KAPITOLA

12

JavaScript a MooTools v systému Joomla!

Témata kapitoly:

- **Co je to JavaScript**
- **Jak pracuje JavaScript**
- **Co je to MooTools**
- **Jak se v systému Joomla! používají JavaScript a MooTools**
- **Zabudované funkce jazyka JavaScript**
- **Použití rozšíření MooTools**
- **Používání technologie AJAX v systému Joomla!**
- **Použití jiných javaskriptových frameworků**
- **Shrnutí**

Jazyk JavaScript umožňuje lepší uživatelskou interakci ve webových aplikacích a stává se stále důležitější součástí systému Joomla!. JavaScript je rovněž klíčovou komponentou technologie AJAX (asynchronous JavaScript and XML). MooTools je javaskriptový framework, který je zahrnut do systému Joomla!. Použití MooTools usnadňuje proces tvorby nových funkčností a údržby těch existujících, které pracují spolehlivě u mnoha různých prohlížečů.

V této kapitole budeme probírat některé základní koncepty, které mají vztah k jazyku JavaScript a MooTools, a vysvětlíme, jak jsou do systému Joomla! integrovány. Pak si probereme specifické funkce jazyka JavaScript, které jsou zabudovány do systému Joomla!, a řekneme si, jak byste tyto funkce mohli použít ve svých programech. Nakonec si ukážeme příklad použití JavaScriptu k tvorbě aktualizace webové stránky Joomla! na bázi technologie AJAX.

Jazyk JavaScript je plně funkční programovací jazyk se svými vlastními jemnostmi a svou složitostí. MooTools je výkonný framework s mnoha klíčovými funkcemi a mnoha dalšími rozšířeními. Studium jazyka JavaScript nebo MooTools je mimo rozsah a možnosti této knihy. Naštěstí je systém Joomla! navržen tak, abychom mohli využít výhody mnoha funkcí jak v jazyce JavaScript, tak i v MooTools bez hlubokých znalostí JavaScriptu.

Co je to JavaScript

JavaScript je programovací jazyk řídící události, které běží v prohlížeči. Podobně jako jazyk PHP je i JavaScript interpretovaný jazyk, což znamená, že jeho skriptovací soubory se provádějí přímo a bez kompilace. Na rozdíl od jazyka PHP, který běží na webovém serveru, JavaScript běží na počítači klienta nebo

uživatelském PC. Každý moderní prohlížeč obsahuje zabudovaný interpret jazyka JavaScript, který umí provést jeho libovolný kód, jenž je součástí webových stránek. Protože běží lokálně na klientském počítači, JavaScript může na události odpovídat okamžitě, aniž by vyžadoval jakoukoliv komunikaci se serverem. Z tohoto důvodu programy napsané v JavaScriptu vypadají spíše jako programy na stolním počítači napsané v jazyku C, Java nebo Basic než jako webové aplikace.

Jak pracuje JavaScript

V systému Joomla! pracuje kód jazyka JavaScript souběžně s kódem PHP. Nenahrazuje kód PHP ani jeho funkčnost. Naopak jeho funkčnost zvyšuje. JavaScript u webových aplikací Joomla! zvyšuje jejich interakčnost ve srovnání s tím, kdyby používaly pouze jazyk PHP nebo HTML.

JavaScript pracuje na klientské straně a modifikuje dokument HTML za pochodu. To může znamenat přidávání, změnu nebo mazání elementů a atributů HTML. JavaScript reaguje na události – na něco, co dělá uživatel nebo webový prohlížeč. Mezi události patří následující činnosti:

- Přesun myši na specifický element HTML nebo z něho pryč
- Kliknutí na nějaký objekt
- Změna hodnoty ve formuláři
- Odeslání formuláře
- Ukončení procesu načítání nového dokumentu HTML

Když se například nad nějakým elementem stránky pohybuje myš, JavaScript může modifikovat dokument HTML v paměti prohlížeče a zobrazit popis tlačítka. Javaskriptová funkce přidá popis tlačítka do dokumentu a zobrazí jej na stránce. Když se myš přesune z elementu pryč, je kód HTML modifikován tak, aby byly elementy popisu tlačítka odstraněny a aby popis zmizel. K tomu dojde uvnitř pracovní paměti prohlížeče okamžitě, aniž by bylo zapotřebí se serverem jakkoliv komunikovat.

