KAPITOLA 3

Základy knihovny jQuery

Dříve v této knize jsme si řekli, že knihovna jQuery UI se zakládá na knihovně jQuery. Rozhodně bychom tedy neměli na knihovnu jQuery zapomenout, protože základní teoretické znalosti této knihovny bychom měli mít, abychom mohli nerušeně pracovat s knihovnou jQuery UI. V této kapitole se proto dozvíte:

- Co to je knihovna jQuery.
- Kde lze sehnat tuto knihovnu.
- Kde se nachází její oficiální dokumentace.
- Z čeho se knihovna jQuery skládá.
- Jak používat selektory knihovny jQuery.

Co je knihovna jQuery

Knihovna jQuery je velmi oblíbenou javascriptovou knihovnou. Základním cílem této knihovny je smazat rozdíly mezi implementacemi jazyka JavaScript v různých webových prohlížečích. Pokud jste někdy psali rozsáhlejší skript v jazyce JavaScript, zajisté víte, jak úmorné může být dosáhnout toho, aby tento skript fungoval ve všech hojně používaných webových prohlížečích stejně. Někdy se jedná až o téměř nadlidský úkol.

Knihovna jQuery se nesnaží zavést objektově-orientovaný způsob programování v jazyce JavaScript, ale místo toho se drží programování řízeného událostmi. Současně s tím se snaží o minimalizaci délky kódu, proto zavádí princip řetězení volání svých metod. Brzy zjistíte, že to, co byste normálně napsali v jazyce JavaScript na deset řádků, můžete s knihovnou

jQuery napsat na jeden řádek kódu. Co je však důležitější – přestože se vám zápis kódu takto zkrátí, bude fungovat (většinou) ve všech moderních webových prohlížečích stejně.

První verzi knihovny jQuery vydal známý programátor John Resig 26. srpna 2006. Tehdy ji vydal především proto, aby usnadnil práci sám sobě. Stejně jako nikoho jiného ho totiž nebavilo bojovat s rozdíly mezi webovými prohlížeči a také si chtěl ulehčit hledání a manipulaci s elementy v dokumentech HTML. Od té doby se rozrostla nejen samotná knihovna, ale rovněž její vývojový tým. K vývoji knihovny jQuery, jakožto knihovny s otevřeným zdrojovým kódem, se přidala řada dobrovolníků. Opět platí, že ačkoliv svou práci dělají rádi a zdarma, dobrovolným příspěvkem jistě nepohrdnou (*http://jquery.org/donate/*).

Autoři knihovny jQuery nabízejí tuto knihovnu pod licencemi MIT a GPL. Jedná se tedy o stejné licence jako u knihovny jQuery UI. Více informací a plné texty těchto licencí najdete na adrese *http://jquery.org/license/*. Nedělitelnou součástí knihovny jQuery je i selektorové jádro Sizzle, jehož autorem je společnost Dojo Foundation a nabízí jej po licencemi MIT, GPL a BSD. Chcete-li se dozvědět více informací o licenci BSD, najdete je kupříkladu zde: *http://cs.wikipedia.org/wiki/BSD_licence*.

UPOZORNĚNÍ

Je důležité si uvědomit, že pokud použijete ve své webové aplikaci knihovnu jQuery UI, používáte ve skutečnosti tři samostatné produkty a u každého z nich musíte přijmout některé licenční ujednání a řídit se jím. Krokem správným směrem je tedy nemazat z příslušných skriptů těchto produktů komentářové bloky s autorskými právy na jejich začátku, a to ani z produkčních (minifikovaných) verzí.

Stažení knihovny jQuery

Již víme, že každý balík s knihovnou jQuery UI obsahuje také knihovnu jQuery, a to jak v produkční, tak ve vývojové verzi. Knihovnu jQuery však můžete stáhnout a používat také samostatně. Všechny nezbytné informace a odkazy ke stažení knihovny jQuery včetně archivu se staršími verzemi najdete na adrese *http://docs.jquery.com/Downloading_jQuery*.

