

Tvorba internetových aplikací s využitím framework jQuery

Bakalářská práce

Michal Oktábec

Vedoucí bakalářské práce: PaedDr. Petr Pexa

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2010

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejich internetových stránkách.

V Českých Budějovicích dne

Anotace

Tato bakalářská práce se zabývá využití frameworku jQuery pro vytváření webových stránek. Nejdříve bude vysvětleno, co to framework jQuery je a čím se liší od klasického JavaScriptu. V dalších částech práce se zaměřím na syntaxi a způsoby používání jQuery. Cílem je vytvořit první uživatelskou příručku v českém jazyce.

Abstract

This work deals using framework jQuery for creating web pages. At first it will be explained what is framework jQuery and how is it different from classic JavaScript. In next parts of work I will specialize in syntax and ways of using jQuery. The goal of this work is create first Czech user guide.

Poděkování

Rád bych poděkoval vedoucímu práce PaedDr. Petru Pexovi, za umožnění zpracování tohoto tématu bakalářské práce a cenné rady při jejím zpracovávání. Také bych rád poděkoval své přítelkyni, která mi hodně pomáhala a podporovala mě, své rodině za umožnění studovat tuto univerzitu a zázemí a v neposlední řadě také Mgr. Michaelu Cífkovi, díky kterému jsem pochytil spoustu cenných zkušeností z praxe.

Obsah

| | | |
|----------|---|-----------|
| 1 | ÚVOD | 7 |
| 2 | CÍLE PRÁCE | 8 |
| 3 | SOUČASNÝ STAV PROBLEMATIKY | 9 |
| 4 | METODIKA ZPRACOVÁVÁNÍ PRÁCE | 10 |
| 5 | TEORETICKÝ ÚVOD | 11 |
| 5.1 | POUŽÍVÁME JAVASCRIPT NA WEBOVÝCH STRÁNKÁCH | 11 |
| 5.2 | CO TO JE FRAMEWORK JQUERY A PROČ JE NUTNÝ? | 12 |
| 6 | UŽIVATELSKÁ PŘÍRUČKA | 13 |
| 6.1 | MOTIVAČNÍ PŘÍKLAD | 13 |
| 6.2 | SYNTAXE JQUERY | 14 |
| 6.3 | JAVASCRIPT VS. JQUERY..... | 15 |
| 6.3.1 | <i>Srovnání: příklad 1 – Výběr elementu</i> | <i>15</i> |
| 6.3.2 | <i>Srovnání: příklad 2 – Bublínková nápověda</i> | <i>16</i> |
| 6.3.3 | <i>Srovnání: příklad 3 – AJAX.....</i> | <i>17</i> |
| 6.4 | ZAČÍNÁME POUŽÍVAT JQUERY | 18 |
| 6.5 | ZÁKLADNÍ VÝBĚR ELEMENTŮ STRÁNKY..... | 19 |
| 6.5.1 | <i>Příklad 1 – Ukázka použití výběru – skrytí elementu.....</i> | <i>20</i> |
| 6.6 | JQUERY A ATRIBUTY ELEMENTŮ..... | 20 |
| 6.6.1 | <i>Příklad 1 – Vypsání obsahu textového pole (input).....</i> | <i>21</i> |
| 6.6.2 | <i>Příklad 2 – Zakázání úpravy textového pole (input).....</i> | <i>22</i> |
| 6.7 | UDÁLOSTI V JQUERY | 23 |
| 6.7.1 | <i>Výběr událostí.....</i> | <i>24</i> |
| 6.7.2 | <i>Možnosti napojení události.....</i> | <i>25</i> |
| 6.7.3 | <i>Zrušení napojení na událost</i> | <i>25</i> |
| 6.7.4 | <i>Příklad 1 – Opuštění textového pole.....</i> | <i>26</i> |
| 6.7.5 | <i>Příklad 2 - Změny barev tlačítka</i> | <i>26</i> |
| 6.7.6 | <i>Příklad 3 – Pozice kliknutí v bloku DIV</i> | <i>27</i> |
| 6.8 | JQUERY A CSS..... | 28 |
| 6.8.1 | <i>Příklad 1 – Hover efekt pomocí jQuery</i> | <i>29</i> |
| 6.8.2 | <i>Příklad 2 – Změna velikosti textu</i> | <i>30</i> |
| 6.9 | JQUERY A RELATIVNÍ VÝBĚRY ELEMENTŮ | 31 |
| 6.9.1 | <i>Dvojtečka a klíčové slovo ve výběru elementu.....</i> | <i>32</i> |
| 6.9.2 | <i>Speciální funkce pro výběr elementu</i> | <i>33</i> |
| 6.9.3 | <i>Příklad 1 – Rodiče a děti</i> | <i>36</i> |
| 6.9.4 | <i>Příklad 2 – Přebarvování čtverců.....</i> | <i>38</i> |
| 6.10 | ROZMĚRY V JQUERY..... | 40 |

| | | |
|----------|---|-----------|
| 6.10.1 | Příklad 1 – Vypsání aktuálních rozměrů body..... | 41 |
| 6.10.2 | Příklad 2 – Zobrazení textu dprostřed okna | 42 |
| 6.10.3 | Příklad 3 – Pokrytí pozadí plátnem | 44 |
| 6.11 | EFEKTY V JQUERY | 46 |
| 6.11.1 | Speciální funkce animate() | 47 |
| 6.11.2 | Příklad 1 – Zobrazení elementu..... | 48 |
| 6.11.3 | Příklad 2 – Modernější hover efekt..... | 50 |
| 6.11.4 | Příklad 3 – Vysouvací menu | 53 |
| 6.12 | PRÁCE S FORMULÁŘI V JQUERY | 60 |
| 6.13 | ÚPRAVY KÓDU STRÁNKY POMOCÍ JQUERY (MANIPULATION) | 61 |
| 6.13.1 | Různé možnosti vkládání HTML kódu | 61 |
| 6.13.2 | Formátovaný a neformátovaný HTML kód..... | 62 |
| 6.13.3 | Příklad 1 – Vypsání obsahu elementu..... | 63 |
| 6.13.4 | Příklad 2 – Zmizení tlačítka, zobrazení textu..... | 63 |
| 6.13.5 | Příklad 3 – Příprava formuláře pro nahrání souboru na internet..... | 64 |
| 6.14 | JQUERY A POZICOVÁNÍ ELEMENTŮ (OFFSET) | 67 |
| 6.14.1 | Příklad 1 – Vypsání umístění elementu..... | 68 |
| 6.15 | UŽITEČNÉ FUNKCE JQUERY | 70 |
| 6.15.1 | Práce s řetězcem..... | 70 |
| 6.15.2 | Příklad 1 – Použití funkce trim() | 71 |
| 6.15.3 | Práce s poli a kolekcemi..... | 71 |
| 6.15.4 | Příklad 2 – Práce s poli | 72 |
| 6.15.5 | Zjištění použitého prohlížeče | 75 |
| 6.15.6 | Příklad 3 – Zjištění použitého prohlížeče | 76 |
| 6.16 | AJAX A JQUERY | 77 |
| 6.16.1 | Rozdíly mezi funkcemi load() a ajax() | 78 |
| 6.16.2 | Příklad 1 – Nahrání obsahu select boxu..... | 79 |
| 6.16.3 | Příklad 2 – Hlasování v anketě..... | 82 |
| 6.17 | JQUERY UI..... | 85 |
| 6.17.1 | Kalendář..... | 85 |
| 6.17.2 | Taby (záložky)..... | 88 |
| 6.18 | POZNÁMKA NA OKRAJ | 90 |
| 7 | ZÁVĚREČNÉ SHRNU TÍ..... | 92 |
| 7.1 | POUŽITELNOST PŘÍRUČKY | 92 |
| 7.2 | KOMPATIBILITA S PROHLÍŽEČI | 92 |

1 Úvod

Moderní a uživatelsky příjemné stránky nelze vytvářet bez jisté části JavaScriptu. Framework jQuery je v tomto ohledu velmi účinný pomocník pro tvorbu webových stránek.

Osobně jsem se s frameworkem poprvé setkal, když jsem nastupoval na brigádu do jedné firmy – původně sice jako PHP programátor, ale nakonec pracuji i na ostatních částech webových stránek.

Od chvíle, kdy jsem byl s jQuery seznámen, jsem si v něm začal prohlubovat znalosti a používat ho i v osobních projektech.

Velmi rád bych všechny čtenáře seznámil s tímto kvalitním frameworkem, díky kterému lze vytvářet velmi pěkné efekty, ale i příjemné uživatelské prostředí a to relativně jednoduše.

2 Cíle práce

Cílem práce je seznámit čtenáře s možnostmi využití frameworku jQuery pro tvorbu interaktivních webových aplikací a především vytvořit uživatelskou příručku právě pro práci s frameworkem jQuery.

V práci bude čtenář seznámen nejen se základy práce s frameworkem jQuery, ale i s pokročilými technikami - například s technologií AJAX. Všechny teoretické části budou doplněny množstvím příkladů, na kterých bude vysvětlena praktická část a které budou otestovány v nejpoužívanějších prohlížečích – Opera, Mozilla Firefox, Internet Explorer, Google Chrome a Safari.

3 Současný stav problematiky

Framework jQuery není novinkou. Začal vznikat v srpnu v roce 2005 a o rok později byla vydána první stabilní verze (verze 1.0). Ve světě je framework jQuery hodně známý, o čemž svědčí i výčet některých webových stránek, které jej používají. Mezi nimi jsou stránky vyhledávače Google, společnost DELL, televize CBS a mnoho dalších.

V angličtině lze nalézt nejvíce zdrojů a i několik knih. Osobně nejvíce využívám stránky jquery.com^[1] a jqueryui.com^[2], ale pokud vyhledávám řešení přímo pro konkrétní situaci, nejlépe poslouží zdroje, které nalezne vyhledávač Google.

V České republice se využívání jQuery rozrůstá, ale prozatím nebyla vydána žádná ucelená publikace. Zmínku o jQuery lze pouze nalézt na diskusních fórech nebo v podobě článků, na několik webových stránkách. Příkladem je server programujte.com, kde vyšlo několik článků.

A právě vznik první příručky o jQuery v českém jazyce je hlavním účelem této práce.

4 Metodika zpracování práce

S frameworkem jQuery už jistě zkušenosti mám a bylo nutné poskládat informace do logických celků s návazností a předat je čtenáři formou příručky.

Při tvorbě příručky je vycházeno především z dokumentace frameworku jQuery^[1], kde je k dispozici kromě dokumentace i návod pro začátečníky a která sloužila jako vzor pro vytvoření kapitol, seskupení témat a hlavičky funkcí. Dalším zdrojem jsou také vlastní zkušenosti, podle kterých je tvořena většina praktických příkladů, o které jsou kapitoly doplněny kvůli lepšímu porozumění a vysvětlení aktuálního tématu.

Využívána je i stránka jqueryui.com^[2], což je knihovna základních aplikací, které jsou postaveny na jQuery. Namátkou můžeme zmínit například kalendář.

Na porovnání klasického AJAXu a jeho alternativy v jQuery je použito knihy Ajax a PHP: tvoříme interaktivní webové aplikace profesionálně^[3]. Jako zástupce klasického JavaScriptu je použito knihy JavaScript: Programujeme internetové aplikace^[4].

5 Teoretický úvod

Když vytváříte webové stránky, většinou si přejete, aby byli úspěšní, navštěvovala je spousta uživatelů a v ideálním případě jste z nich měli velký příjem.

Pokud toho chcete dosáhnout, nestačí pouze kvalitní obsah, protože i sebekvalitnější stránky mohou odradit uživatele svým chováním a designem.

Webové stránky můžete vytvořit klasickým spojením HTML/XHTML a CSS, případně také pomocí PHP či ASP. Ale PHP i ASP jsou programovací jazyky zpracovávané na straně serveru, takže aby se uživateli projevila změna, či jinak upravila stránka – například změnila barva textu, je nutné webovou stránku "odeslat".

Webový server ji pak přijme, zpracuje a zpět odešle vygenerovanou stránku. Pokud byste chtěli takhle vytvořit například vysouvací menu, bylo by velmi nepraktické.

5.1 Používáme JavaScript na webových stránkách

A teď příklad z druhé strany. Odeslání požadavku nebude potřeba a vše se provede v reálném čase. Touto dovedností disponuje skriptovací jazyk JavaScript, který se zpracovává na straně prohlížeče.

JavaScriptem tedy lze nejen udělat pěkné vysouvací menu, ale také stránky obohatit o hodiny, anketu (resp. možnost hlasování v anketě bez nutnosti znovu načítat stránku), základní validaci formulářů (zjištění jestli jsou zadány povinné položky), jezdící text a další procesy, probíhající v reálném čase - tedy bez toho aniž by se musela stránka načíst znovu.

Díky JavaScriptu lze také využít možností AJAXu¹, který zařídí zavolání libovolného, například PHP, skriptu asynchronně a tedy opět uživatel nebude muset

¹ AJAX = Asynchronous JavaScript and XML
Více na stránce <http://www.adaptic.cz/znalosti/slovnicek/ajax.htm>

čekat na nové načtení stránky, ale pouze na zpracování. To lze využít například pro zmiňované hlasování v anketě, nebo třeba telefonní seznam.

Díky JavaScriptu tak můžeme vytvořit vizuální efekty a prvky, které zjednodušují či zrychlují práci se stránkami.

5.2 Co to je framework jQuery a proč je nutný?

Každý, kdo již s JavaScriptem pracoval, si je vědom toho, že má jeden velký nedostatek - je velmi obtížné přimět ho ke stejnému chování ve všech prohlížečích. Abychom jsme se dostali ke kompatibilitě napříč webovými prohlížeči, je nutné velmi dlouho ladit, pročítat diskusní fóra a ani tehdy to někdy není možné. JQuery tento problém řeší, ale není první. Před jQuery jste se mohli setkat s knihovnou Prototype.

Framework² jQuery je optimalizován a zjednodušen takovým způsobem, že velmi pěkných efektů lze dosáhnout i použitím jediného řádku kódu a navíc chování je stejné ve všech prohlížečích. Je naprogramován v klasickém JavaScriptu a jedná se o knihovnu spousty funkcí, které velmi ulehčují programování v JavaScriptu.

V této práci budou vysvětleny způsoby používání frameworku jQuery, ale také ukázány jeho široké možnosti.

² Co to je framework viz: <http://cs.wikipedia.org/wiki/Framework>

6 Uživatelská příručka

Tato část práce bude věnována tvorbě uživatelské příručky. Aby bylo možné teoretické informace vysvětlit na příkladech, bude se používat jednoduchá stránka, která byla vytvořena pro tyto účely.

Příručka se nezabývá podrobnými detaily programování, a proto je nutné, abyste měli základy programování za sebou. Zároveň by bylo vhodné mít základní zkušenosti s tvorbou webových stránek – hlavně se jedná o HTML/XHTML, CSS a JavaScript.

Příručka nemá být českým překladem dokumentace. Pouze ukazuje možnosti a popisuje příklady tvorby webových stránek s využití frameworku jQuery. Příručka zdaleka neobsahuje a nepopisuje všechny možnosti frameworku jQuery.

Téměř všechny příklady, které jsou označeny jako **Příklad X**, kde **X** je nějaké číslo, naleznete na přiloženém CD. Neplatí to pouze pro příklady číslo 1 a 3 v kapitole o srovnání klasického JavaScriptu a frameworku jQuery.

6.1 Motivační příklad

Jakmile si otevřete oficiální stránky jQuery frameworku, přibližně uprostřed stránky naleznete jednoduchou ukázkou toho, jak je jQuery framework mocný.

Je to určitě pěkný motivační příklad, protože jQuery tím ukazuje svou největší zbraň – jednoduchost a proto bude použit pro stejný účel i zde. Bude pouze mírně upraven, aby byla zajištěna kompletní funkčnost. Jedná se o vzhledově pěkné a postupné vysunutí obdélníku s textem, po kliknutí na odkaz.

I když by bylo složité tuto funkci naprogramovat v klasickém JavaScriptu, v jQuery ji vytvoříte pomocí několika řádků kódu:

```
$("#a.mpriklad").bind("click", function() {  
    $("#a.mpriklad").hide();  
    $("#p.mpriklad").show("slow");  
    return false;  
});
```

Pro příklad je nutné mít připravený jeden odkaz se třídou *mpriklad* a také skrytý odstavec se třídou *mpriklad*. Po kliknutí na odkaz dojde ke skrytí odkazu a postupnému "vysunutí" odstavce. *Return false* pouze způsobí, že nedojde k přesměrování na odkaz.

6.2 Syntaxe jQuery

Framework jQuery je naprogramovaný v klasickém JavaScriptu a zdědil po něm jednoduchou syntaxi. Hlavní je použití znaku dolaru (\$) při odkazování na funkce jQuery nebo nějaký element. Případně lze vyměnit znak dolaru za řetězec *jQuery* v případě, že se jQuery používá ve spojení s Prototypem³ – aby nedocházelo ke kolizím.

S jQuery pracujete, pouze pokud využíváte jeho funkcí – především pro tvorbu efektů. Co se týče ostatních operací – například vytváření proměnných, nepracujete již s jQuery, ale s klasickým JavaScriptem. Příklad již zmíněného vytváření proměnné:

```
var promenna = "text";
```

Stejně se přistupuje i k funkcím JavaScriptu:

```
promenna = promenna.toUpperCase();
```

Po vzoru JavaScriptu je nutné každou funkci/řádek ukončit středníkem (;). Systém komentářů je také stejný. Jednořádkový komentář pomocí dvou lomítek a víceřádkový komentář pomocí lomítka a hvězdičky.