JavaScript lze také použít pro požadavky technologie AJAX. To jsou požadavky na webový server podobné těm, které požadují načtení obyčejné stránky. Rozdíl spočívá v tom, že požadavky AJAX se provádějí na pozadí, aniž by přerušovaly chod programu. Uživatel pokračuje v práci s aplikací, zatímco je požadavek odesílán na server nebo odtud přichází odpověď.

Tyto dvě schopnosti jazyka JavaScript – okamžitá interakce s uživatelem a komunikace s webovým serverem na pozadí – umožňují vývojářům zahrnout do aplikací bohaté zkušenosti uživatelů, což by jinak nebylo možné.

Programy psané v JavaScriptu jsou součástí stránky HTML pomocí elementu script. Preferovaná metoda pro přidávání kódu JavaScript do stránky HTML je umístit na dané stránce jeden nebo více elementů script do elementu head. V systému Joomla! se to často dělá pomocí JHTML metod typu behavior. Například kód:

```
JHtml::_('behavior.tooltip');
```

přidává do stránky HTML požadovaný kód JavaScript pro zobrazování popisů tlačítek.

Co je to MooTools

MooTools je javaskriptový framework. Použití frameworků, jako je MooTools, má dvě zásadní výhody. První výhodou je, že framework poskytuje mnoho zabudovaných funkčností. Obsahuje zabudované funkce pro prvky, jako jsou posuvníky, popisy tlačítek, rozbalitelné stromy a jiné běžně používané vlastnosti uživatelského rozhraní. Tyto vlastnosti můžeme v systému Joomla! používat spolu s jednoduchým voláním aplikačního programového rozhraní (API), aniž bychom k tomu museli psát základní javaskriptový kód.

Druhou výhodou je kompatibilita prohlížeče. Ačkoliv moderní prohlížeče mohou JavaScript spouštět, existuje mnoho malých rozdílů v tom, jak JavaScript v různých prohlížečích pracuje. To má za následek, že psát v jazyce JavaScript kód, který by spolehlivě pracoval ve všech prostředích, je obtížné a pracné. S frameworkem, jako je MooTools, mohou jeho vývojáři zajistit, že všechny jeho funkce budou pracovat na všech podporovaných prohlížečích konzistentně. Pokud existují nějaké problémy s kompatibilitou některých specifických prohlížečů, je do frameworku zabudován kód, který tyto problémy řeší. Vývojáři MooTools jsou na jemnosti interpretů JavaScriptu u různých prohlížečů odborníci. To znamená, že uživatelé MooTools (jako jsme my) odborníky být nemusí. Kód, který píšeme pomocí MooTools API, bude pracovat korektně na všech podporovaných prohlížečích.

Stručně řečeno: používání frameworku, jako je MooTools, nám při vývoji aplikace šetří čas, protože můžeme využít výhod jeho zabudované funkčnosti. Rovněž nám šetří čas a problémy s udržováním aplikace, protože se nemusíme starat o testování a o problémy s laděním, které vznikají kvůli rozdílům ve zpracování JavaScriptu u různých prohlížečů.

Existuje řada jiných oblíbených frameworků pro JavaScript, včetně jQuery, Dojo, a Prototype. Každý z nich má své silné a slabé stránky a každý má své příznivce. Počínaje verzí 1.5 bylo v rámci projektu Joomla! přijato rozhodnutí používat MooTools. Zde jsou hlavní výhody tohoto frameworku:

- Je jednoduchý (lightweight) a modulární. Můžeme používat jen ty části, které potřebujeme, a načítání stránek není nijak viditelně zpomaleno.
- Podobně jako u systému Joomla! používá MooTools objektově orientované programování (OOP), takže to z pohledu softwarového inženýra dobře zapadá do systému.
- MooTools je těsně svázán s tím, jak JavaScript pracuje nativně. Rozšiřuje JavaScript, ale nemění způsob, jakým píšete kód JavaScriptu. Studium MooTools je přirozeným pokračováním studia jazyka JavaScript.

Tyto výhody hrály velmi důležitou roli v úvahách roku 2007, když bylo ve vedení projektu Joomla! rozhodnuto používat MooTools. Vedoucí produkční tým systému Joomla! (Joomla! Production Leadership Team) je toho názoru, že uvedené výhody budou platit i v roce 2012 a dále. V současnosti je MooTools zdravý projekt s otevřeným zdrojovým kódem, který je ve vývoji a který se stále vylepšuje. V dohledné budoucnosti se na jiný javaskriptový framework nebude přecházet.