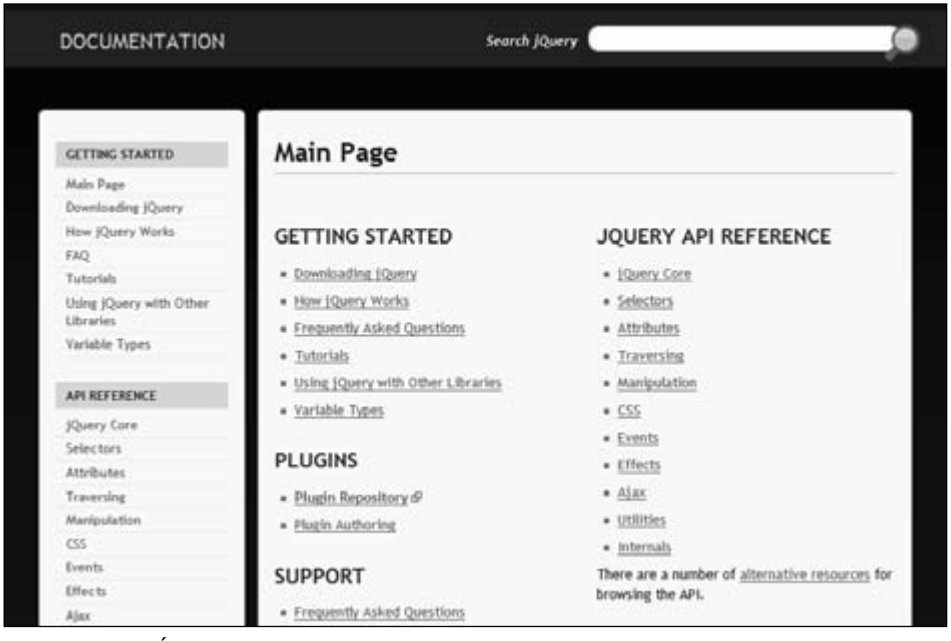
Jestliže nechcete stahovat tuto knihovnu, můžete použít některou z dostupných sítí CDN – síť Google Ajax API CDN, síť Microsoft CDN nebo síť jQuery CDN (jednoduše se pomocí elementu `script` odkážete na soubor přímo na síti CDN). Drobnou nevýhodou sítí CDN je, že jste vázáni na dostupnost připojení k Internetu a dostupnost těchto sítí, což vzhledem ke stabilitě těchto sítí a dostupnosti připojení není až tak zlé (u produkčních verzí webových aplikací nelze dostupnost připojení považovat za nevýhodu vůbec, protože musí být k dispozici stejně).

Další nevýhodou sítí CDN je o něco větší zpoždění při požadavku na příslušný skript, protože ten se přece jen nachází na jiném serveru než webová aplikace. Tato nevýhoda se však stává výhodou (a dosáhnete tedy menšího zpoždění požadavku), pokud nasazujete svou webovou aplikaci pro mezinárodní použití. Síť CDN se totiž skládá z mnoha serverů v různých částech světa a váš skript se vždy načte ze zdroje, který má fyzicky nejblíže ke klientovi.

K dispozici je rovněž celá řada zásuvných modulů pro knihovnu jQuery. Tyto zásuvné moduly jsou přehledně rozdělené do různých kategorií, jak se můžete sami přesvědčit na adrese *http://plugins.jquery.com/*. Nechybí zde ani vyhledávání zásuvných modulů.

Oficiální dokumentace

Oficiální dokumentaci knihovny jQuery najdete na adrese *http://docs.jquery.com/Main_Page*. Tato dokumentace je opravdu dobře zpracovaná, jak už jsme si zvykli z knihovny jQuery UI. Všechny informace jsou rozdělené do kategorií a u každé dokumentované části knihovny jQuery se kromě popisu nacházejí ukázky kódu, náhledy výsledků a komentáře uživatelů. Jestliže potřebujete rychle najít konkrétní informaci, můžete použít vyhledávání. Jak vypadá úvodní strana dokumentace knihovny jQuery, ukazuje obrázek 3.1.



Obrázek 3.1: Úvodní strana dokumentace knihovny jQuery

Základní stavební kameny knihovny jQuery

Nejdůležitější součástí knihovny jQuery, bez které se neobejdeme, je funkce se stejnojmenným názvem `jQuery()`. Tato funkce současně vyvolává nejvíce zmatení jak mezi začínajícími, tak mezi pokročilými programátory. Vzhledem k tomu, že má více druhů využití a různé pojmenování, někteří programátoři považují kód napsaný v knihovně jQuery za nový programovací jazyk. Není tomu tak – stále se jedná o kód jazyka JavaScript; jen to chce dobrou orientaci v terminologii a schopnostech knihovny jQuery a jazyka JavaScript.