³ Prototype je předchůdce jQuery. Také se jedná o framework pro JavaScript. Oficiální stránky (v angličtině): <http://www.prototypejs.org/>

```
// jednořádkový komentář
/* tento komentář již
Může pokračovat na více
řádcích */
```

6.3 JavaScript vs. jQuery

Jak už bylo zmíněno, jQuery je naprogramovaný v JavaScriptu, tudíž lze srovnat pouze to, proč byl jQuery vytvořen. A to je jednoduchá tvorba efektů a kompatibilita napříč prohlížeči.

Co lze v jQuery udělat jednoduše, pomocí několika řádků, je v JavaScriptu potřeba mnohem delšího kódu. A na co je třeba v jQuery využít delšího kódu, k tomu bychom se v klasickém JavaScriptu jen těžko dostávali.

V následujících několika příkladech bude prezentován hlavní "rozdíl" mezi jQuery a JavaScriptem. Příklady by Vám měly jasně ukázat důvody, proč je mnohem efektivnější používat právě jQuery.

6.3.1 Srovnání: příklad 1 – Výběr elementu

Prvním příkladem na porovnání výhod jQuery oproti klasickému JavaScriptu může být hned jedna ze základních dovedností – výběr elementů. V klasickém JavaScriptu existují funkce `getElementById` a `getElementsByTagName`, čímž můžeme vybrat elementy podle identifikátoru nebo jména tagu.

Nejpraktičtější je však vybírat elementy podle přiřazené třídy, protože jednu třídu může mít více elementů, ale identifikátor (`id`) je povolen pouze u jednoho elementu. V JavaScriptu oficiálně neexistuje funkce pro výběr elementu podle třídy. I když v Opeře či Firefoxu bude fungovat funkce `getElementsByClassName`, v Internet Exploreru ne a je potřeba si naprogramovat funkci vlastní.

Oproti tomu jQuery tuto schopnost má. Jednoduše se tak můžeme odkázat například na všechny elementy, které mají třídu `menu` a pomocí funkce `show()` je zobrazit.

```
$( ".menu" ).show();
```

6.3.2 Srovnání: příklad 2 – Bublínková nápověda

Následující příklad vychází z bublinkové nápovědy, řešené pomocí klasického JavaScriptu. Řešení je použito ze stránek DynamicDrive.com⁴, kde lze nalézt spoustu volně použitelných JavaScriptových skriptů.

Stručně řečeno se jedná o možnost zobrazit nápovědu k právě vyplňovanému poli (například registrace) při přejetí myši přes otazník.

Srovnání bylo provedeno tak, že úplně stejná funkčnost byla napsána v jQuery a umístěna na stránku – vlevo je řešení pomocí klasického JavaScriptu, vpravo pomocí jQuery. V jQuery stačil pro tuto funkčnost tento krátký kód:

```
$(document).ready(function() {  
    $(".napoveda a").bind("mouseover", function() {  
        var offset = $(this).offset();  
  
        var text_napovedy = $(this).parent().children("p");  
        $(text_napovedy).css({ top: offset.top-10, left:  
offset.left + 25});  
        $(text_napovedy).show();  
    });  
    $(".napoveda a").bind("mouseout", function() {  
        $(this).parent().children("p").hide();  
    });  
});
```

Zatímco v JavaScriptu bylo nutné použít přibližně 75 řádků nepřehledného klasického JavaScriptu. Zde je příklad několika řádků:

```
var horizontal_offset="9px" //horizontal offset of hint box  
from anchor link
```

⁴ <http://www.dynamicdrive.com/dynamicindex16/showhint.htm>


```

var vertical_offset="0" //horizontal offset of hint box from
anchor link. No need to change.
var ie=document.all
var ns6=document.getElementById&&!document.all

function getposOffset(what, offsettype){
    var totaloffset=(offsettype=="left"? what.offsetLeft :
what.offsetTop;
    var parentEl=what.offsetParent;
    while (parentEl!=null){
        totaloffset=(offsettype=="left"?
totaloffset+parentEl.offsetLeft :
totaloffset+parentEl.offsetTop;
        parentEl=parentEl.offsetParent;
    }
    return totaloffset;
}

function iecompattest(){
    return (document.compatMode &&
document.compatMode!="BackCompat"? document.documentElement :
document.body
}
...

```

6.3.3 Srovnání: příklad 3 – AJAX

Jednou z největších výhod frameworku jQuery oproti klasickému JavaScriptu je právě práce s AJAXem. Pokud chcete dobře pracovat s AJAXem v klasickém JavaScriptu, musíte používat různé objekty, ošetřovat spoustu výjimek a ani tehdy není zaručeno, že vše bude fungovat.

Následující část zdrojového kódu pochází ze zdrojových kódů z knihy *AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně*^[3]. Jedná se o část příkladu hned z první kapitoly. Celý zdrojový kód si můžete stáhnout z adresy: <http://www.zonerpress.cz/download/ajax.zip>.

...

```
// make asynchronous HTTP request using the XMLHttpRequest
object
function process()
{
    // proceed only if the xmlhttp object isn't busy
    if (xmlhttp.readyState == 4 || xmlhttp.readyState == 0)
    {
        // retrieve the name typed by the user on the form
        name =
encodeURIComponent(document.getElementById("myName").value);
        // execute the quickstart.php page from the server
        xmlhttp.open("GET", "quickstart.php?name=" + name, true);
        // define the method to handle server responses
        xmlhttp.onreadystatechange = handleServerResponse;
        // make the server request
        xmlhttp.send(null);
    }
    else
        // if the connection is busy, try again after one second
        setTimeout('process()', 1000);
}
...
```

Tento příklad mi nefungoval ani v jednom z testovaných prohlížečů.

Pokud však použijeme jQuery a jeho funkce pro práci s AJAXem, máme zaručenou jak kompatibilitu s nejpoužívanějšími prohlížeči, tak velkou úsporu kódu. Myslím, že stačí pohled na jednu z funkcí, která stačí ke spolehlivému fungování AJAXu, a uznáte, že se jedná o velké zjednodušení. Funkce se jmenuje *load()*.

```
load( adresa, [parametry], [funkce_dokonceno]
```

Práce s AJAXem pomocí frameworku jQuery bude popsána ke konci příručky a je jí věnována samostatná kapitola 6.16.

6.4 Začínáme používat jQuery

Abyste mohli začít jQuery používat, je nutné provést dva kroky.

1. Vložit do stránky jádro jQuery, neboli soubor `jquery-cisloverze.js`, který si můžeme stáhnout například ze stránek jQuery.com^[1] – pro účely příručky bude sloužit jQuery ve verzi 1.4.2.
2. Do JavaScriptového kódu vložit funkci jQuery jménem `ready()`.

```
$(document).ready(function() {  
    // váš kód  
});
```

Veškerý kód, který vytvoříte, vkládáte do funkce `ready()`, která zajistí jeho vykonání po kompletním načtení stránky.

6.5 Základní výběr elementů stránky

Pokud chceme něco na stránce vůbec vytvářet, musíte samozřejmě manipulovat s něčím, co už na stránce existuje – buď přesunujete/skrýváte/zobrazujete nějaký `DIV` či jiný element stránky nebo pokud chceme vložit něco nového, musíte to opět vložit za nebo před nějaký existující element.

Pro výběr elementu/prvku stránky využijete jQuery jednoduše. Pokud chceme vybrat odkazy, které mají třídu `modry`, stačí použít následující kód:

```
$(".modry")
```

Pro výběr odkazů s identifikátorem (`ID`) `modry`, zaměňte pouze "tečku" za "křížek".

```
$("#modry")
```

A pokud byste chtěli vybrat všechny odkazy, stačí odebrat z výběru třídu/identifikátor.

```
$(".a")
```

Pokud pracujete s CSS⁵, setkali jste se již s tímto způsobem odkazování a určitě bude pro Vás přínosem.

Tímto způsobem se můžeme odkazovat na celé tělo (*body*) stránky, nebo naopak pouze na obrázky, které mají třídu *vlevo* a zároveň *maly* – třídy lze totiž připojovat.

```
$ ("img.vlevo.maly")
```

6.5.1 Příklad 1 – Ukázka použití výběru – skrytí elementu

Pouhý výběr elementu nám samozřejmě ničemu neposlouží. Pokud chceme s elementem něco provést, můžeme využít spoustu funkcí, které jQuery nabízí – příkladem může být skrytí prvku pomocí funkce *hide()*, která je ekvivalentem CSS vlastnosti *display:none*.

```
$ ("a.modry").hide();
```

Odkaz nebude vidět, protože se ihned po načtení stránky aplikuje funkce *hide()*. Pokud bychom chtěli, aby se funkce aplikovala až po kliknutí na nějaké tlačítko, museli bychom využít událostí, ale o těch až dále.

6.6 JQuery a atributy elementů

JQuery umožňuje práci s atributy elementů – např. *style*, *name*, *value*, *disabled*, *selected* a další. Můžeme přidávat, ubírat i upravovat atributy elementů.

Za dobu, co s jQuery pracuji, jsem měl pouze problém s úpravou atributu *name* v prohlížeči Internet Explorer. Ignoroval totiž úpravu jména, ale úprava jména rozhodně nepatří mezi časté nutnosti. Při klasickém používání – přidání a práce např. s atributem *disabled*, jsem se neseťkal s problémy.

⁵ CSS (Cascading Style Sheets, Kaskádové styly) slouží pro formátování elementů na stránce – viz [7]

S atributy se v jQuery pracuje především pomocí funkcí `attr()` a `removeAttr()`. První zmiňovaná atributy přidává nebo upravuje, druhá je odebírá.

Funkce `attr()` má dvě možnosti využití, prvním je pouze získání hodnoty v atributu a druhou možností je atribut nastavit na námi zvolenou hodnotu.

```
attr( jmeno_atributu )  
attr( jmeno_atributu, hodnota )
```

Dále můžeme využít funkce `val()`, která vrací, nebo nastavuje hodnotu textového pole, select boxu⁶ a u dalších elementů, které mají nějakou hodnotu.

Nejen, že touto funkcí uspoříte několik znaků, které byste museli napsat při použití `attr("value")`, ale také vám tato funkce dokáže vypsát zvolenou položku např. v select boxu, což by pouze s funkcí `attr()` nešlo.

Funkce `val()` má opět dvě možnosti použití. Buď jí můžeme využít pro výpis hodnoty (1. řádek) nebo pro nastavení (2. řádek).

```
val()  
val( hodnota )
```

jQuery má ještě další funkce, které souvisejí s atributy a také s kaskádovými styly, ale o těch více až v kapitole, která se zabývá právě prací jQuery s CSS.

6.6.1 Příklad 1 – Vypsání obsahu textového pole (input)

Jako první příklad si ukážeme jednu z možností, jak si nechat vypsát obsah textového pole. Využijeme k tomu dva formulářové prvky, JavaScriptovou funkci `alert()`, která zajišťuje vypsání zprávy do vyskakujícího okna a samozřejmě jQuery.

Nejdříve je nutné vytvořit zmiňované dva formulářové prvky typu `text` a `button`.

⁶ <http://www.jakpsatweb.cz/html/formulare.html#select>

```
<input type="text" name="pole" id="pole" value="" />
<input type="button" name="obsah_pole" id="obsah_pole"
value="Obsah pole" />
```

To je vše, co je potřeba. Pomocí jQuery už pouze naprogramujeme jednoduchý kód:

```
$("#obsah_pole").bind("click", function() {
    alert($("#pole").val());
    //alert($("#pole").attr("value"));    2.způsob
});
```

jQuery nám zařídí, že po kliknutí na tlačítko s identifikátorem *obsah_pole* se do vyskakujícího okna (*alert*) vypíše obsah elementu (v tomto případě atribut *value*) elementu *#pole*, což je náš požadovaný řetězec.

6.6.2 Příklad 2 – Zakázání úpravy textového pole (input)

K tomuto příkladu budeme potřebovat dva nové formulářové prvky. Jedná se o textové pole a tlačítko.

```
<input type="text" name="pole2" id="pole2" value="" />
<input type="button" name="zakazat" id="zakazat"
value="Zakázat" />
```

Tlačítka jsou nejen pojmenována, ale jsou jim přiřazeny i identifikátory (*id*), aby bylo možné se na ně dobře⁷ odkazovat v jQuery.

V jQuery obstaráme všechno ostatní:

```
$("#zakazat").bind("click", function() {
    if ($("#zakazat").val() == "Zakázat") {
        $("#pole2").attr("disabled", "disabled");
        $("#zakazat").val("Povolit");
        //$("#zakazat").attr("value", "Povolit");
    }
});
```

⁷ Teoreticky se lze odkazovat i podle jména elementu, ale kvůli přehlednosti a kompatibilitě se staršími prohlížeči doporučuji používat identifikátory.

```
    }  
    else {  
        $("#pole2").removeAttr("disabled");  
        $("#zakazat").val("Zakázat");  
        //$("#zakazat").attr("value", "Zakázat");  
    }  
});
```

Po kliknutí na tlačítko zakázat (samozřejmě by se dal použít i obyčejný text či odkaz) porovnáme pojmenování tlačítka. Pokud se jedná o zakázání editace textového pole (tlačítko se jmenuje Zakázat), zakážeme editaci pole přidáním atributu *disabled* a přejmenujeme tlačítko na "Povolit".

Pokud chceme naopak povolit editaci textového pole – odebereme atribut *disabled* a opět přejmenujeme tlačítko.

Samozřejmě toto řešení není ideální, protože by se nemělo využívat řetězců s diakritikou – hlavně kvůli kódování, kdy si musíte dát pozor na správné nastavení kódování a také protože řetězce na stránce se mohou měnit. Nejlepší je využívat identifikátorů nebo tříd.

V tomto případě bychom mohli využít služeb tříd, kdy při zakázání editace textového pole by pole dostalo specifickou třídu – například *zakazano* a při povolování by se tato třída odebrala.

Při testování, co s textovým polem zamýšlíme, by se využila funkce na zjištění existence třídy. V tuto chvíli se ale ještě věnovat třídám nebudeme – práce s nimi je efektivnější právě se speciálními funkcemi, které budou popsány v kapitole JQuery a CSS.

6.7 Události v jQuery

Než se budeme věnovat práci jQuery s CSS, podíváme se na práci s událostmi. Už jste se s nimi setkali v přecházející kapitole, ale nyní si je blíže vysvětlíme.

Pokud nechceme, aby se provedla nějaká funkce nebo posloupnost příkazů ihned, ale například teprve po kliknutí na nějaké tlačítko, musíte využít tzv. událostí

(anglicky Event). Můžeme se s tím setkat např. ve vývojovém prostředí MS Visual Studio. Událost se napojí na nějaký objekt (v našem případě element) a čeká, až se s ním stane to, co jsme definovali.

Když si vezmeme například tlačítko, tak u něj můžeme "odchytávat" události typu kliknutí na tlačítko, přejetí myši přes tlačítko (přesunutí myši na místo, kde je tlačítko), opuštění prostoru myši, kde je tlačítko a spoustu dalších.

Kompletní přehled událostí si můžeme projít na stránkách dokumentace jQuery, konkrétně události v sekci Events⁸.

6.7.1 Výběr událostí

V následující tabulce jsou vybrány některé události. V levém sloupci je anglický název, který se používá při definování události, a v pravém sloupci naleznete popis události.

| Událost | Popis funkce |
|------------------|--|
| Click | Kliknutí levým tlačítkem myši na element (odkaz, tlačítko, ...) |
| Dblclick | Dvojklik myši na element |
| Mouseover | Přejetí myši přes element |
| Mouseout | Opuštění prostoru, kde se nachází element |
| Mousedown | Stisknutí levého tlačítka myši |
| Keydown | Stisknutí klávesy |
| KeyUp | Uvolnění klávesy |
| Blur | Opuštění textového pole (kliknutí mimo textové pole nebo pomocí klávesy Tab) |

Tabulka 1 - Vybrané události

⁸ <http://api.jquery.com/category/events/>

6.7.2 Možnosti napojení události

jQuery nabízí dvě možnosti, jak napojit událost na element. První možností je využít funkce `bind()`, které jako jeden parametr předáte název události a jako druhý parametr ji předáte funkci, kterou chcete vykonat po proběhnutí Vámi zvolené události.

```
bind( nazevUdalosti, funkce )
```

Ukázku použití jste už viděli výše např. ve zdrojovém kódu příkladu pro zobrazení obsahu textového pole. Pro přehlednost bude ale základní struktura uvedena.

```
$("#tlacitko").bind("click", function() {  
    // kód funkce  
});
```

Pokud jste se však dívali do dokumentace jQuery, určitě jste si v sekci událostmi všimli, že tam jsou funkce jako `click()`, `dblclick()` apod. Jaký je rozdíl mezi `bind("click", ...)` a `click(...)`?

Odpověď je jednoduchá. Žádný. Funkce `click()` je pouze zkrácený zápis pro funkci `bind("click")`. Používat tedy můžeme, jakoukoliv chceme.

Při napojování události a sestavování kódu funkce se do její hlavičky může zapsat jeden parametr, do kterého pak prohlížeč předává užitečná data. Pojmenovat si ho můžeme libovolně, nejčastěji se používá `event` nebo `e`. Použití tohoto parametru uvidíte ve třetím příkladu.