Jak si ještě řekneme dále v této kapitole, je možné, aby návrháři nebo vývojáři rozšíření zahrnuli jiné javaskriptové frameworky do šablon a rozšíření pro systém Joomla!. Změny, kterými prošel MooTools a Joomla! od verze 1.6, tento proces relativně zjednodušují. Avšak pro-

tože je MooTools již součástí každé instalace Joomla! a protože je do platformy Joomla! a do nástroje pro správu obsahu (CMS) integrován, vývojáři v systému Joomla! jsou vedeni k tomu, aby používali MooTools.

Jak se v systému Joomla! používají JavaScript a MooTools

Systém Joomla! je navržen tak, aby vyhovoval standardům jazyka JavaScript, který neobtěžuje. To znamená následující:

- Kód jazyka JavaScript je od značek HTML separován a vybírá si elementy HTML, které se mají měnit, na základě jejich atributů, jako je id nebo třída.
- Většina stránek v systému Joomla! je navržena tak, aby se snižování funkčnosti odehrávalo „kultivovaně". Pokud je v prohlížeči JavaScript neaktivní, uživatel by měl stále mít přístup ke všem funkcím webových stránek, i když možná v méně elegantním uživatelském rozhraní.
- Stránky a funkčnost v systému Joomla! by měly být přístupné lidem se zrakovým či jiným postižením. Například se zvyšováním uživatelských zkušeností s jazykem JavaScript by nemělo docházet ke vzájemnému rušení s čtečkami obsahu obrazovky pro zrakově postižené. Rovněž by mělo být umožněno používat aplikaci přímo z klávesnice (bez myši).

Implementace jazyka JavaScript obvykle znamená volání některé z metod třídy `JHTMLBehavior` – například `JHtml::_('behavior.tooltip')`. Metody třídy `JHTMLBehaviour` načítají požadovaný kód JavaScriptu do záhlaví dokumentu HTML, takže je k dispozici v okamžiku jeho volání. Pro některé funkce, jako je třeba „keep alive", je to vše, co potřebujeme udělat. U jiných, jako je popis tlačítka, potřebujeme zajistit, že máme správnou značku HTML, abychom mohli vyvolat správnou javaskriptovou metodu typu behavior.

Zabudované funkce jazyka JavaScript

Tabulka 12.1 ukazuje standardní metody typu behavior dostupné v systému Joomla! verze 2.5. Javaskriptové soubory se nacházejí ve složce `media/system/js`. Všimněte si, že dvě kopie každého javaskriptového souboru jsou zahrnuty do balíčku Joomla!. Soubory vyjmenované v tabulce 12.1 jsou soubory, které Joomla! skutečně používá. Jsou komprimované, aby jejich načítání trvalo co nejkratší dobu. Například všechny mezery a komentáře jsou odstraněny. Druhá kopie každého souboru s textem „uncompressed" (nekomprimováno), který je přidán k jeho názvu, je v balíčku rovněž zahrnuta. Nekomprimovaná kopie obsahuje komentáře a běžné formátování a používá se pro ladění, čtení a pro práci s kódem.

V následujících kapitolách budeme probírat každou z těchto metod typu behavior.

Calendar

Tato metoda vytváří rozvírací kalendář, který uživateli umožňuje zadat datum pomocí vizuální kalendářové pomůcky. Obrázek 12.1 ukazuje příklad kalendářové pomůcky z obrazovky pro editaci článku.

Pomůcku lze přidat na stránku jedním ze dvou způsobů. Nejjednodušším způsobem je použít pole `JForm`, jehož atribut `type` je nastaven na hodnotu „calendar“. Následující kód pochází například ze souboru `administrator/components/com_content/models/forms/article.xml`:

```
<field name="created" type="calendar"
label="COM_CONTENT_FIELD_CREATED_LABEL"
   description="COM_CONTENT_FIELD_CREATED_DESC" class="inputbox" size="22"
   format="%Y-%m-%d %H:%M:%S" filter="user_utc" />
```