Nesprávně tuto funkci popisuje i řada knih věnovaných přímo knihovně jQuery, takže si v tom pojďme udělat jasno.

POZNÁMKA

Pokud vám v této části kapitoly nebude všechno zcela jasné, nezoufejte – v klidu si projděte příklady v následujících kapitolách, a až podlehnete své zvědavosti, jak to celé vlastně funguje, vraťte se sem znovu.

OBJEKTY V JAZYCE JAVASCRIPT

Pokud jste nikdy nepracovali s objekty v jazyce JavaScript, zastavte se na chvíli a přečtěte si tento popis – velmi vám to usnadní pochopení následujících termínů a hlavně se vám bude lépe pracovat s příklady v této knize.

Ve většině objektově orientovaných programovacích jazyků se setkáváme s pojmy jako **třída** a **objekt** (nebo také **instance třídy**). Jazyk JavaScript je trochu zvláštní programovací jazyk – třídy v něm nenajdeme, zato se hemží objekty. Pomineme fakt, že v tomto jazyce lze simulovat rovněž třídy, jelikož toto téma je už nad rámec této knihy, a zaměříme se čistě na objekty.

Jako příklad objektu můžeme vzít jakýkoliv objekt z reálného světa – například představme si, že jsme si nedávno koupili automobil Opel Corsa (nemůžeme říct prostě jen „automobil“, protože toto slovo označuje celou třídu objektů, ale my máme na mysli jen náš jediný konkrétní objekt). Naše smyšlené auto má mnoho vlastností se specifickými hodnotami – jeho výrobcem je společnost Opel, název modelu je Corsa A, kód VIN je WOL00009994022569, motor má výkon 59 kW atd. Dále má auto řadu funkcí – za všechny jmenujme kupříkladu, že se umí rozjet a zastavit.

Takový automobil by vypadal v jazyce JavaScript například následovně:

```
var nasAutomobil = {
  vyrobce: 'Opel',
  model: 'Corsa A',
  VIN: 'WOL00009994022569',
  vykon: 59,
  rozjedSe: function() {
    ...
  },
  zastavSe: function() {
    ...
  },
  ...
};
```

K **vlastnostem** a **metodám** objektu přistupujeme tak, že jejich název napíšeme za název objektu následovaný tečkou. Kdybychom chtěli kupříkladu vypsat hodnotu vlastnosti `model` objektu `nasAutomobil` do standardního výstražného okna webového prohlížeče, udělali bychom to takto:

```
window.alert(nasAutomobil.model);
```

Vlastnost je tudíž proměnná (nebo konstanta), která náleží objektu. Alternativně lze k vlastnostem objektu přistupovat stejně jako k prvkům pole (tento postup je vhodný zejména tehdy, když neznáme název vlastnosti předem):

```
window.alert(nasAutomobil[model]);
```

Stejným způsobem můžeme změnit hodnotu vlastnosti – stačí uvést daný výraz na levé straně příkazu přiřazení:

```
nasAutomobil.model = 'Insignia';
```

nebo také:

```
nasAutomobil[model] = 'Insignia';
```

Takto můžeme dokonce přidávat nové vlastnosti objektu:

```
nasAutomobil.rokVyroby = 2009;
```

Předchozím řádkem jsme rozšířili definici našeho objektu o vlastnost `rokVyroby` s hodnotou `2009`.

Metoda je funkce, která patří k nějakému objektu. Od funkce se liší tedy jen tak, že ji nevoláme samostatně, ale uvedeme před ní název objektu s tečkou. V našem objektu jsme si ukázali dvě metody – metodu `rozjedSe()` a metodu `zastavSe()`. Kdybychom chtěli zavolat například metodu `rozjedSe()`, napsali bychom:

```
nasAutomobil.rozjedSe();
```