6.7.3 Zrušení napojení na událost

Někdy může být užitečné, že v určitém okamžiku napojíte událost a za určitých okolností ji budete chtít odpojit. I tuto možnost jQuery nabízí a to v podobě funkce `unbind()`.

Tato funkce se používá podobně jako samotná funkce `bind()`.

```
$("#tlacitko").unbind("click");
```

Tímto kódem jsme tlačítku odebrali událost reagující na kliknutí levým tlačítkem myši a tlačítko tak již nebude na kliknutí reagovat.

6.7.4 Příklad 1 – Opuštění textového pole

První příklad na vyzkoušení událostí bude jednoduchý a tentokrát se nebude jednat o událost `click()`.

Využijeme událost `blur()`, díky které si necháme vypsat řetězec do vyskakovacího okna - funkce `alert()`, při opuštění textového pole.

HTML kód je opět jednoduchý a skládá se pouze z jednoho textového pole.

```
<input type="text" name="pole3" id="pole3" value="" />
```

V jQuery si vystačíme s událostí `blur()` a funkcí `alert()`.

```
$("#pole3").bind("blur", function() {  
    alert("Opuštěno textové pole.");  
});
```

Nyní se přepněte do textového pole. Následně pokud kliknete mimo textové pole, nebo stisknete klávesu TAB a tím opustíte textové pole, spustí se událost `blur()` a je vypsána zpráva s nastaveným textem.

6.7.5 Příklad 2 - Změny barev tlačítka

Druhý příklad se týká už výše několikrát použité události `click()`. V tuto chvíli Vám bude ukázáno, jak lze jednoduše manipulovat například s pozadím tlačítka.

Vytvořte si jedno tlačítko, kterému budeme měnit barvu.

```
<input type="button" name="tlacitko" id="tlacitko"  
value="Tlačítko" class="modra" />
```

Poprvé jsou také důležité kaskádové styly, a proto zde také budou uvedeny.

```
.modra { background-color: blue; }  
.cervena { background-color: red; }
```

V jQuery si vypomůžeme klasickým JavaScriptem, který nám zajistí vytvoření proměnné *modra*. Tato proměnná nám bude říkat, jestli je tlačítko modré a podle toho mu budeme moci nastavit barvu. Zbytek již zařídí funkce *attr*.

```
var modra = true;  
$("#tlacitko").bind("click", function() {  
    if (modra) {  
        $("#tlacitko").attr("class", "cervena");  
        modra = false;  
    }  
    else {  
        $("#tlacitko").attr("class", "modra");  
        modra = true;  
    }  
});
```

Tento postup samozřejmě není jediný. Porovnání by se mohlo sice provést na základě atributu *class*, ale tento příklad měl také ukázat možnost použití právě takovéto proměnné, kde si lze udržovat informace o stavu některých elementů.

6.7.6 Příklad 3 – Pozice kliknutí v bloku DIV

Někdy může být užitečné, když si můžeme zjistit, kde na stránce uživatel kliknul – například pokud chcete něco zobrazit v závislosti na poloze myši (kliku). Pro tyto případy Vám poslouží parametr funkce, která se spouští v případě, že proběhne událost. My využijeme událost *click()* a parametr pojmenujeme *event*. Po kliknutí se do tohoto parametru přidávají vlastnosti *pageX* a *pageY*, které reprezentují souřadnice kliku.

Událost bude napojena na kliknutí v námi definovaném rámečku – DIV⁹ s id *ramecek*.

```
<div id="ramecek" style="width: 300px; height: 150px;
background-color: orange;"></div>
```

Do souboru s JavaScriptem, respektive jQuery vložíme následující krátký kód:

```
$("#ramecek").bind("click", function(event) {
    alert("X: " + event.pageX + "\nY: " + event.pageY);
});
```

A máme hotovo. Pokud nyní kliknete levým tlačítkem myši na plochu oranžového divu, vypíše se vám souřadnice pozice, na kterou jste v rámci stránky klikli.

6.8 JQuery a CSS

Kaskádové styly (CSS) se většinou píše do odděleného souboru s koncovkou *css* a poté se už o ně nestaráme. Pomocí jQuery je ale můžeme efektivně upravovat při běhu stránek a díky tomu docílit velmi pěkných efektů.

K práci s CSS slouží v jQuery funkce *css()*. Může mít různé druhy parametrů a podle toho nastavovat nebo vypisovat CSS vlastnost.

⁹ DIV = Blokový HTML element pro formátování odstavců. Více viz: <http://www.jakpsatweb.cz/div-span.html>

```
css( nazevVlastnosti )           // vypsat vlastnost  
css( nazevVlastnosti, hodnota ) // nastavit vlastnost
```

Už když jsme se zabývali prací s atributy, bylo zmíněno, že existují speciální funkce pro práci s kaskádovými styly. Jsou to funkce `addClass()`, `removeClass()`, `hasClass()`.

Prvku přidáte třídu pomocí `addClass()`, odeberete pomocí `removeClass()` a pokud budete chtít zjistit, zdali danou třídu obsahuje, zjistíte to pomocí `hasClass()`. Následuje ještě přehled výše zmíněných funkcí i s parametry:

```
addClass( jmenoTridy )  
removeClass( jmenoTridy )  
hasClass( jmenoTridy )
```

6.8.1 Příklad 1 – Hover efekt pomocí jQuery

Pokud pracujete s CSS, určitě znáte pseudotřídu `hover`¹⁰. V jQuery lze docílit stejného efektu.

Do HTML kódu stránky si vložte obrázek:

```

```

V jQuery hover efekt ošetříme pomocí událostí `mouseover` a `mouseout` a zbytek obstará funkce `css()` a CSS vlastnost `opacity`.

¹⁰ Pseudotřída `hover` dovoluje v CSS zapsat formát (barvu, řez apod.) elementům, přes které přejedeme myší. Zápis je např. `a:hover {color: red}` -- přejížděné odkazy budou červené.^[8]

```
$("#smajlik").bind("mouseover", function() {
    $("#smajlik").css("opacity", 0.5);
});

$("#smajlik").bind("mouseout", function() {
    $("#smajlik").css("opacity", 1.0);
});
```

6.8.2 Příklad 2 – Změna velikosti textu

Při nastavování CSS vlastností ale v jQuery nejste omezeni možností nastavit pouze jednu vlastnost. Stačí využít jiné parametry ve funkci `css()`.

Tento příklad by se dal použít na všech stránkách, kde chceme uživateli umožnit zvětšit nebo zmenšit text. Na některých stránkách tato možnost je.

Na stránce tedy vytvoříme tři odkazy – malé písmo, normální písmo a velké písmo a pomocí těchto odkazů budeme upravovat velikost textu.

Velikost textu je samozřejmě nastavena pouze do znovu načtení stránky. Aby se uchovala i poté, museli bychom použít např. cookies¹¹ a ukládat si aktuální velikost písma a po načtení stránky ji aplikovat. Pro názornou ukázkou však bude postačovat verze bez ukládání.

Vytvoříme zmiňované tři odkazy:

```
<a href="#" id="male_pismo">Malé písmo</a> | <a href="#"
id="normalni_pismo">normální písmo</a> | <a href="#"
id="velke_pismo">velké písmo</a>
```

A také si vytvoříme speciální odstavec, na který budeme velikost písma aplikovat¹².

¹¹ <http://www.microsoft.com/cze/athome/security/computer/whatis/whatiscookie.msp>

¹² Samozřejmě na reálné stránce můžete velikost aplikovat např. na celé HTML.

```
<p id="aplikovat_velikost">Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Proin ac ornare urna. Curabitur
posuere ornare odio, et eleifend lorem hendrerit ut. Duis justo
enim, interdum sit amet vehicula quis, euismod eu purus.
Vestibulum et sapien molestie dolor auctor fermentum. Etiam
eget nisi eget nulla dapibus volutpat dignissim nec metus.
In.</p>
```

A opět, jQuery obstará zbylou funkčnost.

```
$("#male_pismo").bind("click", function() {
    $("#aplikovat_velikost").css("font-size", "10px");
    return false;
});

$("#normalni_pismo").bind("click", function() {
    $("#aplikovat_velikost").css("font-size", "12px");
    return false;
});

$("#velke_pismo").bind("click", function() {
    $("#aplikovat_velikost").css("font-size", "14px");
    return false;
});
```

Využívá se opět události `click()`, funkce `css()` a nyní vlastnosti `font-size`. Po kliknutí na každý odkaz se provede úprava velikosti písma. Příkaz `return false;` je zde opět pouze z důvodu, aby se odkaz neprovedl. Ale pokud by se nepoužívaly odkazy, ale např. obrázky, samozřejmě by zde být nemusel.

6.9 JQuery a relativní výběry elementů

Při výběru elementu si lze většinou postačit se třídami či identifikátory. Pokud je však na stránce více elementů stejného druhu a my potřebujeme pracovat s právě s jedním, určitým elementem, využijete právě relativní výběry, které Vám umožní odkazovat se na následující element, rodičovský element a další.

Prvním způsobem je přidání dvojtečky a klíčového slova do výběru elementu (selektoru¹³). Druhým způsobem je využití některé ze speciálních funkcí.

6.9.1 Dvojtečka a klíčové slovo ve výběru elementu

Při výběru elementu, případně elementů, lze k elementu přidat dvojtečku a napsat nějaké klíčové slovo. Jejich seznam naleznete v dokumentaci na stránce Selectors¹⁴. V následující tabulce jsou vybrána některá klíčová slova a v pravém sloupci je popis jejich funkce.

| Klíčové slovo | Popis |
|------------------------|---|
| :button | Výběr elementů input typu button nebo elementů button |
| :checked | Výběr zaškrtnutých checkboxů |
| :contains(text) | Výběr elementů, v jejichž obsahu se nachází <i>text</i> |
| :disabled | Výběr elementů input s atributem disabled |
| :even | Výběr lichých elementů |
| :first-child | Výběr prvního potomka |
| :first | Výběr prvního elementu |

Tabulka 2 - Vybraná klíčová slova pro výběr elementu/elementů

Tyto klíčová slova se dají výhodně použít. Například zrovna klíčové slovo *even* nám může ušetřit spoustu práce při vybarvování řádků tabulky, pokud chceme, aby liché řádky byly jinou barvou než sudé. Následující příklad naleznete na stránce s popisem klíčového slova *even*¹⁵ a ukazuje jeho jednoduché použití právě na zmiňovaný problém.

HTML kód:

¹³ Selektor = závorka, kde definujete element, který chcete vybrat. V kódu: `$("ul.trida").hide()` je selektor: `("ul.trida")`

¹⁴ <http://api.jquery.com/category/selectors/>

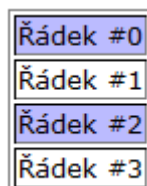
¹⁵ <http://api.jquery.com/even-selector/>


```
<table border="1">
  <tr><td>Řádek #0</td></tr>
  <tr><td>Řádek #1</td></tr>
  <tr><td>Řádek #2</td></tr>
  <tr><td>Řádek #3</td></tr>
</table>
```

JavaScript:

```
$("tr:even").css("background-color", "#bbbbff");
```

Výsledek:



| |
|----------|
| Řádek #0 |
| Řádek #1 |
| Řádek #2 |
| Řádek #3 |

Obrázek 1 - Tabulka po aplikování podbarvování

6.9.2 Speciální funkce pro výběr elementu

Jako příklad můžeme vyjmenovat některé nepoužívanější funkce. Rozhodně by se mezi ně zařadily funkce *prev()*, *next()*, *children()*, *parent()* a *find()*.

```
prev( [element] )
next( [element] )
children( [element] )
parent( [element] )
find( element )
```

6.9.2.1 Prev() a next()

První dvě zmíněné ukazují na jeden prvek, který je před zvoleným elementem – funkce *prev()*, nebo za zvoleným elementem – funkce *next()*. Pokud bychom se

chtěli odkázat na všechny prvky před (případně za) zvoleným elementem, použili bychom funkce `nextAll()`, případně `prevAll()`.

6.9.2.2 Children() a parent()

Funkce `children()` a `parent()` mají mírně odlišnou funkci, kterou lze ale odvodit z jejich názvu. Pomocí funkce `children()` se můžeme odkázat na elementy, které jsou v rámci našeho vybraného elementu a pomocí funkce `parent()` se naopak můžeme odkázat na rodičovský element, tedy element nadřazený našemu vybranému.

6.9.2.3 Prev() a next() vs. children() a parent()

Zjednodušeně by se dalo říct, že rozdíl mezi funkcemi `next()`, `prev()` a `children()`, `parent()` je takový, že `next()` a `prev()` vybírají elementy ve stejné úrovni, přičemž `children()` a `parent()` se zanořují hlouběji do elementu.

Pokud bychom měli `DIV` a uvnitř něho by byl seznam `UL` s položkami `LI`, tak `DIV` je nadřazený element (rodič) `UL` a `UL` je rodič `LI`. Z toho lze usoudit, že `UL` je potomek `DIV`.

HTML struktura předcházejícího příkladu by byla tato:

```
<div>
  <ul>
    <li>Položka</li>
    <li>Položka</li>
    <li>Položka</li>
  </ul>
</div>
```

Všechny výše zmíněné funkce mají jeden nepovinný parametr, kterým lze zpřesnit vyhledávaný element. Pokud budeme chtít vybrat pouze potomky typu `LI`, použijeme funkci `children()` následovně.

```
children("li");
```

6.9.2.4 Find()

Tuto funkce byste použili, pokud by bylo zapotřebí vyhledat element nezávisle na jeho úrovni, či zanoření.

V dokumentaci jQuery, na stránce o funkci `find()`¹⁶, můžeme najít následující příklad. V HTML kódu máme několik seznamů *UL* zanořených do sebe. Každá položka seznamu má třídu v závislosti na tom, jak hluboko je v celém seznamu.

```
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
```

V tuto chvíli by bylo obtížné podbarvit od určité položky všechny následující položky, jak je záměrem příkladu. Nelze totiž využít výběr pomocí položky `LI`, protože jsou v seznamu položky, které zbarvené být nemají, a zároveň nemůžeme použít výběr podle třídy, protože každá položka má jinou třídu.

¹⁶ <http://api.jquery.com/find/>

Nemůžeme použít ani funkci `children()`, protože ta nám sice vrací potomky našeho elementu, ale vyhledává je pouze v první úrovni a dále se nezanořuje.

Řešení spočívá v použití funkce `find()`, které jako parametr předáme element `li` a aplikujeme ji na element, od kterého chceme následující položky podbarvit.

```
$('.li.item-ii').find('li').css('background-color', 'red');
```

6.9.3 Příklad 1 – Rodiče a děti

Princip tohoto příkladu bude takový, že si vytvoříme si několik seznamů, které budou představovat vždy jednoho rodiče a jeho děti. Děti budou ve výchozím stavu skryté a zobrazí se až po kliknutí na příslušného rodiče. Při znovu kliknutí budou opět skryty.

Seznam rodičů a jejich dětí může být například takový jako je seznam níže. Hlavní seznam má navíc identifikátor `rodice_a_deti`, díky kterému se na něj odkážeme.

```
<ul id="rodice_a_deti">
  <li>Pavla
    <ul>
      <li>Dominika</li>
      <li>Martina</li>
    </ul>
  </li>
  <li>Taťána
    <ul>
      <li>Miroslav</li>
      <li>Tomáš</li>
    </ul>
  </li>
  <li>Zuzana
    <ul>
      <li>Daniela</li>
    </ul>
  </li>
```

```
<li>Tereza
  <ul>
    <li>Petr</li>
    <li>Pavel</li>
    <li>Přemysl</li>
  </ul>
</li>
</ul>
```

Děti musí být ve výchozím stavu skryté, a proto jim musíme v CSS nastavit `display: none`. Položkám seznamu je také nastaven hezčí kurzor, ale to už samozřejmě není podmínka.

```
ul#rodice_a_deti li {
  cursor: pointer;
}
ul#rodice_a_deti li ul {
  display: none;
}
```

JavaScriptový kód nám příliš práce nezabere.

```
$('#ul#rodice_a_deti li').bind("click", function() {
  $(this).children("ul").toggle();
});
```

Využívá se události `click()`, která reaguje na kliknutí na položky seznamu `rodice_a_deti`. Pomocí funkce `children()` se odkážeme na ty správné potomky.

Využitá je také funkce `toggle()`, kterou prozatím neznáte. Jedná se o funkci z kategorie efektů. Díky ní se nám položky zobrazí, pokud jsou skryté nebo skryjí, pokud jsou zobrazené. Nemusíme tak nikam ukládat stav potomků a podle toho používat funkce `show()` nebo `hide()`.

6.9.4 Příklad 2 – Přebarvování čtverců

V tomto příkladu si ukážeme použití funkcí `nextAll()` a `prevAll()`. Vytvoříme si v HTML kódu několik oranžově vybarvených čtverců pomocí elementu `DIV`. Funkce `nextAll()` a `prevAll()` nám budou sloužit pro přebarvení čtverce, na který klikneme a také všech "za ním", pokud klikneme levým tlačítkem myši nebo všech před ním, pokud klikneme pravým tlačítkem myši.

Barva čtverců se také bude lišit podle toho, pokud klikneme levým nebo pravým tlačítkem myši. Při kliknutí levým tlačítkem, bude barva čtverců modrá, při kliknutí pravým tlačítkem, bude barva červená.