Tabulka 12.1 Zabudované metody typu behavior v jazyce JavaScript

Název metody typu behavior	Popis	Soubor JavaScript
calendar	Poskytuje rozvírací kalendář pro zadávání dat pomocí myši	`calendar.js`
caption	Vytváří obrazová záhlaví během načítání stránky	`caption.js`
colorpicker	Poskytuje rozvírací pomůcku pro výběr barev	`mooRainbow.js`
formvalidation	Poskytuje validaci pro formuláře JForm	`validate.js`
framework	Načítá jádro frameworku MooTools, což je potřeba, když vlastní skripty v jazyce JavaScript používají jádro MooTools; jedním z příkladů může být kontrola typu „check-all“	`mootools.js`
highlighter	Zvýrazní slova, která používají barvu pozadí	`highlighter.js`
keepalive	Udržuje relaci v chodu, například během časově dlouhého editování	`none (uses mootools.js)`
modal	Poskytuje rozvírací modální okno bez nutnosti aktualizace stránky	`modal.js`
multiselect	Poskytuje možnost výběru řady zaškrtávacích polí na obrazovce pomocí kombinace Shift+Levý klik	`multiselect.js`
noframes	Zabrání načtení stránky dovnitř prvku Iframe (například proto, aby se zabránilo převzetí kontroly nad webovými stránkami)	`none (uses mootools.js)`
switcher	Poskytuje na obrazovkách tabulky s oušky (například v Globálním nastavení)	`switcher.js`
tooltip	Poskytuje rozvírací popisy tlačítek na základě pohybu myši	`none (uses mootools.js)`
tree	Poskytuje ovládání rozbalitelného stromu	`mootree.js`
uploader	Poskytuje prostředek pro odesílání více souborů (například pro Media Manager)	`uploader.js`

Obrázek 12.1 Rozvírací kalendář

Zvýrazněný kód říká objektu `JForm`, aby toto pole vykreslil jako rozvírací kalendář. Vzpomeňte si, že atribut `type` rozhoduje o tom, která třída `JFormField` se pro dané pole použije. Zde voláme metodu `getInput()` třídy `JFormFieldCalendar`. Všimněte si, že poslední řádek metody `getInput()` vypadá takto:

```
return JHtml::_('calendar', $this->value, $this->name, $this->id, $format,
➥ $attributes);
```

Tento řádek volá metodu `JHtml::calendar()`, aby vykreslila pole kalendáře.

Pokud bychom potřebovali použít kalendář mimo formulář `JForm`, můžeme volat metodu `JHtml::calendar()` přímo. V tomto případě předáme této metodě argumenty pro hodnotu data, název pole, identifikátor pole, formát data a pole dodatečných atributů HTML, které u tohoto pole chceme použít. Například následující kód ze souboru `administrator/components/com_banners/views/tracks/tmpl/default.php` vytváří počáteční datum pro datový filtr ve Správci bannerů:

```
<?php echo JHtml::_('calendar',$this->state->get('filter.begin'),
➥ 'filter_begin','filter_begin','%Y-%m-%d' , array('size'=>10,'onchange'
➥ =>"this.form.fireEvent( 'submit');this.form.submit()"));?>
```

V argumentu pole atributů HTML stanovujeme v tomto případě velikost 10 a specifikujeme atribut `onchange`. To znamená, že formulář bude odeslán, kdykoliv dojde ke změně tohoto pole. Takto při změně pole `filter_begin` načteme stránku znovu a se správnými položkami.

Ať už budeme vytvářet kalendář kterýmkoliv z obou způsobů, není nutné používat příkaz `JHtml::_('behavior.calendar')`. Toho dosáhneme pomocí metody `JHtml::calendar()`.

Caption

Tato vlastnost přidává automaticky popisek pod obrázek. Abychom ji mohli použít, přidáme do elementu img dva atributy:

- `class="caption"`: Používá se jako selektor pro javaskriptovou funkci.
- `title="<desired caption title>"`: Text, který se má v popisu zobrazit.

Je-li behavior caption aktivní, je na stránku přidán tento element script:

```
<script type="text/javascript"
src="/joomla_development/j16_trunk/media/system/js/caption.js">
```

Script automaticky vybere všechny elementy img se třídou „caption“ a pod obrázkem vytvoří centrovaný popisek se zněním podle hodnoty atributu title pro text popisku.

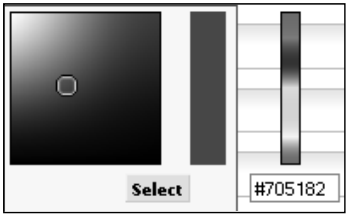
Caption je dobrým příkladem skriptu, který neobtěžuje. K jeho vyvolání se nepožaduje nic než přítomnost těchto atributů. Je-li JavaScript neaktivní, tato vlastnost neuspěje kultivovaným způsobem. Jejím jediným dopadem je, že se popisky nezobrazí.