Jistou zvláštností jazyka JavaScript je, že všechny funkce a metody jsou současně také objekty. Co to znamená? Znamená to, že kromě toho, že je můžeme volat, můžeme jim přiřazovat vlastnosti a metody a pak s nimi pracovat. Přepíšeme si tedy náš ukázkový objekt `nasAutomobil` jako funkci:

```
function nasAutomobil() {
  window.alert(nasAutomobil.vyrobce + ' ' + nasAutomobil.model);
}

nasAutomobil.vyrobce = 'Opel';
nasAutomobil.model = 'Corsa A';
nasAutomobil.VIN = 'WOL000009994022569';
nasAutomobil.vykon = 59;
nasAutomobil.rozjedSe = function() {
  ...
};
nasAutomobil.zastavSe = function() {
  ...
};
...
```

Jak je patrné, s takto vytvořeným objektem `nasAutomobil` pracujeme stejně jako s původním objektem, ale navíc ho můžeme volat jako funkci `nasAutomobil()`. Tato funkce vypíše text s výrobcem a modelem našeho smyšleného auta do výstražného okna.

PARAMETRY A ARGUMENTY FUNKCÍ A METOD

Přestože některé odborné články a knihy nerozlišují termíny **argument** a **parametr**, v této knize je rozlišovat budeme. Pokud definujeme funkci nebo metodu, mluvíme zásadně o parametrech. Mějme kupříkladu následující definici funkce:

```
function mojeFunkce(parametr1, parametr2) { ... }
```

V definici funkce/metody říkáme, že daná funkce/metoda má parametry, takže v tomto případě řekneme, že funkce `mojeFunkce()` má dva parametry – parametr s názvem `parametr1` a parametr s názvem `parametr2`. **Parametr** je tudíž lokální proměnná dané funkce/metody, jejíž hodnotu můžeme nastavit zvenku.

Hodnotou parametru je právě **argument**. Ten nastavujeme při volání dané funkce/metody. Například takto:

```
mojeFunkce(5, 'ahoj');
```

Jak je patrné, argument je už konkrétní hodnota, kterou předáváme při volání funkce/metody (případně hodnota některé proměnné/konstanty). V tomto případě bychom mohli tedy prohlásit, že předáváme číslo `5` jako první argument a textový řetězec `'ahoj'` jako druhý argument funkci `mojeFunkce()`. Občas nebudeme označovat argument pořadím, ale názvem parametru, ke kterému patří. Nyní bychom mohli tudíž říct, že předáváme číslo `5` jako argument `parametr1` a textový řetězec `'ahoj'` jako argument `parametr2` funkci `mojeFunkce()`.

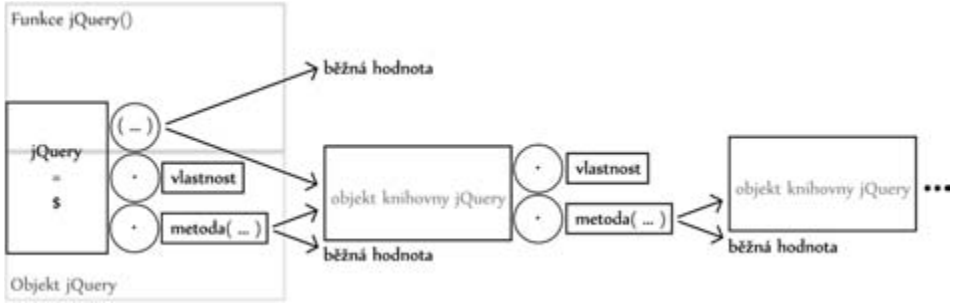
Začneme tak, že si uvedeme tři termíny, jež tvoří základ knihovny jQuery a které si lidé často pletou:

- **Funkce** `jQuery()` – hlavní funkce knihovny jQuery. Pracujeme s ní jako s jakoukoliv běžnou funkcí – tj. zavoláme ji a předáme jí nějaké argumenty, přičemž očekáváme, že něco provede a případně nám vrátí nějakou hodnotu. Tato funkce má více variant; to znamená, že podle počtu a typu argumentů, které jí předáváme při volání, provede různé akce. Různé varianty funkce `jQuery()` si popíšeme za malý okamžik.
- **Globální objekt** `jQuery` (někdy také jen **objekt** `jQuery`) – ve skutečnosti se jedná opět o funkci `jQuery()`, ale vzhledem k tomu, že v jazyce JavaScript je každá funkce také objektem, tentokrát na ni budeme pohlížet jako na objekt (toto je obdoba našeho objektu `nasAutomobil`, jejž jsme definovali jako funkci). S jakýmkoliv objektem můžeme dělat v základu tři věci – načítat hodnoty jeho vlastností, ukládat hodnoty jeho vlastností a volat jeho metody. Objekt `jQuery` se nijak neliší. Ukázku práce s tímto objektem si předvedeme dále v této části kapitoly.
- **Objekt knihovny jQuery** – je velmi důležité si všimnout, že v předchozím bodu tohoto seznamu (globální objekt `jQuery`) označujeme slovo „jQuery“ jako zdrojový kód ve větě. Není to náhoda – jednalo se totiž o jeden konkrétní objekt s názvem `jQuery`. Tentokrát však mluvíme o typu objektu. Většina variant funkce `jQuery()` a celá řada metod globálního objektu `jQuery` vrací objekt knihovny jQuery jako svou návratovou hodnotu. Objekt knihovny jQuery je opět jen obyčejný objekt, takže můžeme přistupovat k jeho vlastnostem a volat jeho metody. Převážná část jeho metod vrací

rovněž další objekt knihovny jQuery – z toho vychází princip řetězení volání metod, o kterém jsme se již bavili.

Funkce `jQuery()` má také svůj kratší alias `$()`. Stejně tak můžeme objekt `jQuery` zapisovat jako `$` (jelikož víme, že objekt `jQuery` je ve skutečnosti funkce `jQuery()`, na kterou nahlížíme z jiného úhlu pohledu).

Obrázek 3.2 znázorňuje právě popsané termíny.



Obrázek 3.2: Základní stavební kameny knihovny jQuery

Teorie už bylo až nad hlavu – pojďme si ukázat nějaké příklady.

Funkce jQuery()

Základní způsob použití (a také nejčastější způsob) funkce `jQuery()` spočívá ve výběru elementů dokumentu, na které potom můžeme aplikovat metody knihovny jQuery. Kdybychom kupříkladu chtěli zjistit, jaké jméno uživatel vyplnil do formuláře na stránce, provedli bychom to takto:

```
var jmenoUzivatele = jQuery('input[name=jmeno]').attr('value');
```

Nebo bychom tento řádek mohli napsat stručněji:

```
var jmenoUzivatele = $('input[name=jmeno]').attr('value');
```

Přestože je tento kód velmi krátký, odehrává se na něm hned několik věcí. Nejprve v části `$('input[name=jmeno]')` voláme funkci `jQuery()` a předáváme jí selektor jazyka CSS, což je obyčejný textový řetězec. O selektorech jazyka CSS, které můžeme používat v knihovně jQuery, si povíme více informací později v této kapitole. V tomto případě vybíráme pomocí selektoru `'input[name=jmeno]'` element `input`, jehož atribut `name` obsahuje hodnotu `jmeno`.

Volání `$('input[name=jmeno]')` nám vrátí bezejmenný objekt knihovny jQuery. Tento objekt může obalovat element `input` se jménem `jmeno`, pokud se daný element na stránce skutečně nachází, nebo žádný element v případě, že takový element neexistuje (proto se objekt knihovny jQuery někdy označuje také jako **obalená skupina knihovny jQuery**, přičemž může obalovat i více elementů). Následně voláme metodu `attr()` tohoto objektu knihovny jQuery, která nám vrátí hodnotu uvedeného atributu – v tomto případě atributu `value`, a to pro první element obsažený v daném objektu, je-li nějaký. Získanou hodnotu ukládáme do proměnné `jmenoUzivatele`.

Většina volání funkce `jQuery()` vrací objekt knihovny jQuery. Výsledek se však liší v tom, co tento objekt obaluje. Tato funkce se totiž chová jinak podle toho, kolik argumentů jí předáme a jakého jsou typu.

Jeden způsob využití už jsme viděli – v následující tabulce se nachází seznam všech možných způsobů použití. Tento seznam se může v průběhu vývoje knihovny jQuery měnit, proto je vhodné sledovat informace na adrese *http://api.jquery.com/jQuery/*. Kromě toho, že jsou tyto informace aktuální, jsou také mnohem podrobnější a doplněné o příklady.