Příklad bude také obsahovat tlačítko `Reset`, které bude nastavovat čtvercům výchozí (oranžovou) barvu.

Začneme nejdříve HTML kódem, který bude tvořit sedmkrát zopakovaný element `DIV` s přiřazenou třídou `ctverec` a také zmiňované tlačítko pro navrácení výchozí barvy.

```
<div class="ctverec"></div>
<div class="ctverec"></div>
...
<input type="button" name="reset_ctverec" id="reset_ctverec"
value="Reset" />
```

Zmíníme také nastavení kaskádových stylů, které jsou neméně důležité. Čtvercům je třeba nastavit obtékání, šířku, výšku, barvu a další. Pro atraktivnější ovládání je čtvercům přiřazen kurzor ukazující ruky. Tlačítko má zakázané obtékání, aby bylo na novém řádku.

```
div.ctverec {
    width: 80px;
    height: 80px;
    float: left;
    display: block;
    background-color: #ff9900;
    margin: 10px;
```

```
    cursor: pointer;
}
#reset_ctverec {
    display: block;
    clear: both;
}
```

Následující kód JavaScriptu (a jQuery) nám zajistí požadovanou funkcionalitu.

```
$(".ctverec").bind("click", function() {
    $(this).css("background-color", "#009");
    $(this).nextAll("div").css("background-color", "#000099");
});
$(".ctverec").bind("contextmenu", function() {
    $(this).css("background-color", "#c00");
    $(this).prevAll("div").css("background-color", "#cc0000");
    return false;
});
$("#reset_ctverec").bind("click", function() {
    $(".ctverec").css("background-color", "#ff9900");
});
```

V příkladu je použita událost *click()*, která reaguje na kliknutí levým tlačítkem myši a také událost *contextmenu*, která reaguje na kliknutí pravého tlačítka myši. Změna barvy se provádí díky funkcím *css()* a *nextAll()* nebo *prevAll()*.

Použití *nextAll()* místo *next()* a *prevAll()* místo *prev()* je nutné, protože při použití druhých zmíněných funkcí by se přebarvil pouze jeden následující (příp. předcházející) element.

Je také nutné omezit výběr elementů ve funkcích *nextAll()* a *prevAll()* pouze na element *DIV*, protože jinak by se přebarvili všechny ostatní element a tedy i tlačítko *reset*.

6.10 Rozměry v jQuery

Při vytváření interaktivních webových aplikací někdy narazíte na nutnost zjistit aktuální výšku některého z prvků na stránce. V tomto případě vám nepomohou kaskádové styly, protože pokud rozměry nenastavíte, pomocí CSS se k nim nedostanete.

Může to být kvůli umístění nějakého nového prvku, anebo třeba pokrytí určité plochy nějakou barvou – příkladem mohou být některé lightboxy¹⁷, které při zobrazení obrázku potáhnou pozadí stránky "plátnem" které pozadí ztmaví.

Aktuální rozměry lze zjišťovat pomocí funkcí `height()` a `width()`, pokud nezadáte žádné parametry. V případě, že byste chtěli rozměry nastavit, využijete jeden parametr, který obě funkce mohou mít.

```
height()           // vrací aktuální výšku
height( hodnota ) // nastavuje aktuální výšku
width()            // vrací aktuální šířku
width( hodnota )  // nastavuje aktuální šířku
```

Při použití těchto funkcí však můžeme narazit na problém. A to tehdy, pokud nebude mít element, se kterým pracujeme, vynulovaný `padding` a `margin`. Funkce `height()` a `width()` totiž vrací rozměry bez `paddingu` a `marginu`.

Abychom získané rozměry zpřesnili, můžeme použít funkce `innerHeight()` nebo `outerHeight()`. Funkce `innerHeight()` nemá žádný parametr a vrací výšku elementu, včetně vlastnosti `padding`.

Funkce `outerHeight()` má jeden nepovinný parametr, který určuje, jestli se má do výšky započítat také `margin`. Pokud je parametr nastaven na `false` nebo je vynechán, vrací funkce výšku elementu + `padding` + `border`.

¹⁷ Lightbox je moderní způsob, jak zobrazit na stránkách fotografii ve větším rozlišení. Pokud uživatel klikne na zmenšenou verzi, není mu do stávajícího/nového okna otevřena originální velikost, ale stránka zašedne a v popředí se zobrazí pěkný rámeček přes celou plochu stránky a v něm je fotka ve větším rozlišení.

Pokud chceme zjistit šířku, použijeme stejné funkce, pouze vyměníme řetězec *height* v názvu funkce za *width*.

```
innerWidth()  
innerHeight()  
outerWidth( [zapocitatMargin] )  
outerHeight( [zapocitatMargin] )
```

6.10.1 Příklad 1 – Vypsání aktuálních rozměrů body

V příkladu číslo 1 se budeme zabývat jednoduchým využitím funkcí *width()* a *height()*. Pomocí těchto funkcí a tlačítka si necháme vypsát do vyskakovacího okna šířku a výšku elementu *body*, který budeme moci pomocí dalšího tlačítka rozšířit, protože zobrazíme skryté odstavce textu.

Nejdříve si vytvoříme zmiňovaná dvě tlačítka. Jedno bude sloužit k zobrazení šířky a výšky elementu *body*, druhé bude zobrazovat skryté odstavce, aby se nám *body* rozšířilo a názorně jste tak viděli změnu v rozměrech.

```
<input type="button" name="vypsati_rozmeru_body"  
id="vypsati_rozmeru_body" value="Vypsati rozmery body" />  
<input type="button" name="zobraziti_odstavce"  
id="zobraziti_odstavce" value="Zobraziti odstavce" />
```

Skryté odstavce jsou naplněny textem z generátoru lipsum.com. Texty jsou zkrácené, aby nezabíraly příliš místa a v celé podobě je naleznete na příloženém CD.

```
<p class="neviditelny">  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam  
sit amet tempus tellus. Mauris iaculis massa non urna laoreet  
at rhoncus lectus vestibulum. Sed vitae venenatis turpis. In ut  
...  
</p>  
  
<p class="neviditelny">  
Vestibulum malesuada semper diam in interdum. Proin faucibus  
tellus ut lorem venenatis vel aliquet metus semper. Phasellus  
tincidunt lobortis turpis non dignissim. Nam porta, est et  
posuere elementum, libero metus lobortis purus, vel volutpat
```

```
...
</p>

<p class="neviditelny">
Suspensisse sit amet purus vitae odio dapibus laoreet nec a
nisl. Duis pellentesque pharetra nisi, in dignissim odio
dapibus nec. Nulla tristique orci sed leo elementum et
...
</p>
```

JavaScript bude následující:

```
$("#vypsats_rozmary_body").bind("click", function() {
    var sirka = $("body").width();
    var vyska = $("body").height();
    alert("Šířka: " + sirka + "\nVýška:" + vyska);
});
$("#zobrazit_odstavce").bind("click", function() {
    $("p.neviditelny").show();
});
```

Pomocí jQuery napojíme na tlačítka události `click()`, které zajistí potřebnou funkcionalitu. Kliknutí na tlačítko `vypsats_rozmary_body` vyvolá vypsání aktuálních rozměrů body pomocí funkcí `width()` a `height()`.

Kliknutí na tlačítko `zobrazit_odstavce` pouze zobrazí skryté odstavce – funkce `show()` bude popsána v následující kapitole.

6.10.2 Příklad 2 – Zobrazení textu doprostřed okna

Tento příklad je o trochu složitější než předchozí. Opět však budeme využívat především funkce pro zjištění šířky a výšky. V některých případech budete potřebovat zobrazit text přímo doprostřed okna. I když tuto funkcionalitu příliš nevyužijete, někdy se může hodit.

Využijeme tlačítko `zobrazit_text_uprostred`, která nám po kliknutí zobrazí zmiňovaný text.

```
<input type="button" name="zobrazit_text_uprostred"
id="zobrazit_text_uprostred" value="Zobrazit text" />
```

Uprostřed okna budeme zobrazovat následující element DIV. Většinou bude nutné, abyste element, který se má zobrazit přes celou stránku, nebo v něm nemají být započítány žádné rozměry z předchozích elementů, umístili hned za element body.

```
<div id="text_uprostred">Text uprostřed okna</div>
```

Tomuto elementu DIV musíme nastavit některé parametry v kaskádových stylech. Jedná se především o skrytí elementu (`display:none`), šířku, aby se dal element dobře vycentrovat, z-index, aby se element zobrazil opravdu nad všemi ostatními elementy a samozřejmě také absolutní pozicování. Dále je mu nastaveno ještě několik méně podstatných vlastností.

```
#text_uprostred {
    display: none;
    font-weight: bold;
    font-size: 20px;
    border: 1px black solid;
    width: 250px;
    text-align: center;
    position: absolute;
    z-index: 999;
    background-color: #ff9900;
}
```

Kód JavaScriptu:

```
$("#zobrazit_text_uprostred").bind("click", function() {
    var sirkaOkna = $("body").width();
    var delkaTextu = $("#text_uprostred").outerWidth();

    var vyskaTextu = $("#text_uprostred").outerHeight();
    var dostupnaVyska = $(window).height();
```

```
$("#text_uprostred").css("left", (sirkaOkna/2 -  
delkaTextu/2));  
$("#text_uprostred").css("top", (dostupnaVyska/2 -  
vyskaTextu/2));  
  
$("#text_uprostred").show();  
});
```

Abychom mohli zobrazit text doprostřed okna, potřebujeme zjistit zaprvé šířku okna a za druhé jeho výšku. Šířku okna můžeme zjistit pomocí šířky elementu *body*, ale výška elementu *body*, je jeho kompletní výška, což není to, co my potřebujeme.

My potřebujeme znát výšku plochy, kterou prohlížeč dokáže zobrazit na jednu stránku (bez nutnosti posouvání). Na konkrétní rozměry okna se můžeme odkázat pomocí objektu *window*¹⁸.

Když už známe konkrétní šířku i výšku okna, stačí už pouze napozicovat požadovaný element na střed, k čemuž využijeme funkci *css()* a nakonec *DIV* zobrazit pomocí funkce *show()*.

Rozměry textu je dobré zjišťovat pomocí *outerWidth()* a *outerHeight()*, aby do rozměrů byly započítány i vlastnosti *border*, *padding* a *margin*.

Funkce *show()* pro Vás ještě není známa a bude popsána v následující kapitole. I přesto bude použita také v následujícím příkladu, protože je výhodnější, když element není zobrazen od začátku, ale až poté, co se klikne na tlačítko.

6.10.3 Příklad 3 – Pokrytí pozadí plátnem

Posledním příkladem této kapitoly bude ukázka toho, jak překrýt celou stránku nějakým elementem – plátnem. Plátno pak můžeme použít například pod text z předchozího příkladu, který je zobrazen uprostřed okna.

¹⁸ http://www.w3schools.com/jsref/obj_window.asp

Aby nebylo plátno vidět od začátku, i v tomto příkladu použijeme tlačítko, které nám ho zobrazí.

```
<input type="button" name="pokryt_platnem" id="pokryt_platnem" value="Pokrýt plátmem" />
```

Plátno nám bude představovat element DIV, který bude opět nejlepší umístit přímo za element *body*.

```
<div id="platno"></div>
```

Než se budeme věnovat JavaScriptu, musíme plátneu, stejně tak jako elementu *DIV* z předchozího příkladu, nastavit některé kaskádové styly. Jedná se o barvu, *z-index*, absolutní pozicování, raději také vynulování *marginu* a *paddingu* a samozřejmě skrytí elementu, abychom ho mohli zobrazit až po kliknutí na tlačítko.

```
#platno {
  background-color: #999;
  position: absolute;
  z-index: 999;
  display:none;
  top: 0;
  left: 0;
  margin: 0;
  padding: 0;
}
```

Nyní už se budeme věnovat JavaScriptu, který bude podobný tomu z předcházejícího příkladu.

```
$("#pokryt_platnem").bind("click", function() {  
    var sirkaOkna = $(window).width();  
    var vyskaOkna = $(window).height() - 2;  
  
    $("#platno").css("width", sirkaOkna);  
    $("#platno").css("height", vyskaOkna);  
    $("#platno").css("opacity", 0.5);  
  
    $("#platno").show();  
});
```

Rozměry okna zjistíme opět díky objektu *window*. Od výšky ale musíme odečíst ještě 2 pixely, protože jinak by bylo plátno příliš vysoké a zbytečně by se nám zobrazil scrollbar (posouvátko).

Pomocí funkce *css()* nastavíme plátneu šířku a výšku, ale také průhlednost, aby to vypadalo lépe. Nakonec je plátno zobrazeno pomocí funkce *show()*.

Tento skript funguje ve všech prohlížečích, pouze v Internet Exploreru se po zobrazení plátneu zruší CSS vlastnost *margin: 10px auto*, která je aplikována na *DIV stranka*. Tento *DIV* obepíná celý obsah a ten díky tomu mírně uskočí nahoru.

6.11 Efekty v jQuery

Nyní se dostáváme k jedné z nejzajímavějších dovedností jQuery. Jsou to ty mnohokrát zmiňované efekty, které lze vytvářet velmi rychle a snadno.

Použitím jediné funkce lze dosáhnout postupného vysunutí menu, efektu *fade in/out* neboli postupné zmizení apod.

Příkladem mohou být funkce *fadeIn()*, *fadeOut()*, *slideDown()*, *slideUp()*, *show()*, *hide()* nebo *animate()*. Přestože je jich více, např. *fadeIn()*, *slideDown()* i *show()* (a další) fungují na podobném principu. Názornější bude, když se podíváme na hlavičky funkcí.

```
fadeIn( [delkaEfektu], [callbackFunkce] )  
fadeOut( [delkaEfektu], [callbackFunkce] )  
slideDown( [delkaEfektu], [callbackFunkce] )  
slideUp( [delkaEfektu], [callbackFunkce] )  
show( [delkaEfektu], [callbackFunkce] )  
hide( [delkaEfektu], [callbackFunkce] )
```

Jak vidíte, všechny mají stejné parametry. Je možné nastavit jak rychle se má efekt vykonat a je také možné nastavit funkci, která se provede po skončení funkce, což je tzv. callback funkce.

Liší se pouze v typu efektu, který se použije – *fade* používá efekt postupně zvětšující se průhlednosti, až nakonec prvek zmizí. *Slide* využívá efektu rolety, kdy se stahuje (*slideUp()*) nebo zatahuje (*slideDown()*) apod.

Rychlost provedení efektu lze nastavit slovně – *slow*, *normal*, *fast* nebo pomocí čísla, které udává celkovou délku efektu a uvádí se v milisekundách.

V návratové (callback) funkci si jednoduše můžeme nechat vypsat okno s textem: "Hotovo" anebo provádět další operace, spouštět další funkce apod.

Funkce *show()* a *hide()* se také častou používají k jednoduchému skrytí nebo zobrazení elementu – stačí ponechat funkci bez parametrů nebo nastavit délku na 0 a element se skryje/zobrazí ihned.

6.11.1 Speciální funkce *animate()*

Funkce *animate()* je jiná než předešlé. Už z definice hlavičky funkce je patrné, že se něčím liší.

```
animate( vlastnosti, [delkaTrvani], [typEasingEfektu], [  
callbackFunkce ])
```

Tato funkce umožňuje zanimovat přidání, úpravu anebo odebrání nějakého kaskádového stylu. Můžeme si nastavit vlastnosti, které se budou měnit a poté také nepovinně délku trvání, typ efektu a samozřejmě také callback funkci.

6.11.2 Příklad 1 – Zobrazení elementu

První příkladem na vyzkoušení efektů bude jednoduché, ale vzhledově hezké zobrazení prvku na stránce.

Pokud si vzpomínáte na motivační příklad, tak ten prezentoval přesně tuto funkci. Ale aby byl tento příklad přeci jen trochu odlišný, vyzkoušíme si více typů efektů, alespoň je uvidíte v praxi.

Nejdříve je nutné si vytvořit jednoduchou tabulku, která bude obsahovat odkazy, které budou zobrazovat obsah a právě zmíněný obsah.

```
<table id="zobrazeni">
  <tr>
    <td>
      <a href="#" class="zobrazeniShow">Show</a>
      <p class="zobrazeniShow neviditelny">Zobrazený
text</p>
    </td>
    <td>
      <a href="#" class="zobrazeniFadeIn">FadeIn</a>
      <p class="zobrazeniFadeIn neviditelny">Zobrazený
text</p>
    </td>
    <td>
      <a href="#"
class="zobrazeniSlideDown">SlideDown</a>
      <p class="zobrazeniSlideDown
neviditelny">Zobrazený text</p>
    </td>
  </tr>
</table>
```

Aby se vše zobrazilo tak jak má, je nutné provést i nastavení CSS. Všechny kaskádové styly vždy naleznete na příloženém CD, ale v některých případech je nutné jejich správné nastavení a proto budou v případě potřeby vypsány i na tomto místě.

```
#obsah .neviditelny { display: none; }
```



```
#obsah p.zobrazeniShow, #obsah p.zobrazeniFadeIn, #obsah
p.zobrazeniSlideDown {
    background-color: blue; padding: 5px; width: 180px; color:
white; margin: 0; text-indent: 0;
}
a.zobrazeniShow, a.zobrazeniFadeIn, a.zobrazeniSlideDown {
    display: block; width: 180px; padding: 5px;
}
table#zobrazeni {
    height: 60px;
    border: 1px black solid;
}
table#zobrazeni td {
    vertical-align: top;
}
```

Třidu *neviditelny*, kterou si na tomto místě definujeme, budeme využívat i v následujících příkladech.