Colorpicker

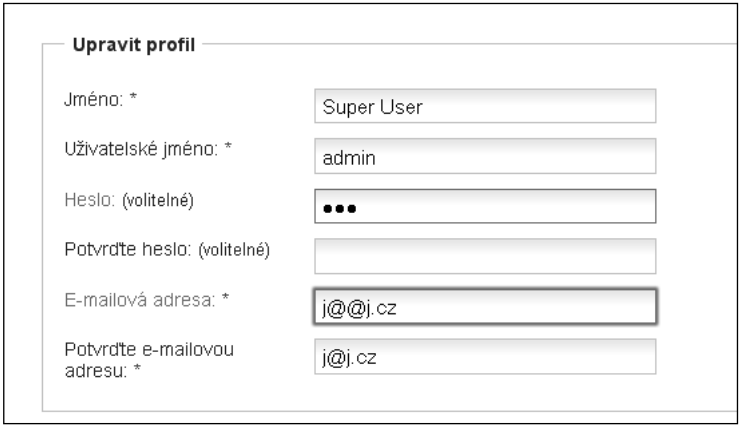
Tato vlastnost, která byla zavedena ve verzi 2.5, vytváří rozvírací ovládací prvek, který umožňuje uživateli vybírat hodnotu barvy. Obrázek 12.2 ukazuje příklad tohoto ovládacího prvku. Barva se vybírá vizuálně posouváním ovládacího prvku po paletě barev. Výsledkem je standardní šesticiferný šestnáctkový kód barvy.

Abyste vytvořili ovládací prvek colorpicker jednoduchým způsobem pomocí třídy `JForm`, proveďte tyto dva kroky:

1. Přidejte kód `JHtml::_('behavior.colorpicker');`, abyste načetli požadovaný javaskriptový soubor.
2. Vytvořte element input s typem „text“ a třídu CSS „input-colorpicker“.



Obrázek 12.2 Příklad rozvíracího ovládacího prvku colorpicker



Obrázek 12.3 Příklad validace formuláře

Validace formuláře

Tato vlastnost umožňuje validovat pole formulářů pomocí jazyka JavaScript. Výhoda tohoto druhu validace spočívá v tom, že můžeme uživatele okamžitě upozornit na neplatné pole, aniž bychom museli čekat, až bude formulář odeslán uživatelem.

Obrázek 12.3 zobrazuje příklad validace formuláře ve frontendu obrazovky Upravit profil.

Zde jsme vložili neplatné heslo a neplatný e-mail. V obou případech jsou pole zvýrazněna okamžitě, jakmile dojde ke změně. Nemusíme klikat na tlačítko Odeslat, abychom tyto chyby uviděli.

Podívejme se nyní, jak to funguje. V souboru XML pro třídu JForm (`components/com_users/models/forms/profile.xml`) uvidíme pro pole Password následující kód:

```
<field name="password1" type="password"
   autocomplete="off"
   class="validate-password"
   description="COM_USERS_DESIRED_PASSWORD"
   field="password2"
   filter="raw"
   label="COM_USERS_PROFILE_PASSWORD1_LABEL"
   message="COM_USERS_PROFILE_PASSWORD1_MESSAGE"
   size="30"
   validate="equals"
/>
```

Nastavením atributu class na `validate-password` vyvoláme validaci hesla v JavaScriptu.

Kód XML pro pole typu „email“ vypadá takto:

```
<field name="email1" type="email"
   description="COM_USERS_PROFILE_EMAIL1_DESC"
   filter="string"
   label="COM_USERS_PROFILE_EMAIL1_LABEL"
   message="COM_USERS_PROFILE_EMAIL1_MESSAGE"
   required="true"
   size="30"
   unique="true"
   validate="email"
/>
```

Zde nastavujeme atribut type na email. Ten říká třídě `JForm`, aby k vykreslení tohoto pole použila třídu `JFormFieldEMail`. V tomto souboru (`libraries/joomla/form/fields/email.php`) nastavíme atribut class tohoto pole na „validate- email“. Takto dojde k aktivaci validace e-mailu v JavaScriptu.