Tabulka 3.1: Varianty funkce jQuery()

Účel	Varianta funkce	Popis parametrů
Hledání skupiny elementů v dokumentu	`jQuery(selektor[, kontext])`	■ `selektor` – textový řetězec, který reprezentuje selektor ■ `kontext` – element, dokument nebo objekt knihovny jQuery
	`jQuery(element)`	■ `element` – element, jejž chceme zabalit do objektu knihovny jQuery
	`jQuery(objekt)`	■ `objekt` – obyčejný objekt, který chceme zabalit do objektu knihovny jQuery
	`jQuery(poleElementu)`	■ `poleElementu` – pole obsahující skupinu elementů, které chceme zabalit do objektu knihovny jQuery
	`jQuery(objektKnihovnyjQuery)`	■ `objektKnihovnyjQuery` – objekt knihovny jQuery, který chceme klonovat
	`jQuery()`	Tato varianta žádný parametr nemá a vrací prázdný objekt knihovny jQuery
Tvorba elementů dokumentu z textového řetězce s kódem jazyka HTML	`jQuery(kodHTML[, dokument])`	■ `kodHTML` – textový řetězec obsahující kód jazyka HTML, z nějž chceme vytvořit elementy za běhu ■ `dokument` – dokument, v němž nové elementy vzniknou
	`jQuery(kodHTML, vlastnosti)`	■ `kodHTML` – textový řetězec s kódem jazyka HTML, ve kterém definujeme jediný element; například `'<div/>'` nebo `'<div></div>'` ■ `vlastnosti` – objekt, s jehož vlastnostmi specifikujeme atributy, události a metody, které by se měly zavolat na nově vzniklý element.
Spuštění funkce po načtení modelu DOM	`jQuery(funkceZpetnehoVolani)`	■ `funkceZpetnehoVolani` – funkce zpětného volání, kterou chceme zavolat po načtení modelu DOM

Velmi důležitým způsobem volání funkce `jQuery()`, který budeme používat v této knize, je poslední uvedený způsob (`jQuery(funkceZpetnehoVolani)`), jehož účelem je spuštění funkce po načtení modelu DOM. Model DOM je objektový model dokumentu, který webový prohlížeč skládá za účelem práce s dokumentem jazyka HTML v kódu jazyka JavaScript,

a to postupně, když načítá jednotlivé jeho elementy. Jakmile webový prohlížeč takto zpracuje celý dokument, můžeme v jazyce JavaScript hledat všechny elementy daného dokumentu a pracovat s nimi.

```
$(function() {
  ...
});
```

Do těla výše uvedené anonymní funkce můžeme tudíž napsat místo tří teček jakýkoliv kód, v němž chceme pracovat s elementy dokumentu – například vyhledat element, ze kterého bychom chtěli vytvořit dialogové okno pomocí ovládacího prvku Dialog, jak zjistíme v kapitole „Navigace v obsahu stránky". Díky tomu, že počkáme na načtení modelu DOM, máme jistotu, že element najdeme, pokud v dokumentu existuje. Jednoduše řečeno – jedná se o vhodný vstupní bod do programu (zvlášť do toho, v němž používáme komponenty knihovny jQuery UI), který tímto způsobem budeme používat napříč celou touto knihou.

Globální objekt jQuery

Globální objekt `jQuery` nabízí celou řadu vlastností a metod. Přehled všech jeho vlastností a metod lze najít na adrese *http://api.jquery.com/*. Snadno je poznáme tak, že před jejich názvem se nachází předpona `jQuery.`, přičemž za metodami následuje dvojice kulatých závorek, ale za vlastnostmi nikoliv. Nebudeme se zabývat tím, k čemu slouží všechny tyto vlastnosti a metody, ale začneme kupříkladu stručným popisem vlastnosti `jQuery.fx.off`.

Vlastnost `jQuery.fx.off` globálně zakazuje všechny animace:

```
jQuery.fx.off = true;
```

Předchozím příkazem přiřazení nastavujeme vlastnosti `fx.off` globálního objektu `jQuery` hodnotu `true`, čímž zakážeme všechny animace – to znamená, že animované elementy se okamžitě dostanou do svého cílového stavu, aniž by se animace jakkoliv projevila. Animace je možné zase zapnout přidělením hodnoty `false`. Znak tečky v názvu vlastnosti `fx.off` značí, že vlastnost `fx` je také objektem se svou vlastností `off`. Tento objekt nás však nezajímá – důležité je, že jsme si ukázali, jak změnit hodnotu vlastnosti globálního objektu `jQuery`.