Následně si nastavíme okraje, pozadí a rozměry odstavců, které budeme zobrazovat. A to tak, abychom se vyvarovali nepěknému posouvání textu a podobně. Z tohoto důvodu je také nastavena výška tabulky a zarovnání textu v sloupci nahoru.

V jQuery už pouze stačí napojit potřebné události a efekty.

```
$("#a.zobrazeniShow").bind("click", function() {
    $("#p.zobrazeniShow").show("slow");
    return false;
});

$("#a.zobrazeniFadeIn").bind("click", function() {
    $("#p.zobrazeniFadeIn").fadeIn("slow");
    return false;
});

$("#a.zobrazeniSlideDown").bind("click", function() {
    $("#p.zobrazeniSlideDown").slideDown("slow");
    return false;
});
```

Po kliknutí na jednotlivé odkazy se spustí požadovaný efekt a to tak, že se tímto efektem zobrazí prozatím skrytý odstavec. Odkazy i odstavce napojujeme pomocí speciální třídy a všem efektům byla nastavena pomalá rychlost, aby bylo názorně vidět, jak efekt pracuje.

6.11.3 Příklad 2 – Modernější hover efekt

V kapitole, která se zabývá prací jQuery s kaskádovými styly jsme si ukázali, že je možné udělat hover efekt nad nějakým elementem pomocí jQuery bez nutnosti definovat CSS pseudotřídou hover.

Ale jQuery toho umí ještě více. Zmiňovaný hover efekt lze ještě zdokonalit, pokud použijeme funkce pro tvorbu efektů. Následující příklad je inspirován článkem v Profi Magazínu¹⁹, ale rozhodně by na něj mohl přijít kdokoliv, kdo už zná funkci `animate()`.

V HTML kódu stránky využijeme obrázek smajlíka:

¹⁹ <http://www.profimagazin.cz/jquery/jquery-zmena-pruhlednosti-pri-hoveru-changing-opacity-on-hover>

```

```

JavaScriptový kód opět není nijak dlouhý, ačkoliv nám zajistí moc pěkný efekt.

```
$("#smajlik2").bind("mouseover", function() {
    $("#smajlik2").stop().animate({ opacity: 0.5 }, 500);
});

$("#smajlik2").bind("mouseout", function() {
    $("#smajlik2").stop().animate({ opacity: 1.0 }, 500);
});
```

Události *mouseover* a *mouseout* nám zaručí, že se daný kód bude provádět při přejetí/odjetí myši z obrázku.

V předchozí verzi (implementace podle CSS), bylo využito funkce *css()*. Nyní využíváme funkce *animate()*, která dokáže, jak již bylo řečeno, zanimovat průběh přidání, odebrání nebo úpravy kaskádového stylu.

Stačí si zvolit styl, který chceme aplikovat (případně je možné aplikovat i více stylů) a také délku trvání efektu (v milisekundách). Při *mouseover* tedy zvolíme přidání průhlednosti (CSS *opacity*) a při *mouseout* jej opět vrátíme na původní hodnotu, jinak by nám obrázek zůstal zprůhledněný.

Tím je efekt hotový. Avšak při událostech si musíte dát také pozor na zahlcení fronty. JQuery zpracovává všechny události postupně za sebou, takže pokud byste v tuto chvíli přejížděli myši přes obrázek rychle, vzniklo by spoustu událostí, které by JQuery chtělo provést, takže by Vám ještě dlouho obrázek "blikal."

6.11.3.1 Využití funkce *stop()* v příkladu

Tohoto nepříjemného efektu se můžeme vyhnout použitím užitečné funkce *stop()*.

```
stop( [vycistitFrontu], [skocitNaKonec] )
```

Funkce `stop()` způsobí zastavení animace, která na prvku právě probíhá. Pokud tedy rychle přejedeme myší přes obrázek a hned ho zase opustíme, spustí se událost `mouseover`, ale než se stihne dokončit, vyvolá se událost `mouseout`. Tím, že vložíme funkci `stop` před funkci `animate()` způsobíme to, že ať už probíhá cokoli s prvkem, se kterým chceme pracovat, zastaví se to a bude probíhat funkce, kterou jsme zadali za funkcí `stop()`.

Funkce `stop` má dva nepovinné parametry. Pokud by Vám nepostačovala samotná funkce `stop()` – pokud by Vám šlo například o zastavení více funkcí/animací, nastavte parametr `vyčistitFrontu` na `true`. Parametr `skocitNaKonec` naopak způsobí provedení poslední přidané funkce/animace do fronty.

Oba parametry jsou nepovinné, a pokud je nezádáte, budou nastaveny na `false`. Pokud si chcete vyzkoušet využití těchto parametrů v praxi, upravte příklad, který zde byl popisován. Stačí například za funkci `animate()` v `mouseover` přidat ještě třeba `fadeOut()` (kvůli tomu, aby ve frontě bylo více funkcí) a funkci `stop` v události `mouseout` nastavte na `true` parametr `vyčistitFrontu`.

Pokud budete rychle přejíždět myší přes obrázek a byl by parametr `vyčistitFrontu` ve funkci `stop()` v události `mouseout` nastaveno na `false` (výchozí nastavení), provedou se postupně všechny funkce – částečné zprůhlednění obrázku (funkce `animate()` v `mouseover`), postupné zmizení obrázku (`fadeOut()` v `mouseover`) a pak až zrušení průhlednosti (`animate()` v `mouseout`).

Avšak výsledek poslední funkce už nevidíte, protože funkce `fadeOut()` obrázek úplně skryla. Je nutné tedy před aplikování funkce `animate()` v `mouseout` úplně vyčistit frontu a začít provádět přesně to, co chcete.

6.11.3.2 Alternativa k funkcím s efekty a funkci `stop()`

Použití funkce `stop` může být však někdy problematické. V případě, že se Vám nedaří vyladit správné použití funkce `stop`, můžeme si vypomocet funkcí `animate()`.

Na funkci `animate()` jsme již narazili právě v příkladu vytvoření hezčího hover efektu. Pomocí funkce `animate()` můžeme napodobit funkce vytvářející efekty (`fadeIn/Out`, `slideDown/Up` apod.). Stačí si uvědomit, že funkce `fadeIn()` (zjednodušeně řečeno) využívá průhlednosti, `slideDown()` využívá výšky elementu a tak bychom mohli pokračovat dále.

Pokud nenastane problém s CSS, máme při použití `animate()` efekt plně pod kontrolou. A v případě, že se obáváte zacyklení/zahlcení fronty, můžeme použít jiného zápisu parametrů.

V příkladu vytvoření hezčího hover efektu bylo použito následujícího zápisu:

```
animate({ opacity: 1.0 }, 500);
```

Ale jQuery umožňuje zápis funkce `animate()` i jiným způsobem.

```
animate( vlastnosti, nastaveni )
```

V parametru `nastaveni` pak můžeme nastavovat nejen délku/rychlost efektu (duration), ale také ovládat frontu (queue), vložit návratovou funkci, která se provede v případě úspěšného dokončení a další.

Návratovou funkci lze nastavit i v prvním případě nastavení parametrů, ale frontu nastavit nelze.

Stejný zápis předchozí ukázky i se zakázáním fronty, by byl následující:

```
animate({ opacity: 1.0 }, { duration:500, queue:false })
```

6.11.4 Příklad 3 – Vysouvací menu

Jako poslední příklad ke kapitole o práci s jQuery efekty bude vytvoření horizontálního vysouvacího menu. Menu bude samozřejmě obohacené o efekt vysunutí.

Základem všeho je menu vytvořené pomocí seznamu, zaobaleného do jednoho divu. Podpoložky menu se udělají jednoduše přidáním nového seznamu k potřebné položce. Viz HTML kód.

```
<div id="prikladmenu">
  <ul>
    <li>Menu položka
      <ul>
        <li><a href="#">Podpoložka</a></li>
      </ul>
    </li>
    <li>Menu položka
      <ul>
        <li><a href="#">Podpoložka</a></li>
        <li><a href="#">Podpoložka</a></li>
        <li><a href="#">Podpoložka</a></li>
      </ul>
    </li>
    <li><a href="#">Menu položka</a></li>
    <li>Menu položka
      <ul>
        <li><a href="#">Podpoložka</a></li>
        <li><a href="#">Podpoložka</a></li>
      </ul>
    </li>
  </ul>
</div>
```

Bylo by možné se obejít i bez divu, ale nastaly by problémy se správným zobrazením menu a funkčností efektů.

Právě kvůli správnému zobrazení a funkčnosti menu je nutné také vyladit kaskádové styly. Styly vezmeme postupně. Nejdříve nadefinujeme vlastnosti pro obalovací *DIV*.

```
#prikladmenu {
  width: 480px;
```

```
height: 30px;
border: 1px black solid;
margin: 0 auto;
}
```

Nepodařilo se mi přijít na způsob, jak se obejít bez pevného nastavení šířky, takže podle počtu prvků se bude muset vždy změnit, pokud by Vám také vadilo prázdné místo v případě, že by se nastavila záměrně větší šířka.

Seznamu (*UL*) se pouze vynulují *margin* a *padding*, ale to na menu příliš vliv nemá.

```
#prikladmenu ul {
margin: 0;
padding: 0;
}
```

Ovšem mnohem důležitější je už nastavení samotného prvku seznamu (*LI*). Tomu se musí nastavit obtékání (*float*), šířka a další vlastnosti. Jako další nastavíme také barvu odkazů.

```
#prikladmenu ul li {
list-style-type: none;
margin: 0;
padding: 0;
display: block;
float: left;
width: 120px;
background-color: #7d9cdf;
text-align: center;
line-height: 30px;
}
#prikladmenu ul li a {
display: block;
}
#prikladmenu ul li a:hover {
```

```
background-color: #99b2df;
}
```

Pokud by se nejednalo o více úrovněvé menu, bylo by v tuto chvíli vystaráno. Ale jelikož se nám jedná o vysouvací menu, musíme ještě nastavit vlastnosti podpoložek.

```
#prikladmenu ul ul {
    margin: 0;
    padding: 0;
    border: 0;
    width: 120px;
    display: none;
    position: relative;
    left: -1px;
}
#prikladmenu ul ul li {
    /*float: none;*/
    clear: both;
    border-left: 1px black solid;
    border-bottom: 1px black solid;
    border-right: 1px black solid;
    display: block;
}
```

Proč je zakomentované přepsání vlastnosti *float* je vysvětleno v následující podkapitole. V tuto chvíli máme vše připraveno a zbývá pouze ošetřit jQuery, což nebude nijak složité. Stačí napojit události *mouseenter* a *mouseleave* (viz další podkapitola) a přidat efekt *slideDown()*, příp. *slideUp()*.


```
$("#prikladmenu ul li").bind("mouseenter", function() {  
    $(this).children("ul").css("top", "30").slideDown("slow");  
});  
  
$("#prikladmenu ul li").bind("mouseleave", function() {  
    $(this).children("ul").stop(true, true).slideUp();  
});
```

Princip tohoto příkladu je v podstatě jednoduchý. Pomocí pseudoproměnné *this* se odkazujeme na aktuální položku, přes kterou jsme přejeli myší, případně kterou jsme opustili. Funkce *children()* nám zajistí vybrání správného podelementu typu *UL*.

V ošetření události *mouseenter* následuje po použití funkce *children()* pouze zobrazení podpoložek pomocí efektu *slideDown()*.

V události *mouseleave* je však před efektem zasunutí (*slideUp*) ještě funkce *stop()*. Pomocí této funkce a obou jejích parametrů nastavených na *true*, menu ochráníme před zahlcením fronty událostí při rychlém přejíždění myší přes položky menu.

Největší problém při tvorbě pěkného menu, využívajícího jQuery efekty je správně nastavit kaskádové styly a proto byly probrány v úvodu příkladu.

6.11.4.1 Problémy při tvorbě vysouvacího menu

V následujících podkapitolách jsou popsány některé problémy, na které byste mohli také narazit.

Internet Explorer a absolutní pozicování

Položkám menu, které se vysouvají, je nutné nastavit absolutní pozicování. Kdyby tomu tak nebylo, posouval by se obsah pod nimi, při jejich vysouvání. Při nastavení absolutního pozicování podpoložkám a relativního hlavním položkám vše fungovalo správně ve všech prohlížečích, mimo Internet Exploreru.

Díky tomu bylo nutné nastavit, před efektem vysunutí, pozici top na 30px, což je výška hlavní položky. Díky tomu už začalo menu správně fungovat i v Internet Exploreru.

Internet Explorer a CSS float

Zvláštní chybu při tvorbě vysouvacího menu vykazoval Internet Explorer. Ačkoliv všechny ostatní prohlížeče (Opera, Google Chrome, FireFox i Safari) zobrazovali podpoložky menu správně, v Internet Exploreru byly podpoložky od sebe odsazené. A to i přesto, že kaskádové styly *margin* i *padding* byly vynulované.

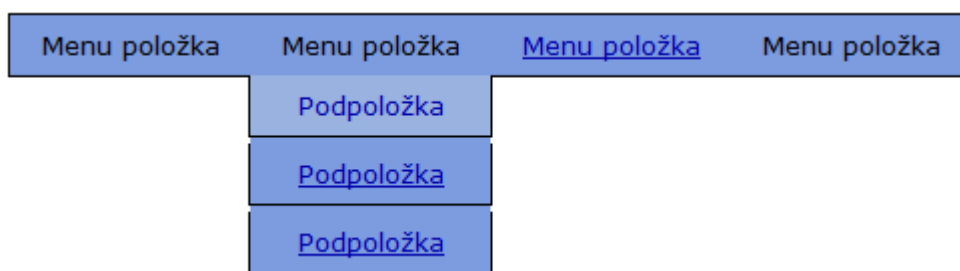
Stačilo však použít Developer toolbar, který je zabudován v Internet Exploreru a otestovat vliv nastavených kaskádových stylů.

Jelikož hlavní položky menu mají být zobrazeny vedle sebe, bylo jim nastaveno obtékání, zatímco jejich podpoložky mají být pod sebou a jelikož by obtékání zdědily, muselo jim být přepsáno vlastností, která obtékání zakazuje.

Použito bylo pro jistotu *float:none* i *clear:both*. Avšak právě *float:none* způsobilo v Internet Exploreru problémy. Stačilo tuto vlastnost odstranit. Následným otestováním se ukázalo, že dědění *float:left* nevádí, protože *clear:both* obtékání přesto zakáže.



Obrázek 2 - Float:left (prohlížeč Opera)



Obrázek 3 - Float:left (prohlížeč Internet Explorer)

MouseOver vs. MouseEnter

Ve všech případech, kdy bylo třeba ošetřit událost při přejetí myši přes nějaký element, jsem používal událost *mouseover* a *mouseenter* jsem si nevšímal. Problém však nastal právě při tvorbě vysouvacího menu.

Projevoval se tak, že i když byla událost vysunutí (a zasunutí) napojena na prvek *li*, přesto se při pohybu v rámci jeho oblasti spouštěla událost znovu. Stačilo, aby se myš přesunula na "Podpoložku" menu, neboli další prvek *li*, který však stále spadá do oblasti rodičovského *li*, a menu se zasunulo a poté znovu vysunulo.

Tuto chybu však lze jednoduše odstranit. Stačí, když se podíváte do dokumentace, do sekce s událostmi (Events). Naleznete tam výše zmiňovanou událost, která je na první pohled shodná s *mouseover*. Když si však prohlédnete příklad, který na stránce s událostí *mouseenter* naleznete²⁰, zjistíte, v čem je rozdíl.

Zatímco událost *mouseenter* je spuštěna pouze při vstupu do oblasti daného elementu, *mouseover* je spuštěna několikrát, i když ve skutečnosti prostor elementu neopustíte. Událost *mouseover* je také na rozdíl od *mouseenter* vyvolána i v případě přesunutí se na potomka elementu, na který je událost napojena.

²⁰ <http://api.jquery.com/mouseenter/>

Safari a uskakování hlavních položek

Bohužel se mi nepodařilo odstranit "uskakování" hlavních položek ze Safari. V žádném jiném prohlížeči tento "efekt" není. Pouze v prohlížeči Safari, když se vysouvají podpoložky, text hlavní položky "uskočí" o pár pixelů doleva.

6.12 Práce s formuláři v jQuery

Framework jQuery obsahuje také funkce, které můžeme využít pro formuláře. Některé však už byly vysvětleny v jiných kapitolách – jedná se například o funkci `val()`, která vrací hodnotu například textového pole nebo událost `blur()`.

Můžeme také použít funkci `submit()`, která odešle formulář (nemusíte tedy čekat na odeslání pomocí tlačítka typu `submit`) anebo funkci `focus()`. Avšak funkce `submit()` i `focus()` zastávají roli také události a to tehdy, pokud jako parametr použijeme callback funkci.