Tuto vlastnost můžeme rovněž použít k validaci podmínky, zda byla do položky zadaná povinná hodnota. Za tímto účelem přidáme atribut required s logickou hodnotou true. To můžeme udělat v XML souboru pro pole `JForm` nebo ji prostě přidat jako atribut do našeho elementu input. Všechny tyto validace se dělají v javaskriptovém souboru `media/system/js/validate.js`. Aby fungovala vlastnost pro validaci formuláře, musíme udělat následující kroky.

- Volat metodu `JHtml::_('behavior.formvalidation')`, která načte JavaSript do záhlaví stránky.
- Zajistit, že element form obsahuje atribut class s hodnotou „form-validate".
- Zajistit, že pole obsahuje buď atribut required s hodnotou true, nebo atribut class s hodnotou uvedenou na seznamu v tabulce 12.2. Toho dosáhneme pomocí souboru XML pole `JForm`. (Jak již bylo řečeno výše, atribut class se přidává tehdy, když typem pole `JForm` je „email".)

Všimněte si, že tato javaskriptová validace je oddělená od validace, kterou provádí třída a která je aktivovaná atributem validate položek `JForm`. Jak jsme již probírali v kapitole 6, kde jsme přidali atribut validate do položky `JForm`, dojde k vyvolání metody `test()` odpovídající třídy `JFormRule`. V našem příkladu s e-mailem jsme nastavili atribut validate na „email", takže vyvoláme metodu `test()` třídy `JFormRuleEMail`. Tato metoda se spustí, když stiskneme tlačítko Odeslat. Rovněž si všimněte, že se tato činnost liší od filtrování. Filtrování ve skutečnosti mění hodnoty během procesu ukládání a odstraňuje znaky, které by mohly být škodlivé.

Vzpomeňte si na jiná místa v této knize, kde říkáme, že se nelze spoléhat na žádný typ validace, který by nás ochránil před hackery. Uvedený typ validace je důležitý k tomu, aby nám poskytl dobré uživatelské rozhraní, ale stále potřebujeme validovat a filtrovat data i po odeslání formuláře, aby nás chránil před škodlivým kódem.

Deaktivování činnosti Submit nebo Save

Jak jsme si již řekli, javaskriptová validace pracuje v prohlížeči lokálně. Aby došlo k validaci polí ve formuláři, není nutné použít tlačítka Odeslat nebo Uložit. Pokud na základě javaskriptové validace víme, že některá data našeho formuláře nejsou platná, můžeme požadovat, aby činnost odesílání byla deaktivovaná, dokud veškerá data formuláře nebudou platná.

Pokud odešleme formulář s neplatnými daty, měli bychom obdržet chyby validace ze serveru a tento formulář by neměl být přijat. Není proto důvod odesílat formulář s neplatnými daty a pro uživatele je pochopitelnější, když uvidí pouze jeden druh validačních chyb (spíše než kdyby viděl jeden druh chybových hlášení před odesláním formuláře a jiný druh hlášení po jeho odeslání).

V klíčových programech systému Joomla! obvykle deaktivujeme funkci odesílání v okamžiku, když byly během validace formuláře nalezeny chyby. Toho dosáhneme dvěma různými způsoby:

- Pokud má formulář element button s atributem typu „submit" a atributem třídy „validate", bude validace formuláře validovat automaticky celý formulář a deaktivovat tlačítka Odeslat, bude-li formulář obsahovat jakákoliv neplatná data.
- Pokud formulář nemá tlačítko submit, můžeme vytvořit javaskriptovou funkci, aby zachytila činnost odesílání, a odeslat formulář jen tehdy, je-li validní. To je případ obrazovek pro správce v backendu, který mají panely nástrojů s více tlačítky pro různé činnosti.

Tabulka 12.2 Zabudované typy validace polí

Atribut class	Popis
`validate-username`	Nepovoluje následující znaky: menší nebo větší než (< nebo >), jednoduché nebo dvojité uvozovky (" nebo '), procentuální znaménko (%), středník (;), závorky („(" nebo „)") nebo ampersand (&)
`validate-password`	Musí mít nejméně čtyři znaky (a ne více než 196 znaků) a nesmí začínat prázdným znakem
`validate-numeric`	Může obsahovat pouze číslice (0– 9), znaménko minus (–), čárku (,) nebo tečku (.)
`validate-email`	Formát řetězce musí být ve tvaru xxx@yyy.zzz, kde xxx, yyy, a zzz mohou obsahovat pouze písmena, čísla, tečky a pomlčky. xxx a yyy musí mít minimálně jeden znak. zzz musí mít dva až čtyři znaky.