Stejně tak bychom neměli zapomínat, že je možné napsat tento kód stručněji:

```
$.fx.off = true;
```

Jako ukázkovou metodu globálního objektu `jQuery` si zvolíme například metodu `jQuery.isArray()`. Tato metoda vrací hodnotu `true`, jestliže je jí předaný argument skutečným polem jazyka JavaScript. Pokud tedy spustíme tento kód:

```
var jeToPole = jQuery.isArray(['jablko', 'hruška']);
```

proměnná `jeToPole` bude obsahovat hodnotu `true`. A zde je kratší zápis tohoto kódu s naprosto stejným výsledkem:

```
var jeToPole = $.isArray(['jablko', 'hruška']);
```

Objekty knihovny jQuery

Už víme, že pod pojmem objekt knihovny jQuery si nemáme představovat žádný konkrétní objekt, ale celou skupinu (typ) objektů. Jak takový objekt získáme, již rovněž víme – většina variant funkce `jQuery()` a některé metody globálního objektu `jQuery` nám vracejí objekt knihovny jQuery jako svou návratovou hodnotu.

Objekt knihovny jQuery má stejně jako globální objekt `jQuery` spoustu vlastností a metod, které najdeme na adrese *http://api.jquery.com/*. Opět je velmi jednoduše rozlišíme – jejich název začíná tečkou (bez předpony `jQuery`), přičemž za metodami následuje dvojice kulatých závorek, ale za vlastnostmi nikoliv. S jednou metodou objektu knihovny jQuery jsme se už setkali při popisu funkce `jQuery()` – jednalo se o metodu `.attr()`.

```
var jmenoUzivatele = jQuery('input[name=jmeno]').attr('value');
```

Na tomto řádku jsme volali metodu `attr()` bezejmenného objektu knihovny jQuery, který jsme získali voláním `jQuery('input[name=jmeno]')`. Tato metoda vrací pro vybraný element `input` konkrétní textovou hodnotu (existuje-li) atributu `value`, kterou ukládáme do proměnné `jmenoUzivatele`. Na takovou hodnotu už nemůžeme volat další metody a řetězit volání metod, jak jsme si slibovali dříve. Jaká metoda objektu knihovny jQuery tedy kupříkladu vrací jiný objekt knihovny jQuery, na který můžeme volat další metodu, a tak stále dokola?

Uvedeme si například hned dvě takové metody – metodu `addClass()` a metodu `removeClass()`. První jmenovaná metoda (`addClass()`) přidá k vybrané skupině elementů třídy předané jako argument (více tříd oddělujeme mezerou) a metoda `removeClass()` je naopak odebírá. Obě tyto metody vracejí objekt knihovny jQuery; tj. stejnou skupinu elementů, které však mají upravený atribut `class`. Dejme tomu, že máme dokument s mnoha elementy `p`, které mají třídu `cernyOdstavec`. Budeme chtít všem těmto odstavcům v dokumentu odebrat třídu `cernyOdstavec` a přidat k nim třídu `modryOdstavec`:

```
$('p').removeClass('cernyOdstavec').addClass('modryOdstavec');
```

Jak je patrné, díky řetězení volání metod je řešení opravdu jednoduché. Nejprve vybíráme všechny elementy `p` v dokumentu voláním `$('p')`. Tím získáme objekt knihovny jQuery, na nějž voláme metodu `removeClass()` s argumentem `'cernyOdstavec'` a ta nám vrátí další objekt knihovny jQuery, jenž reprezentuje všechny elementy `p` v dokumentu, ale bez třídy `cernyOdstavec`.

Jelikož se opět jedná o objekt knihovny jQuery, můžeme na něho zavolat metodu `addClass` s argumentem `modryOdstavec`. Takto vznikne konečně náš finální objekt knihovny jQuery, který představuje všechny elementy `p` v dokumentu bez třídy `cernyOdstavec`, avšak doplněné o třídu `modryOdstavec`.

Kontrolní otázka: „Kolik funkcí jsme v tomto případě volali?“

Správná odpověď: „Volali jsme funkci `jQuery()` a dvě metody objektu knihovny jQuery, takže celkem tři funkce“.

