageBody");
        fooReader.divLoading = document.getElementById("divLoading");
        //zde bude další kód
    }
    var divPaneContainer = document.getElementById("divPaneContainer");
    var divReadingPane = document.getElementById("divReadingPane");
    var divMessageContainer = document.getElementById("divMessageContainer");
    var divMessageHeader = document.getElementById("divMessageHeader");
    var iDocHeight = document.documentElement.clientHeight;
    divPaneContainer.style.height = iDocHeight -
        divPaneContainer.offsetTop - 12 + "px";
    var iFeedsListHeight = divPaneContainer.offsetHeight -
        fooReader.divViewItem.offsetHeight -
        getStyleNumber(fooReader.divFeedList, "paddingTop") -
        getStyleNumber(fooReader.divFeedList, "paddingBottom");
    fooReader.divFeedList.style.height = iFeedsListHeight + "px";
    //zde bude další kód
};
window.onload = fooReader.init;
window.onresize = fooReader.init;
```

Zvýrazněný kód začíná získáním výšky oblasti pro zobrazení pomocí `document.documentElement.clientHeight`. Tato hodnota je pak společně s `divPaneContainer.offsetTop` použita pro nastavení výšky prvku s identifikátorem `divPaneContainer`. Číselná konstanta 12 je použita pouze pro vizuální účely, protože poskytuje 12 pixelů volného prostoru mezi spodní hranou kontejneru a spodní hranou okna.

Následuje přiřazení proměnné `iFeedsListHeight`, která je použita pro nastavení výšky `divFeedList`. Výška tohoto prvku je nastavena tak, aby prvek vyplnil celý dostupný prostor panelu. Výpočet se udělá tak, že se vezme výška kontejneru pro panely a prostřednictvím vlastnosti `offsetHeight` prvku `divViewItem` od ní odečte velikost hlavičky panelu. Nesmíme také zapomenout na odečtení hodnot `paddingTop` a `paddingBottom` prvku `divFeedList`. Tyto dvě hodnoty se totiž rovněž podílejí na celkové výšce prvku `divFeedList`, takže musejí být ve výpočtu zahrnuty. Získaná velikost v kombinaci s pravidlem `overflow: auto` způsobí, že panel bude obsahovat posuvníky pro případ, kdy se do něj obsah nevejde.

Naprosto stejný postup je použit i pro druhý (prostřední) panel s položkami – `fooReader.divFeedList` je pouze nahrazen za `fooReader.divItemList`, takto:

```
//kód objektu fooReader
fooReader.init = function (evt) {
    evt = evt || window.event;
    if (evt.type == "load") { // Tyto inicializace proběhnou
        // jen při události load
        fooReader.divFeedList = document.getElementById("divFeedList");
        fooReader.divViewItem = document.getElementById("divViewItem");
        fooReader.divItemList = document.getElementById("divItemList");
        fooReader.divMessageTitle = document.getElementById("divMessageTitle");
        fooReader.aMessageLink = document.getElementById("aMessageLink");
        fooReader.divMessageBody = document.getElementById("divMessageBody");
        fooReader.divLoading = document.getElementById("divLoading");
        //zde bude další kód
    }
    var divPaneContainer = document.getElementById("divPaneContainer");
    var divReadingPane = document.getElementById("divReadingPane");
    var divMessageContainer = document.getElementById("divMessageContainer");
    var divMessageHeader = document.getElementById("divMessageHeader");
    var iDocHeight = document.documentElement.clientHeight;
    divPaneContainer.style.height = iDocHeight -
        divPaneContainer.offsetTop - 12 + "px";
    var iFeedsListHeight = divPaneContainer.offsetHeight -
        fooReader.divViewItem.offsetHeight -
        getStyleNumber(fooReader.divFeedList, "paddingTop") -
        getStyleNumber(fooReader.divFeedList, "paddingBottom");
    fooReader.divFeedList.style.height = iFeedsListHeight + "px";
    var iItemListHeight = divPaneContainer.offsetHeight -
        fooReader.divViewItem.offsetHeight -
        getStyleNumber(fooReader.divItemList, "paddingTop") -
        getStyleNumber(fooReader.divItemList, "paddingBottom");
    fooReader.divItemList.style.height = iItemListHeight + "px";
```