Příklad druhé metody nalezneme v backendu na editační obrazovce Správce uživatelů (`administrator/components/com_users/views/user/tmpl/edit.php`). Tam uvidíme následující kód:

```
<script type="text/javascript">
Joomla.submitbutton = function(task)
{
   if (task == 'user.cancel' || document.formvalidator.isValid(document.id
➥('user-form'))) {
      Joomla.submitform(task, document.getElementById('user-form'));
   }
}
</script>
```

Tento kód vytváří funkci v jazyce JavaScript s názvem `Joomla.submitbutton`. Když uživatel klikne na kteroukoliv ikonu z panelu nástrojů, která odesílá formulář, jako třeba Uložit, Zavřít nebo Uložit & Zavřít, vyvolá tuto funkci. Pokud se úkol rovná „user.cancel" nebo pokud je formulář validní, bude formulář odeslán vyvoláním funkce `Joomla.submitform` (která je definovaná v souboru `media/system/js/core.js`). Jinak se nestane nic a soubor odeslán není. Takto musí uživatel opravit veškerá neplatná data ještě před odesláním formuláře.

Všimněte si, že tohle je jiný příklad nenápadnosti jazyka JavaScript. Aplikace nezávisí na javaskriptové validaci, která má zajistit platnost polí. Existují jiné metody, které kontrolují platnost dat – například třídy `JFormRule` a validační metody třídy `JTable`. Javaskriptové metody pouze vylepšují uživatelské rozhraní tím, že poskytují uživateli okamžitou odezvu, když vloží neplatná data. Formulář bude funkční i v případě, že JavaScript bude neaktivní, ale uživatelské prostředí nebude tak elegantní.

Framework

Tato vlastnost prostě načte framework MooTools jako tag typu script umístěný v záhlaví stránky. To uvidíte, když přidáte vlastní kód JavaScriptu, jehož činnost je závislá na MooTools. Není propojen na žádnou specifickou javaskriptovou funkci. Kód vypadá takto:

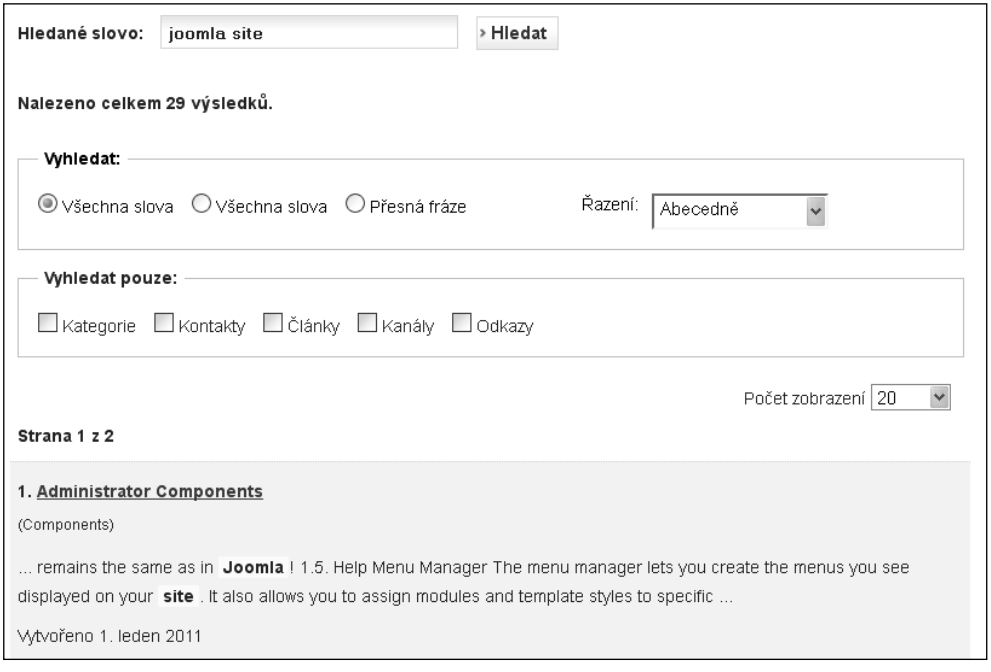
```
JHtml::_('behavior.framework');
```

Highlighter

Vlastnost highlighter vám umožní na stránce zvýraznit jednoduchým způsobem vybraná slova. To se využívá u komponenty Smart Search ke zvýrazňování hledaných výrazů v článcích, jak ukazuje obrázek 12.4.