Funkci `submit()` tak můžeme použít buď k vyvolání odeslání formuláře, nebo jako událost, která reaguje na odeslání formuláře. Funkci `focus()` můžeme použít pro přepnutí na vybraný element nebo jako událost, která reaguje, pokud se na daný element přepnete.

```
.submit( funkce(udalost) ) // událost odeslání formuláře
.submit()                 // odeslání formuláře
.focus( funkce(udalost) ) // událost přepnutí se na element
.focus()                  // přepnutí na element
```

Při vstupu na stránku s přihlašovacími údaji tak můžeme například rovnou vybrat textové pole, kde se zadává přihlašovací jméno a tím urychlit přihlášení, protože uživatel nebude muset hledat formulář a hned může začít psát přihlašovací jméno.

```
// Příklad vybrání přihlašovacího jména
$("#login").focus();
```

6.13 Úpravy kódu stránky pomocí jQuery (Manipulation)

Práce s elementy, které jsou již v kódu stránky, Vám určitě časem nebude postačovat. Někdy je nutné vložit do stránky nějaký kód/element a nebo ho naopak odebrat.

K tomuto účelu slouží v jQuery funkce, které jsou popsány v dokumentaci v části nazvané Manipulation²¹.

Tyto funkce umožňují přidávat, upravovat nebo odebírat HTML kód stránky v reálném čase, tedy během používání stránky uživatelem.

Příkladem mohou být funkce `append()`, `prepend()`, `html()` a `text()`.

```
append( html_kod )
prepend( html_kod )
html( html_kod )
text( html_kod )
```

6.13.1 Různé možnosti vkládání HTML kódu

Nejdříve bude představena funkce `append()`. Tato funkce přidá zvolený obsah (HTML kód) na konec elementu, který byl vybrán. Takže například²² pokud máme takovýto HTML kód:

```
<h2>Uvitání</h2>
<div class="obsah">
  <div class="uvnitr">Ahoj</div>
  <div class="uvnitr">Čau</div>
</div>
```

A použijeme funkci `append()` následovně:

```
$( '.uvnitr' ).append( '<p>Test</p>' );
```

²¹ <http://api.jquery.com/category/manipulation/>

²² Příklad je převzat ze stránky <http://api.jquery.com/append/>

Výsledek bude vypadat takto:

```
<h2>Uvítání</h2>
<div class="obsah">
  <div class="uvnitř">
    Ahoj
    <p>Test</p>
  </div>
  <div class="uvnitř">
    Čau
    <p>Test</p>
  </div>
</div>
```

Funkce *prepend()* obsah vloží naopak na začátek elementu, na který je funkce aplikována.

Mohli byste se zeptat, jak vložit nějaký kód za zvolený element a ne na jeho konec. Na tuto otázku je jednoduchá odpověď: "Použít funkci *before()* nebo *after()*."

```
before( html_kod )
after( html_kod )
```

6.13.2 Formátovaný a neformátovaný HTML kód

Funkce *html()* a *text()* fungují na trochu jiném principu. Pracují totiž s obsahem celého vybraného elementu a ten Vám buď mohou vypsát, nebo ho pomocí nich můžeme upravit.

Obě funkce jsou na první pohled stejné. Je v nich pouze jeden rozdíl. Jak už možná tušíte, funkce *html()* vrací obsah elementu včetně HTML značek – tedy například **, ** apod., ale *text()* vrací pouze čistý text – a to i kdyby původní obsah elementu nějaké značky obsahoval.

6.13.3 Příklad 1 – Vypsání obsahu elementu

Nejdříve si v praxi ukážeme rozdíl mezi funkcemi `html()` a `text()`. Vytvoříme si následující odstavec, kterému přiřadíme identifikátor (*ID*) a jeho obsah budeme vypisovat do vyskakovacího okna. Je důležité, aby odstavec obsahoval i HTML tagy, jinak by rozdíl mezi funkcemi nebyl zřejmý.

```
<p id="odstavec_vypsat">Text text text text text <strong>TUČNÝ  
TEXT</strong> text text text text</p>
```

Protentokrát budeme vypisovat pomocí tlačítek místo odkazů. Pro tuto funkčnost to vypadá lépe a nebudeme muset přidávat do události další řádek v podobě zakázání přesunutí na odkaz (`return false`).

```
<input type="button" name="vypsat_text" id="vypsat_text"  
value="Vypsat obsah pomocí text()" />  
<input type="button" name="vypsat_html" id="vypsat_html"  
value="Vypsat obsah pomocí html()" />
```

Následně už pouze ošetříme tlačítka událostmi, které reagují na kliknutí a použijeme funkce `text()` nebo `html()` pro získání obsahu odstavce.

```
$("#vypsat_text").bind("click", function() {  
    alert($("#odstavec_vypsat").text());  
});  
  
$("#vypsat_html").bind("click", function() {  
    alert($("#odstavec_vypsat").html());  
});
```

V otevřeném okně s vypsánými řetězci je názorně vidět rozdíl mezi funkcemi.

6.13.4 Příklad 2 – Zmizení tlačítka, zobrazení textu

Na tomto jednoduchém příkladu si ukážeme způsob, jak můžeme zamezit vícenásobnému hlasování, opětovnému odeslání formuláře a určitě by se proto našla další spousta využití.

Příklad bude opět v principu jednoduchý. Po kliknutí na tlačítka dojde k jeho skrytí a zobrazí se místo toho námi definovaný text.

Skrytí tlačítka provedeme pomocí funkce `remove()`, o které jsme zatím nemluvili, ale spadá stejně jako například `append()` a `html()` a pod kategorií Manipulation a slouží tedy k úpravě kódu stránky.

Do HTML kódu stránky si vložíme jednoduché tlačítko a `div`, který bude ze začátku skrytý a zobrazíme ho až po skrytí tlačítka. Skrytý `div` bude mít své `id` a také třídu `neviditelný`, pro kterou je definováno v CSS `display: none`.

```
<input type="button" name="hlasovat" id="hlasovat"
value="Hlasovat" />
<div id="dekujeme_hlasovani" class="neviditelný">Děkujeme za
hlasování.</div>
```

Nic víc není potřeba. A jediné, co obstará jQuery, je právě skrytí tlačítka `hlasovat` pomocí funkce `remove()` a zobrazení divu pomocí funkce `show()`.

```
$("#hlasovat").bind("click", function() {
    $("#hlasovat").remove();
    $("#dekujeme_hlasovani").show();
});
```

Samozřejmě by bylo možné tento příklad napsat různými způsoby. Bylo by možné využít efektů a tlačítko skrýt pomocí například `fadeOut()` a div zobrazit pomocí `fadeIn()`. Také by bylo možné tlačítko pouze zakázat díky přidání atributu `disabled` pomocí funkce `attr()`.

Anebo byste `div` (či jakýkoliv jiný element s textem) nemuseli vůbec mít v HTML kódu stránky a přidávat jej až po kliknutí na tlačítko `hlasovat` pomocí funkce `append()`. Možností je celá řada.

6.13.5 Příklad 3 – Příprava formuláře pro nahrání souboru na internet

Tento formulář bude představovat pouze základ pro skript určený k nahrání souboru na webové stránky – skript ve skutečnosti nic nahrávat nebude. Pouze si

ukážeme, jak jednoduše lze zajistit přidání nových políček pro nahrání více souborů najednou. Tento postup by pak bylo možné aplikovat i třeba na přikládání příloh k e-mailu.

Opět začneme HTML kódem, který bude tentokrát o něco složitější. Vytvoříme si formulář, který nastavíme tak, jak by bylo potřeba ve skutečném formuláři pro nahrání souboru - nastavíme `enctype="multipart/form-data"`. Parametr `method` nastavíme na `post` a `action` vyplníme pouze křížkem (#).

Aby byl příklad o trochu složitější, stanovíme si, že souborů může být maximálně 5 a u každého souboru bude napsáno, kolikátý je v pořadí.

```
<form action="#" method="post" enctype="multipart/form-data">
  <em>Maximálně 5 souborů.</em>
  Soubor 1: <input type="file" name="soubory[]" id="soubor1"
class="soubory" />
  <a href="#" id="pridat_soubor">Přidat další soubor</a>
</form>
```

Je důležité, aby jméno (name) políčka pro výběr souboru obsahovalo hranaté závorky. Pokud by se jmenovalo pouze `soubory`, nebylo by možné nahrát více souborů, protože by se pomocí POSTu předal pouze jeden soubor. Při použití hranatých závorek se předává celé pole souborů.

Nyní se podíváme na zdrojový kód JavaScriptu. Veškeré přidávání a ošetřování množství souborů se bude provádět v události `click()` tlačítka `pridat_soubor`. Jelikož používáme odkaz, na který se ale nechceme přesměrovat, opět využijeme `return false`.

```
$("#pridat_soubor").bind("click", function() {
  // kód
  return false;
});
```

Počet souborů/políček lze snadno získat pomocí JavaScriptové vlastnosti *length*, kterou aplikujeme na výběr všech souborů. Abychom je snadno identifikovali, je každému políčku přiřazena třída *soubory*.

```
$("#pridat_soubor").bind("click", function() {  
    var soubory = $(".soubory");  
    var pocetSouboru = $(soubory).length;  
    // kód  
    return false;  
});
```

Po přidání podmínky *if...else* a vyskakovacího okna tak už máme ošetřen maximální počet souborů.

```
$("#pridat_soubor").bind("click", function() {  
    var soubory = $(".soubory");  
    var pocetSouboru = $(soubory).length;  
    if (pocetSouboru < 5) {  
        // kód  
    }  
    else {  
        alert("Maximální počet souborů je 5.");  
    }  
});
```

Nyní už stačí pouze ošetřit přidání HTML kódu do stránky. HTML kód si sestavíme přesně tak, jak má vypadat ve výsledku, pouze vyměníme číslo souboru za proměnnou *aktualniCislo*, kterou si v sekci *if* vytvoříme. Do HTML kódu si nový element vložíme pomocí funkce *after()*.

Funkce *after()* umožňuje vložit libovolný HTML kód za zvolený element. Nelze zde použít funkci *append()*, protože ta vkládá HTML kód do zvoleného elementu.

```
$("#pridat_soubor").bind("click", function() {
    var soubory = $(".soubory");
    var pocetSouboru = $(soubory).length;
    if (pocetSouboru < 5) {
        var aktualniCislo = (pocetSouboru*1)+1;

        var html = 'Soubor '+aktualniCislo+' : <input
type="file" name="soubory[]" id="soubor'+aktualniCislo+'
class="soubory" />';

        $("#soubor"+pocetSouboru).after(html);
    }
    else {
        alert("Maximální počet souborů je 5.");
    }
    return false;
});
```

Tímto by byl připravený formulář pro nahrání max. 5 souborů na nějaký web. Pro úplnou funkčnost už stačí pouze přidat skript pro nahrání souboru. Ten už sice překračuje rámec této práce, ale můžete se inspirovat skriptem v PHP pro nahrání souboru na web od Forpsi²³. Jedná se ale o nahrání pouze jednoho souboru, takže by bylo nutné skript upravit pro podporu více souborů.

6.14 JQuery a pozicování elementů (Offset)

V kapitole o rozměrech v jQuery bylo poukazováno na to, že někdy je nutné zeptat se jQuery, jakou má element aktuálně šířku/výšku, a přes kaskádové styly to nejde zjistit.

Kromě této situace nastává podobná a to ta, že potřebujeme zjistit, kde se prvek nachází. Jeho pozice může být ovlivněna vlastnostmi *padding*, *margin*, anebo absolutním či relativním pozicováním. Navíc různé prohlížeče si započítávají různé vzdálenosti.

²³ <http://kb.forpsi.com/article.php?id=372>

JQuery pomocí funkce `offset()` umožňuje zjistit aktuální polohu a my pomocí tohoto údaje můžeme upravit vzdálenost elementu přesně tak, jak to potřebujeme. Funkce `offset()` vrací dva údaje: `top` a `left`, ke kterým se lze dostat skrze "tečku". Vlastnost `top` je vzdálenost od horního okraje stránky a tedy ekvivalent souřadnice y a `left` je ekvivalentem souřadnice x.

6.14.1 Příklad 1 – Vypsání umístění elementu

Následující příklad bude jednoduchou ukázkou, jak vypsát umístění elementu na stránce. Využijeme k tomu opět obrázek smajlíka a tlačítko pro zobrazení informace o pozici.

```
<div id="prikkladdiv">
  
</div>
```

Abyste názorně viděli, jak se pozice elementu mění, vytvoříme také další dvě tlačítka, která budou v rámci jednoho elementu `DIV` obrázek posunovat doprava a doleva.

```
<input type="button" name="vypsat_umisteni"
id="vypsat_umisteni" value="Vypsát umístění" />
<input type="button" name="posunout_doleva"
id="posunout_doleva" value="Posunout doleva" />
<input type="button" name="posunout_doprava"
id="posunout_doprava" value="Posunout doprava" />
```

V JavaScriptu využijeme následujícího kódu:

```
$("#posunout_doleva").bind("click", function() {
    $("#smajlik3").animate({left: '-=30'});
});
$("#posunout_doprava").bind("click", function() {
    $("#smajlik3").animate({left: '+=30'});
});
$("#vypsat_umisteni").bind("click", function() {
```

```
var offset = $("#smajlik3").offset();
alert("Top: " + offset.top + "\nLeft: " + offset.left);
});
```

Funkce tlačítka *vypsat_umisteni* není nijak složitá. Pouze si načteme umístění (offset) našeho elementu, konkrétně elementu *smajlik3* a jeho vlastnosti *top* a *left* vypíšeme do vyskakovacího okna (alertu).

Tlačítka *posunout_doleva* a *posutnout_doprava* jsou založené opět na události *click()* a využívají efektu *animate()*. Tato funkce totiž umožňuje posouvat element, aniž bychom znali jeho původní pozici a navíc je pohyb plynulý a pěkný. Kvůli jednomu kaskádovému stylu sice využití této funkce dělalo v Opeře problémy, ale problém byl vyřešen a v následující podkapitole je vysvětlen.

Díky tomu, že elementu *DIV* je nastaveno *overflow: hidden*, nebude obrázek při posouvání vyjíždět ze stránky a místo toho se bude ve chvíli, kdy opustí *DIV*, zasouvat pod stránku.

```
#prikladdiv {
  border: 1px #000 solid;
  overflow: hidden;
  position: relative;
}
```

6.14.1.1 Poznámka k příkladu

K testování všech příkladů je využíván jeden soubor s kaskádovými styly, který obsahuje jak styly nutné k správnému zobrazení stránky, tak styly pro jednotlivé příklady.

V tomto příkladu bylo nutné přidat do definice stylu *prikladdiv* ještě relativní pozicování, protože jinak se z nějakého důvodu aplikoval na obrázek *margin* právě z definice *#stranka*, což je *DIV*, který obklopuje celou stránku.

Obrázek pak při prvním požadavku o posunutí doprava, případně doleva, nejdříve odskočil přibližně o 260px požadovaným směrem a až poté se plynule posunul o zvolený počet pixelů. Tato chyba se projevovala pouze v Opeře.

6.15 Užitečné funkce jQuery

Framework jQuery nedisponuje pouze funkcemi pro tvorbu efektů nebo jednodušší správu obsahu webové stránky. Vylepšuje také některé funkce klasického JavaScriptu a některé přidává.

Tyto funkce lze používat pomocí tečky, kterou umístíte za znak dolaru nebo řetězec jQuery.

6.15.1 Práce s řetězcem

Ať už jste se setkali s programovacím jazykem PHP, C# nebo Javou, ve všech zmíněných jazycích určitě využíváte pro zpracování řetězce funkci `trim()`. Tato funkce ořízne řetězec o počáteční a koncové mezery, které mohou způsobovat chyby, nepřesnosti nebo špatné zobrazování.

Klasický JavaScript funkci `trim()` nemá a je tak nutné ji doprogramovat²⁴.

```
String.prototype.trim = function()
{
    return this.replace(/(^\\s*)|(\\s*$)/g, "");
};

// Použití
vysledek = vysledek.trim();
```

I když kód není nijak složitý, určitě je vhodnější použít jednoduchou, vestavěnou, funkci frameworku jQuery o jejíž implementaci se nemusíte starat.

```
jQuery.trim( retezec )
```

²⁴ <http://diskuse.jakpsatweb.cz/?action=vthread&forum=8&topic=33574>

Funkce `trim()` má jediný parametr, kterým je řetězec, který chceme zbavit počátečních a koncových mezer. Následně je nám vrácen upravený řetězec.

6.15.2 Příklad 1 – Použití funkce `trim()`

Pro ukázkou použití funkce `trim()`, vytvoříme stránku, kde bude jedno textové pole a jedno tlačítko. Po kliknutí na tlačítko bude do vyskakovacího okna vypsán řetězec, který uživatel napíše do textového pole a také jeho "otrimovaná" verze.

HTML kód:

```
<input type="text" name="text" id="text" value="" />
<input type="button" name="otrimovat_text" id="otrimovat_text"
value="Otrimovat text" />
```

Kód JavaScriptu:

```
$("#otrimovat_text").bind("click", function() {
    var retezec = $("#text").val();
    var otrimovatnyRetezec = jQuery.trim(retezec);
    alert("Řetězec: \n" + retezec + "\n" +
        "\nOtrimovaný řetězec: \n" + otrimovatnyRetezec);
});
```

Přes funkci `val()` je načten řetězec, který uživatel zadal. Řetězec je otrimován a obě verze (původní i upravená) jsou pak vypsány do vyskakovacího okna.

6.15.3 Práce s poli a kolekcemi

Klasický JavaScript má také nedostatky při práci s polem. Nelze jednoduše zjistit, jestli daná proměnná je pole, jestli v ní je nebo není nějaký prvek apod. JQuery také umožňuje z pole odstranit duplicitní záznamy a také ho projít.