```
var iMessageBodyHeight = divReadingPane.offsetHeight -
    divMessageHeader.offsetHeight -
    getStyleNumber(divMessageContainer, "paddingTop") -
    getStyleNumber(divMessageContainer, "paddingBottom");
fooReader.divMessageBody.style.height = iMessageBodyHeight + "px";
};
window.onload = fooReader.init;
window.onresize = fooReader.init;
```

Výpočet pro nastavení výšky pro prvek `divMessageBody` je založen na podobném vzoru, který jsme vám právě ukázali. Rozdíl je pouze v tom, že je použita výška panelu pro čtení (`divReadingPane`) a výška hlavičky zprávy (`divMessageHeader`) místo kontejneru pro panely a jeho hlavičky. Co se týče vertikálního odsazení, tak to se získává z prvku `divMessageContainer` (a nikoliv z prvku `divMessageBody`). Nicméně výsledný efekt je stejný jako v minulých případech – jakmile je nastavena výška těla pro zprávy, bude možné rolovat obsahem v případě potřeby.

Zobrazování a skrývání ukazatele s průběhem načítání

Objekt `fooReader` poskytuje dvě metody pro zobrazení a skrytí ukazatele s průběhem načítání:

```
hideLoadingDiv : function () {
    this.divLoading.style.display = "none"; },
showLoadingDiv : function () {
    this.divLoading.style.display = "block"; },
```

Tyto metody jednoduše změní hodnotu vlastnosti `display` z `none` na `block`, čímž tento ukazatel se stavem načítání buď skryje, nebo zobrazí.

Nastavení obsahu pro třetí panel

Následující metoda `setMessage()` nastavuje obsah třetího panelu. Tato metoda přidává obsah do prvků s identifikátory `divMessageTitle`, `aMessageLink` a `divMessageBody`. Přijímá tři argumenty – nadpis zprávy, odkaz související s článkem a tělo zprávy.

```
setMessage : function (sTitle, sHref, sMessageBody) {
    this.divMessageTitle.innerHTML = sTitle;
    this.aMessageLink.href = sHref;
    this.divMessageBody.innerHTML = sMessageBody;
},
```

Metody pro položky

Následující čtyři metody souvisí s panelem položek (prostřední panel). Konkrétně jsou odpovědné za naplnění tohoto panelu položkami, za změnu hlavičky tohoto panelu, za odstraňování položek z panelu a samozřejmě za programový výběr položek.

Přidávání položek

První metoda je `addItem()` a dynamicky vytváří HTML kód položky, který následně připojuje do panelu položek (tzn. do prostředního panelu). Přijímá dva argumenty: objekt `XParser` položky a číslo asociované s položkou.

```
addItem : function (oItem, iNum) {  
    var aItem = document.createElement("A");  
    aItem.className = "itemlink";  
    aItem.href = oItem.link.value;  
    aItem.setAttribute("frFeedItem", iNum);  
    aItem.id = "item" + iNum;  
    var divHeadline = document.createElement("DIV");  
    divHeadline.className = "itemheadline";  
    divHeadline.innerHTML = oItem.title.value;  
    var divDate = document.createElement("DIV");  
    divDate.className = "itemdate";  
    divDate.appendChild(document.createTextNode("Date: " + oItem.date.value));  
    aItem.appendChild(divHeadline);  
    aItem.appendChild(divDate);  
    //zde bude další kód  
    this.divItemList.appendChild(aItem);  
},
```

Tento kód používá informace obsažené v objektu `XParser`, aby vytvořil HTML kód pro položku. Všimněte si, že pro prvek `aItem` je vytvořen atribut s názvem `frFeedItem`. Tento atribut obsahuje číslo asociované s touto položkou a je později použit pro přidání obsahu do třetího panelu.

Kliknutí na položku zatím nic nedělá. Ve skutečnosti ovšem kliknutí na položku přenese uživatele na URL zadanou ve vlastnosti `href` prvku `aItem`. Tohle ale není žádoucí, takže musíme zajistit ovládání události `click`. Kliknutí na nějakou položku by mělo způsobit dvě věci.

- Za prvé – aktuálně vybraná položka by se měla vrátit do normálního stavu a položka, na kterou bylo teď nově kliknuto, by se měla stát vybranou položkou.
- Za druhé – třetí panel pro obsah by měl být naplněn obsahem této položky.

Ovladač události `onclick` se vykoná v rozsahu prvku `<a>`. Kód proto potřebuje použít API `fooReader` pro přístup k částem UI:

```
addItem : function (oItem, iNum) {
```