Highlighter lze používat, když provedete následující kroky:

1. Přidejte do kódu příkaz `JHtml::_('behavior.highlighter', $wordArray)`, kde `$wordArray` je pole slov, která chcete zvýraznit.
2. V dokumentu přidejte HTML kód `<br id="highlighter-start" />` bezprostředně před sekcí, ve které chcete něco zvýraznit.
3. Přidejte HTML kód `<br id="highlighter-end" />` hned za sekci, ve které chcete něco zvýraznit.



Obrázek 12.4 Příklad vlastnosti highlighter

Keepalive

Tato vlastnost umožňuje systému Joomla! udržovat relaci aktivní po neomezenou dobu. Když uživatel pracuje se systémem Joomla!, za normálních okolností chceme, aby se relace ukončila po určitém daném čase. Výchozí nastavení je 15 minut. Je to z důvodu bezpečnosti. Když je například uživatel přihlášen v nějaké citlivé části a na svém počítači se neodhlásí, chceme, aby se relace sama odhlásila a aby se snížila pravděpodobnost, že by se na webové stránky dostala neautorizovaná osoba.

Avšak v některých případech může časový limit způsobit problémy, například kdyby uživatel editoval dlouhý článek. Je jednoduché si představit případ, kdy někdo pracuje 15 minut či déle, aniž by svoji práci uložil. Pokud by se relace ukončila během tohoto času, uživatel by mohl veškerou svoji práci ztratit. Ještě horší by bylo, kdyby uživatel přišel na tento problém až poté, co stiskl tlačítko Uložit (když už by bylo příliš pozdě!).

Abychom tomu zabránili, můžeme vyvolat vlastnost keepalive. K tomu potřebujeme jen jediný řádek kódu:

```
JHtml::_('behavior.keepalive');
```

Tento kód používá zabudovanou metodu MooTools, která automaticky vyvolá požadavek AJAX na server minutu předtím, než by se jinak relace ukončila. Požadavek se provede na pozadí a na serveru způsobí obnovení časového limitu relace. Výsledkem je, že se relace bude udržovat donekonečna. Toho využívá mnoho komponent pro editační obrazovky v systému Joomla!.

Modal

Vlastnost modal se používá, když potřebujeme z hlavního okna zobrazit nezávislé rozevírací okno. Zde máme dva příklady z jádra systému Joomla!:

- Kliknutím na ikonu Možnosti v panelu nástrojů na obrazovce Správce uživatelů nebo na jiných obrazovkách pro editaci možností konfigurace na úrovni komponenty.
- Kliknutím na tlačítko Vybrat/Změnit pro výběr článku pro položku nabídky Jednoho článku.

V těchto případech potřebujeme poslat na server požadavek, abychom obdrželi data, a poté chceme tato data zobrazit. Avšak nechceme ztratit aktuální text, se kterým pracujeme. Když modální okno zavřeme, chceme dále pokračovat v naší práci na obrazovce, ze které jsme jej otevřeli, a chceme pokračovat přesně tam, kde jsme práci přerušili.

Vlastnost modal nám umožňuje otevírat okna z našeho dokumentu, pracovat s tímto oknem a zavřít jej, když jsme práci ukončili. Po uzavření modálního okna se vracíme zpět do nadřazeného okna. Dokud je modální okno otevřené, v nadřazeném okně pracovat nemůžeme.

V mnoha případech zobrazujeme modální okno v elementu iframe. To nám umožňuje mít v nadřazeném dokumentu další dokument.

K vytvoření modálního okna v systému Joomla! potřebujeme provést tyto kroky:

- Doplňte příkaz `JHthml::_('behavior.modal')` ke vložení JavaScriptu do záhlaví stránky. Jako volitelný druhý argument můžeme předat selektor třídy CSS. Jeho výchozí hodnota je „a.modal“, která vybere element anchor s atributem „modal“.
- Vytvořte element anchor s atributem class „modal“ (nebo takovým atributem, který odpovídá druhému argumentu příkazu behavior.command) a atributem href, který ukazuje na adresu URL, odkud se bude modální okno zavádět. Pokud bude modální okno vykreslené v elementu iframe, použijte atribut rel s obslužnou rutinou a s informacemi o rozměrech.
- Vytvořte pohled a rozvržení pro obsah modálního okna. Ty jsou stejné jako kód pro normální pohled a rozvržení.