V následujících několika odstavcích budou popsány funkce, které výše zmíněné dovednosti mají a jsou součástí jádra JQuery.

Hledání v poli

Hledat v poli můžeme díky funkci `inArray()`. Funkce přijímá dva povinné parametry, přičemž prvním je hodnota, kterou v poli hledáme a druhým je pole, ve kterém se bude hledat.

```
jQuery.inArray( hodnota, pole )
```

Testování pole

Pomocí funkce `isArray()` můžeme zjistit, jestli proměnná (objekt) je pole. Podle toho se můžeme zachovat a provést nějakou akci. Tato funkce má pouze jediný (povinný) parametr a tím je objekt, který chceme otestovat.

```
jQuery.isArray( objekt )
```

Foreach v jQuery

Framework jQuery má ve svém jádru také funkci, která plní úlohu cyklu `foreach`. Název této funkce je `each()` a má dva povinné parametry, přičemž prvním je kolekce hodnot. Kolekce proto, že se nemusí jednat přímo o pole, ale stačí právě kolekce klíčů a hodnot. Druhým je funkce, která představuje tělo cyklu.

```
jQuery.each(kolekce, navratovaFunkce(indexPrvku, hodnotaPrvku))
```

6.15.4 Příklad 2 – Práce s poli

Tento příklad Vám ukáže výše popisované funkce v praxi. V JavaScriptu budou definovány dvě proměnné, se kterými se bude pracovat po kliknutí na tlačítko. Proměnné se nejdříve otestují, jestli se jedná o pole, poté se proměnná typu pole vypíše a nakonec se využije textové pole, ze kterého si načteme hodnotu a tu se pokusíme, v proměnné typu pole, vyhledat.

Začneme klasicky HTML kódem. Do něj si vložíme textové pole a tlačítko.


```
<input type="text" name="prvek_v_poli" id="prvek_v_poli"
value="" />
<input type="button" name="pole_vypis" id="pole_vypis"
value="Výpis" />
```

JavaScriptový kód bude o něco delší. Nejdříve si definujeme dvě proměnné. Jedna bude typu pole, druhá bude jiného typu – například řetězec (string).

```
var pole_promennal = new Array("červená", "modrá", "oranžová",
"růžová", "fialová");
var pole_promenna2 = "Řetězec";
```

Po kliknutí na tlačítko si začneme sestavovat závěrečnou zprávu, která bude na konci události vypsána do vyskakovacího okna.

```
$("#pole_vypis").bind("click", function() {
    var zprava = "";
    // ...
    alert(zprava);
});
```

Prvním úkolem bude zjistit, jestli některá z proměnných je typu pole. Tuto vlastnost zjistíme díky funkci `isArray()`.

```
$("#pole_vypis").bind("click", function() {
    var zprava = "";

    // Jedná se pole?
    zprava += "Proměnná pole_promennal je pole: " +
jQuery.isArray(pole_promennal) + "\n";
    zprava += "Proměnná pole_promenna2 je pole: " +
jQuery.isArray(pole_promenna2) + "\n";
    zprava += "\n";

    alert(zprava);
});
```

Druhým krokem bude vypsání proměnné typu pole pomocí funkce `each()`.

```
$("#pole_vypis").bind("click", function() {
    var zprava = "";

    // Jedná se pole?
    zprava += "Proměnná pole_promennal je pole: " +
jQuery.isArray(pole_promennal) + "\n";
    zprava += "Proměnná pole_promenna2 je pole: " +
jQuery.isArray(pole_promenna2) + "\n";
    zprava += "\n";

    // Výpis
    zprava += "--- Výpis pole ---\n";
    zprava += "pole_promennal: \n{ ";
    jQuery.each(pole_promennal, function(index, hodnota) {
        if (index > 0){
            zprava += ", ";
        }
        zprava += hodnota;
    });
    zprava += " }\n\n";

    alert(zprava);
});
```

Nakonec použijeme textové pole, které jsme vytvořili. Do něj uživatel může napsat nějakou hodnotu, která se následně pokusí vyhledat v proměnné typu pole. Vyhledávání v poli nám zajistí funkce *isArray()*.

```
$("#pole_vypis").bind("click", function() {
    var zprava = "";

    // Jedná se pole?
    zprava += "Proměnná pole_promennal je pole: " +
jQuery.isArray(pole_promennal) + "\n";
    zprava += "Proměnná pole_promenna2 je pole: " +
jQuery.isArray(pole_promenna2) + "\n";
    zprava += "\n";
```

```
// Výpis
zprava += "--- Výpis pole ---\n";
zprava += "pole_promennal: \n{ ";
jQuery.each(pole_promennal, function(index, hodnota) {
    if (index > 0){
        zprava += ", ";
    }
    zprava += hodnota;
});
zprava += " }\n\n";

// Existence prvku v poli?
var prvek = $("#prvek_v_poli").val();
zprava += "--- Existence prvku [" + prvek + "] v poli ---
\n";
zprava += "pole_promennal: " + jQuery.inArray(prvek,
pole_promennal) + "\n";
zprava += "\n";

alert(zprava);
});
```

6.15.5 Zjištění použitého prohlížeče

Webové stránky by měly být tvořeny tak, aby fungovaly stejně ve všech prohlížečích. Prohlížeče to však tvůrcům stránek příliš nezjednodušují a tak je někdy nutné provést v kódu jiný postup pro jeden prohlížeč a jiný pro ostatní. Někdy to ale může být pouze potřeba zachovat se podle toho, jaký prohlížeč uživatel používá a podle toho mu například do hlavičky zobrazit ikonku ohnivě lišky, pokud používá Firefox a podobně.

Klasický JavaScript sice také umí poznat prohlížeč, jeho verzi a operační systém, ale v jQuery je zjištění těchto informací jednodušší. Stačí nám k tomu jediná vlastnost, která se jmenuje *browser*.

```
jQuery.browser
```

Přes vlastnost `browser` se můžeme ptát, jestli uživatel používá námi zvolený prohlížeč, přičemž nám je vráceno buď `true` nebo `false`. Nevýhodou je právě nutnost se zeptat přímo na prohlížeč místo toho, aby nám byl rovnou vrácen prohlížeč, který uživatel používá. Pokud bychom chtěli zjistit, jestli uživatel používá Internet Explorer, udělali bychom to následovně.

```
if (jQuery.browser.msie) {  
    alert("Vítej uživateli Internet Exploreru.");  
}
```

Podporované parametry

```
jQuery.browser.webkit  
jQuery.browser.safari // Prohlížeč Safari  
jQuery.browser.opera // Prohlížeč Opera  
jQuery.browser.msie // Prohlížeč Internet Explorer  
jQuery.browser.mozilla // Prohlížeč Mozilla Firefox
```

6.15.6 Příklad 3 – Zjištění použitého prohlížeče

Jak jinak otestovat funkčnost vlastnosti `browser`, než právě zjištěním uživatelského prohlížeče. Tento příklad, po kliknutí na tlačítko, vypíše uživateli pozdrav, který je závislý na používaném prohlížeči.

V HTML kódu budeme mít pouze tlačítko. Nic jiného není třeba.

```
<input type="button" name="zjistit_prohlizec"  
id="zjistit_prohlizec" value="Zjistit prohlížeč" />
```

JavaScript ale také nebude příliš složitý. Jediné co musíme udělat je, že postupně otestujeme všechny prohlížeče, které jQuery umí rozeznat a těm zbylým oznámíme, že je nedokážeme rozpoznat.

```
$("#zjistit_prohlizec").bind("click", function() {
    if (jQuery.browser.msie) {
        alert("Vítej uživateli Internet Exploreru.");
    }
    else if (jQuery.browser.opera) {
        alert("Vítej uživateli Opery.");
    }
    else if (jQuery.browser.mozilla) {
        alert("Vítej uživateli Mozilly.");
    }
    else if (jQuery.browser.safari) {
        alert("Vítej uživateli Safari.");
    }
    else {
        alert("Vítej uživateli. Bohužel nemůžu identifikovat
tvůj prohlížeč.");
    }
});
```

Poznámka

Jelikož jQuery neumí prozatím rozpoznat prohlížeč Google Chrome, vypíše při jeho použití stejný pozdrav, jako kdyby uživatel použil prohlížeč Safari.

6.16 AJAX a jQuery

Kromě efektů je další významnou funkcí jQuery jeho pohodlná a jednoduchá práce s AJAXem. AJAX znamená anglicky Asynchronous JavaScript and XML neboli asynchronní JavaScript a XML. Pokud bychom to chtěli jednoduše vysvětlit, tak umožňuje zavolání nějakého skriptu (PHP, ASP) bez nutnosti znovu načítat stránku, jako je tomu například u odesílání formulářů.

Klasický JavaScript potřebuje pro správnou funkčnost AJAXu napříč nepoužívanějšími prohlížeči spoustu kódu, ale v jQuery si vystačíme s dvěmi jednoduchými funkcemi. Jedná se o funkce `load()` a `ajax()`. Každá však pracuje na mírně odlišném principu.

6.16.1 Rozdíly mezi funkcemi `load()` a `ajax()`

Prvním velkým rozdílem je, že funkci `load()` použijeme pro načtení obsahu nějakého elementu. Tuto funkci při použití musíte aplikovat na nějaký element a po jejím úspěšném dokončení je obsah elementu nahrazen obsahem, který se načetl ze skriptu.

Oproti tomu funkce `ajax()` slouží k provedení nějakého skriptu, přičemž nejste vázáni na nějaký element. Po úspěšném dokončení můžeme otevřít nějaké okno, vypsát zprávu o tom, že vše proběhlo v pořádku, vymazat, či přidat nějaké elementy a spoustu dalšího.

Funkce `ajax()` má také na rozdíl od funkce `load()` více parametrů. Můžeme například zakázat cache, ale hlavně můžeme definovat typ dat, se kterými se pracuje. Ze skriptu tak výsledek může být například ve formátu XML nebo JSON. Oběma funkcím lze nastavit funkci, která se provede v případě úspěchu, ale pouze `ajax()` umožňuje nastavit funkci, která se provede v případě neúspěchu.

6.16.1.1 Funkce `load()`

```
load( adresa, [parametry], [funkce_dokonceno] )
```

Parametr `parametry` umožňuje předat adrese parametry. Pokud se jedná například o PHP skript, můžeme toho využít a pomocí parametrů například volit, co se stane.

Funkce `funkce_dokonceno()` se provede po úspěšném zavolání a provedení adresy. Můžeme v ní například vypsát zprávu o dokončeném procesu.

6.16.1.2 Funkce `ajax()`

```
jQuery.ajax( nastaveni )
```

Funkce `ajax()` už má více možností než `load()`. Parametry předáváte ve složených závorkách a to například následovně:

```
jQuery.ajax({
    url: "skript.php",
    type: "POST",
    data: "ukazka=1",
    cache: false,
    dataType: "xml",
    success: function(vracenaData) {
        alert(vracenaData);
    }
});
```

V předcházející ukázce jsme zavolali soubor skript.php, pomocí metody POST jsme mu předali parametr *ukazka* s hodnotou *1*, zakázali jsme cachování, data, která bude skript vracet, budou ve formátu XML a v případě úspěšného dokončení se vypíše do vyskakovacího okna data, která vrátil skript.

Funkce *ajax()* může ale mít ještě více parametrů. Příkladem může být funkce, která se spustí v případě chyby – parametr *error*, časový limit, který je stanoven pro provedení skriptu – parametr *timeout* a spousty dalších

6.16.2 Příklad 1 – Nahrání obsahu select boxu

Nejdříve si ukážeme možnost, jak lze pracovat v jQuery s funkcí *load()*. Na stránce budou dva select boxy, které mohou být například součástí registračního formuláře a s jejich pomocí by tvůrci internetových stránek zjišťovali, odkud se o nich uživatel dozvěděl.

Druhý select box bude měnit své hodnoty v závislosti na vybrané možnosti z prvního selectu. Hodnoty z prvního select boxu bude zpracovávat AJAXově zavolaný skript *ajax_load.php*.

Začneme tedy s HTML kódem. Do něho si vložíme label, který bude popisovat, co uživatel vybírá a také si naplníme první select box základními hodnotami. Pro účely příkladu bylo zvoleno, že uživatel se mohl o stránce dozvědět z televize, rádia nebo internetu.

```
<label for="odkud">Odkud jste se o nás dozvěděli?</label>
<select name="odkud_select" id="odkud_select">
  <option value="tv">Z televize</option>
  <option value="radio">Z rádia</option>
  <option value="internet">Z internetu</option>
</select>
```

Druhý select box se sice bude naplňovat pomocí AJAXu, ale nebylo by hezké ani praktické, kdyby při načtení stránky byl prázdný a tak mu vyplníme stejné hodnoty, jako kdyby byla vybrána první položka v prvním select boxu.

```
<select name="odkud_select2" id="odkud_select2">
  <option value="ct1">ČT 1</option>
  <option value="nova">Nova</option>
  <option value="prima">Prima</option>
</select>
```

PHP skript také naleznete na příloženém CD, jeho kód ale není nijak složitý. Obsahuje pouze otestování, jestli je předán parametr *vybrano*, pomocí kterého se určí obsah druhého select boxu. Rozhodování se provádí v rozhodovací podmínce *switch*, kde se v závislosti na vybrané hodnotě sestaví požadovaný HTML kód. Nakonec je HTML kód vypsán pomocí *echo*.

```
<?php

$html = "";
if (isset($_GET["vybrano"])) {
    switch ($_GET["vybrano"]) {
        case "tv":
            $html .= '<option value="ct1">ČT 1</option>';
            $html .= '<option value="nova">Nova</option>';
            $html .= '<option value="prima">Prima</option>';
            break;
        case "radio":
            $html .= '<option value="kiss">Kiss Jižní
Čechy</option>';
```



```

        $html .= '<option value="faktor">Hitrádio
Faktor</option>';
        $html .= '<option
value="evropa2">Evropa2</option>';
        break;
        case "internet":
            $html .= '<option
value="google">Google</option>';
            $html .= '<option
value="seznam">Seznam.cz</option>';
            $html .= '<option
value="centrum">Centrum.cz</option>';
            $html .= '<option value="idnes">iDNES</option>';
            break;
    }
}
echo $html;?>

```

Nyní se dostáváme k JavaScriptu. I přesto, že JavaScript musí zavolat PHP skript a zpracovat jeho výsledek, není dlouhý, ani složitý.

```

$("#odkud_select").bind("change", function() {
    $("#odkud_select2").load("ajax_load.php",
"vybrano="+$("#odkud_select").val(), function(odpoved, status,
XMLHttpRequest) {
        if (status == "error") {
            alert("Chyba: \n" + XMLHttpRequest.status + " "
+ XMLHttpRequest.statusText + "\n\n");
        }
    });
});

```

Základem je použití události *change()*, která reaguje na změnu výběru v select boxu. Následuje použití funkce *load()*. Do parametrů funkce vložíme nejdříve skript, který bude zpracovávat hodnotu prvního select boxu a vrátet HTML kód.

Druhým parametrem je nastavení proměnné *vybrano* na hodnotu, vybranou v první select boxu. Tato proměnná bude předána skriptu pomocí komunikace *GET*.

Posledním, třetím, parametrem funkce `load()` je funkce, která se zavolá po dokončení nebo nedokončení operace. V této funkci pouze vypíšeme chybovou zprávu o tom, co se nepodařilo v případě, že zpracování souboru neproběhne v pořádku.

O navrácení HTML kódu a jeho naplnění do druhého select boxu se už postará funkce `load()` a skript `ajax_load.php`.

6.16.3 Příklad 2 – Hlasování v anketě

Pro příklad použití funkce `ajax()` byla zvolena tematika hlasování v anketě, která už byla jednou zmíněna. Opět se nebude hlasování nikam ukládat, ale tentokrát půjde o reálné vybrání jedné z možností, následného zpracování pomocí AJAXu a nakonec vypsání poděkování a zobrazení možnosti, kterou uživatel vybral.

Jako první si vytvoříme v HTML kódu element `DIV` s identifikátorem `vyber_druhu_dovolene`. Do tohoto elementu vložíme poznámku, oč se jedná a hlavně také tři možnosti v podobě radio buttonů. Předvybraná bude první možnost.

```
<div id="vyber_druhu_dovolene">
  <strong>Zvolte Váš oblíbený typ dovolené:</strong><br />
  <input type="radio" name="dovolena" id="dovolena_more"
value="more" checked="checked" /><label
for="dovolena_more">Koupání v moři</label><br />
  <input type="radio" name="dovolena" id="dovolena_hory"
value="hory" /><label for="dovolena_hory">Hory a
turistika</label><br />
  <input type="radio" name="dovolena" id="dovolena_doma"
value="doma" /><label for="dovolena_doma">Relaxování v místě
bydliště</label><br />
  <input type="button" name="dovolena_zvolit"
id="dovolena_zvolit" value="Zvolit" />
</div>
```

Možnosti jsou obklopené elementem `DIV` z toho důvodu, aby se s nimi dalo snadno manipulovat a bylo možné je jednoduše odstranit a zobrazit požadovaný text.