Mezi nejznámější vlastnost objektu knihovny jQuery patří vlastnost `length`. Tato vlastnost nám sděluje, kolik elementů daný objekt obsahuje. Zde je příklad:

```
var pocetOdstavcu = $('p').length;
```

V předchozím kódu ukládáme počet elementů `p` v dokumentu do proměnné `pocetOdstavcu`, a to tak, že vybereme tyto elementy do objektu knihovny jQuery a načteme hodnotu jeho vlastnosti `length`.

Jak používat selektory knihovny jQuery

V minulé části této kapitoly jsme zjistili, že funkci `jQuery()` můžeme předat jako její první argument selektor v podobě textového řetězce. Tyto selektory vycházejí z jazyka CSS, přičemž knihovna jQuery navíc přidává spoustu vlastních selektorů. To je velmi výhodné, protože se nemusíme učit novou syntaxi selektorů od začátku.

Co to vlastně je selektor? **Selektor** je textový výraz, s jehož pomocí vyhledáváme elementy v dokumentu. V jazyce CSS obvykle na vyhledané elementy aplikujeme nějaká pravidla stylů, ale v knihovně jQuery si s nimi můžeme dělat, co se nám zlíbí – dokonce je můžeme z dokumentu zcela odstranit.

Zde jsou příklady standardních selektorů z jazyka CSS (do verze 3, a to včetně):

- `#identifikator` – odpovídá jedinému elementu s identifikátorem `identifikator`.
- `znacka` – vyhovuje všem elementům se značkou `znacka` (například `div`).
- `.trida` – reprezentuje všechny elementy s třídou `trida`.
- `[atribut="hodnota"]` – odpovídá všem elementům, jejichž atribut `atribut` má hodnotu `hodnota`, přičemž uvozovky musíme doplňovat pouze k víceslovným hodnotám nebo hodnotám se speciálními znaky.
- `znacka.trida` – vyhovuje všem elementům `znacka` s třídou `trida`.
- `znacka1, znacka2` – reprezentuje všechny elementy, které mají značku `znacka1` nebo `znacka2`.
- `znacka1 znacka2[atribut="hodnota"]` – odpovídá všem elementům `znacka2`, jejichž atribut `atribut` má hodnotu `hodnota`, a zároveň jsou potomky elementů `znacka1`.
- `:disabled` – reprezentuje všechny zakázané elementy.
- Atd.

A tady jsou ukázky nových selektorů knihovny jQuery:

- `:submit` – odpovídá všem elementům `input` s typem `submit` (odesílací tlačítko formuláře).
- `:password` – reprezentuje všechny elementy `input` typu `password`.
- `:file` – vyhovuje všem elementům `input` s typem `file`.
- Atd.

Kompletní seznam všech selektorů knihovny jQuery spolu s popisy a příklady lze najít na adrese *http://api.jquery.com/category/selectors/*.

Jakmile předáme selektor funkci `jQuery()`, vybere elementy z dokumentu a zabalí je do objektu knihovny jQuery. Co například dělá níže uvedené volání?

```
var pocetPriloh = $('#zprava :file.priloha').length;
```

Na tomto řádku vybíráme všechny elementy `input` s typem `file` a třídou `priloha`, které se nacházejí ve struktuře dokumentu pod elementem s identifikátorem `zprava` (tj. jsou jeho potomky). Vybrané elementy se obalují do objektu knihovny jQuery, jehož hodnotu vlastnosti `length` ukládáme do proměnné `pocetPriloh`.

Shrnutí

V této kapitole jsme se dozvěděli, co to je knihovna jQuery, kdo ji vytvořil a odkud ji můžeme stáhnout. Potom jsme si ukázali, kde se nachází její oficiální dokumentace, v níž najdeme všechny nezbytné informace rozdělené do kategorií.

Popsali jsme si, z jakých základních stavebních kamenů se knihovna jQuery skládá. Kromě toho, že umíme volat funkci `jQuery()`, víme také, jak používat globální objekt jQuery a objekty knihovny jQuery. Mimo jiné jsme zjistili, kde se bude nacházet vstupní bod do programů, které si popíšeme v této knize.

Nakonec jsme si představili selektory, s nimiž můžeme pracovat v knihovně jQuery. Tyto základní znalosti stačí pro důkladné pochopení zdrojových kódů v této knize.

Knihovna jQuery však velmi ulehčuje práci v jazyce JavaScript, a pokud vás tato kapitola zaujala, určitě si toto téma nastudujte podrobněji.