Kód PHP skriptu je hodně podobný tomu z předcházejícího příkladu. S tím rozdílem, že proměnná se jmenuje volba a generujeme také jiný HTML kód.

```
<?php

$volba = "";
if (isset($_GET["vybrano"])) {
    switch ($_GET["vybrano"]) {
        case "more":
            $volba .= 'Koupání v moři';
            break;
        case "hory":
            $volba .= 'Hory a turistika';
            break;
        case "doma":
            $volba .= 'Relaxování v místě bydliště';
            break;
    }
}
echo $volba;

?>
```

JavaScriptový kód je o něco složitější než předcházející, ale na druhou stranu máme díky funkci ajax() větší možnosti nastavení.

```
$("#dovolena_zvolit").bind("click", function() {
    var hodnota = $("input[name=dovolena]:checked").val();

    jQuery.ajax({
        url: "ajax_ajax.php",
        data: "vybrano="+hodnota,
        cache: false,
        success: function(html) {
            if (html == "") {
                alert("Špatně jste zvolili hodnotu.");
            }
        }
    });
});
```

```
        else {
            alert("Děkujeme za hlasování.");

            $("#vyber_druhu_dovolene").before("<p>Zvolena možnost:
<strong>"+html+"</strong></p>");
            $("#vyber_druhu_dovolene").remove();
        }
    },
    error: function(XMLHttpRequest, textStatus,
errorThrown) {
        alert("Chyba.\n" + textStatus + " " +
errorThrown);
    }
});
});
```

Veškerý kód se nachází ve funkci, která ošetřuje událost kliknutí na tlačítko *zvolit*. Nejdříve je nutné zjistit, který radio button máme vybraný. To zjistíme pomocí jména radio buttonů a selektoru *:checked*.

```
var hodnota = $("input[name=dovolena]:checked").val();
```

Funkci *ajax()* nastavíme následující parametry:

- Skript, který budeme volat (parametr *url*)
- Hodnotu proměnné *vybrano*, která bude předána skriptu (*data*)
- Zakázanou cache (*cache*)
- Funkci, která se provede v případě úspěšného dokončení (*success*)
- Funkci, která se provede v případě neúspěšného dokončení (*error*)

Jelikož jsme nezvolili typ komunikace, bude použit *GET*. Zároveň jsme také nenastavili typ vrácených dat (*dataType*). V tomto případě jQuery rozezná typ automaticky a podle toho se zachová.

Funkce, která se provede v případě úspěšného dokončení, má jeden parametr, kterým je HTML kód vrácený ze skriptu. Pokud není HTML kód prázdný (vše proběhlo v pořádku), vypíšeme do vyskakovacího okna poděkování, před element

DIV s možnostmi vložíme název možnosti, kterou uživatel zvolil a následně *DIV* odstraníme.

6.17 JQuery UI

Dostáváme se ke konci příručky. Do teď jste byli seznamováni se základy používání jQuery. Ale díky tomu, že je jQuery zdarma, je na internetu k dispozici velké množství různých rozšíření (pluginy), které jsou většinou zdarma. Vytvořená rozšíření lze rozdělit do dvou základních skupin:

- Oficiální rozšíření, vytvořené vývojáři jQuery
- JQuery rozšíření od jednotlivců a firem

A právě oficiální rozšíření od vývojářů jQuery Vám budou v této kapitole představeny. Naleznete je na stránce jQuery UI²⁵, neboli jQuery User Interface²⁶.

Základní rozšíření, která v jQuery UI můžeme nalézt, jsou kalendář, dialog, taby (záložky), tlačítko, posouvátko (slider). V následujících dvou příkladech si ukážeme postup při použití kalendáře a tabů (záložek) na Vašich stránkách. Stejně jednoduché použití mají ale i ostatní dostupná rozšíření.

Navíc jQuery UI se neomezuje pouze na připravené objekty. Můžeme využít rozšíření pro přesouvání bloků po stránce – tzv. drag&drop, řazení bloků na stránce, výběr bloků a další.

6.17.1 Kalendář

S příklady začneme nejdříve u kalendáře. Vytvořte si v HTML kódu stránky jedno textové pole, které bude sloužit pro nastavení data. Aby byl příklad praktičtější, přidáme také *label*, který bude popisovat funkci textového pole.

```
<label for="kalendar">Výběr data z kalendáře:</label>  
<input type="text" name="kalendar" id="kalendar" value="" />
```

²⁵ <http://jqueryui.com>

²⁶ Český překlad: User Interface = Uživatelské rozhraní

Nyní máte dvě možnosti. Buď si rozšíření kalendář stáhnout přímo ze stránek jqueryui.com nebo jej naleznete na přiloženém CD.

ZIP archiv na CD

Potřebný ZIP archiv naleznete v adresáři **jqueryui** pod názvem **datepicker.zip**.

Stažení ZIP archivu z internetu

1. Otevřete si stránku **jqueryui.com**
2. Klikněte na menu položku **Download**
3. Klikněte na odkaz **Deselect all components** (je to z toho důvodu, abyste si zbytečně nestahovali všechna rozšíření)
4. Sjedťte na stránce o něco dolů a v sekci **Widgets** zaškrtněte položku **Datepicker**
5. Po zaškrtnutí položky Datepicker by se Vám také měla zaškrtnout položka **Core** (jádro) ze sekce **UI Core** (úplně nahoře) – případně ji zaškrtněte
6. **Vpravo**, nad tlačítkem Download **vyberte verzi 1.8** (měla by být vybrána automaticky)
7. Stáhněte si vygenerovaný ZIP archiv pomocí tlačítka **Download** a uložte si jej k sobě na disk

Ať už jste si archiv stáhli nebo jste si jej našli na přiloženém CD, rozbalte si jej a otevřete si jeho obsah. Uvnitř je několik adresářů:

- **css** – zde naleznete kaskádové styly, které ovlivňují vzhled kalendáře
- **development-bundle** – zde naleznete kompletní zdrojové kódy v případě, že byste se chtěli dozvědět více o tom, jak je objekt naprogramován a také zde jsou příklady, které naleznete na stránce jqueryui
- **js** – zde naleznete JavaScriptové kódy připravené k použití

a soubor **index.html**.

Použití kalendáře

JavaScriptové soubory si nakopírujte ke stránce, která bude zobrazovat kalendář a vložte je do HTML kódu stránky. Například takto:

```
<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
<script type="text/javascript" src="js/kalendar/jquery-ui-1.8.custom.min.js"></script>
```

Adresář **ui-lightness** z adresáře **css** si překopírujte ke svým kaskádovým stylům a do stránky si vložte styl, který naleznete uvnitř adresáře.

```
<link rel="stylesheet" type="text/css" href="css/kalendar/ui-lightness/jquery-ui-1.8.custom.css" media="screen" />
```

Aby Vám kalendář začal fungovat, musíte udělat poslední krok. Tím je vložení následujícího kódu do JavaScriptu.

```
$('#kalendar').datepicker();
```

Tímto JavaScriptem jste zajistili napojení kalendáře na Vaše textové pole. Ve výchozím nastavení se kalendář zobrazí po kliknutí do textového pole.

Pokud by byl kalendář příliš velký, může to být používáním relativní velikosti písma. V kaskádových stylech, které jsou vytvořené pro příklady příručky tomu tak je a kalendář je tak příliš velký.

Stačí však, abyste v souboru *jquery-ui-1.8.custom.css* přepsali velikost písma ve stylu *.ui-widget* z *font-size: 1.1em;* na *font-size: 0.8em;* a kalendář bude mít přijatelnější rozměry.

Kalendář také můžeme dále nastavovat podle parametrů, které naleznete na stránce jqueryui.com v sekci Datepicker. Můžeme například nastavit formát data.

```
$('#kalendar').datepicker({
    dateFormat: 'dd.mm.yy'
});
```

Ale také zobrazení ikonky pro zobrazení kalendáře, určení povoleného rozsahu kalendáře, nastavit jazyk kalendáře a spoustu dalšího.

6.17.2 Taby (záložky)

Taby (záložky) na svých stránkách můžeme používat například pro přehledné třídění textu do sekcí bez nutnosti znovu obnovovat stránku, protože přepínání mezi taby probíhá v reálném čase.

Pro použití záložek opět potřebujeme ZIP archiv s nutnými zdrojovými kódy a kaskádovými styly.

ZIP archiv na CD

Potřebný ZIP archiv naleznete v adresáři **jqueryui** pod názvem **taby.zip**.

Stažení ZIP archivu z internetu

8. Otevřete si stránku **jqueryui.com**
9. Klikněte na menu položku **Download**
10. Klikněte na odkaz **Deselect all components** (je to z toho důvodu, abyste si zbytečně nestahovali všechna rozšíření)
11. Sjeďte na stránce o něco dolů a v sekci **Widgets** zaškrtněte položku **Tabs**
12. Po zaškrtnutí položky Tabs by se Vám také měly zaškrtnout položky **Core** a **Widget** ze sekce **UI Core** (úplně nahoře) – případně je zaškrtněte
13. **Vpravo**, nad tlačítkem Download **vyberte verzi 1.8** (měla by být vybrána automaticky)
14. Stáhněte si vygenerovaný ZIP archiv pomocí tlačítka **Download** a uložte si jej k sobě na disk

Po stažení a následném rozbalení archivu naleznete i zde níže zmíněné tři adresáře a jeden soubor:

- **css** – zde naleznete kaskádové styly, které ovlivňují vzhled tabů

- **development-bundle** – zde naleznete kompletní zdrojové kódy v případě, že byste se chtěli dozvědět více o tom, jak je objekt naprogramován a také zde jsou příklady, které naleznete na stránce jqueryui
- **js** – zde naleznete JavaScriptové kódy připravené k použití

a soubor **index.html**.

Použití tabů (záložek)

JavaScriptové soubory si nakopírujte ke stránce, která bude zobrazovat kalendář a vložte je do HTML kódu stránky. Například takto:

```
<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
<script type="text/javascript" src="js/taby/jquery-ui-1.8.custom.min.js"></script>
```

Adresář **ui-lightness** z adresáře **css** si překopírujte ke svým kaskádovým stylům a do stránky si vložte styl, který naleznete uvnitř adresáře.

```
<link rel="stylesheet" type="text/css" href="css/taby/ui-lightness/jquery-ui-1.8.custom.css" media="screen" />
```

V tuto chvíli nemáme na stránce žádný obsah, na který by bylo možné taby aplikovat a tak si můžeme vypůjčit část zdrojového kódu ze souboru **index.html**, který hned při otevření staženého archivu.

Aby taby správně fungovaly, je nutné mít v HTML kódu stránky jeden seznam s položkami, které budou ve finále plnit funkci tabů a pak několik elementů DIV, které budou tvořit obsah tabů. Celý obsah – taby i jejich obsah bude uzavřen v jednom elementu DIV, na který budou taby aplikovány.

```
<div id="taby">
  <ul>
    <li><a href="#tab1">První</a></li>
    <li><a href="#tab2">Druhý</a></li>
    <li><a href="#tab3">Třetí</a></li>
  </ul>
```

```

    <div id="tab1">Lorem ipsum dolor sit amet, consectetur
    adipiscing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea commodo
    consequat.</div>

    <div id="tab2">Phasellus mattis tincidunt nibh. Cras orci
    urna, blandit id, pretium vel, aliquet ornare, felis. Maecenas
    scelerisque sem non nisl. Fusce sed lorem in enim dictum
    bibendum.</div>

    <div id="tab3">Nam dui erat, auctor a, dignissim quis,
    sollicitudin eu, felis. Pellentesque nisi urna, interdum eget,
    sagittis et, consequat vestibulum, lacus. Mauris porttitor
    ullamcorper augue.</div>
</div>

```

Pro napojení funkce tabů na náš seznam už nyní pouze stačí následující, krátký JavaScriptový kód:

```
$('#taby').tabs();
```

Pro účely této příručky bylo stejně jako v předchozím příkladu zmenšeno písmo v souboru *jquery-ui-1.8.custom.css*, protože taby byly příliš velké. Přepsána byla velikost písma ve stylu *.ui-widget* z *font-size: 1.1em;* na *font-size: 0.9em;* a kalendář bude mít přijatelnější rozměry.

Stejně tak jako kalendář, i taby můžeme dále nastavovat. Veškeré možnosti naleznete na stránce jqueryui.com v Tabs. Příkladem úpravy může být změna způsobu otevírání tabů. Tab se bude otevírat bez nutnosti kliknutí – bude stačit přejet přes něj myší.

```

$('#taby').tabs({
    event: 'mouseover'
});

```

6.18 Poznámka na okraj

V této příručce nebyly popsány všechny funkce frameworku jQuery. Dokonce i některé funkce mají ještě více možností, než bylo v příručce popsáno – například funkce *css()* má spoustu možností variant parametrů.

Avšak účelem nebylo přeložit kompletně dokumentaci. Účelem této příručky, respektive bakalářské práce bylo seznámení s frameworkem jQuery a popsání jeho možností vytvořením příručky.

7 Závěrečné shrnutí

7.1 Použitelnost příručky

Během tvorby bakalářské práce byla vytvořena příručka pro práci s frameworkem jQuery a jeho využití při tvorbě webových aplikací. Příručka je psána pro uživatele, kteří již mají alespoň základní zkušenosti s tvorbou webových stránek. Je to z toho důvodu, že se práce nezabývá základy tvorby webových stránek, ale využívá základní znalosti jazyku HTML nebo XHTML, kaskádových stylů a JavaScriptu.

Příručku může použít jakýkoliv správce stránek, který bude chtít zjednodušit, zrychlit a zpříjemnit ovládání webové stránky. Jelikož byly všechny vytvořené příklady otestovány v nejpoužívanějších prohlížečích, nemusí se správci stránek obávat o nepředvídatelné problémy spojené s nekompatibilitou prohlížečů.

7.2 Kompatibilita s prohlížeči

Příklady byly otestovány v prohlížečích, které jsou vypsány v následujícím seznamu. Za jménem prohlížeče je uvedena jeho verze.

- Internet Explorer 8
- Opera 10.51
- Google Chrome 4.1
- Mozilla Firefox 3.6.3
- Safari 4.0.4

Framework jQuery pracoval správně ve všech uvedených prohlížečích. Chyby nebo nedokonalosti, které byly objeveny, jsou především chybou zobrazování kaskádových stylů a byly popsány v podkapitolách, následujících za příklady, kde se chyba objevila.

Jediné nedostatky byly zpočátku zaznamenány v Opeře, která do verze 10.50 neuměla ošetřit pravé tlačítko myši. Od verze 10.50 však už pravé tlačítko lze ošetřit pomocí události stejně tak jako v ostatních prohlížečích.

Přehled použité literatury

- [1] RESIG, John. *JQuery: The Write Less, Do More, JavaScript Library* [online]. c2009 [cit. 2009-12-14]. Dostupný z WWW: <<http://jquery.com/>>.
- [2] BAKAUS, Bakaus. *JQuery UI* [online]. c2009 [cit. 2009-12-14]. Dostupný z WWW: <<http://jqueryui.com/>>.
- [3] DARIE, Christian, BRINZAREA, Bogdan, CHERECHES-TOSA, Filip, BUCICA, Mihai. *AJAX a PHP : tvoříme interaktivní webové aplikace profesionálně*. [s.l.] : Zoner Press, 2006. 320 s. ISBN 80-86815-47-1.
- [4] ŠKULTÉTY, Rastislav. *JavaScript : Programujeme internetové aplikace*. 2. aktualiz. vyd. [s.l.] : Computer Press, 2004. 224 s. ISBN 80-251-0144-4.
- [5] *W3Schools Online Web Tutorials* [online]. 1999 [cit. 2010-01-20]. Dostupný z WWW: <<http://www.w3schools.com/>>.
- [6] *Learning jQuery - Tips, Techniques, Tutorials* [online]. 2006 [cit. 2010-01-20]. Dostupný z WWW: <<http://www.learningjquery.com/>>.
- [7] JANOVSKEÝ, Dušan. *Jak psát web* [online]. 2010 [cit. 2010-04-20]. CSS styly - úvod. Dostupné z WWW: <<http://www.jakpsatweb.cz/css/css-uvod.html>>.
- [8] JANOVSKEÝ, Dušan. *Jak psát web* [online]. 2010 [cit. 2010-04-20]. Hover. Dostupné z WWW: <<http://www.jakpsatweb.cz/enc/hover.html>>.
- [9] JANOVSKEÝ, Dušan. *Jak psát web* [online]. 2010 [cit. 2010-04-20]. Dostupné z WWW: <<http://www.jakpsatweb.cz>>.
- [10] MELČÁK, Jiří. *Profi magazín* [online]. 2009 [cit. 2010-03-08]. JQuery: změna průhlednosti při hoveru (changing opacity on hover). Dostupné z WWW: <<http://www.profigazin.cz/jquery/jquery-zmena-pruhlednosti-pri-hoveru-changing-opacity-on-hover>>.
- [11] *Dynamic Drive* [online]. 2008 [cit. 2010-04-20]. Show Hint script. Dostupné z WWW: <<http://www.dynamicdrive.com>>.
- [12] *JQuery* [online]. 2010 [cit. 2010-04-20]. JQuery API. Dostupné z WWW: <<http://api.jquery.com>>.

- [13] *Microsoft* [online]. 2010 [cit. 2010-04-20]. Co je soubor cookie?. Dostupné z WWW:
<<http://www.microsoft.com/cze/athome/security/computer/whatis/whatiscookie.mspx>>.
- [14] *Diskusní fórum o webdesignu* [online]. 2006 [cit. 2010-04-20]. Funkce trim v js?. Dostupné z WWW:
<<http://diskuse.jakpsatweb.cz/?action=vthread&forum=8&topic=33574>>.

Seznam příloh

[1] Zdrojové kódy příkladů na přiloženém CD